

David Naccache · Shouhuai Xu
Sihan Qing · Pierangela Samarati
Gregory Blanc · Rongxing Lu
Zonghua Zhang · Ahmed Meddahi (Eds.)

LNCS 11149

Information and Communications Security

20th International Conference, ICICS 2018
Lille, France, October 29–31, 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>


David Naccache · Shouhuai Xu
Sihan Qing · Pierangela Samarati
Gregory Blanc · Rongxing Lu
Zonghua Zhang · Ahmed Meddahi (Eds.)

Information and Communications Security


20th International Conference, ICICS 2018
Lille, France, October 29–31, 2018
Proceedings


Editors


David Naccache 
Département d'Informatique
Ecole Normale Supérieure
Paris, France

Shouhuai Xu 
University of Texas
San Antonio, TX, USA

Sihan Qing
Chinese Academy of Sciences
Beijing, China

Pierangela Samarati 
Dipto di Informatica
Univ degli Studi di Milano
Crema, Italy

Gregory Blanc 
Télécom SudParis
Evry, France

Rongxing Lu 
University of New Brunswick
Fredericton, NB, Canada

Zonghua Zhang 
IMT Lille Douai
Villeneuve-d'Ascq, France

Ahmed Meddahi 
IMT Lille Douai
Villeneuve-d'Ascq, France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-01949-5 ISBN 978-3-030-01950-1 (eBook)
<https://doi.org/10.1007/978-3-030-01950-1>

Library of Congress Control Number: 2018939454

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

Welcome to the proceedings of ICICS 2018, the 20th edition of the International Conference on Information and Communications Security! The ICICS conference was created in 1997 in Beijing, with the aim of bringing together researchers and practitioners from academia and industry to discuss and exchange their experiences, lessons learned, and ideas related to computer and communications security. ICICS has been held in different countries and cities. In particular, Beijing takes the lead and has hosted the event seven times (2017, 2015, 2013, 2011, 2009, 2005, and 1997), followed by another three venues in China (Zhengzhou 2007, Hohhot 2003, Xi'an 2001). ICICS took place twice, respectively, in Singapore (2016, 2002), Hong Kong (2014, 2012), and Spain (Barcelona 2010, and Malaga 2004). UK (Birmingham 2008), USA (Raleigh 2006), and Australia (Sydney 1999) hosted ICICS once.

ICICS 2018 was organized by IMT Lille Douai, which is the largest engineering graduate school in northern France and was held during October 29–31, 2018. The conference received 202 submissions from 38 countries and regions (the number of submissions was among the top in ICICS history). The Program Committee carefully selected 39 full papers and 11 short papers for presentation, resulting in an acceptance rate of 19.3% for full papers and 24.75 overall. This year, the Program Committee was composed of 69 security experts, including both experienced academics and rising stars in the security community, who are from 14 different countries and regions. Each Program Committee member reviewed about ten papers and each paper was assigned to three to four reviewers and evaluated in a double-blind manner. We believe that the conference program encompasses many papers advancing the current state of the art in information and communications security, including several cryptography-related topics (e.g., signature schemes, encrypted computing, searchable encryption, applications) as well as non-crypto topics such as attack detection and analysis, malware and network security, botnet security, and privacy. It is worth noting that, for the first time, a whole session was dedicated to blockchain technology. The program also featured three keynote speeches delivered by Prof. Adrian Perrig, Dr. Eric Freyssonet, and Prof. Kui Ren, representing three different security perspectives and research domains.

The success of ICICS 2018 relied on the tremendous efforts of many contributors. First of all, we would like to thank the authors for submitting their quality work to ICICS 2018, the members of the Program Committee, as well as the 97 external reviewers, for their hard work and contributions in selecting papers, which was no doubt a time- and energy-consuming process. In particular, we are grateful to the honorary general chair, Alain Schmitt, president of IMT Lille Douai, who gave strong encouragement and full support to organize this event on campus. Thanks also go to the general chair, Zonghua Zhang, the co-chair Ahmed Meddahi, and their local organizing team members, Qipeng Song, Montida Pattaranantakul, Fatima Semmoudi, Christine Conreur, and Isabelle Fabresse, who made significant contributions to ensure that ICICS 2018 ran smoothly. Especially Qipeng and Montida who were heavily involved

in the entire management lifecycle of the conference, from setting up the website to handling registrations to arranging logistics. We are also indebted to Martine Ducornet, the industry liaison chair, who extensively solicited sponsorship from the industrial partners. We had five sponsors from both industry and academia, including Axians, Ernst & Young, the I-Site Université Lille Nord Europe (ULNE), the CNRS SAMOVAR laboratory, and the Centre d'Innovation des Technologies sans Contact (CITC). We would like to express our gratitude not only for their generous financial support, but also for their encouragement of the research community in security.

We hope that all the participants enjoyed ICICS 2018 and its program!

October 2018

David Naccache
Shouhuai Xu
Sihan Qing
Pierangela Samarati
Gregory Blanc
Rongxing Lu

Organization

Program Committee

Man Ho Au	The Hong Kong Polytechnic University, SAR China
Joonsang Baek	University of Wollongong, Australia
Gregory Blanc	Télécom SudParis, Institut Mines-Télécom, France
Zhenfu Cao	East China Normal University, China
Stéphane Cauchie	equensWorldline, France
Qian Chen	University of Texas at San Antonio, USA
Songqing Chen	George Mason University, USA
Xiaofeng Chen	Xidian University, China
Jin-Hee Cho	U.S. Army Research Laboratory, USA
Gouenou Coatrieux	IMT Atlantique, Institut Mines-Télécom, France
Frédéric Cuppens	IMT Atlantique, Institut Mines-Télécom, France
Nora Cuppens-Boulahia	IMT Atlantique, Institut Mines-Télécom, France
Jean-Luc Danger	Télécom ParisTech, Institut Mines-Télécom, France
Haixin Duan	Tsinghua University, China
Doudou Fall	Nara Institute of Science and Technology, Japan
Anthony Fleury	IMT Lille Douai, Institut Mines-Télécom, France
Xinwen Fu	University of Central Florida, USA
Debin Gao	Singapore Management University, Singapore
Joaquin Garcia-Alfaro	Télécom SudParis, Institut Mines-Télécom, France
Angelo Genovese	University of Milan, Italy
Dieter Gollmann	Hamburg University of Technology, Germany
Yuan Hong	Illinois Institute of Technology, USA
Hongxin Hu	Clemson University, USA
Shouling Ji	Zhejiang University, China
Dong Jin	Illinois Institute of Technology, USA
Sokratis Katsikas	Norwegian University of Science and Technology, Norway
Nizar Kheir	Thales Group, France
Houda Labiod	Télécom ParisTech, Institut Mines-Télécom, France
Maryline Laurent	Télécom SudParis, Institut Mines-Télécom, France
Wonjun Lee	University of Texas at San Antonio, USA
Shujun Li	University of Kent, UK
Xiaodong Lin	University of Ontario Institute of Technology, Canada
Yang Liu	Nanyang Technological University, Singapore
Yao Liu	University of South Florida, USA
Zhe Liu	Nanjing University of Aeronautics and Astronautics, China
Giovanni Livraga	University of Milan, Italy

Rongxing Lu	University of New Brunswick, Canada
Di Ma	University of Michigan, USA
Jean-Yves Marion	Loria, Université de Lorraine and Institut Universitaire de France, France
Atsuko Miyaji	Japan Advanced Institute of Science and Technology, Japan
Aziz Mohaisen	University of Central Florida, USA
David Naccache	Ecole Normale Supérieure, France
Takashi Nishide	University of Tsukuba, Japan
Sihan Qing	Chinese Academy of Sciences and Peking University, China
Pierangela Samarati	University of Milan, Italy
Sandra Scott-Hayward	Queen's University Belfast, UK
Zhiyong Shan	Wichita State University, USA
Seungwon Shin	KAIST, South Korea
Chunhua Su	University of Aizu, Japan
Kun Sun	George Mason University, USA
Mario Südholt	IMT Atlantique, Institut Mines-Télécom, France
Qiang Tang	Luxembourg Institute of Science and Technology, Luxembourg
Qiang Tang	New Jersey Institute of Technology, USA
Juan Tapiador	Universidad Carlos III de Madrid, Spain
Boyang Wang	University of Cincinnati, USA
Wei Wang	Beijing Jiaotong University, China
Zhi Wang	Florida State University, USA
Lei Xu	University of Houston, USA
Shouhuai Xu	University of Texas at San Antonio, USA
Guanhua Yan	State University of New York at Binghamton, USA
Qiben Yan	University of Nebraska-Lincoln, USA
Min Yang	Fudan University, China
Yanfang Ye	West Virginia University, USA
Yong Yu	Shaanxi Normal University, China
Fengwei Zhang	Wayne State University, USA
Yongbin Zhou	Institute of Information Engineering, Chinese Academy of Sciences, China
Haojin Zhu	Shanghai Jiaotong University, China
Sencun Zhu	Pennsylvania State University, USA

Additional Reviewers

Berri, Sara	Chen, Jiageng
Castiglione, Arcangelo	Chen, Long
Chaudhuri, Sumanta	Chen, Shiqing
Chen, Hua	Cheng, Wei

Coronado, Andres Hernandez
Dey, Rajib
Ding, Fei
Ding, Wenbo
Duc, Guillaume
Fan, Chun-I
Fang, Song
Gao, Chao
Gao, Fei
Gao, Zhimin
Graba, Tarik
Gunnam, Ganesh Reddy
Guo, Jingjing
Hammi, Badis
Hao, Jialu
Hu, Kexin
Huang, Cheng
Huang, Hui
Huo, Wei
Ito, Ryoma
Jia, Keting
Jin, Shuyuan
Karamchandani, Neeraj
Kühne, Ulrich
Lee, Yongjae
Li, Hongda
Li, Meng
Li, Ming
Li, Yannan
Li, Yue
Liang, Jinjin
Liu, Bingyu
Liu, Songsong
Lu, Xingye
Lu, Yuan
Ma, Hui
Ma, Xu
Mamun, Mohammad
Markwood, Ian
Matsuzaki, Natsume
May, Alexander
Mimoto, Tomoaki
Ming, Tang
Moore, Ciara
Msahli, Mounira
Nadim, Mohammad
Okumura, Shinya
Rafferty, Ciara
Ren, Bingfei
Shen, Dakun
Sun, Jianhua
Sun, Siwei
Sun, Xin
Sun, Yuanyi
Tan, Gaosheng
Tang, Xiaoxiao
Tao, Yang
Tochikubo, Kouya
Villain, Jonathan
Vishwamitra, Nishant
Wan, Shengye
Wang, Han
Wang, Jianfeng
Wang, Kunpeng
Wang, Qingju
Wang, Tao
Wang, Xiaolei
Wang, Xiaoshan
Wang, Xinda
Wang, Yunling
Wei, Feng
Xie, Shangyu
Xu, Li
Yanai, Naoto
Yang, Changsong
Yang, S. J.
Yu, Yu
Yuan, Haoran
Yue, Qinggang
Zhang, Hailong
Zhang, Jun
Zhang, Peng
Zhang, Qian
Zhang, Xiaoyu
Zhang, Yinghui
Zhang, Yue
Zhao, Yongjun
Zhou, Wenxuan
Zhu, Youwen

Contents

Full Paper Session I: Blockchain Technology

- Blockchain-Based Secure Data Provenance for Cloud Storage. 3
Yuan Zhang, Xiaodong Lin, and Chunxiang Xu
- uMine: A Blockchain Based on Human Miners 20
Henning Kopp, Frank Kargl, Christoph Bösch, and Andreas Peter

Full Paper Session II: Malware, Botnet and Network Security

- LagProber: Detecting DGA-Based Malware by Using Query Time Lag
of Non-existent Domains 41
Xi Luo, Liming Wang, Zhen Xu, and Wei An
- Study on Advanced Botnet Based on Publicly Available Resources. 57
Jie Yin, Heyang Lv, Fangjiao Zhang, Zhihong Tian, and Xiang Cui
- Deep Packet Inspection with Delayed Signature Matching
in Network Auditing 75
Yingpei Zeng and Shanqing Guo
- Low-Rate DoS Attack Detection Based on Two-Step Cluster Analysis. 92
Dan Tang, Rui Dai, Liu Tang, Sijia Zhan, and Jianping Man

Full Paper Session III: Real-World Cryptography

- On the Weakness of Constant Blinding PRNG in Flash Player 107
Chenyu Wang, Tao Huang, and Hongjun Wu
- Random Number Generators Can Be Fooled to Behave Badly 124
George Teşeleanu
- Utilizing GPU Virtualization to Protect the Private Keys of GPU
Cryptographic Computation 142
Ziyang Wang, Fangyu Zheng, Jingqiang Lin, and Jiankuo Dong

Full Paper Session IV: Encrypted Computing

- Accelerating Integer Based Fully Homomorphic Encryption
Using Frequency Domain Multiplication 161
Shakirah Hashim and Mohammed Benaissa

Comparison-Based Attacks Against Noise-Free Fully Homomorphic Encryption Schemes 177
Alessandro Barenghi, Nicholas Mainardi, and Gerardo Pelosi

On Security in Encrypted Computing 192
Peter T. Breuer, Jonathan P. Bowen, Esther Palomar, and Zhiming Liu

Full Paper Session V: Privacy Protection

PPOIM: Privacy-Preserving Shape Context Based Image Denoising and Matching with Efficient Outsourcing 215
Meng Zheng, Jun Zhou, Zhenfu Cao, and Xiaolei Dong

Linking Differential Identifiability with Differential Privacy 232
Anis Bkakra, Nora Cuppens-Boulahia, and Frédéric Cuppens

PP-MCSA: Privacy Preserving Multi-channel Double Spectrum Auction 248
Zhili Chen, Sheng Chen, Hong Zhong, Lin Chen, and Miaomiao Tian

Full Paper Session VI: Signature Schemes

Hierarchical Group Signatures with Verifier-Local Revocation 271
Lin Hou, Renzhang Liu, Tian Qiu, and Dongdai Lin

Achieving Full Security for Lattice-Based Group Signatures with Verifier-Local Revocation 287
Maharage Nisansala Sevewandi Perera and Takeshi Koshiba

Towards Practical Lattice-Based One-Time Linkable Ring Signatures 303
Carsten Baum, Huang Lin, and Sabine Oechsner

Full Paper Session VII: Attack Analysis and Detection

Slop: Towards an Efficient and Universal Streaming Log Parser 325
Zhiyuan Zhao, Chenxu Wang, and Wei Rao

Community Discovery of Attribution Trace Based on Deep Learning Approach 342
Jian Xu, Xiaochun Yun, Yongzheng Zhang, and Zhenyu Cheng

Examine Manipulated Datasets with Topology Data Analysis: A Case Study 358
Yun Guo, Daniel Sun, Guoqiang Li, and Shiping Chen

Full Paper Session VIII: Searchable Encryption and Identity-Based Cryptography

Efficient Multi-keyword Searchable Encryption Based on Multi-input Inner-Product Functional Encryption 377
Yunong Liang, Zhenfu Cao, Xiaolei Dong, and Jiachen Shen

Fast Two-Server Multi-User Searchable Encryption with Strict Access Pattern Leakage 393
Cédric Van Rompay, Refik Molva, and Melek Önen

Identity-Based Functional Encryption for Quadratic Functions from Lattices 409
Kelly Yun, Xin Wang, and Rui Xue

Key Dependent Message Security for Revocable Identity-Based Encryption and Identity-Based Encryption 426
Rui Zhang and Yang Tao

Full Paper Session IX: Verifiable Storage and Computing

Publicly Verifiable Data Transfer and Deletion Scheme for Cloud Storage . . . 445
Changsong Yang, Jianfeng Wang, Xiaoling Tao, and Xiaofeng Chen

Publicly Verifiable Static Proofs of Storage: A Novel Scheme and Efficiency Comparisons 459
Henning Kopp, Frank Kargl, and Christoph Bösch

Verifiable Single-Server Private Information Retrieval 478
Xingfeng Wang and Liang Zhao

Full Paper Session X: Applied Cryptography

Cryptographic Password Obfuscation 497
Giovanni Di Crescenzo, Lisa Bahler, and Brian Coan

CCA Secure Multi-recipient KEM from LPN 513
Haitao Cheng, Xiangxue Li, Haifeng Qian, and Di Yan

Witness-Indistinguishable Arguments with Σ -Protocols for Bundled Witness Spaces and Its Application to Global Identities 530
Hiroaki Anada and Seiko Arita

Full Paper Session XI: Supporting Techniques

Automatic Identification of Industrial Control Network Protocol Field Boundary Using Memory Propagation Tree 551
Chen Kai, Zhang Ning, Wang Liming, and Xu Zhen

PCA: Page Correlation Aggregation for Memory Deduplication in Virtualized Environments 566
Min Zhu, Kun Zhang, and Bibo Tu

Role Recognition of Illegal Online Gambling Participants Using Monetary Transaction Data 584
Xiaohui Han, Lianhai Wang, Shujiang Xu, Dawei Zhao, and Guangqi Liu

Certifying Variant of RSA with Generalized Moduli 598
Yao Lu, Noboru Kunihiro, Rui Zhang, Liqiang Peng, and Hui Ma

Full Paper Session XII: Formal Analysis and Cryptanalysis

Attack Trees in Isabelle 611
Florian Kammüller

Automated Verification of Noninterference Property 629
Fan Zhang, Cong Zhang, Mingdi Xu, Xiaoli Liu, Fangning Hu, and HanChieh Chao

Automatic Method for Searching Integrals of ARX Block Cipher with Division Property Using Three Subsets 647
Ya Han, Yongqiang Li, and Mingsheng Wang

Improved Automatic Search Algorithm for Differential and Linear Cryptanalysis on SIMECK and the Applications 664
Mingjiang Huang, Liming Wang, and Yan Zhang

Short Paper Session I: Attack Detection

MalHunter: Performing a Timely Detection on Malicious Domains via a Single DNS Query 685
Chengwei Peng, Xiaochun Yun, Yongzheng Zhang, and Shuhao Li

Detecting Intrusion in the Traffic Signals of an Intelligent Traffic System . . . 696
Abdullahi Chowdhury, Gour Karmakar, Joarder Kamruzzaman, and Tapash Saha

A Linguistic Approach Towards Intrusion Detection in Actual Proxy Logs. . . 708
Mamoru Mimura and Hidema Tanaka

Short Paper Session II: Security Management

Simau: A Dynamic Privilege Management Mechanism for Host
in Cloud Datacenters 721
Lin Wang, Min Zhu, Qing Li, and Bibo Tu

USB Packets Filtering Policies and an Associated Low-Cost
Simulation Framework. 732
*Xiaoshu Ji, Gurvan Le Guernic, Nora Cuppens-Boulahia,
and Frédéric Cuppens*

Short Paper Session III: Applied Cryptography

FPGA-Based Assessment of Midori and GIFT Lightweight Block Ciphers. 745
*Carlos Andres Lara-Nino, Arturo Diaz-Perez,
and Miguel Morales-Sandoval*

Simpler CCA Secure PKE from LPN Problem Without Double-Trapdoor 756
Haitao Cheng, Xiangxue Li, Haifeng Qian, and Di Yan

PoS: Constructing Practical and Efficient Public Key Cryptosystems
Based on Symmetric Cryptography with SGX 767
Huorong Li, Jingqiang Lin, Bingyu Li, and Wangzhao Cheng

Short Paper Session IV: Applied Cryptography

Application of Public Ledgers to Revocation in Distributed
Access Control 781
Thanh Bui and Tuomas Aura

Micropaying to a Distributed Payee with Instant Confirmation 793
Peifang Ni, Hongda Li, and Dongxue Pan

Revisiting Anonymous Two-Factor Authentication Schemes
for Multi-server Environment 805
Ping Wang, Zijian Zhang, and Ding Wang

Author Index 817

**Full Paper Session I: Blockchain
Technology**



Blockchain-Based Secure Data Provenance for Cloud Storage

Yuan Zhang^{1(✉)}, Xiaodong Lin², and Chunxiang Xu¹

¹ University of Electronic Science and Technology of China, Chengdu, China
zy_loye@126.com

² Wilfrid Laurier University, Waterloo, Canada

Abstract. Data provenance, which records the history of the ownership and process of a document during its lifecycle, is essential for the success of cloud storage systems. However, it also inevitably incurs some challenging security and privacy issues. In this paper, to address these challenging issues, we present an efficient and secure data provenance scheme and realize it in a system called ESP. ESP is characterized by employing a blockchain-based provenance record chain and can provide a secure and efficient system for data outsourcing, where the correctness, integrity, and timeliness of provenance records can be ensured. Furthermore, we introduce a concept of window of latching (WoL) to assess the practicality of secure provenance schemes. We analyze the security of ESP and evaluate the performance of ESP via implementation, which shows WoL of ESP is short and demonstrates ESP is secure and practical.

Keywords: Secure provenance · Blockchain · Cloud storage
Digital investigation

1 Introduction

In today's big data era, digital data are explosively generated and people are increasingly outsourcing their data to cloud servers so as to enjoy efficient data management services without bearing heavy local storage costs [1]. While cloud storage is very helpful in many aspects, public cloud service could put user information in danger [2]. Meanwhile, it is also very important to keep track of what happens to these data throughout its lifecycle (from creation to ownership transfer to destruction or deletion), such as its ownership and custodial history as well as how it has been accessed by its users, which is also known as data provenance [3,4]. For example, in a digital investigation, digital evidences must be strictly secured and clearly documented about its ownership transfer as well as how it was handled during its lifecycle. It is usual that the defendant challenges the authenticity of a digital evidence during the trial. The most common types of

Yuan Zhang—This work is supported by National Key R&D Program of China (No. 2017YFB0802000) and China Scholarship Council.

digital evidence are hard disk images, and the defendant may question the hard disk image that investigators are working on and presented in the courtroom is not the same one acquired from the hard disk found at the crime scene.

In the past decades, many security mechanisms have been developed to ensure security and privacy of sensitive information, as well as achieve accountability and auditability through data access logging or audit trails [5], such as logging activities on data creation, modification, and access. Much of the focus has been on protecting digitally stored information from unauthorized access or modification. However, despite extensive research on information security and privacy, little attention has been paid to securing provenance information and providing assurance that a data document is trustworthy. It is worth mentioning that as the current best of practice, log files are also protected from tampering. In the banking industry, any activities, such as bank transfers, can only be recorded by creating a new log, and past logs cannot be modified or deleted for security reasons. Nevertheless, another important question still needs to be answered about whether the provenance information can be trusted to make sure that the corresponding data document is a trusted one after a series of user activities on the document which have been detailed in the provenance information.

Unlike the traditional file access auditing where file access activities are logged, provenance information contains the ownership history of data documents as well as activities occurred on them by their owners or users. Furthermore, such information is organized in chronological order during the lifecycles of the data documents, and allows accesses and activities of data documents to be tracked. As a result, it not only improves accountability and reliability [6], but also meets the requirements of emerging applications, such as maintaining the digital chain of custody in a digital forensics investigation [7–9], as well as regulatory compliance requirements and industry standards, such as HIPAA [10].

Actually, provenance information is not useful if it cannot be trusted, it is inadvisable to trust provenance information without proper protection. For example, whenever a Microsoft office document is created, Microsoft office automatically embeds an author name into the document. It has been proved very useful to solve crime. A good example of it is the BTK killer case [11]. Nevertheless, the assigned name for the document can be easily changed. This problem is further exacerbated by the fact that data documents and the corresponding provenance records are outsourced to the cloud storage which cannot be fully trusted, since the data documents and the provenance information would not be physically owned by data owners and they are transmitted over an insecure network [12]. Hence, it is crucial to ensure provenance information security.

Provenance information can be modeled as a sequence of records (each known as provenance record) which present details about how a data document was processed at every stage of its lifecycle. To guarantee the security of provenance information, similar to protection of data document itself, we can protect individual records of provenance information from unauthorized use or modification. However, the integration of security mechanisms in current clouds to ensure the security of the individual records would incur additional costs on both the cloud

provider and users. As such, it is vital to point out what affects the practicality of secure provenance schemes and define how to evaluate the practicality.

In existing schemes [3, 13], an identity manager is introduced to efficiently secure the outsourced data provenance information, where any operation on the data performed by a user is required to be authorized by the identity manager. By doing so, the provenance records actually reflect the state information on the data during its lifecycle. Nevertheless, such mechanism suffers from a strong assumption that the identity manager is honest and reliable. Once the identity manager is compromised, the security of these schemes are broken: if the cloud server colludes with the identity manager, the outsourced provenance records can be modified without detection. In reality, compromising the identity manager is feasible for adversaries, since an adversary (e.g., a malicious cloud server) can perpetually incentivize the identity manager to deviate from the prescribed scheme over a long period of time. Furthermore, existing schemes do not consider the timeliness of provenance records.

In this paper, we propose an efficient and secure data provenance scheme for cloud storage systems called ESP. ESP is secure against provenance record forgery, removal, modification attacks. The security of ESP is guaranteed in the case that the identity manager is compromised, even if the malicious cloud server colludes with it. The key technique behind ESP is the blockchain-based currencies which provide a secure way to conduct transactions without a central authority. In ESP, each provenance record is integrated into a transaction on the blockchain, and all provenance records corresponding to one data form a record chain such that any one of them is corrupted, the chain is broken. With the integration of a provenance record in a transaction on the blockchain, the provenance record is time-stamped, and the time when the provenance record was generated can be extracted. As such, the provenance records in ESP not only keep track of what happened to the data, but also reflect when the data was processed. Detailed security analysis proves that ESP is secure against various attacks, even if the malicious cloud server/user colludes with the identity manager. Moreover, we introduce a concept of window of latching (WoL) which is one of the most important factors that affects the practicality of secure provenance schemes. We implement ESP and evaluate its performance. Experiment results show that WoL of ESP is short and is acceptable in reality, which demonstrates ESP is efficient and practical. Specifically, the contributions of this work are as follows:

- We formalize a model of data provenance, where the lifecycle of data documents is formally formulated. We also introduce the concept of WoL to measure the practicality of secure provenance schemes.
- We propose an efficient and secure data provenance scheme (ESP) for cloud storage systems. ESP employs a provenance record chain which is built on blockchain-based currencies, e.g., Ethereum, this ensures the secure auditability of provenance records in terms of correctness, integrity, and timeliness, even if the identity manager is compromised.

- We present security analysis to demonstrate that ESP can be secure and robust from various attacks. We implement ESP and conduct a comprehensive performance evaluation, which shows that ESP is highly efficient and practical.

2 Related Work

Data provenance provides sufficient information about target data that what happened to the data from creation to destruction. As we are moving into the age of big data where digital data are explosively generated nowadays and most of data are managed via the Internet with the aid of cloud systems, data provenance is pretty important to digital investigations [14, 15]. Once a dispute arises in outsourced data, provenances serve as the most vital evidences for post investigation.

Lynch [16] first pointed out the need for trust and provenance in information retrieval. Hasan et al. [2] first defined the problem of secure provenance and argued that it is of vital importance in practice. Prior work on secure data provenance in cloud storage systems was proposed by Lu et al. [3], where the basic security requirements were first enumerated, i.e., unforgeability and conditional privacy preservation. The unforgeability ensures that a provenance record reflects the corresponding state of data, even if the data and the provenance record are outsourced to an untrusted environment; The conditional privacy preservation guarantees that only an authenticated entity can reveal the real identity recorded in the provenance, while anyone else cannot.

Following the Lu et al.'s work, several secure data provenance schemes have been proposed [13, 17]. These schemes mainly focus on enhancing the functionality of secure provenance for cloud storage systems. However, in existing schemes, a trusted identity manager is introduced to secure the provenance records. If the identity manager is compromised, the security would be broken. Moreover, in existing schemes, lifecycles of data documents in cloud storage are not considered, the timeliness of provenance records has not been explored, and how to measure the practicality of secure provenance schemes is also not well investigated. In this paper, we propose ESP, an efficient and secure data provenance scheme that ensures the correctness, integrity, and timeliness of provenance records against the malicious identity manager.

3 Preliminaries

3.1 Basic Cryptographic Primitives

Secure Hash Function. A secure hash function h has the following three properties: h can take a message of arbitrary length as input, and output a short fixed-size message digest; Given x , it is easy to compute $h(x) = y$. However, given y , it is hard to calculate $h^{-1}(y) = x$; Given x , it is computationally infeasible to find $x' \neq x$ such that $h(x') = h(x)$.

Bilinear Maps. Let G be an additive group and G_T be a multiplicative group, they have the same prime order p . A bilinear map $e: G \times G \rightarrow G_T$ has the following properties. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G$, $a, b \in \mathbb{Z}_p^*$; Non-degeneracy: for $P, Q \in G$ and $P \neq Q$, $e(P, Q) \neq 1$; e can be computed efficiently.

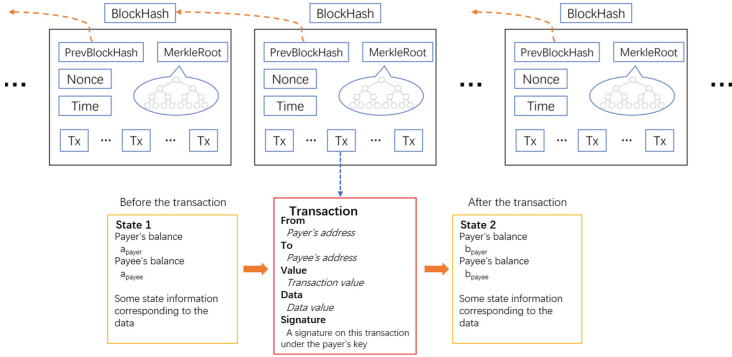


Fig. 1. A simplified Ethereum blockchain

3.2 Blockchain

We defer a brief introduction to the blockchain technique to **Appendix A. Blockchain.** We construct ESP on the Ethereum blockchain, since Ethereum is more expressive than other blockchain-based currencies. We show a simplified Ethereum blockchain in Fig. 1, where Tx denotes the transaction, BlockHash denotes the hash value of current block, PrevBlockHash denotes the hash value of the last block, Time denotes the time when the block is chained to the blockchain, and MerkleRoot denotes the root value of a Merkle hash tree formed by all transactions recorded in the block. The value token of the Ethereum blockchain is called Ethers.

In Ethereum, the state is made up of objects called “account”. Generally, there are two types of accounts: externally owned accounts and contract accounts. Externally owned accounts are controlled by private keys and can conduct a transaction. Contract accounts are controlled by their contract code. For a transaction between two external owned accounts, if it is recorded into the blockchain, the balances of these two accounts are updated, where the user who conducts the transaction can set the “data” field to be any binary data she/he chooses. Therefore, Ethereum blockchain ensures the timeliness of the data state: when a payer transfers Ethers to a payee, a string Δ can be set to be the Data value of the transaction; After the block containing this transaction is added into the blockchain, Δ is recorded, which means that Δ is generated no later than the time when the block is chained to the block.

4 Models and Design Goals

4.1 A Model of Data Provenance

Lifecycle of a Data Document and Its Users. In this work, a data document lifecycle is viewed as a sequence of stages from creation to modification, destruction, and ownership transfer, which is shown in Fig. 2. After a data document has been created, it may go through many stages due to the document modification or ownership transfer. Finally, a document may be destroyed or deleted, becoming unavailable to the users. Hence, an individual state of a data document can be uniquely identified by its content and owner, and can be represented as $St_i = \mathcal{H}(F_i, O_i)$, where St_i stands for a state where a document has been at, F_i means the content of the document at state of St_i , O_i is the owner of the document at state of St_i , and \mathcal{H} stands for a secure hash function.

During the lifetime of a data document, users can play different roles in it, and can be classified into four types in general: *creator*, *owner*, *editor*, and *viewer*.

Creator: showing a user is the creator of a data document.

Owner: showing a user is the owner of the data document. By taking ownership of a data document, the user can assign other users permissions to the data document, including editing and viewing a data document, and transferring ownership of a data document. By default, the creator of a data document is also the owner, but document ownership can be transferred to another user by its current owner.

Editor: identifying a user is able to edit the data document.

Viewer: identifying a user can only view the data document.

Documents can have many editors and viewers, but only one creator during its lifetime and one owner at a time. In addition to the four aforementioned types of users, we assume there exists an *auditor* who can verify the validity and trustworthiness of any provenance information but without any knowledge of the user’s identity who generates each individual provenance record [2,3].

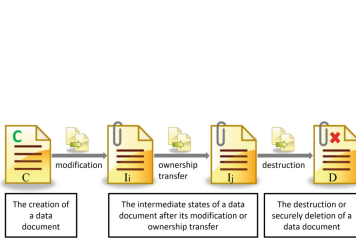


Fig. 2. Document lifecycle

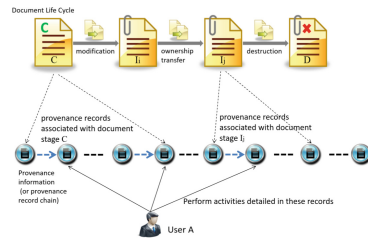


Fig. 3. Provenance model

Provenance Model. As shown in Fig. 3, in data provenance, provenance information is organized into a chain in chronological order, where each chain item represents a provenance record which details how a data document was processed

at every stage of its lifecycle. Each provenance record is also associated with a specific document stage, and a legitimate user (e.g., editors) may perform many actions on data documents. A typical provenance record consists of a specially formatted data block that contains information related to how a data document is processed at a time as well as its ownership information, which usually can be classified into two types: *Essential provenance data (EPD)*: information related to activities performed on the data document; *Nonessential provenance data (NPD)*: security overhead which has been generated by security mechanisms that are used to protect provenance information.

Measure the Practicality of Secure Provenance. With the provenance model, we introduce a concept of window of latching (WoL) to evaluate the practicality of secure provenance schemes.

Definition 1. Window of latching (WoL) means the time-interval between two successive provenance records that are accepted and published. The shorter WoL, the more practical the secure provenance scheme is.

4.2 Threat Model

In our threat model, we mainly consider the following security and privacy threats against data provenance.

Provenance Record Forgery Attack. A malicious user may collude with others to forge a valid provenance record in terms of the record's content and its timeliness.

Provenance Record Removal Attack. A malicious user colludes with others to remove one or several existing provenance records that have been generated due to the operations performed on data documents.

Modification Attack. Similar to the two above threats, a malicious user who may collude with others may attempt to tamper with provenance information by modifying the provenance records.

Repudiation Attack. A malicious user may deny that he performed an action on a data document.

Privacy Violation. Privacy violation refers to the attack that the identity of user who generates a provenance record is leaked out. Recall that in secure provenance schemes [3, 13], only conditional privacy preservation can be ensured, where the identity manager has the ability to reveal the real identity recorded in the provenance record, while anyone else cannot.

4.3 System Model and Design Goals

As shown in Fig. 4, there are four different entities in ESP: users, an authenticated server, a cloud server, and an independent auditor. The authenticated server is used to authorize the users and control who can access the data. It also

assists users in preserving their identity against adversaries. Data and the corresponding provenance information are generated by the users, and are stored in the cloud server. The auditor can check the provenance records' validity including their correctness, integrity, and timeliness.

Different from existing schemes [3, 13], the authenticated server is not fully trusted by others, and thereby should be responsible for all its authorizations. As long as the authenticated server remains inaccessible to adversaries, we ensure both the security and privacy preservation. If both the authenticated server and cloud storage server are compromised, we retain the security guarantees on the provenance records in existing schemes.

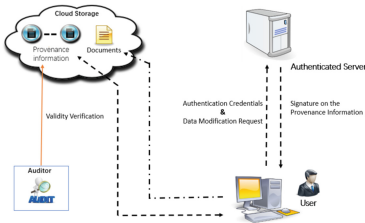


Fig. 4. The system model of ESP

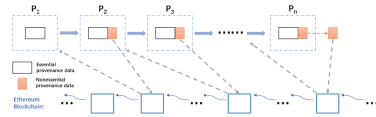


Fig. 5. Blockchain-based provenance record chain

Aiming at the above security challenges, our design goal is to develop an efficient and secure data provenance scheme in the cloud storage system. Specifically, the following goals should be achieved.

Security and Privacy Preservation. The validity of provenance records can be audited by authorized auditors with resistance against various attacks. The conditional privacy can be ensured.

Efficiency. It should efficiently work without introducing too much extra storage space caused by introducing security mechanisms, for example, digital signatures and cryptographic hashes. Its WoL should be as short as possible such that it can be applied in reality.

5 The Proposed Scheme

5.1 Overview

ESP consists of three parts: *system setup*, *secure provenance generation*, and *secure provenance verification*.

In the first part, the authenticated server assigns a human-memorable password to each user, and maintains a list that records the assigned passwords and the corresponding identities. With the list, the authenticated server can authenticate each user securely and efficiently.

When a user wants to process a document, she/he needs to be authorized by the authenticated server. Then the authenticated server assists the user in generating a provenance record, this enables the user to prove herself/himself to the cloud server that she/he is qualified to process the document. Each provenance record is integrated into a transaction on the blockchain, where the user transfers a service charge to the authenticated server. This also time-stamps the provenance record.

To achieve the security, all provenance records are chained together with the aid of the Ethereum blockchain. This is shown in Fig. 5, where the provenance record chain is indicated by dashed gray lines. Assume that there currently are n provenance records, P_1, P_2, \dots, P_n , each of them stands for a state of the underlying document at the corresponding stage during its lifecycle as modeled in Sect. 4.1. They are chained together as follows: from the second provenance record P_2 , each record contains a data field that points to a block on the Ethereum blockchain, this block relates to the last provenance record. Each record is appended to the last one until it reaches the last one of the current provenance information, P_n . Finally, the last record will be signed by the authenticated server and the signature becomes the tail of the provenance record chain. In this case, if any existing record is modified or removed, the provenance record chain is broken. The computational costs to verify provenance records mainly depend on the hashing operation along with one signature verification for the last element or the tail of the provenance record chain. As a result, the verification is very fast.

5.2 Description of ESP

A set of user $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, \dots\}$, an authenticated server \mathcal{AS} , a cloud storage server \mathcal{C} , and a third-party auditor \mathcal{A} are involved in ESP.

System Setup:

- With the security parameter ℓ , the system parameters $\{p, G, G_T, P, e, E(\cdot), h, H\}$ are determined, where G is an additive group whose generator is P , $e : G \times G \rightarrow G_T$, G and G_T have the same prime order p , $E(\cdot)$ is a secure symmetric encryption algorithm, $h : \{0, 1\}^* \rightarrow Z_p^*$, and $H : \{0, 1\}^* \rightarrow G$.
- \mathcal{AS} randomly chooses $s \in Z_p^*$, and computes $P_{pub} = sP$, and $k = h(s)$.
- \mathcal{AS} 's secret keys are (s, k) , the corresponding public key is P_{pub} .
- For each $\mathcal{U}_i \in \mathcal{U}$ with identifier ID_i , \mathcal{AS} assigns a human-memorable password pwd_i to her/him, and stores (ID_i, pwd_i) locally.

Secure Provenance Generation:

Once a user \mathcal{U}_i processes a document at \mathcal{C} and generates a provenance record P_j , she/he will request \mathcal{AS} to generate secure provenance on the document process.

Phase 1: With the identifier ID_i and password pwd_i , \mathcal{U}_i makes mutual authentication with \mathcal{AS} to establish a secure channel as follows.

- \mathcal{U}_i randomly chooses $r_1, a \in Z_p^*$, and fetches the current timestamp ct .
- \mathcal{U}_i computes C_1, C_2 , where $C_1 = r_1P$, $C_2 = E_k(ID_i||pwd_i||aP||ct)$, and $k = r_1P_{pub}$.
- \mathcal{U}_i sends (C_1, C_2) to \mathcal{AS} .
- After receiving (C_1, C_2) , \mathcal{AS} computes $k = sC_1 = sr_1P = r_1P_{pub}$, and extracts $ID_i||pwd_i||aP||ct$ from C_2 with k .
- \mathcal{AS} checks the validity of the timestamp ct to resist the replay attack.
- \mathcal{AS} authenticates \mathcal{U}_i by checking whether (ID_i, pwd_i) is stored locally.
- \mathcal{AS} randomly chooses $b \in Z_p^*$ and computes $sk = b(aP)$ as the session key.
- \mathcal{AS} calculates \mathcal{U}_i 's pseudonym $PID_j = E_k(ID_i||ct||b)$, C_3 , and C_4 , where $C_3 = bP$, $C_4 = E_{sk}(ID_i||aP||bP||ct||PID_j)$.
- \mathcal{AS} sends (C_3, C_4) to \mathcal{U}_i .
- With (C_3, C_4) , \mathcal{U}_i computes the session key $sk = aC_3 = abP$.
- \mathcal{U}_i extracts $ID_i||aP||bP||ct||PID_j$ from C_4 with sk .
- \mathcal{U}_i authenticates \mathcal{AS} and confirms the correctness of sk by verifying the correctness of $ID_i||ct||aP||bP$.
- Since the session key sk is shared between \mathcal{U}_i and \mathcal{AS} , a secure channel between them is established for secure provenance.

Different roles of \mathcal{U}_i require different execution between \mathcal{U}_i and \mathcal{AS} .

Creator: \mathcal{U}_i creates a new document

If \mathcal{U}_i creates a new document F , i.e., the provenance record P_j is P_1 , where $P_1 = h(F_1||ID_i)$ and F_1 denotes the content of the document at the first stage, she/he requests a secure provenance from \mathcal{AS} as follows.

- \mathcal{U}_i sends P_1 to \mathcal{AS} via the secure channel.
- \mathcal{AS} extracts PID_1 from local storage (i.e., $j = 1$).
- \mathcal{AS} generates a signature on P_1 and PID_1 as $\sigma_{T_1} = sH(P_1||PID_1)$, and sends σ_{T_1} to \mathcal{U}_i .
- \mathcal{U}_i verifies $e(\sigma_{T_1}, P) \stackrel{?}{=} e(H(P_1||PID_1), P_{pub})$. If the verification fails, reject.
- \mathcal{U}_i creates a transaction T_{x_1} shown in Fig. 6, where \mathcal{U}_i transfers service charge to the \mathcal{AS} 's account, and the data field of the transaction is set to $h(h(P_1||PID_1)||\sigma_{T_1})$.
- After T_{x_1} is recorded into the Ethereum blockchain, $(P_1||PID_1||Bl_1, \sigma_{T_1})$ is sent to \mathcal{C} , and is published as the provenance record.

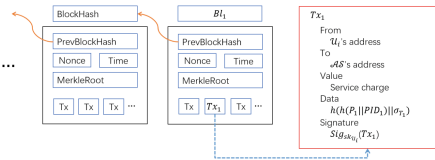


Fig. 6. The transaction conducted by the creator

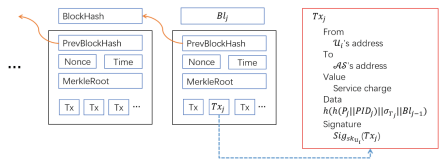


Fig. 7. The transaction conducted by the editor/viewer

Editor/Viewer: \mathcal{U}_i edits/views an existing document

If \mathcal{U}_i edits/views an existing document, without loss of generality, we assume the underlying document is F and the provenance record $P_j = h(F_j || ID_i)$ with $j \geq 2$, where F_j denotes the content of the document at the j -th stage. \mathcal{U}_i interacts with \mathcal{AS} as follows.

- \mathcal{U}_i sends $(P_j, Bl_{j-1}, \sigma_{T_{j-1}})$ to \mathcal{AS} via the secure channel, where Bl_{j-1} denotes the hash value of the block that contains the transaction whose data field is $h(h(P_{j-1} || PID_{j-1}) || \sigma_{T_{j-1}})$.
- \mathcal{AS} checks the validity of Bl_{j-1} , if the checking fails, reject.
- \mathcal{AS} extracts PID_j from local storage.
- \mathcal{AS} computes $\Theta(P_j) = H(P_j || PID_j || Bl_{j-1})$, generates a signature $\sigma_{T_j} = s \cdot \Theta(P_j)$, and \mathcal{AS} sends σ_{T_j} to \mathcal{U}_i .
- \mathcal{U}_i verifies σ_{T_j} by checking whether $e(\sigma_{T_j}, P) \stackrel{?}{=} e(\Theta(P_j), P_{pub})$.
- \mathcal{U}_i creates a transaction Tx_j shown in Fig. 7, where \mathcal{U}_i transfers service charge to the \mathcal{AS} 's account, and the data field of the transaction is set to $h(h(P_j || PID_j) || \sigma_{T_j} || Bl_{j-1})$.
- After Tx_j is recorded into the blockchain, $(P_j || PID_j || Bl_j, \sigma_{T_j}, Bl_{j-1})$ is sent to \mathcal{C} , and is published as the provenance record.

Finally, the secure provenance becomes

$$(P_1 || PID_1 || Bl_1, P_2 || PID_2 || Bl_2, \dots, P_j || PID_j || Bl_j, \sigma_{T_j}).$$

Secure Provenance Verification: Given the provenance

$$(P_1 || PID_1 || Bl_1, P_2 || PID_2 || Bl_2, \dots, P_j || PID_j || Bl_j, \sigma_{T_j}),$$

the auditor \mathcal{A} checks its correctness as follows:

- Locate the last block Bl_j on the Ethereum blockchain, and verifies the validity of the last recorded provenance record $P_j || PID_j || Bl_j$.
- Compute $\Theta(P_j) = H(P_j || Bl_{j-1})$.
- Check whether $e(\sigma_{T_j}, P) \stackrel{?}{=} e(\Theta(P_j), P_{pub})$.
- Extract the data information from blockchain according to Bl_1, \dots, Bl_j .
- Verify the integrity and timeliness of provenance by checking whether the hash value of provenance matches the extracted data. Here, the physical time when the provenance record was generated is derived from the height of the corresponding block. Specifically, assuming the time when $P_j || PID_j || Bl_j$ was generated is denoted by τ_j and the height of the corresponding block is denoted by ρ_j . $\tau_j = \tau_0 + \gamma \cdot \rho_j$ (seconds), where τ_0 is the physical time when the genesis block of Ethereum was generated (i.e., 2015-07-30, 03:26:13 PM +UTC) and γ is the average time that a block is mined in Ethereum.

If all the provenance records pass all the above checking, it can be accepted.

5.3 Remark

In ESP, \mathcal{A} can check the time when a provenance record was generated by extracting the timestamp of the corresponding block from the blockchain. However, in Ethereum, the timestamp of a block cannot accurately reflect when transactions included in the block were generated, since the timestamp of the block might be confronted with up to 900 seconds errors. To overcome the time errors, in ESP, the auditor derives the transaction time from the height of the block including the transaction. The key observation is that the average time that a block is mined is deterministic and can be counted, and the blockchain height can be trusted to increase with respect of either short or long term, which is formalized as the chain-growth of blockchain [18]. By doing so, the time when a provenance record was generated suffers from around 15 seconds errors in ESP, which has improved the accuracy of timestamp significantly.

6 Security Analysis

ESP is secure against provenance record forgery, removal, modification, and repudiation attacks, even if the authenticated server is compromised. ESP also guarantees conditional privacy preserving. We defer the detailed security analysis to **Appendix B. Security Analysis of ESP**.

7 Implementation and Evaluation

We implement ESP by using JAVA, and the experiments are conducted on a laptop with Window 7 system, an Intel Core 2 i5 CPU and 8GB DDR3 of RAM. The security level is chosen to 80 bits and the hash function h is selected to SHA3-256. The implementation of ESP is illustrated in Fig. 8 and is described below. For clarity, we prefix calls with \mathcal{AS} when they are made by \mathcal{AS} and with \mathcal{U} when they are made by \mathcal{U} .

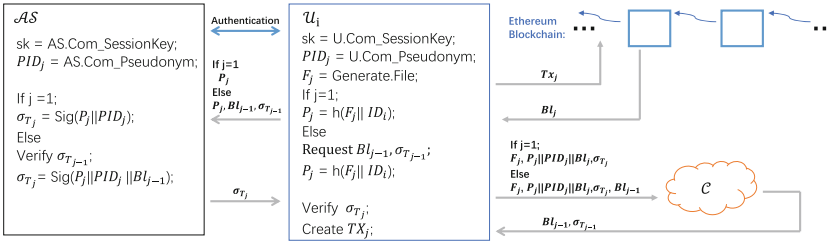


Fig. 8. Implementation of ESP

Com_SessionKey is an interactive algorithm to compute the session key between \mathcal{U} and Com_Pseudonym is an algorithm to compute a pseudonym for \mathcal{U} .

The signature on the provenance record P_j and the pseudonym PID_j is implemented by $\text{Sig}(P_j||PID_j)$. The verification of provenance record T_{j-1} is implemented by $\text{Verify } \sigma_{T_{j-1}}$. Generating/editing the target file on \mathcal{U} is implemented by Generate.File .

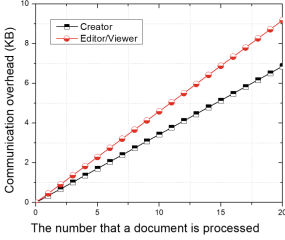


Fig. 9. Communication overhead of creator and editor/viewer

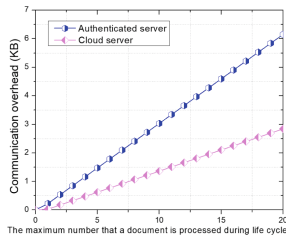


Fig. 10. Communication overhead of authenticated server and cloud server

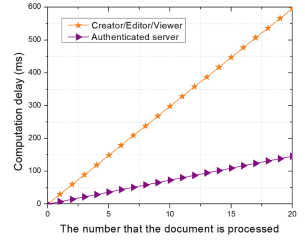


Fig. 11. Computational overhead

We show the communication overhead of the creator and editor/viewer in Fig. 9, where the size of human-memorable password is set to 120 bits. We also show the communication overhead of the authenticated server and the cloud server in Fig. 10.

In ESP, generating the system parameters is a one-time computation, here we would not show the computation costs in initializing ESP. Instead, we show the computation delay on the users and authenticated server in Fig. 11. In ESP, generating a provenance record takes within 50 ms. ESP is constructed on the Ethereum blockchain. In Ethereum, a block as well as its transactions is considered *confirmed* if at least 12 consecutive blocks are mined following it. The average time that a block is mined is 15 seconds and hence a transaction takes averagely 15 seconds to be chained to the Ethereum blockchain. As such, publishing a new provenance record takes average 3.25 min in ESP, and the time interval between two successive provenance records only requires around 3.25 min. Another user may have to wait at least 3.25 min to work on the same document. It is the most important factor that affects the practicality of a secure provenance scheme, which is called window of latching (WoL) and is defined in *Definition 1*.

Another factor that affects the practicality of ESP is the costs to publish provenance record. The transaction fee in Ethereum can be set to be values from 0.000021 Ether to 0.000756 Ether, and the averagely fee is 0.000378 Ether. As of May, 2018, publishing a provenance record requires a user to pay average 25 US cents, which is acceptable to users with respect to the value of the data that ESP protects.

The above experiment results demonstrate that ESP is efficient in terms of communication and computation overhead. We have evaluated WoL of ESP, the evaluation results show that WoL is short and is acceptable in reality. The above

analysis also indicates that WoL of ESP is mainly subject to the transaction confirmation time in the blockchain system, and the costs to publish provenance records are at the mercy of transaction fees in the blockchain system.

8 Conclusion

In this paper, we have proposed an efficient and secure data provenance scheme (ESP) for cloud storage systems. ESP employs the blockchain-based provenance record chain to ensure the correctness, integrity, and timeliness of provenance records. ESP protects users' real identities against the cloud storage server, which preserves users' privacy. Detailed security analyses have shown that ESP is secure and robust from various attacks with privacy preservation. Compared with existing schemes, ESP can resist the malicious identity manager. We have introduced the concept of window of latching (WoL) to evaluate the practicality of secure provenance schemes. We also have implemented ESP and show that WoL of ESP is short and can be acceptable in reality, which has demonstrated ESP is practical and efficient.

Appendix A. Blockchain

A blockchain is a shared immutable ledger for recording the history of transactions, it provides a tamper-proofing and distributed way to conduct transactions without a central authority. Technically, the blockchain is a linear collection of data elements, where each data element is called a *block*. All blocks are linked in chronological order to form a chain and secured using cryptography. Typically, each block contains a hash pointer, a timestamp, and transaction data, where the hash pointer points to the previous block as a link, and the timestamp indicates when the current block is chained to the blockchain [19]. Only valid transactions would be recorded into the blockchain.

The most prominent manifestation of blockchain is blockchain-based currencies, such as Bitcoin [19] and Ethereum [20], wherein the blockchain is used to serve as an open and distributed ledger that records transactions between two entities. The ledger here is verifiable and inherently resistant to modification of chained blocks. Participants who perform the transaction verification and maintain the blockchain are called *miner*.

The ledger of Ethereum blockchain can be considered as a state transition system. When a payer conducts a new transaction, she/he broadcasts the transaction to all miners. Each miner first verifies the validity of received transaction, and collects multiple new transactions into a block. Then each miner computes a valid nonce such that the hash value of the block is less than or equal to a value provided by the Ethereum system. The first miner who finds the nonce broadcasts the block including the nonce and a timestamp. Other miners accept the block only if the nonce and all transactions in it are valid. More details can be found in [20].

Appendix B. Security Analysis of ESP

ESP is secure against provenance record forgery attacks. In ESP, when a user U_i processes a document and requests \mathcal{AS} to generate secure provenance on the document process, \mathcal{AS} needs to authenticate U_i . Without knowing the password pwd_i , an attacker cannot generate $C_1 = r_1P$, $C_2 = E_k(ID_i || |pwd_i| |aP| |ct)$ with $k = r_1P_{pub}$. Note that this authorization is integrated into the corresponding provenance record and would be recorded into the blockchain, \mathcal{AS} cannot deny the authorization and should be responsible for it. In addition, since the timestamp ct has also been included, it can resist the replay attack. Due to this time-sensitive credential, an attacker cannot forge a secure provenance.

ESP is secure against provenance record removal/modification attacks. In the provenance record chain (shown in Fig. 5), if P_{j-1} is removed or modified, all successive hash values along the chain will be affected, and thereby the hash value of the last record $P_n || PID_n || Bl_n$ will be changed. In ESP, the provenance record is built on the BLS signature [21] which is existentially unforgeable. Even if the attacker colludes with \mathcal{AS} to forge a signature, the transaction recorded into the blockchain cannot be removed/modified, collusion between any two entities in ESP can not remove/modify the published provenance records.

ESP also provides the non-repudiation. In ESP, the generation of a provenance record P_j for a user U_i requires the \mathcal{AS} 's assistance, where the pseudonym of U_i , i.e., PID_i , is integrated into the provenance record. PID_i is derived from U_i 's identity ID_i , and \mathcal{AS} can "open" PID_i to prove the relationship between ID_i and PID_i . Furthermore, before publishing the secure provenance $(P_j || PID_j || Bl_j, \sigma_{T_j}, Bl_{j-1})$, U_i conducts a transaction Tx_j that transfers service charge to \mathcal{AS} , where the hash value of $(P_j || PID_j || \sigma_{T_j} || Bl_{j-1})$ is integrated into the transaction. By the security of Ethereum, anyone cannot impersonate others to conduct a transaction. Therefore, the auditor is able to confirm that P_j is generated by PID_i by checking the creator of the transaction related to P_j with the aid of \mathcal{AS} .

ESP also provides the conditional privacy preservation. To resist the privacy violation, the pseudonym $PID_j = E_k(ID_i || ct || b)$ in place of the real identity ID_i is included in the signature. Due to the security of $E(\cdot)$, the real identity ID_i cannot be disclosed from PID_j , the user privacy is preserved. The privacy preservation is also conditional, since $PID_j = E_k(ID_i || ct || b)$ is derived from ID_i with the master key k , once a provenance record P_j is in dispute, \mathcal{AS} can use k to recover the real identity.

ESP enables auditors to securely check the timeliness of provenance records. In ESP, before a provenance record P_j is published by a user U_i , a transaction Tx_j should be created and recorded into the Ethereum blockchain. The data value of Tx_j is set to the EPD and NPD related to P_j , if the block containing Tx_j is added to the blockchain, the EPD and NPD related to P_j is stored in the transaction. As such, integrating P_j into Tx_j essentially time-stamps P_j , and the time when the block containing Tx_j is chained to the blockchain represents the time when P_j is generated. This enables the auditor to check the timeliness of provenance records without introducing any trusted entity. Due to

the security (chain-growth [18]) of Ethereum, anyone cannot modify the height-derived timestamp of P_j .

References

1. Zhang, Y., Xu, C., Liang, X., Li, H., Mu, Y., Zhang, X.: Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. *IEEE Trans. Inf. Forensics Secur.* **12**(3), 676–688 (2017)
2. Hasan, R., Sion, R., Winslett, M.: Introducing secure provenance: problems and challenges. In: *Proceedings of StorageSS*, pp. 13–18 (2007)
3. Lu, R., Lin, X., Liang, X., Shen, X.: Secure provenance: the essential of bread and butter of data forensics in cloud computing. In: *Proceedings of ASIACCS*, pp. 282–292 (2010)
4. Xu, S., Ni, Q., Bertino, E., Sandhu, R.S.: A characterization of the problem of secure provenance management. In: *Proceedings of ISI*, pp. 310–314 (2009)
5. Audit trails. <http://csrc.nist.gov/publications/nistbul/it197-03.txt>
6. Madden, B.A., Adams, I.F., Storer, M.W., Miller, E.L., Long, D.D., Kroeger, T.M.: Provenance based rebuild: using data provenance to improve reliability. Technical report, UCSC (2011)
7. Ashcroft, J., Daniels, D.J., Hart, S.V.: Forensic examination of digital evidence: a guide for law enforcement. <https://www.ncjrs.gov/txtfiles1/nij/199408.txt> (2011)
8. Lin, X., Chen, T., Zhu, T., Yang, K., Wei, F.: Automated forensic analysis of mobile applications on android devices. In: *Proceedings of DFRWS USA* (2018)
9. Zhang, Y., Xu, C., Li, H., Yang, K., Zhou, J., Lin, X.: HealthDep: an efficient and secure deduplication scheme for cloud-assisted eHealth systems. *IEEE Trans. Inf. Forensics Secur.*, to appear. <https://doi.org/10.1109/TII.2018.2832251>
10. Health insurance portability and accountability act. https://en.wikipedia.org/wiki/Health_Insurance_Portability_and_Accountability_Act
11. BTK killer. https://en.wikipedia.org/wiki/Dennis_Rader
12. Zhang, Y., Xu, C., Yu, S., Li, H., Zhang, X.: SCLPV: secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. *IEEE Trans. Comput. Soc. Syst.* **2**(4), 159–170 (2015)
13. Chow, S.S.M., Chu, C.-K., Huang, X., Zhou, J., Deng, R.H.: Dynamic secure cloud storage with provenance. In: Naccache, D. (ed.) *Cryptography and Security: From Theory to Applications*. LNCS, vol. 6805, pp. 442–464. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28368-0_28
14. Hasan, R., Sion, R., Winslett, M.: Preventing history forgery with secure provenance. *ACM Trans. Storage* **5**(4), 12 (2009)
15. Wang, Q., Hassan, W.U., Bates, A., Gunter, C.: Fear and logging in the Internet of Things. In: *Proceedings of NDSS* (2018)
16. Lynch, C.A.: When documents deceive: trust and provenance as new factors for information retrieval in a tangled web. *J. Assoc. Inf. Sci. Technol.* **52**(1), 12 (2001)
17. Martin, A.P., Lyle, J., Namiluko, C.: Provenance as a security control. In: *Proceedings of TaPP* (2012)
18. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: a composable treatment. In: Katz, J., Shacham, H. (eds.) *CRYPTO 2017*. LNCS, vol. 10401, pp. 324–356. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_11

19. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. <https://bitcoin.org/bitcoin.pdf>
20. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, vol. 151 (2014)
21. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Proceedings of ASIACRYPT, pp. 514–532 (2001)



uMine: A Blockchain Based on Human Miners

Henning Kopp^{1(✉)}, Frank Kargl¹, Christoph Bösch¹, and Andreas Peter^{2(✉)}

¹ Institute of Distributed Systems, Ulm University, Ulm, Germany
{henning.kopp, frank.kargl, christoph.boesch}@uni-ulm.de

² Services, Cybersecurity and Safety Group, University of Twente,
Enschede, The Netherlands
a.peter@utwente.nl

Abstract. Blockchain technology like Bitcoin is a rapidly growing field of research which has found a wide array of applications. However, the power consumption of the mining process in the Bitcoin blockchain alone is estimated to be at least as high as the electricity consumption of Ireland which constitutes a serious liability to the widespread adoption of blockchain technology. We propose a novel instantiation of a proof of human-work which is a cryptographic proof that an amount of human work has been exercised, and show its use in the mining process of a blockchain. Next to our instantiation there is only one other instantiation known which relies on indistinguishability obfuscation, a cryptographic primitive whose existence is only conjectured. In contrast, our construction is based on the cryptographic principle of multiparty computation (which we use in a black box manner) and thus is the first known feasible proof of human-work scheme. Our blockchain mining algorithm called uMine, can be regarded as an alternative energy-efficient approach to mining.

Keywords: Blockchain · Applied cryptography · Peer-to-Peer Proof of work

1 Introduction

The last few years have seen a rising interest in the use of blockchain technology. Originally, blockchain architectures emerged from the design of the cryptographic cash system Bitcoin [27] to construct alternative cryptocurrencies. Nowadays, there are applications besides cryptocurrencies, like secure and fair multiparty computations [2, 7, 19] or smart contracts [23, 24, 33], though decentralized cryptocurrencies are still the main driving force behind the blockchain

A major part of this work was conducted during a research stay at the University of Twente. This work was partially funded by the Baden-Württemberg Stiftung and by the Netherlands Organisation for Scientific Research (NWO) in the context of the CRIPTIM project. The full version of this article can be found at IACR [22].

trend. In a nutshell, blockchains provide an immutable distributed ledger and thus can potentially be used to record various forms of asset ownership in different domains.

One of the major drawbacks of blockchain technology is its huge energy consumption. According to de Vries [32] Bitcoin alone consumes at least 2.55 gigawatts of energy making it comparable to countries such as Ireland’s electricity consumption (3.1 gigawatts). We identify this problem to be one of the main challenges of scaling blockchains and allowing for their widespread adoption.

In this article we tackle the issue of the huge energy consumption of blockchains by introducing uMine, a mining algorithm based on a novel proof of human-work construction. Proofs of human-work are cryptographic mechanisms where a prover can convince a verifier that it has spent some amount of human work. In particular, proof of human-work puzzles can only be solved by humans and not by computers under the hardness assumption of some underlying AI problem. This allows us to lower the energy consumption of the blockchain by exchanging the costly proof of work mining algorithm by a proof of human-work which can only be provided by humans.

Proofs of human-work were originally developed by Blocki and Zhou [8] but their construction relies on indistinguishability obfuscation, a theoretical cryptographic primitive where no realization is known. Our new construction in contrast is based on multiparty computation where multiple feasible instantiations exist [5, 9, 10, 17, 28].

Our contributions can be summarized as follows:

- We provide a novel instantiation of a proof of human-work which does not rely on indistinguishability obfuscation but instead uses multiparty computation as a black box.
- We prove the security of our proof of human-work given a secure captcha.
- We use our proof of human-work to construct uMine, a novel energy efficient mining algorithm where the mining is performed by human miners creating proofs of human-work.

2 Building Blocks

Notation: We write $a \leftarrow \mathcal{A}(x)$ to assign to a the output of running the randomized algorithm \mathcal{A} on the input x . We denote with $a \leftarrow \mathcal{A}(x; r)$ the deterministic result of running \mathcal{A} on input x with the fixed randomness r . We say that an algorithm \mathcal{A} is ppt if it runs in probabilistic polynomial time.

2.1 Blockchain

A blockchain is a distributed append-only database together with a consensus algorithm where nodes decide which data is persisted. Usually blockchains are frequently used in the design of cryptographic currencies, to agree on the order

of transactions and provide a single immutable log where all transactions are recorded. The most prominent example is Bitcoin [27], which was the first to introduce the idea of a blockchain. The participants in the consensus protocol bundle transactions into blocks and try to append them to the blockchain by partially inverting a hash function, a process which is called proof of work [11, 12]. Since each node is granted a financial reward in the underlying cryptocurrency for finding a new correct block, the so called mining reward, proof of work achieves alignment of incentives. If these nodes, called miners, solve a proof of work to include wrong transactions in the chain, their financial reward is annihilated, since the other nodes reject wrong blocks. Thus it is rational for miners to only persist valid information in the blockchain.

The proof of work mining algorithm includes a difficulty parameter which in Bitcoin is adjusted every 2016 blocks (approximately two weeks) such that one block is expected to be found every ten minutes assuming no changes in the hash rate. This mechanism allows the global hash rate to change while preserving the block creation rate. While the optimal adjustment of the difficulty parameter is not well understood it is clear that a difficulty parameter needs to be supported when designing alternative mining algorithms.

For further reading regarding blockchains we refer the reader to the survey by Tschorsch et al. [31] or the book by Antonopoulos [3].

While the original vision of blockchains [27] was that each processor has the same chance to mine a block, nowadays the mining industry is dominated by few corporations with specialized mining hardware. Due to this commercialized mining arms race the power consumption of the whole Bitcoin network has increased significantly. For the scalability and the further development of blockchain technology this clearly constitutes a problem.

2.2 Slow Hash Functions

Slow hash functions are a special kind of hash function. While usual hash functions \mathcal{H} are designed to be easy to compute, the evaluation of a slow hash function $\overline{\mathcal{H}}$ in contrast is computationally costly. Normally the evaluation of a slow hash function like `bcrypt` [30] or `scrypt` [29] is on the order of several hundred milliseconds, thus slowing down brute-force attacks significantly. The intuition behind slow hash functions is that an authorized user needs to evaluate them only once, and thus the overhead is negligible.

2.3 Captchas

Captcha is an acronym for a **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part. They are challenge response tests to determine if the user is a human or a program. One major application is to prevent automated registrations of accounts in web services. The most common form of a captcha puzzle consists of a set of warped letters, where the user is requested to recognize the letters, a task which is supposedly hard for computers and easy for humans. There are also other forms like audio-based captchas where the user

is challenged to recognize speech data. To enable automatic verification of a given solution without human assistance the service provider has usually stored a secret set of puzzle-solutions pairs. These pairs are generated by computing a puzzle from a known solution. For verification, access to these puzzle-solution pairs is needed and hence captchas are in general not publicly verifiable.

Since captchas are based on the assumption that some fundamental AI problem is hard to solve, the need to model the human solver as an entity distinct from an algorithm arises. Sometimes this is done in the form of a (yet) unknown algorithm. Since we prefer giving a clearer exposition to giving a philosophically correct one we simply model the human as an oracle that can provide the solutions to a captcha puzzle along the lines of Blocki and Zhou [8].

Definition 1. *Captcha [8]: A Captcha C is a quintuple of algorithms $(\text{Setup}, W, G, \Sigma^{\text{human}}, \text{Verify})$ with the following properties:*

- $PP \leftarrow C.\text{Setup}(1^\lambda)$ is the generation of the public parameters PP given a security parameter λ .
- $\sigma \leftarrow C.W(PP)$ is a randomized algorithm sampling a solution σ given the public parameters.
- $Z \leftarrow C.G(PP, \sigma)$ generates a captcha-puzzle Z with solution σ . We write $C.G(PP, \sigma; r)$ if we fix the randomness r , i.e., if we consider $C.G$ as a deterministic function.
- $\sigma \leftarrow C.\Sigma^{\text{human}}(PP, Z)$ is a solution finding algorithm that takes as input the public parameters and a puzzle Z and outputs a solution σ . It has internal access to a human oracle.
- $b := C.\text{Verify}(PP, Z, \sigma)$ outputs a single bit which is 1 whenever there is a random r , such that $C.G(PP, \sigma; r) = Z$.

The original definition of a captcha by Blocki and Zhou [8] additionally uses a tag which is generated together with the puzzle and needed for the verification of a solution σ . We stress that our construction later also works with the definition of Blocki and Zhou, where the tag is set as undefined. However, the tags are not necessary in our construction and thus we decided to present our work using a simpler definition to aid in the understanding.

If the randomness r which was used to generate the captcha in the algorithm $C.G$ is known it may be possible to invert $C.G$. In the case of image based captchas r determines the chosen transformations, e.g., rotation, addition of noise, and their parameters applied on the solution to yield a puzzle [1]. Knowledge of these may allow an attacker to invert the used transformations and thus recover the solution σ from a puzzle Z without the use of human work. Consequently the security of a captcha puzzle $Z = C.G(PP, \sigma; r)$ is usually based on the secrecy of the random value r , which was used to generate the puzzle [1, Section Who knows What?].

Additional Requirements for Our Construction: In contrast to the original definition by Blocki and Zhou [8] we require the generation of the puzzles $C.G(PP, \sigma; r)$ to be collision-free, i.e., injective, in its randomness r and in its

solutions σ . Regarding the solutions σ it is natural to assume that there can be no two different solutions to the same puzzle. Regular image based captchas do have this property. Regarding injectivity in the randomness r we can assume that it serves as an enumeration of the puzzle space for a given captcha solution. Consider the case of image based captchas where the randomness determines the type of transformations. Different transformations with different parameters yield different puzzles and thus collision freeness can be assumed.

Use in Our Instantiation: In our construction of a proof of human-work the randomness r used in the puzzle generation C.G is set to a deterministic value containing a slow hash of the solution $\mathcal{H}(\sigma)$. This way it is easy to verify a solution publicly, given a puzzle, since one only needs to regenerate the puzzle from the solution σ using the same randomness and check if the given puzzle equals the computed one. The use of a slow hash function is necessary to prevent bruteforcing of the solution using the verification algorithm.

Security Properties: We require that any captcha should be solvable by a human. We use the term human-work unit to denote the effort needed to solve a single instance of a captcha. Although the time needed to solve a captcha may depend on the human and his abilities, we expect these differences to be small and similar to the differences in performance of different computer hardware.

Definition 2. *Honest Human Solvability [8]:* We say that a human-machine solver $C.\Sigma^{human}$ controls m human-work units if it can query its human oracle at least m times. We say that a captcha system $C = (\text{Setup}, \text{W}, \text{G}, \Sigma^{human}, \text{Verify})$ is honest human solvable if for every polynomial $m = m(\lambda)$ and for any human $C.\Sigma^{human}$ controlling m human-work units, it holds that

$$P \left[\begin{array}{l} \forall PP \leftarrow C.\text{Setup}(1^\lambda); \\ \forall i \in [m] (\sigma_i^* \leftarrow C.\text{W}(PP)); \\ \forall i \in [m] (Z_i^* \leftarrow C.\text{G}(PP, \sigma_i^*)); \\ (\sigma_1^*, \dots, \sigma_m^*) \leftarrow C.\Sigma^{human}(PP, Z_1^*, \dots, Z_m^*) \end{array} \right] \geq 1 - \text{negl}(\lambda)$$

Finally we require that captchas are hard for computers to solve without access to a human oracle.

Definition 3. *Captcha Break [8]:* We say that a ppt adversary \mathcal{A} who has at most m human-work units breaks security of a captcha system $C = (\text{Setup}, \text{W}, \text{G}, \Sigma^{human}, \text{Verify})$ if there exist polynomials $m = m(\lambda)$, $n = \text{poly}(\lambda)$ and $\mu(\lambda)$ such that if \mathcal{A} controls at most m human-work units it holds that

$$P \left[\begin{array}{l} \forall PP \leftarrow C.\text{Setup}(1^\lambda); \\ \forall i \in [n] (\sigma_i^* \leftarrow C.\text{W}(PP)); \\ \forall i \in [n] (Z_i^* \leftarrow C.\text{G}(PP, \sigma_i^*)); \\ S \leftarrow \mathcal{A}(PP, Z_1^*, \dots, Z_n^*); \\ \forall i \in [n] (b_i \leftarrow \max_{\sigma \in S} C.\text{Verify}(PP, Z_i^*, \sigma)) : \\ \sum_{i \in [n]} b_i \geq m + 1 \end{array} \right] \geq \frac{1}{\mu(\lambda)}$$

It is debatable, whether in AI research the concept of a security parameter applies [1]. AI research does not deal with asymptotics and thus it can be argued that problem classes are either solvable or unsolvable, independent of the concrete problem in the problem class. This is in contrast to classical cryptography where it may be feasible to solve certain “small” instances of problems without solving all, e.g., factorization of small integers may be possible, without being able to factorize all integers. Thus, if captchas are either solvable or unsolvable in the real world our definitions can be made even stronger by setting the negligible term in the definition of honest human solvability to zero. Without a tunable security parameter, a captcha is called broken if the attacker has a success probability of 1 of finding solutions without access to a human oracle.

2.4 Proof of Human-Work Puzzles

Proof of human-work puzzles (PoH) were first introduced by Blocki and Zhou [8]. Their goal was to construct a publicly verifiable proof that some amount of human work has been exercised. Their construction relies on indistinguishability obfuscation [14] and thus is currently infeasible.

A PoH in contrast to a captcha has a tunable difficulty parameter and is publicly verifiable. That means that no secret knowledge is needed neither to generate nor to verify a PoH. Especially, the solution does not need to be known to generate the puzzle as is the case with captchas. The difficulty parameter enables its use as a mining algorithm in a blockchain as explained above.

Definition 4. *Proof of Human-work Puzzle [8]: A proof of human-work puzzle system POH consists of four algorithms $(\text{Setup}, \text{G}, \Sigma^{\text{human}}, \text{V})$ where:*

- $PP \leftarrow \text{POH.Setup}(1^\lambda, 1^\omega)$ is a randomized system setup algorithm that takes as input a security parameter λ and a difficulty parameter ω and outputs public parameters of the system PP .
- $x \leftarrow \text{POH.G}(PP)$ is a randomized algorithm that takes as input the public parameters PP and outputs a puzzle x .
- $a \leftarrow \text{POH.}\Sigma^{\text{human}}(PP, x)$ is a solution finding algorithm that has access to a human oracle. It takes as input the public parameters and a puzzle x and outputs a solution σ to the puzzle.
- $b := \text{POH.V}(PP, x, a)$ is a deterministic verification algorithm that takes as input the public parameters PP , together with a puzzle x and a solution a and outputs a bit b where $b = 1$ if and only if a is a valid solution to the puzzle x .

Similar to a captcha we require from PoHs that they are solvable by a human, with a success probability depending on the difficulty parameter. Following the notation of Blocki and Zhou [8] and Miller et al. [26] we define $\zeta(m, \omega) := 1 - (1 - 2^{-\omega})^m$. This describes the probability of finding a valid solution using m queries to the human oracle.

Definition 5. *Honest Human Solvability [8]: We say that a PoH system $\text{POH} = (\text{Setup}, \text{G}, \Sigma^{\text{human}}, \text{V})$ is honest human solvable if for every polynomial*

$m = m(\lambda)$, and for any honest human-machine solver $\text{POH}.\Sigma^{\text{human}}$ who controls m human-work units, it holds that

$$P \left[\begin{array}{l} \forall PP \leftarrow \text{POH.Setup}(1^\lambda, 1^\omega); \\ x^* \leftarrow \text{POH.G}(PP); \\ a^* \leftarrow \text{POH}.\Sigma^{\text{human}}(PP, x^*); \\ \text{POH.V}(PP, x^*, a^*) = 1 \end{array} \right] \geq \zeta(m, \omega) - \text{negl}(\lambda)$$

Further, we require that any adversary that controls too few human-work units succeeds in solving a PoH only with negligible probability.

Definition 6. *Adversarial Human Unsolvability [8]:* We say that a ppt algorithm \mathcal{A} breaks security of the PoH system $\text{POH} = (\text{Setup}, \text{G}, \Sigma^{\text{human}}, \text{V})$ if for some polynomials $m = m(\lambda)$ and $\mu(\lambda)$ when \mathcal{A} controls at most m human-work units, it holds that

$$P \left[\begin{array}{l} \forall PP \leftarrow \text{POH.Setup}(1^\lambda, 1^\omega); \\ x^* \leftarrow \text{POH.G}(PP); \\ a^* \leftarrow \mathcal{A}(PP, x^*); \\ \text{POH.V}(PP, x^*, a^*) = 1 \end{array} \right] \geq \zeta(m + 1, \omega) + \frac{1}{\mu(\lambda)}$$

2.5 Multiparty Computation Protocol

Multiparty computation protocols (MPC) are cryptographic protocols that allow a set of mutually distrusting parties to collaboratively compute a function with private input values. For example, the parties evaluate some $f(y_1, \dots, y_n)$, where the input y_i is only known to party i . The participants in the protocol do not learn anything beyond their own inputs and the solution $f(y_1, \dots, y_n)$.

While traditional schemes suffered from severe performance issues, over the last few years, multiple practical solutions that can deal with arbitrary computable functions f have emerged [5, 9, 10, 17, 28]. In our case we require a secure multiparty protocol with k different parties, where $k - 1$ participants can be controlled by an *active* attacker. An active (malicious) attacker can arbitrarily deviate from any protocol execution in an attempt to cheat. This is in contrast to passive (semi-honest) attackers who try to gather as much information about the underlying inputs and (intermediate) outputs but honestly follow the prescribed steps in the given protocol.

We use MPC as a black box in this article, having secret sharing based MPC protocols in mind (such as SPDZ [9]). For this we define an MPC protocol MPC as a triple of ppt algorithms (Setup, Share, Reveal) where:

- $PP \leftarrow \text{MPC.Setup}(1^\lambda)$ is a randomized algorithm that takes a security parameter λ as input and sets up the protocol by distributing the keys and parameters. It outputs the public parameters for the system.
- $\langle y \rangle \leftarrow \text{MPC.Share}(PP, y)$ shares the value y among the k participants using a secret sharing scheme such that each of the k participants receives one share. We use $\langle y \rangle$ to denote the vector of secret shares of y . Note that the Share algorithm can be executed by one of the participants or any other external party with access to the parameters PP .

- $y \leftarrow \text{MPC.Reveal}(PP, \langle y \rangle)$ reconstructs the value y from its secret shares $\langle y \rangle$. The MPC participants send their secret shares $\langle y \rangle$ to an external party who can then execute MPC.Reveal and learn the value y ; consequently being the only party knowing y in the clear.

By abuse of notation, we apply computable functions on secret shares to denote the computation of the secret shares of the result of the function applied to the clear values, i.e., we denote $\langle f(y_1, \dots, y_n) \rangle$ by $f(\langle y_1 \rangle, \dots, \langle y_n \rangle)$. The clear values are not revealed by this operation. Note that knowledge of the public parameters may be needed for this computation, but is left out to simplify our notation.

One possible MPC framework for our use is SPDZ [9], which consists of a preprocessing and an online phase. The preprocessing phase is independent of the function to be computed as well as of the inputs. In the online phase the actual function is evaluated. The online phase has a total computational and communication complexity linear in the number of participants k . The work done by each participant in SPDZ is only a small constant factor larger than what would be required to compute the function in the clear. Thus, SPDZ provides an efficient framework which satisfies our requirements.

3 Our Construction

3.1 Overview

On a high level we are interested in exchanging the proof of work by a PoH. The parties involved in our system are human miners, i.e., miners who control some human-work units, who try to solve the PoH puzzles in order to gain the block rewards, as well as a consortium of k puzzle generators. To mine a new block, each human miner requests a puzzle for a proof of human-work from the puzzle generators. The puzzle is linked to the transactions the human miner wants to persist, as well as to the current block in the blockchain. Throughout the generation of the puzzle, the solution is unknown to any single party, in contrast to regular captchas. If the human miner does not succeed in solving the puzzle it can request a new puzzle from the captcha generators. If the human miner succeeds however, it can publish the new block containing the captcha puzzle, its solution, and the transactions. A node which receives a new block can check the transactions and the proof of human-work for validity. It accepts the block if all of these are correct and mining continues on top of the new block.

3.2 Our Proof of Human-Work

We give a new instantiation of a PoH puzzle which does not rely on indistinguishability obfuscation [14] like the work of Blocki and Zhou [8], but instead on MPC. Our construction is the first PoH which is feasible and does not involve a trusted third party. In contrast to the work of Blocki and Zhou [8], computing the algorithm POH.G in our construction needs interaction with a set of captcha generators $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$. This set can be a fixed consortium of k parties as will

be explained in Subsect. 3.4. The assumption of interaction poses no problem for our use case since for mining on a blockchain the miners are required to be online anyway to receive the latest blocks and transactions.

The intuition behind our construction is that the captcha generators collaboratively compute a captcha puzzle using multiparty computation. This computation is done in such a way that each captcha generator has access to only a secret share of the solution, but not to the solution itself. Consequently, the solution is unknown to any single party.

Nevertheless, since it is a captcha the solution can be found by querying the human oracle. If the hash of the solution σ of the captcha puzzle is above a difficulty parameter the solution is deemed invalid for the PoH. Thus, for creating a valid PoH one may need to solve multiple captchas depending on the difficulty, until a captcha solution with a small hash is found. This captcha solution then constitutes a PoH a .

In order to achieve public verifiability, remember that the generation of a captcha puzzle is a probabilistic algorithm using the solution. If the randomness used in the captcha generation is known it is possible to regenerate the captcha puzzle from the solution. This allows public verification of the solution since the recomputed puzzle can be compared to the given puzzle. In our construction the randomness in the captcha generation is derived from the captcha solution itself.

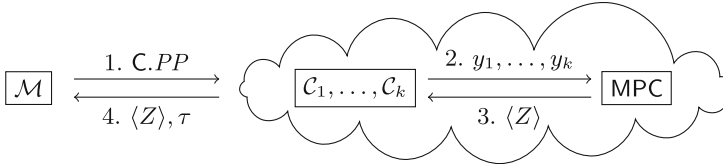


Fig. 1. Simplified Overview of our Proof of Human-work Construction

A standard workflow is shown in Fig. 1. As a first step the captcha generators $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$ are initialized by executing $MPC.Setup$ with an appropriate security parameter. Now, suppose a human \mathcal{M} wants to compute a proof of human-work.

First, the human sets up the public parameters PP for the proof of human-work by executing $POH.Setup$. The public parameters consist of the public keys of the captcha generators, a difficulty parameter ω , and a security parameter λ .

Next, the human queries the captcha generators to obtain a captcha by executing $POH.G$. To this end, he computes public parameters $C.PP$ for the captcha by running the setup algorithm of the captcha with security parameter λ . The public parameters $C.PP$ are distributed to the captcha generators $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$ (Step 1 in Fig. 1). Each captcha generator \mathcal{C}_j chooses a random value y_j and shares it among the other captcha generators, according to the multiparty computation protocol (Step 2 in Fig. 1). Together, the captcha generators sample a solution σ of the captcha by using the sum of their chosen

randomness $y_1 \oplus \dots \oplus y_k$ in the sampling algorithm C.W together with the public parameters of the captcha C.PP. This solution σ is not revealed, but rather stays secret shared between the captcha generators. Thus, no captcha generator knows the solution. From the shared solution $\langle \sigma \rangle$ the shares of the captcha puzzle $\langle Z \rangle$ are computed by the captcha generators using multiparty computation as $\langle Z \rangle \leftarrow \text{C.G}(\text{C.PP}, \langle \sigma \rangle; \overline{\mathcal{H}}(\langle \sigma \rangle))$ (Step 3 in Fig. 1). Here, $\overline{\mathcal{H}}$ denotes a slow hash function. Each captcha generator signs its share of the puzzle and sends it to the human \mathcal{M} (Step 4 in Fig. 1). We call these signed shares τ . The human then reveals the puzzle Z by executing the MPC.Reveal algorithm of the multiparty computation protocol. \mathcal{M} is now the only person knowing the captcha puzzle Z , and the solution σ is unknown to any single party. The puzzle to the PoH is $x = (\text{C.PP}, Z, \tau)$.

In order to create the PoH the human solves the captcha puzzle Z by executing its captcha solving algorithm. This yields a solution σ to the captcha. If $\mathcal{H}(\sigma) < T_\omega$ then this constitutes a valid proof of human-work. Here, \mathcal{H} is a hash function and $T_\omega = 2^{n-\omega}$ analogous to Blocki and Zhou [8], where ω is the difficulty parameter and n is the bit size of the output of \mathcal{H} . If this is not the case, i.e., if the hash of the solution is not small enough, the human has to start again by querying the captcha generators for a new puzzle until he succeeds in solving a captcha with a small solution. The PoH consists of the solution to the captcha puzzle $a = \sigma$.

To verify the PoH, i.e., to execute POH.V, the public parameters PP , the puzzle x , and its solution a are needed. The verifier first needs to check if the puzzle has been computed in a correct way, i.e., by the captcha generators. This can be done by checking τ , the signatures on the shares of the solution which are included in the puzzle x . Next, the verifier checks that the hash of the solution is small enough, i.e., if $\mathcal{H}(\sigma) < T_\omega$. As a final step, the verifier checks that the solution is a correct solution to the captcha. This can be done by simply regenerating a puzzle from the solution and checking equality between the recomputed puzzle and the original puzzle. I.e., it needs to be checked if $\text{C.G}(\text{C.PP}, \sigma; \overline{\mathcal{H}}(\sigma)) = Z$. If any of these three steps fails, the PoH is rejected. Otherwise it is considered valid.

Construction 1. *Let C be a secure human solvable captcha and MPC be a secure MPC scheme initialized with public parameters $\text{MPC.PP} \leftarrow \text{MPC.Setup}(1^\lambda)$. Let $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a hash function. We define $T_\omega = 2^{n-\omega}$ analogous to Blocki and Zhou [8]. We use T_ω to scale our difficulty parameter ω , since $P(\mathcal{H}(r) < T_\omega) = 2^{-\omega}$ for a random r . We now construct a PoH by defining the following operations.*

- $PP \leftarrow \text{POH.Setup}(1^\lambda, 1^\omega)$ outputs the parameters PP containing λ , ω , and the public keys of the captcha generators $\mathcal{C}_1, \dots, \mathcal{C}_k$.
- $x \leftarrow \text{POH.G}(PP)$ is computed by interacting with the set of captcha generators $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$. First we parse λ and ω from PP locally and compute the public parameters for the captcha as $\text{C.PP} \leftarrow \text{C.Setup}(1^\lambda)$. These are then given to the captcha generators. Each captcha generator \mathcal{C}_j chooses

a secret random value y_j and uses $\text{MPC.Share}(\text{MPC.PP}, y_j)$ to distribute shares of its value y_j among the k captcha generators. In a next step the captcha generators compute $\langle \sigma \rangle = \text{C.W}(\text{C.PP}, \langle y_1 \rangle \oplus \dots \oplus \langle y_k \rangle)$ using MPC such that each of the captcha generators now possesses a secret share of the solution σ to the captcha. The solution σ is not revealed but stays in the secret shared domain. Next, the captcha generators compute the puzzle $\langle Z \rangle \leftarrow \text{C.G}(\text{C.PP}, \langle \sigma \rangle; \overline{\mathcal{H}}(\langle \sigma \rangle))$. The captcha generators each sign their shares $\langle Z \rangle$ of the puzzle as τ which later guarantees that each of the captcha generators \mathcal{C}_j has participated in the protocol.

Finally the captcha puzzle Z is revealed by executing $Z = \text{MPC.Reveal}(\text{MPC.PP}, \langle Z \rangle)$ and the PoH puzzle $x = (\text{C.PP}, Z, \tau)$ is output.

- $a \leftarrow \text{POH.}\Sigma^{\text{human}}(\text{PP}, x)$ computes a solution a to a PoH as follows. First, the parameters λ, ω , are parsed from PP . The puzzle is parsed as $(\text{C.PP}, Z, \tau) = x$. Then the captcha solving algorithm is queried $\sigma \leftarrow \text{C.}\Sigma^{\text{human}}(\text{C.PP}, Z)$. If $\mathcal{H}(\sigma) < T_\omega$ we return the solution $a = \sigma$. Otherwise $a = \perp$ is returned.
- $b := \text{POH.V}(\text{PP}, x, a)$ first parses $\sigma = a$ and checks if $\mathcal{H}(\sigma) < T_\omega$. If that is not the case $b = 0$ is returned. Otherwise we parse $(\text{C.PP}, Z, \tau) = x$ and check the signatures and the final shares of the puzzle τ to ensure that the puzzle has been generated in a correct way, i.e., by the captcha generators \mathcal{C}_j , and not by anyone else. This is possible, since the public keys of the captcha generators needed for the verification of the signatures are contained in PP . If τ is invalid, we return $b = 0$. As a third step we need to ensure that the solution σ is a valid solution to the captcha. This can be done by checking if $\text{C.G}(\text{C.PP}, \sigma; \overline{\mathcal{H}}(\sigma)) = Z$. If that is the case, return $b = 1$, otherwise return $b = 0$.

Theorem 1. *If our construction is instantiated with a secure and honest human solvable captcha, then the resulting PoH is honestly human solvable and adversarial human unsolvable under the assumption that at least one of the k captcha generators $\mathcal{C}_1, \dots, \mathcal{C}_k$ is honest.*

Proof. The proof can be found in the full version of the paper [22].

3.3 Block Generation

In this section we describe a design of a blockchain which is based on proofs of human-work. We call the resulting mining process uMine. In order to mine a new block B_i , a human miner \mathcal{M} needs access to the previous block B_{i-1} . Further it needs to have a set of transactions Tx_i , which it wants to persist in the new block B_i .

To mine a new block, first, the algorithm $\text{POH.Setup}(1^\lambda, 1^\omega)$ is run in order to generate the public parameters for the PoH. The security parameter λ is globally fixed but the difficulty parameter ω needs to be adjusted dynamically to ensure a stable block creation rate. In Bitcoin the difficulty parameter is adjusted every 2016 blocks such that the expected block generation interval is

10 min, assuming no changes in the global mining power. Although it is unknown if these parameters are optimal, there is insufficient research covering the choice of parameters and thus we see no reason to deviate from them.

The captcha generators $\mathcal{C}_1, \dots, \mathcal{C}_k$ are initialized by computing the algorithm $\text{MPC.PP} \leftarrow \text{MPC.Setup}(1^\lambda)$. After generating the public parameters for the PoH the human miner \mathcal{M} contacts the set of captcha generators to receive a PoH puzzle x_i for the new block B_i as we will explain in the following.

The human miner splits the hash of the transactions $\mathcal{H}(Tx_i)$, as well as the hash of the current block h_{i-1} into secret shares $\langle \mathcal{H}(Tx_i) \rangle, \langle h_{i-1} \rangle$ which are distributed to the captcha generators.¹ Each captcha generator computes the captcha parameters as $PP \leftarrow \text{C.Setup}(1^\lambda)$. Together they compute a random captcha solution in the secret shared domain as follows. First, each captcha generator \mathcal{C}_j chooses a secret input y_j uniformly at random. This secret randomness is shared among the captcha generators by computing $\langle y_j \rangle = \text{MPC.Share}(\text{MPC.PP}, y_j)$. The shared randomness is used to compute the secret shared random captcha solution as $\langle \sigma_i \rangle = \text{C.W}(PP, \langle y_1 \rangle \oplus \dots \oplus \langle y_k \rangle)$. This way, none of the captcha generators knows the solution σ_i .

To be able to use our PoH construction from above in a blockchain we need to include a reference to the previous block $h_{i-1} = \mathcal{H}(B_{i-1})$ and the new transactions Tx_i in the puzzle. Otherwise, if an already persisted transaction in the blockchain is modified the PoH is still valid, and thus integrity of persisted transactions cannot be guaranteed. In order to connect the hash of the previous block and the transactions with the puzzle the captcha generators compute their secret shares of the captcha puzzle Z_i given their shares of the solution $\langle \sigma_i \rangle$ as $\langle Z_i \rangle = \text{C.G}(PP, \langle \sigma_i \rangle; \overline{\mathcal{H}}(h_{i-1}, \mathcal{H}(Tx_i), \langle \sigma_i \rangle))$. I.e., the hash of the previous block h_{i-1} and the current transactions $\mathcal{H}(Tx_i)$ are included in the randomness of the puzzle generation. At this stage, each captcha generator has a share of a captcha puzzle $\langle Z_i \rangle$ where the solution is effectively unknown to any single party.

Next, the captcha generators send their signed final shares of the captcha puzzle to the human miner \mathcal{M} who assembles them as a PoH $x_i = (\text{C.PP}, Z_i, \tau_i)$, where $Z_i = \text{MPC.Reveal}(\text{MPC.PP}, \langle Z_i \rangle)$ is the revealed captcha puzzle. Here, τ_i is the set of the final signed shares of the puzzle from the multiparty protocol run. It is used to prove that each captcha generator participated in the protocol and thus guarantees that the puzzle has been generated in a correct way.

The human can now try to solve its PoH x_i . If the hash of the solution to the encapsulated captcha is too big, that is, when $\mathcal{H}(\sigma) \geq T_\omega$, the human requests another PoH puzzle from the captcha generators. If the human eventually succeeds to find a solution σ'_i to the PoH, it can locally verify its solution, by checking if $Z_i = \text{C.G}(\text{C.PP}, \sigma'_i; \overline{\mathcal{H}}(h_{i-1}, \mathcal{H}(Tx_i), \sigma'_i))$. If that is the case, it can publish the new block B_i containing the captcha puzzle x_i , its solution $a_i = \sigma'_i$,

¹ We explain our protocol using classical secret sharing based MPC, where a dealer distributes shares of the input and a set of nodes computes on these shares. We hope this makes our explanations more clear. In a practical implementation we suggest the use of SPDZ [9] which is a highly optimized variant thereof.

as well as the transactions Tx_i and a reference to the previous block in form of a hash h_{i-1} .

Each receiving node verifies the solution to the captcha, by running $\text{POH.V}(PP, x_i, a_i)$. More specifically, the captcha puzzle Z_i is recomputed from its solution and it is examined if this leads to the same Z_i , i.e., if $Z_i = \text{C.G}(\text{C.PP}, \sigma_i; \overline{\mathcal{H}}(h_{i-1}, \mathcal{H}(Tx_i), \sigma_i))$. Additionally the signed shares of the puzzle τ_i are checked for correctness of the signature to guarantee that the puzzle was created by the correct parties. Further it is examined if $\mathcal{H}(\sigma) < T_\omega$ holds.

Beyond these steps of verification of the PoH for usage in a blockchain, the difficulty parameter ω and the validity of the transactions in the new block is checked, as in Bitcoin.

If any of these checks fails, the new block is discarded and mining continues on top of the old block B_{i-1} . Otherwise the human miners can continue to generate blocks on top of B_i .

If one of the captcha generators is malicious it may abort the generation of the puzzle to the PoH, thus preventing that new blocks can be mined by a PoH. To remedy this situation we additionally allow blocks to be mined by proof of work as in Bitcoin. However, in order to keep the advantages of the mining with human work, we use a distinct difficulty parameter from the proof of human-work. The difficulty parameter of the proof of work is chosen in such a way that mining a block using proof of work is significantly harder than mining with proofs of human-work. This ensures that the mining process is dominated by PoH and proof of work is only used as a fallback mechanism.

3.4 Choosing the Captcha Generators

One important design consideration is the choice of the captcha generators. In this paper we discuss only a static consortium. However, it is possible to choose captcha generators dynamically based on the randomness contained in the blockchain. This is explained in the full version of the paper [22].

Static Consortium: In the most simple case we can assume a consortium of k fixed entities. If some of them are not online, no proof of work puzzles will be generated. In this case proof of work can be used as a fallback mechanism as explained above. Thus, if the captcha generators are not online, our system collapses to proof of work mining. Due to our use of SPDZ [9] we can tolerate up to $k - 1$ cheaters. However, if all k parties collude, they may be able to generate captchas where they already know the solution and thus mine faster than any human miner, achieving a significant financial gain. We can remedy this situation by providing incentives for captcha generators to expose collusions and then punish the colluding parties and reward the traitor (see next paragraph). Intuitively this provides incentives for the traitor to reveal collusion, thus preventing the formation of collusions in the first place. For this to work the captcha generators additionally publish a signed commit on their shares of the solution σ . These can be included in the information used to verify that the puzzle was

generated by the correct parties τ which consists of the signed shares of the puzzle.

The Traitor Reward Protocol: The traitor reward protocol has two rounds. In a first round any captcha generator can claim that the captcha generators colluded by publishing the particular shares of the puzzle solution of each captcha generator. If collusion influenced the creation of the current block, at least the miner of the block knows this information.²

Other parties are allowed to chime in with their claims of collusion by also publishing commits to the respective shares of the captcha generators. After a fixed timespan the first round ends and the second round starts.

In the second round the captcha generators have to reveal their commitments on their shares of the solution. If any of them does not comply within a fixed time period, collusion can be assumed. The claims of the supposed traitors are handled in the order of their arrival. Note that since we are in a distributed setting there is no global time. However, since we want to reward only some traitor to deter collusion and not necessarily the first traitor, this poses no problem. The claimed shares of the solutions are compared with the real shares of the captcha generators and if they coincide, collusion has occurred. In this case, the witness of collusion can be persisted in the blockchain as a regular transaction and the block reward of the fraudulent block is granted to the traitor. Note that there is no need to invalidate the block which has been mined fraudulently, since it contains only valid transactions and thus, is a valid block.

The time periods for the traitor reward protocol need to be chosen appropriately and the block reward needs to be locked for a fixed amount of time to prevent that it is already spent before collusion claims can be handled.

Consequently each captcha generator can choose to either collude or not collude and orthogonally to betray the other nodes or refrain from doing so, leading to the four strategies (collude, betray), (not collude, claim betrayal), (collude, not betray), and (not collude, not claim betrayal). The incentives need to be designed such that not colluding and not claiming betrayal has to be the strictly dominant strategy in a game-theoretic sense, because this is the behavior we want to support in the captcha generators. Colluding and not betraying the others needs to be a strictly dominated strategy, such that colluding nodes gain a profit from betraying the other conspirators. However, the profit needs to be smaller than if there would have been no collusion at all. Otherwise it may be rational to stage betrayal and share the reward with the other nodes. For the other two strategies there are no restrictions.

² It may be the case that the k colluding parties decide to reveal the solution to the PoH by MPC, such that no one knows the partial solutions of the other captcha generators. Even then $k - 1$ nodes can collude to reveal their particular shares, recompute the missing share of the last captcha generator, and claim betrayal. This increases their reward in contrast to not betraying the last captcha generator. For our cases it is irrelevant if a subset of captcha generators or only a single one claims betrayal. Though for the sake of simplicity we assume a single traitor.

It is interesting to note that under the assumption of rational actors the traitor reward protocol will never be executed. Thus, collusions are prevented by the existence of the traitor reward protocol and not by its execution.

4 Security

Additionally to the trust assumptions in usual blockchain systems, we require that at least one of the k captcha generators is honest due to our use of SPDZ [9]. For the security of our PoH scheme we require that it is instantiated with a secure captcha system. If this is not the case and the captcha can be solved without human work, our uMine construction will not lose its functionality but instead degrade to a form of proof of work. Other than the use of a secure captcha we do not impose any additional trust assumptions.

Since our main focus is to substitute the proof of work by an environmentally friendly alternative, some of the attacks in Bitcoin also affect our scheme. In particular, since we treat forks as in Bitcoin, our construction is vulnerable to 51% attacks and eclipse attacks [16]. Although, to successfully pull off a 51% attack an attacker needs to be in charge of more than 50% of the human work units in the system instead of more than 50% of the computational resources, as in Bitcoin. However, we do not introduce any new security vulnerabilities under our assumptions.

In the following we discuss the infeasibility of selected attacks.

History Rewriting: If old transactions are changed in the blockchain, the solution to the captcha is invalidated, since the transactions are also used in the generation of the puzzle Z_i from the solution σ_i as follows: $Z_i = \text{C.G}(PP, \sigma_i; \overline{\mathcal{H}}(B_{i-1}, \mathcal{H}(Tx_i), \sigma_i))$.

Finding two sets of transactions $Tx_i \neq Tx'_i$ which yield the same puzzle Z_i for the solution σ_i , implies that $\overline{\mathcal{H}}(B_{i-1}, \mathcal{H}(Tx_i), \sigma_i) = \overline{\mathcal{H}}(B_{i-1}, \mathcal{H}(Tx'_i), \sigma_i)$, since C.G is collision-free in its randomness by assumption. So, an attacker would have to find a collision in the slow hash function $\overline{\mathcal{H}}$ to successfully change the old transactions which is assumed to be infeasible. Note that changing the solutions σ_i also does not yield an attack, since they are referenced in the next block.

Thus, the only way left to change transactions already persisted in the blockchain would be to split the chain after this block and redo the human work. This is only possible if an attacker controls more than 50% of the human resources in the network.

Transaction Denial Attack: In a transaction denial attack, the attacker tries to prevent a transaction from being confirmed. If the attacker is a human miner, it can only succeed if his chain grows faster than the chain containing the transaction it wants to censor. This is exactly the case if it has more than 50% of the human power in the system. As soon as that is not the case anymore, the transaction will be included in the blockchain.

If the attacker is one of the captcha generators instead, it cannot prevent inclusion of the transaction in the chain, by not serving a captcha to the miners

which want to include that specific transactions. That is due to the fact that the captcha generators do not see the transactions but only their hash. Identifying if a transaction is included in a set of transactions, given only the hash of the set is infeasible.

Thus, an attacker is unable to target specific transactions for denial.

Bruteforcing of Solutions: Since captchas usually do not have much entropy—image based captchas consists of up to 12 characters—an attacker \mathcal{A} may have the idea to simply brute-force the solution σ_i to a puzzle Z_i . This would possibly allow \mathcal{A} to mine a block without spending human labor on it.

While we can almost never fully prevent brute-forcing, our use of a slow hash function impedes the attempts of the attacker. To brute-force a solution, an attacker needs to guess a σ'_i , compute $Z'_i = \text{C.G}(PP, \sigma'_i; \overline{\mathcal{H}}(B_{i-1}, \mathcal{H}(Tx_i), \sigma'_i))$ and then check if $Z'_i = Z_i$. I.e., \mathcal{A} needs to evaluate a slow hash function for each guess, which is expensive.

5 Related Work

There is a series of related work which suggests an alternative to Bitcoin’s wasteful proof of work.

The most famous among these approaches is probably **proof of stake** [6, 20], where the scarcity used to power the blockchain is the underlying currency itself. In proof of stake the miner of the next block is chosen pseudorandomly among the set of all miners. The probability of a miner being chosen to create a new block is dependent on its wealth which can lead to an undesirable “rich get richer” scenario. A common problem in proposals for proof of stake is that in the case of a blockchain fork miners have nothing to lose by trying to mine on both chains, thus preventing the fork from resolving. Peercoin [20] solves this problem by including centralized checkpoints in the code, thus introducing a trusted third party. Other protocols such as Algorand [15] do not provide incentives for the participants. The first construction to provably solve the proof of stake problem is due to Kiayias et al. [18].

Other approaches to substitute proof of work are **proofs of storage** [21, 25] i.e., proving possession of a specific file, or **proofs of space** [4, 13], where a miner only constructs a proof about the size of its memory resources. However, these approaches will also invariably degrade into a hardware arms race and thus do not solve the problem of the vast energy consumption of blockchain technology.

A spiritual predecessor of our work is **HumanCoin** [8]. However HumanCoin is based on a PoH based on indistinguishability obfuscation [14], a cryptographic principle where no construction is known yet and thus is currently infeasible.

Their construction of a PoH, and consequently HumanCoin requires a trusted setup phase for the generation of the obfuscated programs which are used to generate the captchas without revealing the solutions.

If the unobfuscated programs are known, miners can generate puzzle-solution pairs to the PoH without spending any human work by running the puzzle-generation in the clear. In contrast, in our work we are able to verify that the

puzzles have been created in such a way that no-one knows the solution by publishing and verifying the signed final shares of the captcha generators τ .

In HumanCoin, collision-freeness of the captcha puzzle generation algorithm is not stated, but if this is not assumed their scheme is trivially insecure. Their PoHs are generated by $x_i = \text{C.G}(PP; \mathcal{H}(Tx_i, h_{i-1}))$, where h_{i-1} is the hash of the previous block. If there is no collision resistance in the randomness, old transactions can be changed without changing the puzzle, thus invalidating the integrity of the transactions stored in blockchain.

HumanCoin does not implement any countermeasures against brute-forcing the solution to the PoH from its verification function in contrast to our use of a slow hash function.

In contrast to HumanCoin our PoH puzzle generation phase is online and requires k captcha generators. However, this poses no problem, since to mine new blocks a miner has to receive new transactions and blocks and thus is required to be online anyway.

6 Conclusion

We have introduced uMine, an energy-efficient alternative to proof of work mining which utilizes human workers. Our construction is based on a novel instantiation of proofs of human-work which relies on MPC, thereby answering an open question of Blocki and Zhou [8] whether proofs of human work without indistinguishability obfuscation are possible. Our proof of human-work scheme is introduced as a separate building block and thus may find applications beyond cryptocurrencies.

Our uMine system may share similarities with early manual accounting systems, whose bookkeepers were financially compensated. Additionally, our system decentralizes the ledger and provides anyone with the opportunity to be an accountant, provided it accepts the remuneration. We leave the social and economical implications of our work as an open question.

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_18
2. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multi-party computations on Bitcoin. In: S&P 2014, pp. 443–458. IEEE (2014)
3. Antonopoulos, A.M.: Mastering Bitcoin, unlocking digital cryptocurrencies. O’Reilly Media, December 2014. <https://github.com/aantonop/bitcoinbook>
4. Ateniese, G., Bonacina, I., Faonio, A., Galesi, N.: Proofs of space: when space is of the essence. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 538–557. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_31

5. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 169–188. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_11
6. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 142–157. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_10
7. Bentov, I., Kumaresan, R.: How to use Bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_24
8. Blocki, J., Zhou, H.-S.: Designing proof of human-work puzzles for cryptocurrency and beyond. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 517–546. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_20
9. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_38
10. Damgård, I., Zakarias, S.: Constant-overhead secure computation of Boolean circuits using preprocessing. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_35
11. Dwork, C., Goldberg, A., Naor, M.: On memory-bound functions for fighting spam. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 426–444. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_25
12. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_10
13. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 585–605. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_29
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.* **45**(3), 882–929 (2016)
15. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling Byzantine agreements for cryptocurrencies. In: SOSP 2017, pp. 51–68. ACM (2017)
16. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on Bitcoin’s peer-to-peer network. In: USENIX 2015, pp. 129–144 (2015)
17. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: SIGSAC 2016, pp. 830–842. ACM (2016)
18. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
19. Kiayias, A., Zhou, H.-S., Zikas, V.: Fair and robust multi-party computation using a global transaction ledger. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 705–734. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_25
20. King, S., Nadal, S.: PPCoin: peer-to-peer crypto-currency with proof-of-stake (2012). <https://peercoin.net/whitepaper>

21. Kopp, H., Bösch, C., Kargl, F.: KopperCoin – a distributed file storage with financial incentives. In: Bao, F., Chen, L., Deng, R.H., Wang, G. (eds.) ISPEC 2016. LNCS, vol. 10060, pp. 79–93. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49151-6_6
22. Kopp, H., Kargl, F., Bösch, C., Peter, A.: uMine: a blockchain based on human miners. Cryptology ePrint Archive, report 2018/722 (2018). <https://eprint.iacr.org/2018/722>
23. Kopp, H., Mödinger, D., Hauck, F.J., Kargl, F., Bösch, C.: Design of a privacy-preserving decentralized file storage with financial incentives. In: EuroS&P Workshops, pp. 14–22. IEEE (2017)
24. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: S&P 2016, pp. 839–858. IEEE (2016)
25. Miller, A., Juels, A., Shi, E., Parno, B., Katz, J.: Permacoin: repurposing Bitcoin work for data preservation. In: S&P 2014, pp. 475–490. IEEE (2014)
26. Miller, A., Kosba, A., Katz, J., Shi, E.: Nonoutsourcable scratch-off puzzles to discourage bitcoin mining coalitions. In: SIGSAC 2015, pp. 680–691. ACM (2015)
27. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009). <https://bitcoin.org/bitcoin.pdf>
28. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_40
29. Percival, C.: Stronger key derivation via sequential memory-hard functions. Self-published (2009). <http://www.tarsnap.com/scrypt/scrypt.pdf>
30. Provos, N., Mazieres, D.: A future-adaptable password scheme. In: USENIX 1999, pp. 81–91 (1999)
31. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutor. **18**(3), 2084–2123 (2016)
32. de Vries, A.: Bitcoin’s growing energy problem. Joule **2**(5), 801–805 (2018)
33. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014). <http://gavwood.com/paper.pdf>

Full Paper Session II: Malware, Botnet and Network Security



LagProber: Detecting DGA-Based Malware by Using Query Time Lag of Non-existent Domains

Xi Luo^{1,2}(✉), Liming Wang¹, Zhen Xu¹, and Wei An¹

¹ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{luoxi,wangliming,xuzhen,anwei}@iie.ac.cn

² University of Chinese Academy of Sciences, Beijing, China

Abstract. Domain Generation Algorithm (DGA) has been outfitted by various malware families to extend the resistance to the blacklist-based techniques. A lot of previous approaches have been developed to detect the DGA-based malware based on the lexical property of the random generated domains. Unfortunately, attackers can adjust their DGAs to produce domains by simulating the character distribution of popular domains or words and thus can evade the detection of these approaches.

In this work, we develop an approach from a novel perspective, i.e., the query time lags of non-existent domains (NXDomain), to mitigate DGA-based malware without the lexical property. The key insight is that, unlike the benign hosts, the hosts infected by the same malware families will query a lot of NXDomains in inherent time lags. We design a system, LagProber, to detect infected hosts by analyzing the distribution of time lags. Our experiment with a week of real world DNS traffic reveals that LagProber is able to detect the infected hosts with low false positive rate.

Keywords: Domain Generation Algorithm · DGA-based malware
Time lag · NXDomain queries

1 Introduction

Domain Generation Algorithm (DGA) has been outfitted by various malware families to extend the resistance to the blacklist-based techniques. Cybercriminals utilize DGAs to produce random domains and select a small subset for actual command and control (C&C) use. The randomly generated and short lived C&C domains render detection approaches that rely on static domain lists ineffective.

As the domains generated by the DGA-based malware consist of random and unreadable character concatenations, a lot of researchers have developed detection techniques, e.g., [8, 22, 25, 28, 30–32, 34], based on the lexical property.

However, these random domains can also be generated by simulating the readable strings. For example, in [11], the authors present a method to generate domains based on the character distribution of the words in English dictionary and their experiment proves that their method can significantly degrade the lexical property-based detection techniques such as [8, 34]. In this case, more intrinsic features should be extracted without the lexical property to detect DGA-based malware.

In this work, we develop an approach from a novel perspective, i.e., the query time lags of NXDomains, to mitigate DGA-based malware without the lexical property. The key insight is that, unlike the benign hosts, the hosts infected by the same malware families will query a lot of non-existent domains (NXDomains) in inherent time lags to find the rendezvous points for C&C connection. Motivated by this peculiarity, we design a system, LagProber, to detect the infected hosts by analyzing the query time lags of NXDomains. LagProber extracts features from the distribution of query time lags, and implements a clustering method to identify the infected hosts. In contrast with the other DGA-based malware detection approaches using the time-based features, e.g., periodicity of C&C connections and change points of NXDomain traffic, our features can be extracted in a shorter period and do not rely on specific time patterns. Moreover, the features extracted from time lags can be used compatibly with the periodicity-, change point- or lexical-based detection.

In summary, our research makes following contributions.

- (1) We develop an approach from a novel perspective, i.e., the query time lags of NXDomains, to detect DGA-based malware. Our approach is able to identify the infected hosts without the lexical property which is easy to be obscured by attackers.
- (2) We design a system, LagProber, to identify the DGA-based malware by analyzing the query time lags of NXDomains. LagProber implements an unsupervised algorithm and thus can run without prior knowledge; and the key advantage is that it can detect the DGA-based malware without the lexical property.
- (3) We evaluate LagProber using a week of real world DNS traffic collected from a large ISP network to show the efficacy. The result illustrates that LagProber can accurately detect the DGA-based malware and has scalable performance.

Organization. The rest of the paper is organized as follows. In Sect. 2, we illustrate the background and motivation. The system design is introduced in Sect. 3 and evaluated in Sect. 4. We discuss the limitations in Sect. 5, and summarize the previous works in Sect. 6. At last, in Sect. 7, we conclude this work.

2 Background and Motivation

In this section, we first introduce the background knowledge of the domain generation algorithm and then illustrate the motivation of our approach.

2.1 Domain Generation Algorithm

DGA is an advanced DNS technique used by sophisticated malware families. The attackers periodically generate thousands of domains, which can be used as rendezvous points for C&C communication. Among these domains, only a small number of them are used as actual C&C domains at a certain moment. The real C&C domains only live for short periods before they are replaced with other domains; thus, if the C&C domains are retained by the responders, the C&C communication will persist. The large number of potential C&C domains complicates taking down the C&C servers.

The generated domains are computed based on a given seed, which can consist of numeric constants, the current date/time, or even Twitter trends. In most cases, the character distribution of random generated domains is distinct with that of the benign domains. One can detect DGA-based malware by identifying the random domains. However, as aforementioned in Sect. 1, the domains can be generated by simulating the English dictionary words [11], which can significantly degrade the detection approaches based on the lexical property.

Table 1. The time lags of different malware families.

#	Family	Time lag
1	PadCrypt	0 s between domains
2	Kraken	0 s between domains
3	Proslikefan	0 s between domains
4	Corebot	0 s between domains
5	Pykspa	0 s between domains
6	DirCrypt	0 s between domains
7	Necurs	0 s between domains
8	Symmi	0 s between domains
9	Ramnit	0 s between domains
10	Ranbyus	500 ms between domains
11	newGOZ	1 s between domains
12	Sisron	3 s interval between domains
13	Shiotob	5 s between domains
14	Qadars v3	20 s after 200 domains
15	Banjori	as long as DNS query for www.google.de succeeds

2.2 Query Time Lag of NXDomains

In this work, we develop our approach from a novel perspective, i.e., the query time lags of NXDomains, to detect DGA-based malware without the lexical property. This is motivated by the fact that the infected hosts don't know the

exact C&C domain and have to query a large amount of generated domains until an available response. Therefore, a sequence of query time lags of the NXDomains can be collected to extract the features for detection.

In Table 1, according to the analysis report in Johannes’ blog [5], we summarize the time lags of 15 DGA-based malware families. Nine of them query their domains without waiting, five of them implement invariable intervals and one of them alternatively queries the generated domains and www.google.de. No matter what kind of the time lag the attackers implemented, the hosts infected by the same DGA-based malware will have the same distribution. Particularly, when an infected host query a lot of generated domains, the distribution of time lags is consistent. Motivated by this finding, we design a system, LagProber, to detect DGA-based malware by clustering the hosts with similar distribution of the query time lags of NXDomains. The detail of LagProber will be introduced in the following.

3 System Design

In this section, we present our system, LagProber, to show how it works based solely on the query time lags of NXDomains. It is noticeable that LagProber only analyzes the second level domains (SLDs) and domains served by dynamic DNS such as *ddns.net*. A SLD is a domain directly below a top-level domain (TLD) like *.com* and *.net*, or a ccSLD like *.ac.uk* and *.co.uk*. The dynamic DNS domains analyzed in our work are the same as the ones listed in [18]. The reason for analyzing these two types of domains is that the DGA-based malware families usually map their servers to them. Thus, unless otherwise noted, when we talk about domains or NXDomains we refer to the two types of domains.

3.1 Architecture

The architecture of LagProber is shown in Fig. 1. LagProber takes DNS traffic as input and the *Collector* records the NXDomain queries for analysis. For each host, the *Feature Generator* extracts the features from the distribution of query time lags of NXDomains to generate the vectors. The *Group Analyzer* performs a clustering process on the vectors to gather similar ones and outputs a set of candidate clusters. The *Significance Analyzer* implements a significance detection process to identify if any infected hosts there. In the following, we will introduce the details of the four components, i.e., *Collector*, *Feature Generator*, *Group Analyzer*, *Significance Analyzer*.

We maintain a finite-state machine to manage the whole workflow of our system. The state machine is shown in Fig. 2. LagProber starts from the **Idle** state. If there is no host querying more than 10 NXDomains ($n_{nx} < 10$), LagProber keeps waiting. When a host queries more than 10 NXDomain ($n_{nx} > 10$), LagProber comes to the **Preparing** state. The *Feature Generator* extracts the vector from the distribution of the time lags. If the waiting time t exceeds 1 h (the time window), LagProber comes to the **Detecting** state, or else it comes

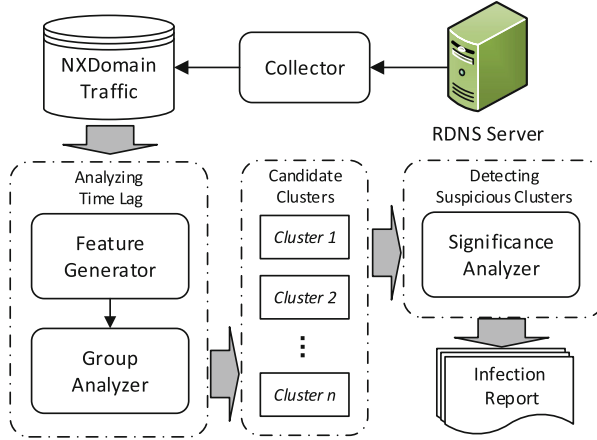


Fig. 1. System Architecture.

back to the **Waiting** State. In the **Detecting** state, the *Group Analyzer* clusters the vectors and *Significance Analyzer* reports the infected hosts. Then, LagProber resets the system (e.g., $t = 0$ and) and goes back to the **Waiting** state.

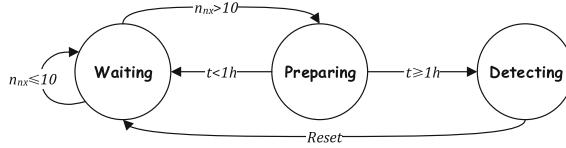


Fig. 2. System Workflow.

3.2 Collector

The *Collector* collects the NXDomain traffic produced in the monitored network and filters out the non-malicious NXDomains to improve the system efficacy. In this work, we consider an NXDomain as non-malicious when it satisfies one of the following conditions.

- Invalid Top Level Domain (TLD): A domain is considered as non-malicious if its TLD is not in the list of registered TLDs presented by IANA [4].
- Irregular characters: A domain contains the characters that should not be included in regular domain (consisting of only letters, numbers and dashes/hyphens). This domain is probably caused by the typing error or mis-configuration.
- Popular domains: We consider the top 100,000 domains in Alexa [1] and websites of world’s biggest 500 companies from Forbes [3] as popular legitimate

domains. These NXDomains are mostly utilized by benign services to transfer disposable signals [10].

3.3 Feature Generator

When host h_i queries more than m NXDomains d_1, d_2, \dots, d_m (in this work we set $m = 10$ to achieve a fast detection) with timestamps t_1, t_2, \dots, t_m , LagProber extracts the time lags of the queries, i.e., $S_t = \{l_k : t_{k+1} - t_k\}, k \in [1, m - 1]$. We only focus on the distinct domains in a single day due to that repeatedly queried domains have no help to find the rendezvous point for C&C connection.

Although most DGA-based malware families in Table 1 prefer to query the domains with constant time lags, some sophisticated ones can still implement the time lags based on some probability distribution, e.g., Gaussian distribution, to obscure the similarity. Anyway, they have the similar statistic values, e.g., mean and standard deviation, of S_t . Therefore, six statistic values, i.e., the mean, variance, median, maximum, minimum and mode (the most frequent value), of S_t are extracted by this component to construct vector v_i .

3.4 Group Analyzer

The *Group Analyzer* performs a clustering process to gather the similar vectors. Since the number of DGA-based malware families in the monitored network is unsure, LagProber implements a hierarchical merging algorithm, which does not require the number of clusters as input. This algorithm is a clustering approach

Algorithm 1. Clustering Process

Require: $S_v = \{v_i\}, i = 0, 1, \dots, p$ containing vectors generated by the Feature Generator Component;

Ensure: $S_c = \{c_k\}, k = 0, 1, \dots, q$ containing the outputted clusters.

```

1:  $C \leftarrow S_v$ 
2: while  $|C| > 0$  do
3:    $c_i, c_j, d_{ij} \leftarrow \text{getClosestPairOfClusters}(S_v)$ 
4:    $a_i = \text{mean}(c_i) + 2 * \text{std}(c_i)$ 
5:    $a_j = \text{mean}(c_j) + 2 * \text{std}(c_j)$ 
6:   if  $d_{ij} < \max\{\sqrt{|v|}, \min\{a_i, a_j\}\}$  then
7:      $C \cup \text{merge}(c_i, c_j)$ 
8:   end if
9:   if  $d_{ij} > a_i$  then
10:     $S_c \cup c_i$ 
11:     $\text{remove}(C, c_i)$ 
12:   end if
13:   if  $d_{ij} > a_j$  then
14:     $S_c \cup c_j$ 
15:     $\text{remove}(C, c_j)$ 
16:   end if
17: end while

```

that merges the most similar pairs of clusters as one moves up the hierarchical until the terminated conditions are satisfied. In our work, the Euclidean distance is selected as the similarity measures.

In Algorithm 1, we present the workflow of the clustering process. First, considering each vector generated by the *Feature Generator* component as a single cluster, the function *getClosestPairOfClusters* extracts the closest pair of clusters (c_i and c_j) and returns their distance d_{ij} . Second, LagProber determines whether d_{ij} is too large for c_i and c_j by comparing d_{ij} with the value of $a_i = \text{mean}(c_i) + 2 * \text{std}(c_i)$, where $\text{mean}(c_i)$ and $\text{std}(c_i)$ represent the mean and standard deviation of the internal distances of the vectors in c_i , respectively. As shown in Table 1, the time lags of most DGA-based malware families are less than 1 second. The distances of the vectors generated by them are very likely less than $\sqrt{|v|} = \sqrt{\sum_{i=1}^{|v|} (1^2 - 0^2)}$, where $|v|$ is the size of vector. Therefore, if $d_{ij} < \max\{\sqrt{|v|}, \min\{a_i, a_j\}\}$, LagProber merges c_i and c_j . Third, if $d_{ij} > a_i$ or $d_{ij} > a_j$, which indicates that there is no cluster similar to c_i , LagProber outputs cluster c_i or c_j . At last, if no cluster can be merged, the clustering process is terminated.

3.5 Significance Analyzer

When the infected hosts try to connect the C&C servers, they will query much more NXDomains than the benign ones and the feature vectors will be more similar. As a result, one or more clusters will contain much more items than the other ones. Hence, LagProber implements a outliers testing algorithm, i.e., one-side Grubbs' test [12], also known as the maximum normalized residual test or extreme studentized deviate test, to search for the significantly larger clusters in the result of *Group Analyzer*.

According to Grubbs' test, we define two hypothesis H_1 and H_0 denoting if there is a significantly large cluster or not, respectively. Assuming q clusters c_1, c_2, \dots, c_q have been outputted by the *Group Analyzer*, the statistic test is:

$$G = \frac{\max_{i=1, \dots, q} \{|c_i| - |c|\}}{s}, \quad (1)$$

where $|c_i|$ is the size of cluster c_i , $|c|$ is the mean size and s is the standard deviation of the sizes. The hypothesis H_0 is rejected at a significance level α (set as 0.001 in our work) if

$$G > \frac{q-1}{\sqrt{q}} \sqrt{\frac{t_{\frac{\alpha}{2q}, q-2}^2}{q-2 + t_{\frac{\alpha}{2q}, q-2}^2}}, \quad (2)$$

where $t_{\frac{\alpha}{2q}, q-2}$ denotes the upper critical value of the t-distribution [33] with $q-2$ degrees of freedom and a significance level of $\frac{\alpha}{2q}$. In Fig. 3, we present the workflow of the significance test in this component. As the Grubbs' test only examines the maximum, when the hypothesis H_0 has been rejected, LagProber

identifies the corresponding cluster as malicious and removes it for the next round of test. If hypothesis H_0 is accepted, the test process is terminated.

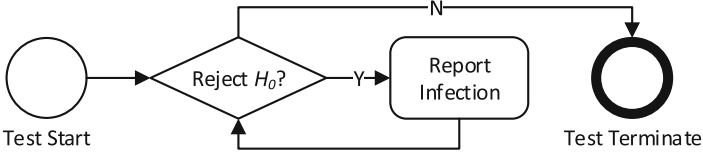


Fig. 3. Workflow of the significance test.

Besides the significantly large clusters, we also consider the clusters containing significantly similar vectors as malicious. Since most of the DGA-based malware families implements 0 s time lag to query the domains, we determine a cluster as malicious when the max distance of the internal vectors is less than $\sqrt{|v|}$. At last, LagProber reports all the hosts generating the vectors in the significantly large or similar clusters as malicious.

4 Evaluation

In this section, LagProber is evaluated on a week of real world DNS traffic. We first introduce the dataset used in our evaluation and analyze the detection result. Then, we present the system performance of LagProber.

4.1 Dataset

Our dataset is collected from an ISP network which offers Internet services to the Chinese education, research, scientific and technical communities, relevant government departments, and hi-tech enterprises. We obtained DNS traffic collected on the edge of this network from May 1st, 2018 to May 7th, 2018. The summary of our dataset is presented in Table 2. Q_{total} and Q_{nx} represent the number of total and NXDomain queries, respectively. Since LagProber only processes the NXDomain traffic (with about 5% of the volume of the total traffic), the exact volume of dataset is significantly reduced. It is noticeable that we rule out the traffic of DNS servers (hosts opening the 53 port for service) in this dataset. This is because LagProber is designed to work under the recursive servers and the behaviors of DNS server are not feasible to represent the human activities.

Labeling. To label the infected hosts in our dataset, we first extracts the response IP addresses mapping to more than 50 distinct SLDs. This is because the C&C IP addresses used by DGA-based malware families are very likely to map with multiple domains. Then, we manually examine the domains with the

Table 2. Summary of our dataset.

Date	Q_{total}	Q_{nx}
2018-05-01	154,548,745	8,352,698
2018-05-02	250,951,430	10,874,362
2018-05-03	226,918,325	10,823,140
2018-05-04	138,368,609	6,085,376
2018-05-05	159,469,521	8,260,969
2018-05-06	154,055,202	8,450,981
2018-05-07	134,670,589	6,940,916
Total	1,218,982,421	59,788,442

help of *Virustotal* [7] and *ThreatCrowd* [6] to ensure the malicious. As a results, we identify 17 infected hosts.

We classify the 17 infected hosts into 3 categories, i.e., *dga-1*, *dga-2* and *dga-3*, based on the character distribution. The domain samples are presented in Fig. 4. The *dga-1* domains are constructed by numerics, the *dga-2* domains are similar with the traditional DGA-based malware families that random select characters, and the *dga-3* domains are very likely generated by the HASH-based DGA [23]. The *dga-1* and *dga-3* infected hosts query domains with no waiting while the *dga-2* infected hosts consecutively query a few domains per 7s.

Although the number of hosts is small, the large amount of DNS traffic gives us a good chance to measure the false positive rate, which is very important for real world usage. In the following, these infected hosts are used as ground truth to analyze the result.

<i>dga-1</i>	<i>dga-2</i>
2682389127-77.com	uodqbomv.net
2682408961-77.com	uzyfzcier.biz
2682416897-77.com	bcmxxmaso.com
2682398078-77.com	riwaaofjze.biz
2682417924-77.com	xpmbjqpv.net
2682417931-77.com	dxplr.cc
<i>dga-3</i>	
sbxrfxowwxzabunktnruoi0jdd.com	
gjneogaqd4enpt2z2usircmyrd.com	
4lf1vuowkzei00ffdoqavnqjzg.com	
3qhdch5s5jvs2sgqqnskhrdmbh.com	
hf5ap34xohz1wospvkqqlnpjqh.com	
osoedr5uhdatiewvzgredivbtzg.com	

Fig. 4. Domain samples queried by the infected hosts.

4.2 Result Analysis

In Table 3, we present the detection result of LagProber. TPr and FPr denote the true and false positive rates, respectively. D_h denotes the ratio between the number of detected infected hosts and the total number N_{inf} of active infected hosts. As one host can generate multiple vectors, the metrics, i.e., TPr and FPr, are calculated based on whether LagProber correctly or falsely classifies a host in a certain time window (1 h). For example, if the vectors generated by an infected host are clustered in a significantly large cluster, we identify a true positive. The true negative, false positive and false negative can be identified similarly. Besides, we also manually examine the classified vectors to ensure the malicious. This is because that the infected hosts can also generate benign queries, and if they fail to connect the C&C servers the positive vectors can not be labeled solely based on the ground truth.

Table 3. Summary of the detection result.

	2018-05-01	2018-05-02	2018-05-03	2018-05-04	2018-05-05	2018-05-06	2018-05-07
TPr	90%	82%	87.5%	90%	100%	100%	88.5%
FPr	3.4%	6.8%	2.9%	0.9%	0.9%	0.6%	0.8%
D_h	100%	100%	100%	100%	100%	100%	100%
N_{inf}	8	9	11	8	5	12	11

As shown in Table 3, the TPr of LagProber is about 90% in average. The false negatives mainly emerges in the case that the number of vectors generated by the infected hosts is too small to construct a significantly large cluster. Anyway, LagProber can detect all the infected hosts (with $D_h = 100\%$).

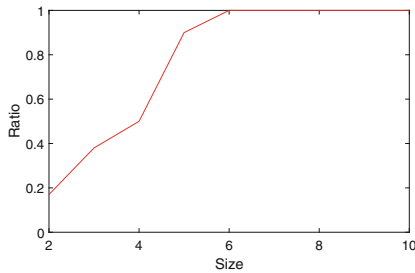


Fig. 5. The relationship between the size and maliciousness of clusters.

The FPr of LagProber is about 2% in average. The false positives are mainly caused by the small clusters being identified as significantly large by Grubbs' test. These small clusters are accidentally by the Network Address Translation

(NAT) routers in the monitored network when they query a sequence of distinct NXDomains in a short period. In Fig. 5, we present the relationship between the size and maliciousness of clusters. The abscissa axis represents the size of clusters and the vertical axis denotes the ratio of malicious ones. When the size of clusters is 5, the ratio of malicious ones is 90% and when the size exceeds 6, all of the clusters are malicious. Hence, to reduce the false positives, one can simply set the threshold as 6 to identify the significantly large clusters.

In conclusion, LagProber can detect all the infected hosts with less false positive rate. This illustrates that the features extracted from the time lags can be utilized to effectively detect the DGA-based malware. Besides, since most hosts in our dataset are the NAT routers, LagProber can achieve a better performance when processing the traffic generated from local or enterprise networks which contain more personal computers.

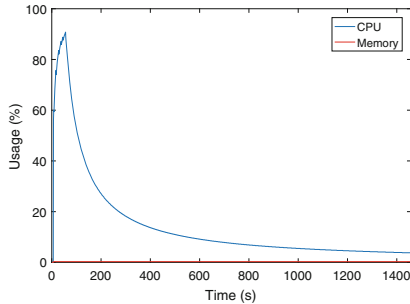


Fig. 6. System performance.

4.3 System Performance

In our experiment, we run LagProber on Ubuntu 12.04 with CPU Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00 GHz and 32 GB memory. As shown in Fig. 6, our system spends about 25 min to analyze the traffics (about 24 GB in bro DNS log format [2]) collected in May 1st, 2018. Since we rule out the repeatedly queried domains in a single day, usage percentage of CPU raises up to 90% at the beginning and then gradually decreases to about 9%. LagProber only stores the vectors generated in an hour so that the memory usage is less and stable (about 0.3%). In summary, when running on the traffic collected from a large scale network, the CPU usage rate is 22.5% in average and the memory usage is about 1 GB. All the results prove that LagProber has scalable performance.

5 Discussion

Limitation. The limitations of this work are as follows. First, the dataset only contains few kinds of DGA-based malware families due to that it is not easy

to find the real world traffic of DGA-based malware families with distinct time lags. However, as we can see in Table 1, the time lag (0s), of most malware families is the same as *dga-1* (Sect. 4.1), which indicates LagProber can detect most of them. Second, the attacker can implement long time lags, which is more similar with benign hosts, to evade our detection. In this case, a longer time window can be used for accurate detection. Moreover, they have to spend much more time to connect the C&C servers, and the utility of the infected hosts to attacker is reduced or limited because the attacker can no longer command his bots promptly and reliably.

Comparison. The most generally time-based features to detect DGA-based malware are the periodicity of C&C connection and the change point of the NXDomain traffic. First, for the periodicity-based detection approaches, multiple C&C connections are needed to extract the features while time lags of the NXDomains can be extracted in a single C&C connection. If the infected hosts do not periodically connect the C&C servers, the periodicity-based approaches are ineffective. Second, for change point detection approaches, it is difficult for them to accurately detect DGA-based malware because of that the benign hosts are also very likely to generate the suddenly increased traffics. Hence, the previous works [9, 34] should utilize other evidences, like lexical or whois features for accurate detection, while LagProber can achieve a low false positive rate merely based on the features extracted from time lags. Last but not least, the features extracted from time lags can be used compatibly with the periodicity, change point or lexical-based detection. For example, when the periodicity or change point is detected, one can also analyze the time lags to improve the accuracy.

6 Related Work

A wealth of researches have been conducted on detecting DGA-based malware. They mainly utilize the lexical property of the generated domains. Antonakakis et al. [8] introduce a system, Pleiades, to detect DGA bots in large scale network by clustering the NXDomains with the similar character distribution generated by the same DGA-based malware families. Sharifnya et al. [26] present a reputation system to detect DGA botnets by periodically clustering DNS queries with similar characteristics. Schiavoni et al. [25] present a system, Phoenix, to track and fingerprint DGA botnets by clustering domains with similar character distributions. Wang et al. [29] present a system, DBod, to detect DGA botnet by searching for the similar set of NXDomains queried by the bots. Thomas et al. [27] present a method to detect DGA domains by clustering the NXDomains with similar character distributions queried by distinct recursive DNS servers. Yadav et al. [32] introduce three metrics, i.e., K-L divergence, Jaccard Index and Edit distance, to detect DGA domains sharing the same postfix or C&C IP addresses. Yadav et al. improve their work [32] using NXDomains and temporal correlation in [31]. Zhang et al. [34] present a system, BotDigger, to detect a single bot by searching for the suddenly increasing and decreasing random generated SLD

queries before and after C&C connection. Mowbrey et al. [22] present an approach to detect a DGA bot by examining the anomaly domain length distribution in a time slot. Luo et al. [20] and Truong et al. [28] present a set of lexical features to separate a DGA domain from a popular one. Wodbridge et al. [30] and Lison et al. [19] present deep learning methods to predict a DGA domain. While existing solutions demonstrate their effectiveness in detecting malicious servers or server infrastructures, they still can be significantly degraded by generating readable domains [11].

Krishnan et al. [15] implement Threshold Random Walk algorithm [13] to identify an infected host in a fast way without analyzing the lexical property. Their approach relies on the assumption that an infected host is more likely to query a previously unseen NXDomain than a benign host. They have to train the parameters based on at least 24 hours traffic every time when they deploy their approach. Besides, since the malicious samples are not easy to achieve, the probability that an infected host queries a previously unseen NXDomain is pretty difficult to be estimate. Conversely, LagProber detects infected hosts needs no malicious samples.

Except the above DGA-based malware techniques, some botnet or malware domain detection approaches without the lexical can also be utilized to detect DGA domains. Manadhata et al. [21] utilize belief propagation algorithm on graphical models to detect malicious domains. Lee et al. [17] develop a malicious domain detection technique using the sequential correlation property of malicious domains. Khalil et al. [14] and Rahbarinia et al. [24] present methods to infer the suspicious domains which have strong relationship with the known malicious ones. Bilge et al. [9] introduce EXPOSURE to detect malicious domains. They extract 15 features and divide the features into Time-based, DNS answer-based, TTL value-based and Domain name-based. Then, a detection model is trained by using decision tree algorithm. Kwon et al. [16] present PsyBoG to detect botnet by analyzing similar periodicity of the bots. The graph-based approaches [17,21], which need plenty of samples and time to build and process a graph structure, are resource consuming. The time-based approaches [9,16] rely on longer term time patterns, e.g., active time in a month [9] or periodicity of C&C connection. In contrast, LagProber detects DGA-based malware families in a short term mode, i.e., the time lag between two NXDomain queries, and does not rely on the periodicity.

The aforementioned systems are mostly limited by the lexical property, and thus work only on random generated domains. LagProber is a novel general detection system that does not have such limitations and can greatly complement existing detection approaches.

7 Conclusion

In this work, we develop a system, LagProber, to detect DGA-based malware that is independent of the lexical property of the generated domains. Our system exploits a new essential property of DGA-based malware, i.e., hosts infected

by the same malware family will exhibit similar patterns of the query time lags of NXDomains. In our experimental evaluation real-world network traces, LagProber shows excellent detection accuracy with a very low false positive rate on normal traffic.

Acknowledgements. The authors are very grateful to the anonymous reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0801900).

References

1. Alexa top 1m. <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>
2. The Bro network security monitor. <https://www.bro.org>
3. Forbes biggest companies. <http://www.forbes.com>
4. IANA top level domains. <http://www.iana.org/domains/root/db/>
5. Johannes Bader's blog. <https://johannesbader.ch/>
6. ThreatCrowd. <http://threatcrowd.org>
7. VirusTotal. <http://www.virustotal.com/>
8. Antonakakis, M., et al.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX Security Symposium, pp. 491–506 (2012)
9. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: EXPOSURE: finding malicious domains using passive DNS analysis. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6–9 February 2011 (2011). <http://www.isoc.org/isoc/conferences/ndss/11/pdf/4-2.pdf>
10. Chen, Y., Antonakakis, M., Perdisci, R., Nadji, Y., Dagon, D., Lee, W.: DNS noise: measuring the pervasiveness of disposable domains in modern DNS traffic. In: 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, 23–26 June 2014, pp. 598–609 (2014). <https://doi.org/10.1109/DSN.2014.61>
11. Fu, Y.: Stealthy domain generation algorithms. *IEEE Trans. Inf. Forensics Secur.* **12**(6), 1430–1443 (2017). <https://doi.org/10.1109/TIFS.2017.2668361>
12. Grubbs, F.E.: Sample criteria for testing outlying observations. *Ann. Math. Stat.* **21**, 27–58 (1950)
13. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: 2004 IEEE Symposium on Security and Privacy (S&P 2004), 9–12 May 2004, Berkeley, CA, USA, pp. 211–225 (2004). <https://doi.org/10.1109/SECPRI.2004.1301325>
14. Khalil, I., Yu, T., Guan, B.: Discovering malicious domains through passive DNS data graph analysis. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, 30 May–3 June 2016, pp. 663–674 (2016). <https://doi.org/10.1145/2897845.2897877>
15. Krishnan, S., Taylor, T., Monroe, F., McHugh, J.: Crossing the threshold: detecting network malfeasance via sequential hypothesis testing. In: 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Budapest, Hungary, 24–27 June 2013, pp. 1–12 (2013). <https://doi.org/10.1109/DSN.2013.6575364>
16. Kwon, J., Lee, J., Lee, H., Perrig, A.: PsyBoG: a scalable botnet detection method for large-scale DNS traffic. *Comput. Netw.* **97**, 48–73 (2016). <https://doi.org/10.1016/j.comnet.2015.12.008>

17. Lee, J., Lee, H.: GMAD: graph-based malware activity detection by DNS traffic analysis. *Comput. Commun.* **49**, 33–47 (2014). <https://doi.org/10.1016/j.comcom.2014.04.013>
18. Lever, C., Kotzias, P., Balzarotti, D., Caballero, J., Antonakakis, M.: A lustrum of malware network communication: evolution and insights. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017, pp. 788–804 (2017). <https://doi.org/10.1109/SP.2017.59>
19. Lison, P., Mavroeidis, V.: Automatic detection of malware-generated domains with recurrent neural models. *CoRR abs/1709.07102* (2017). <http://arxiv.org/abs/1709.07102>
20. Luo, X., Wang, L., Xu, Z., Yang, J., Sun, M., Wang, J.: DGAsensor: fast detection for DGA-based malwares. In: Proceedings of the 5th International Conference on Communications and Broadband Networking, pp. 47–53. ACM (2017)
21. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, AISec 2014, Scottsdale, AZ, USA, 7 November 2014, pp. 59–60 (2014). <https://doi.org/10.1145/2666652.2666659>
22. Mowbray, M., Hagen, J.: Finding domain-generation algorithms by looking at length distribution. In: 25th IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Naples, Italy, 3–6 November 2014, pp. 395–400 (2014). <https://doi.org/10.1109/ISSREW.2014.20>
23. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A comprehensive measurement study of domain generating malware. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 263–278 (2016). <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann>
24. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Efficient and accurate behavior-based tracking of malware-control domains in large ISP networks. *ACM Trans. Priv. Secur.* **19**(2), 4:1–4:31 (2016). <https://doi.org/10.1145/2960409>
25. Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Phoenix: DGA-based Botnet tracking and intelligence. In: Proceedings of Detection of Intrusions and Malware, and Vulnerability Assessment - 11th International Conference, DIMVA 2014, Egham, UK, 10–11 July 2014, pp. 192–211 (2014). https://doi.org/10.1007/978-3-319-08509-8_11
26. Sharifnya, R., Abadi, M.: DFBotKiller: domain-flux botnet detection based on the history of group activities and failures in DNS traffic. *Digit. Investig.* **12**, 15–26 (2015)
27. Thomas, M., Mohaisen, A.: Kindred domains: detecting and clustering Botnet domains using DNS traffic. In: 23rd International World Wide Web Conference, WWW 2014, Seoul, Republic of Korea, 7–11 April 2014, Companion Volume, pp. 707–712 (2014). <https://doi.org/10.1145/2567948.2579359>
28. Truong, D., Cheng, G.: Detecting domain-flux Botnet based on DNS traffic features in managed network. *Secur. Commun. Netw.* **9**(14), 2338–2347 (2016). <https://doi.org/10.1002/sec.1495>
29. Wang, T.S., Lin, H., Cheng, W., Chen, C.: DBod: clustering and detecting DGA-based Botnets using DNS traffic analysis. *Comput. Secur.* **64**, 1–15 (2017). <https://doi.org/10.1016/j.cose.2016.10.001>
30. Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. *CoRR abs/1611.00791* (2016). <http://arxiv.org/abs/1611.00791>

31. Yadav, S., Reddy, A.L.N.: Winning with DNS failures: strategies for faster Botnet detection. In: Security and Privacy in Communication Networks - 7th International ICST Conference, SecureComm 2011, London, UK, 7–9 September 2011, Revised Selected Papers, pp. 446–459 (2011). https://doi.org/10.1007/978-3-642-31909-9_26
32. Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia, 1–3 November 2010, pp. 48–61 (2010). <https://doi.org/10.1145/1879141.1879148>
33. Yamane, T.: Statistics: an introductory analysis (1973)
34. Zhang, H., Gharaibeh, M., Thanasoulas, S., Papadopoulos, C.: BotDigger: detecting DGA bots in a single network. In: Traffic Monitoring and Analysis- 8th International Workshop, TMA 2016, Louvain la Neuve, Belgium, 7–8 April 2016 (2016). <http://dl.ifip.org/db/conf/tma/tma2016/tma2016-final56.pdf>



Study on Advanced Botnet Based on Publicly Available Resources

Jie Yin^{1,2}, Heyang Lv^{3(✉)}, Fangjiao Zhang^{1,2}, Zhihong Tian^{4(✉)},
and Xiang Cui^{1,4}

¹ Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China

² School of Cyber Security,
University of Chinese Academy of Sciences, Beijing, China

³ Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
heyanglv@126.com

⁴ Cyberspace Institute of Advanced Technology,
Guangzhou University, Guangzhou, China
tianzhihong@gzhu.edu.cn

Abstract. In recent years, botnets continue to be an ever-increasing threat on the Internet. To be well prepared for future attacks and ensure the cyberspace security, defenders take more attention on advanced botnet designs that could be used by botmasters. In this paper, we design an advanced botnet based on publicly available resources, and implement its prototype system, which is named as PR-Bot. First of all, in terms of system design, PR-Bot is completely constructed based on the third-party publicly available resources and supports the bidirectional communication between the control end and the controlled end. At the same time, the system's command and control (C&C) channel consists of three sub-channels: command control channel (CC channel), command addressing (CA channel) and result feedback (RF channel), making it extremely robust and concealed. Secondly, in terms of defense technology, this paper proposes the targeted defense strategies from the perspective of detection, measurement and tracking, so as to achieve the goal of combating against such botnets. In short, the ultimate purpose of this paper is not to design a highly harmful botnet, but to accurately predict the techniques that the botnet may adopt in the future and assess its new threats from the point of attack and defense.

Keywords: Publicly available resource · Command and control
Bidirectional communication · Defense technology

1 Introduction

1.1 Background

A botnet refers to a group of compromised computers that are remotely controlled by a botmaster via C&C channels [1]. Based on botnets, multiple types of Internet attacks

can be initiated, such as: DDoS (Distributed Denial of Service), Email Spam, Bitcoin or Monero Mining, etc. At present, the studies on botnets can be summarized into two aspects: attack technology and defense technology. The purpose of studying attack technology is predicting the attack trends and techniques of future botnets, so as to prevent the possible emerging botnet activities; and the purpose of studying on defense technology is improving the detection efficiency of botnets and discovering the botnets that are already in the cyberspace but not yet exposed in a timely manner, so as to reduce the actual harm caused by them.

In the early days, attackers usually controlled the bot based on the IRC [2, 3] or HTTP [4, 5] protocol. This centralized architecture is simple, efficient and highly interactive, but is vulnerable to single point of failure. Although a modified architecture based on the Domain-Flux [6] or Fast-Flux [7] protocol that appeared later can eliminate this problem certainly, it may be attacked by Sinkhole [8]. In order to make up for the deficiency of centralized botnets, botnets using P2P protocols as C&C channels have also evolved. In a P2P botnet, the botmaster can issue commands at any node, so it can hide the real address of the C&C server and effectively solve the single point of failure. However, P2P botnets are not perfect, which still have inherent weakness. For example, the structured P2P botnets, such as Storm [9, 10], are vulnerable to Index Pollution attack and Sybil attack, and its scale is easy to be measured by Crawler and Sybil nodes; the unstructured P2P botnets usually communicate by the way of random scanning or peer-list, the former has the inherent weaknesses of flow anomaly, and the latter is vulnerable to Peer-list Pollution attack.

In recent years, the new generation of botnets based on social network have been proposed, such as: Koobface [11], Stegobot [12], etc. Among the social botnet, each social account is a control node, which is equivalent to a C&C server in the traditional botnet, and is used to transfer the commands between the botmaster and individual bots. Although the social botnet can hide the malicious traffic within the normal legitimate traffic, the social platforms are generally only applicable for the botmaster issuing commands and cannot be used by bots to send back harvested information, especially file information. Moreover, for the botnet based solely on the social platform, its C&C channel is relatively simple, and it can be easily detected and destroyed by the defender. Therefore, in order to make up for the inadequacies of social botnets, this paper proposes an advanced botnet based on multiple publicly available resources.

1.2 Contribution

The goal of this paper is to study the development trends of future botnets, increase the defenders' understanding of the advanced botnet, and promote more effective cyber defense to deal with the possible similar cases.

The contributions of this paper mainly include three aspects:

- (1) Based on the idea of “severless botnet”, an advanced botnet based on publicly available resources is designed. The system adopts a three-channel scheme, and each sub-channel can be supported by multiple publicly available resources and extended in the form of plug-in.

- (2) We have tested five categories and 37 websites, and the application scenarios of each website when constructing C&C channel are discussed. Meanwhile, the operation flow between the botmaster and individual bots is analyzed and described in detail to verify the feasibility of the proposed model.
- (3) We have analyzed the attributes and weaknesses of the PR-Bot, and propose a practical targeted defense scheme that covers detection, measurement, and tracing.

2 The Design of PR-Bot

2.1 System Overview

Definition 1 (Publicly Available Resource Botnet). In this paper, we believe that all botnets that construct C&C channels based on the publicly available resources (including but not limited to: social network, URL shortener, image hosting, online clipboard or cloud disk, etc.) could be called the Publicly Available Resource Botnet (Fig. 1).

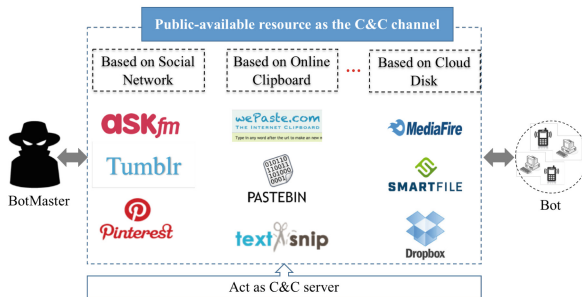


Fig. 1. Basic characteristics of publicly available resource Botnet

The basic characteristics of this type of botnets are: the botmaster no longer relies on the self-built C&C server to control the bots, but uses the open and free website on the Internet to act as the C&C server. All communication flows are transferred through the Internet publicly available resources.

2.2 System Design

In botnets, no matter how complex the control model of botnet is or how powerful the bot program is, the interaction between the control end and the controlled end usually involves only the transfer of text information (that is: string content) or file information (that is: binary content), as shown in Table 1.

Although there are many kinds of publicly available resources on the Internet, due to the limitations of the nature of the publicly available resources, not all publicly

Table 1. Interactive information between the control end and the controlled end

	Text information	File information
Control end	Command or others	Malicious program
Controlled end	Callhome or others	Stolen files

available resources are suitable for issuing both text information and file information. For example, the social platforms used by social botnets are generally only applicable to store the commands issued by the botmaster, but not applicable to store the harvested information sent back by individual bots, making it a one-way communication channel.

The PR-Bot, an advanced botnet designed in this paper, takes into account the limitations of a purely social platform as C&C channels, so it combines multiple publicly available resources and uses their respective advantages to construct C&C channels. The PR-Bot is suitable for transmitting both text information and file information, as well as supporting two-way communication between the botmaster and bots. All in all, the PR-Bot adopts the three-channel scheme, and the interaction information in each channel is distributed in different locations of cyberspace, as shown in Fig. 2.

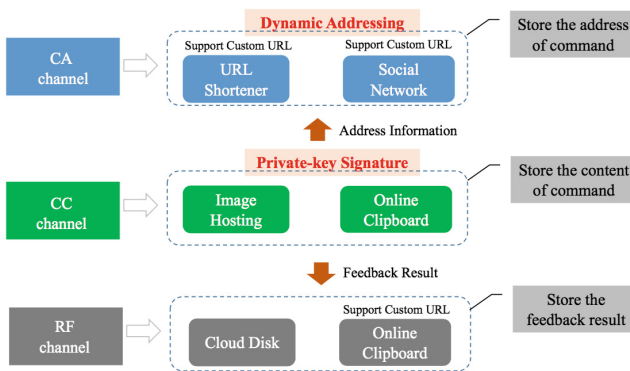


Fig. 2. The control model of PR-Bot

2.3 System Architecture

The architecture of PR-Bot is shown in Fig. 3, whose communication between the botmaster and bots includes the following six stages:

- (1) **Botmaster issues the content of command:** The botmaster issues the content of command to the publicly available resources, such as an online clipboard or image hosting website, and records the URL of the website (abbreviated as PR_Address_A) where the command is located. For the online clipboard, the command is issued in the form of string; for the image hosting, the command is first converted into a picture and then issued in the form of picture.

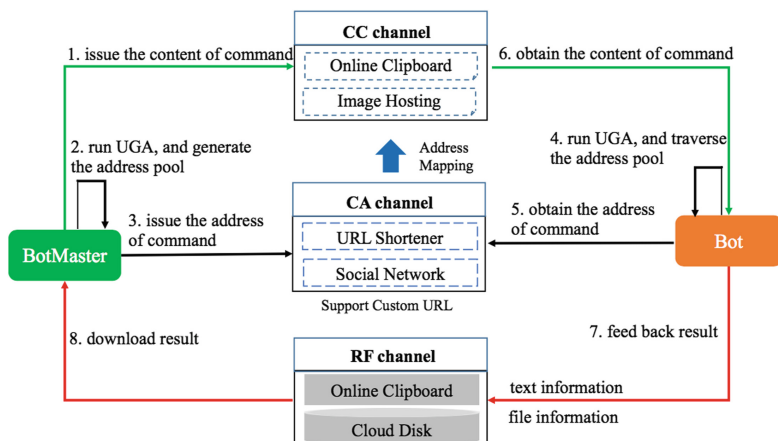


Fig. 3. The system architecture of PR-Bot

(2) Botmaster issues the address of command: First, the botmaster selects a URL shortener or social network that supports customized URL, designs and runs the Username Generation Algorithm (UGA) [13], followed by selecting several candidate addresses (abbreviated as PR_Address_B) from the URL address pool. And then, the botmaster issues the address PR_Address_A as content to the website corresponding to the address PR_Address_B.

(3) Bot obtains the address of command: After the bot infects the controlled end, to establish the communication with the control end, it will first run the UGA algorithm consistent with the botmaster, and then traverse the URL address pool one by one. When an address (take PR_Address_B as an example here) is found to be accessible, it is considered that address of command is stored at this place, and then the address PR_Address_A will be extracted.

(4) Bot obtains the content of command: After obtaining the content of command from PR_Address_A in Stage 3, the bot first verifies its validity and availability, and only the command that passes verification can be executed. Otherwise, the control logic of the bot program will jump to Stage 3 and run again. Among them, “validity” refers to whether the command is within the validity period specified by the botmaster; “availability” refers to whether the command is signed by the private key of the botmaster.

(5) Bot feeds back the result information: For the command that needs to feed back the result information, the botmaster is required to specify the receiving address as a parameter in the issued command. The bot will feed back the relevant information based on this parameter in the command and the customized protocol with the control end.

(6) Botmaster downloads the result information: After a certain period of time, for the command that will feed back the result information, the corresponding result acquisition module will be run. During the operation of the module, it will download the text information (such as callhome information) or file information

(such as stolen files) returned by the bot for further analysis, which is based on the address parameter in the issued command and the customized protocol with the controlled end.

2.4 The Standard for Selecting Publicly Available Resources

As shown in Table 2, this paper tests five major types of publicly available resources and analyzes the specific application scenarios of each when constructing C&C channel. Among them, PR-Bot stores the content of command based on online clipboard or image hosting website; stores the address of command based on a URL shortener or social network that can customized URL address; stores the stolen files based on public cloud disk website; stores the callhome information based on online clipboard that can customized URL address.

Table 2. Application scenarios of publicly available resources

Type	Is it suitable for issuing text?	Is it suitable for issuing file?	Is it suitable for issuing picture?	Application scenarios
Online clipboard	Y	N	N	Store the content of command Store the callhome information
Image hosting	N	N	Y	Store the content of command
URL shortener	Y	N	N	Store the address of command
Social network	Y	N	Y	Store the address of command
Cloud disk	N	Y	Y	Store the stolen files from bot

(1) Publicly available resource for storing the content of command

In the CC channel, when selecting an online clipboard, as shown in Table 3, PR-Bot only considers two factors: one is the size of the space for storing information, and the other is the length of time for storing information. And at this stage, it does not consider whether the website supports customized URL. As long as the storage space can reach 200 KB and the storage time can reach 1 month, it means that the requirements is met. In addition, for the image hosting website, as shown in Table 4, PR-Bot only selects the websites that have no registration required and do not compress the pictures. And the size of signal picture allowed to upload should meet the requirements of 1 MB. Moreover, because the storage time supported by the image hosting website is usually unlimited, so this factor was not taken into account here.

Table 3. Online clipboard and its selection standard

Name	Site	Customized URL	Storage space	Storage time
dpaste	http://dpaste.com/	N	Unknown	1 year
pasted	http://pasted.co/	N	Unknown	Unknown
pastebin	http://pastebin.com/	N	512 KB	Unlimited
wepaste	http://www.wepaste.com	Y	Unknown	Unlimited
cl1p	https://www.cl1p.net/	Y	Unknown	1 time
textsnip	http://www.textsnip.com	Y	70000 characters	Unknown
showtxt	http://showtxt.cn/	Y	Unknown	Unknown

Table 4. Image hosting and its selection standard

Name	Site	Registration	File size limit	Is it compressed?
baidu-pic	http://image.baidu.com	N	5 MB	N
360-pic	https://st.so.com/	N	2 MB	Y
Imgbb	https://imgbb.com/	N	16 M	N
sm.ms	https://sm.ms/	N	5 MB	N
upload.cc	https://upload.cc/	N	5 MB	N
Imgur	https://imgur.com/	N	1 MB	N
sina	http://photo.weibo.com/	Y	20 MB	Y
qiniu	https://www.qiniu.com/	Y	Unlimited	N

(2) Publicly available resource for storing the address of command

In the CA channel, we test some services and show selection standard in Tables 5 and 6. For the URL shortener website, because it itself stores the mapping relationship between the long URL and the short URL, and the storage time is usually unlimited, PR-Bot does not have too many restrictions when selecting such publicly available resources, as long as it supports customized URL for dynamic addressing. Similarly, for social network website, in order to achieve the dynamic addressing, PR-Bot only selects the social platform that can customized homepage URL based on the user name. In addition, priority is given to websites that support temporary mailbox registration in order to avoid exposing too much real information about the botmaster.

Table 5. URL shortener and its selection standard

Name	Site	Customized URL	Storage time
tinyurl	https://tinyurl.com/	Y	Unlimited
is.gd	https://is.gd/	Y	Unlimited
yep.it	http://yep.it/	Y	Unlimited
shorturl	https://shorturl.com/	N	Unlimited
shorl	http://shorl.com/	N	Unlimited
bit.ly	https://bitly.com/	N	Unlimited

(3) Publicly available resource for storing the feedback result information

In the RF channel, for the online clipboard website, except for the storage space of up to 200 KB and the storage time up to 1 month, the website that supports customized URL is required, for making that the botmaster can find the result information fed back from bot by a certain rule (URL + numeric string). For the public cloud disk, as shown in Table 7, PR-Bot mainly considers two factors: one is the file size limit, and the other is the storage time, which is similar to the image hosting. If the website allows 10 M-sized files to be uploaded and storage time can be up to 1 month, it is enough. In addition, the public cloud disk is available without registration.

Table 6. Social network and its selection standard

Name	Site	Customized URL	Temporary mailbox
Tumblr	https://www.tumblr.com	Y (revisable)	Y
Pinterest	https://www.pinterest.com	Y (revisable)	Y
Ask.fm	https://ask.fm	Y	Y
Twitter	https://twitter.com	Y (revisable)	N
Facebook	https://www.facebook.com	N	N
Weibo	https://weibo.com	N	N
Qzone	https://qzone.qq.com/	N	N
Renren	http://sns.renren.com/	N	N

Table 7. Public cloud disk and its selection standard

Name	Site	File size limit	Storage time
Sendspace	https://www.sendspace.com	300 M	30
Fileden	http://fileden.net/	100 M	60
Senduit	http://www.senduit.com/	100 M	7
Zippyshare	https://www.zippyshare.com/	500 M	30
Rapidshare	http://www.rapidshare.com.cn/	100 M	30

3 The Implementation of PR-Bot

3.1 CC Channel

PR-Bot mainly supports two types of commands, the “callhome” and “file stealing”, which parameters are shown in Table 8.

In order to prevent C&C hijacking and replay attacks, before issuing the command, the botmaster will specify the validity period of the command and sign the command based on the private key. The format of the command to be issued is as follows:

*Base64 (Base64 (private key signature (original command ^ validity period))
#original command ^ validity period)*

Table 8. The parameters of commands

Command	Key	Value	Remark
Callhome	cmd_type	callhome	Command type
	paste_name	wepaste	Website name
	address	http://www.wepaste.com/abcde	Url address
File stealing	cmd_type	upload_file	Command type
	cloud_name	sendspace	Website name
	paste_name	wepaste	Website name
	file_type	doc	File type
	address	http://www.wepaste.com/abcde	Url address

Take the “callhome” as an example, its original content is a string in JSON format, which is as follows:

```
{
  "cmd_type":"callhome",
  "paste_name":"wepaste",
  "address":"http://www.wepaste.com/abcde"
}
```

First, after specify validity period, private key signature, base64 encoding and string splicing, the corresponding content is as follows:

```
bqi+YmccF62Li+INvT4xRdO8d6Z7GXy74E8qVKYwHNrCYUY7wGEuLHrm5bdfyeuQ4S7BH5fx
zIQmsQn9xY/+iPjzXv9ap/mZefCY5JCpzQ6X/uLsQPYulvihTJZ52deiQZvPRwWZtPwSCBu3si1Pga
N/mxs8eWSgl7PGuWD1L37/TT7BvG+IrdozFqBQF5kILIX3hahEIqsh7DRJpDpwyfCH5Nz2K5zm7
X3apA34Sz1OL1vNxfHML2TZtNBR2LTCgtaXWa9JNiszHulPEF7pgHdTxWLuqQ/IG5Te9/5LN04y
7evFnJETMPK+WLV4mRaFwkjlx3c6kHkZJ64eyl5qiQ==#{ "cmd_type": "callhome", "paste_name": "w
epaste", "address": "http://www.wepaste.com/abcde"}^2018_05_12-2018_06_12
```

Next, if selecting the online clipboard website to store the command, it only needs to issue the above content in the form of text to the online clipboard website, which corresponding address looks like <http://dpaste.com/0SV8NS5>. But if selecting the image hosting website to store the command, the above content shall be converted into picture firstly and then issued, which corresponding address looks like <http://h.hiphotos.baidu.com/image/pic/item/c8177f3e6709c93db7a0d055933df8dcd00054c6.jpg>. In the process of storing data based on picture, PR-Bot does not embed the command into an existing picture, but directly converts it into pixels, and stores the original content in the form of pixels. The converted image style is shown in Fig. 4.

Normally, each pixel in a colored image is composed of three color information of RGB. Each color information occupies 8 bits, and the three colors are 24 bits, which means that each pixel can store 3 bytes of data. For a 500*500 RGB picture, it can store 75000 bytes of data, about 730 KB, which is enough to meet the space required for storing the command. Take the string “callhome” as an example: its hexadecimal

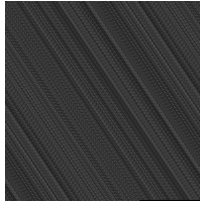


Fig. 4. Image style generated from text

representation is “0x63, 0x61, 0x6c, 0x6c, 0x68, 0x6f, 0x6d, 0x65”. First, it is divided into groups and each group consists of three units, if there are less than three, add 0 at the end. In this way, the original hexadecimal string is divided into three groups of {0x63, 0x61, 0x6c}, {0x6c, 0x68, 0x6f} and {0x6d, 0x65, 0x00}, and the corresponding RGB can produce three pixels. That is, the string “callhome” is converted into three pixel values. In addition, in the process of generating a picture, in addition to recording the content of the original data, the size of the original data also needs to be recorded to restore it normally. For PR-Bot, it uses two pixel units at the beginning of the picture, which is the six-byte space, for recording the data size. Finally, the pixel information in the picture consists of “data size (2 fixed pixels) + data content + 0 (may exist)”.

3.2 CA Channel

When issuing the address of command, the botmaster designs a UGA algorithm, which seed is based on the current date and hottest topics. This method is to prevent the address list generated by the bot from being predicted prematurely by the defender. The generated address list is as follows:

URL Shortener:

<https://tinyurl.com/shehuim>

<https://tinyurl.com/ixniimli>

<https://tinyurl.com/vyowinfc>

...

Social Network:

<https://www.tumblr.com/yfvqwvvi>

<https://www.tumblr.com/nshynnuu>

<https://www.tumblr.com/ldtctknm>

...

For the URL Shortener, when storing the address of command, only the URL address to be converted and the suffix of the alternative URL is needed, as shown in Fig. 5. And for the social network website, the user’s personal homepage address needs to be configured based on the alternative URL suffix, and then the address of command can be issued as a new message. Of course, all operations are automated by the program.

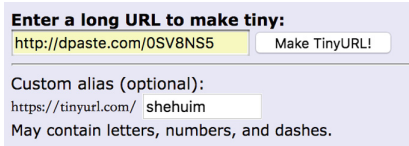


Fig. 5. How to use the URL shortener

As shown in Fig. 6, for the bot, it undergoes two steps when obtaining the content of command, that is the “Secondary Addressing Mechanism” described above. First, the bot will traverse the generated address list based on the hard-coded UGA algorithm. When finding the address PS_Address_B, it will obtain the address PS_Address_A and further extract the content of command. The reason for adopting the “Secondary Addressing Mechanism” is to improve the flexibility and scalability of the PR-Bot. For some publicly available resources that are suitable for storing commands but not support customized URL, the jump relationship provided by the “Secondary Addressing Mechanism” can make it become “a publicly available resource that supports customized URL”. In addition, this mechanism can also improve the robustness and concealment of the C&C channel to some extent.

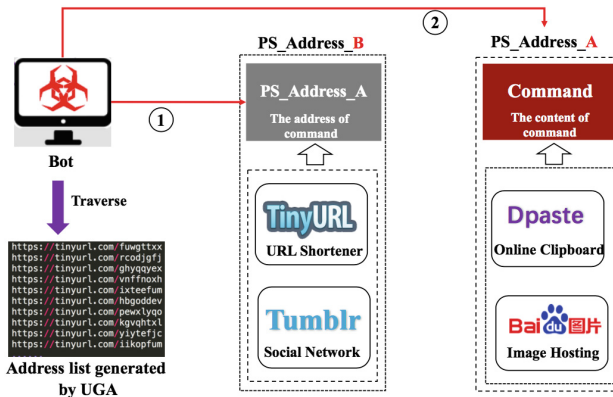


Fig. 6. Secondary addressing mechanism

3.3 RF Channel

Callhome Module. If all bots upload the information of controlled end to the unique URL specified in the command, there is a problem of information loss due to the limited storage space of the online clipboard website. In order to avoid this problem, this paper adopts the strategy of “URL + numeric string”.

As shown in Fig. 7, after the bot extracts the address parameter from the command, the bot will add a numeric string of the specified digits (according to the law from small to large) behind the address, and then it sends the device information to the new

address. Take the address <http://www.wepaste.com/abcde> as an example: After the bot obtains the address parameter, it will add a numeric string from 000000 to 999999 behind it and then traverse the URLs from <http://www.wepaste.com/abcde000000> to <http://www.wepaste.com/abcde999999>. The device information is not fed back until an address with empty content is found.

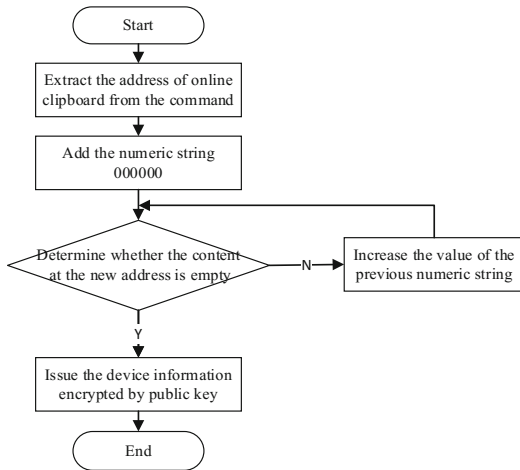


Fig. 7. The brief solution for result feedback

However, if only the above brief solution is adopted, although the problem of the limited storage space can be solved, there is a problem of information coverage due to concurrent operation of bots. That is, if an address with blank content is found by two bots at the same time, they will upload device information to the address, no matter who comes first, there must be a case where one bot overwrites information uploaded by another bot. It is because the content on the online clipboard website is readable and writable.

In order to solve this new problem, this paper uses an enhanced solution, as is shown in Fig. 8. Specifically, after several minutes of uploading device information, the bot reads the uploaded information again and compares it with the locally stored information to verify whether the information is uploaded by itself. If the MD5 of the two are the same, it is considered that the device information is successfully uploaded. Otherwise, the new address is traversed sequentially and the device information is re-uploaded. In this way, even if multiple bots find an address with empty content at the same time, there will be no problem of information coverage.

File Stealing Module. Cloud disk website, also known as cloud storage website, is mainly classified into two categories: the public cloud disk and the private cloud disk. The public cloud disk can be used without registering, and the information on it is public. In the process of feeding back files, PR-Bot selects a public cloud disk to store the stolen files, which is shown in Fig. 9. First, the bot traverses the specific types of

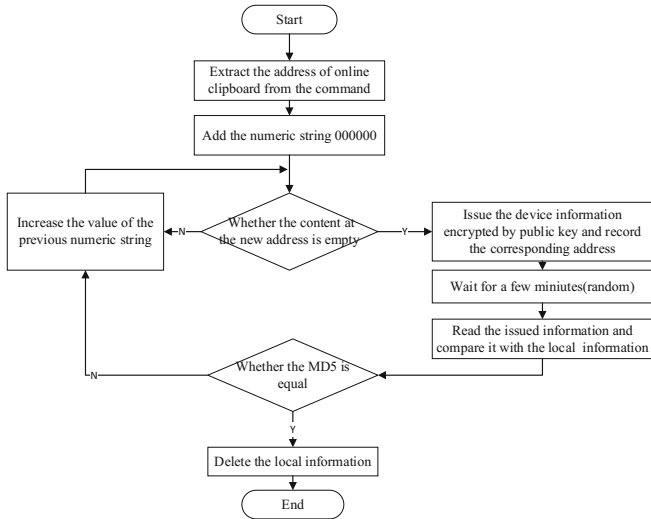


Fig. 8. The enhancement solution for result feedback

files in the controlled end and uploads them to public cloud disks one by one, and records the corresponding URL address at the same time. Then, all URL addresses are issued to the online clipboard website in the form of string. When issuing the address, it is basically the same as the flow of issuing callhome information, including the strategy of “URL + numeric string” and the mechanism of secondary verification.

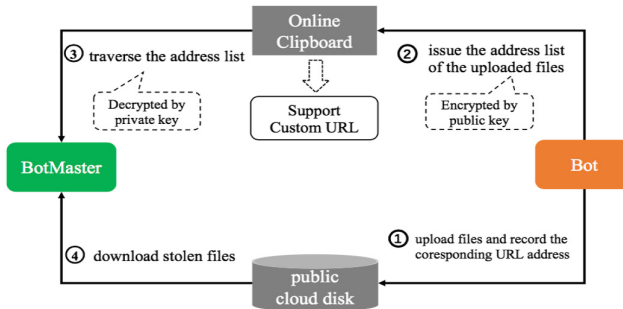


Fig. 9. File feedback process based on public cloud disk

Here, we will discuss the reason why PR-Bot does not choose private cloud storage to store stolen files. If PR-Bot selects the private cloud disk to store stolen files, it is necessary to specify the parameters required by the private cloud disk API interface in the command, which is used for authentication. It will cause two problems:

- (1) Once the defender has mastered the identification information, it is equivalent to obtaining the control authority of the account corresponding to the private cloud disk, so the files in the cloud disk can be viewed and deleted.
- (2) With the increase in the number of bots, the behavior of sharing the same account can easily cause abnormalities on the website, and may expose the entire botnet's activities.

For the public cloud disk, the above two problems are inexistent. First, even if the defender obtains the command and masters the law of “URL address + numeric string” adopted by bots, it cannot locate the network location where the stolen file is located. It is because the uploaded address list is encrypted by the public key, and it can only be decrypted by the private key of the botmaster. Second, when a file is uploaded through a public cloud disk, each bot is equivalent to an independent user and has no necessary correlation, so there is no problem that multiple bots share the same account. Therefore, PR-Bot selects the public cloud disk to store stolen files, whose purpose is to ensure the security of the C&C channel.

4 The Defense Measures

Accurately finding botnets similar to PR-Bot and taking targeted measures to contain them is the ultimate goal of this paper. For the PR-Bot botnet proposed in this paper, this section describes the contents of PR-Bot defense from the aspects of detection, measurement and tracking, in order to take over the control of the botnet or reduce its availability.

4.1 Detection

In the CA channel, PR-Bot uses the UGA algorithm to generate an address pool, and the bot obtains commands by connecting the pseudo-random addresses. This process is similar to DGA. Therefore, some methods for detecting DGA also apply to UGA detection: (1) Character feature detection based on domain name [14]. There are still differences between the addresses generated by UGA and the normal addresses, such as: the use of a large number of URL Shortening services, the use of unusual user names, the use of fixed social network and etc. Therefore, the rules of distribution of domain name strings can be found by constructing the semantic rules and feature vectors, and they can be identified by the methods such as data mining and machine learning. (2) Detection based on domain name activity. In order to obtain commands, the bot will constantly address, and the addressing time will show some regularity, such as addressing at a fixed point of time even early in the morning. These features all show the non-human characteristics, so the domain name activity and spatio-temporal features can be used to detect the malicious addresses [15].

In the CC channel, PR-Bot mainly uses the online clipboard website and image hosting website. Therefore, the detection method based on communication content and network layer anomalies can be used. (1) For the commands issued on an online clipboard website, the bot will obtain the commands in the form of text, which are

encoded and have the specificity, as well as identifiability. Among them, for the transmission content of the HTTP protocol, a feature matching rule may be configured in advance, such as Snort and other intrusion detection systems, to quickly and accurately discover such botnet. (2) For the commands issued on the map bed website, the bots will obtain commands from the downloaded pictures. It can also be detected through the abnormality of the transmitted information. However, the detection method based on the communication content is only applicable to botnets with specific characteristics. The disadvantage is that the unknown botnets cannot be detected, and the signature of the bot program needs to be continuously maintained and updated. The network layer anomaly detection method assumes that the communication mode between the botmaster and bots is quite different from the normal user communication, so that the trail of the botnet can be found through the flow analysis [16]. In the RF channel, PR-Bot uploads the text information according to the address specified by the botmaster, which is similar to the CC channel, so the detection method based on the communication content and the network layer anomaly may also be used.

In addition, public resource service providers should actively improve the security protection of the website to prevent normal services from being abused by attackers. PR-Bot needs to automatically register a large number of accounts and automatically issue control commands. Therefore, service providers can use the verification code-based or speed-limiting method to prevent the account from being registered in batches. Although this method will degrade legitimate users' experience, it increases the cost of the attacker and can effectively avoid creating a potential target for attackers. Besides, the content of the account on the platform can be monitored in real time, and the release of the suspicious character string shall be further traced or handled by the security personnel.

4.2 Measurement

By measuring the PR-Bot botnet, it can portray its topological structure and corresponding scale, so that the defender can understand more about the outline and characteristics of the PR-Bot. However, due to the mechanism characteristics of PR-Bot itself, it is difficult to measure the PR-Bot, and the traditional measurement methods based on Crawler and Sybil cannot be applied. However, in the RF channel, the bot adopts the strategy of "URL + numeric string" to upload the callhome information or the address list of stolen files. The defender can find out the pattern adopted by PR-Bot through reverse analysis or flow monitoring, so that the entire scale of the botnet can be measured through the method of address traversal. Although the PR-Bot measurements are affected by various factors, such as time zone, startup/shutdown, it is difficult to accurately estimate the scale of the entire botnet, but it can estimate the number of bots as much as possible.

4.3 Tracking

If the defenders have mastered the botnet C&C channel, they can run the bot in a controlled environment or join the botnet in an infiltrated form to understand the internal activity of the botnet. In this section, we focus on how to track botnets by

means of infiltration, and the infiltrating agent is called “Infiltrator”. Infiltrator can disguise as an infected controlled device to join the botnet and simulate the real communication protocol of PR-Bot to communicate with the botmaster to observe the internal activities of PR-Bot. Among them, in the RF channel, the infiltrator can intentionally submit a decoy file with tracking watermark or other payloads, so as to track the botmaster. For example: the infiltrator embeds a hidden remote picture URL in a Word document, so if the botmaster downloads and opens the file, it will actively request the URL and load the remote picture, and then the defender can trace the position of the botmaster based on the source of the request.

5 Related Work

To be well prepared for future botnet attacks, security researchers have done many works on studying advanced botnet models and defense technologies.

Sanatinia et al. [17] presented a robust, stealthy botnet that named OnionBots. The botnet use Tor privacy infrastructures for cyber-attacks by completely decoupling their operation from the infected host IP address and by carrying traffic that does not leak information about its source, destination, and nature. Ali et al. [18] presented ZombieCoin which used Bitcoin network for botnet C&C. ZombieCoin is robustness, because common takedown techniques of confiscating suspect web domains, seizing C&C servers or poisoning P2P networks, would not be effective. Yan et al. [19] proposed an anti-pollution P2P botnet called AntBot, which used a tree-like structure to propagate commands in P2P networks. The tree-like structure with the randomness and redundancy in its design, renders it possible that individual bots, when captured, reveal only limited information.

Besides, there are a number of botnet designs are based on publicly available resources. Artturi et al. [20] explores the multitude of ways in which modern malware abuses third-party web services as C&C channels, including Google Docs, Tumblr, Twitter and so on. Lee et al. [21] explore botnets based on USS, and propose alias flux methods that frequently change shortened URLs of C&C servers to hide their existence, which is similar to the domain flux method. Nagaraja et al. [12] exploit image steganography techniques to set up a communication channel within the social network, and use it as the botnet’s C&C channel. However, none of these research works have studied how to design a resilient and efficient bidirectional communication channel. Our study focuses on constructing a three-channel botnet based on multiple publicly available resources and is complementary to the existing research works to some degree.

On the defensive side, there have been many types of approaches to detect botnets, including signature-based, anomaly-based, DNS-based and data mining, machine learning techniques. For public service-based botnets, Chen et al. [22] design an unsupervised system to detect Twitter spam campaigns that use botnets to send duplicate content with embedded URLs. The unsupervised detection approach allows to build a blacklist of malicious email addresses, URLs and Twitter accounts, and to share threat intelligence with the research community in real-time. Guo et al. [23] explore the currently typical C&C server finding schemes as three types: dedicated IP

address, Internet infrastructure and third-party service from a new perspective. Their work indicates that third-party service based C&C presents a better approach in terms of complexity, flexibility, traffic covertness and scale. In this paper, for PR-Bot, we propose the targeted defense strategies from the perspective of detection, measurement and tracking, so as to achieve the goal of combating against such botnets.

6 Conclusions

This paper introduces an advanced botnet based on publicly available resources, which is named PR-Bot. The PR-Bot is constructed by a three-channel scheme, which includes three sub-channels: CC channel, CA channel and RF channel. Each sub-channel can be supported by multiple publicly available resources and can be extended in the form of plug-in. Meanwhile, PR-Bot also uses the technologies, such as information hiding, content encryption and digital signature, to improve the robustness and concealment of C&C channels. In addition, in the face of new challenges, this paper proposes the defense strategies against PR-Bot in terms of detection, measurement and tracking to deal with possible similar cyber threats. We believe that it is of great practical significance to study how to construct a highly antagonistic botnet from the perspective of the attackers and propose the effective defense strategies before the attackers deploy them in practice. In the next step, we will conduct an in-depth study on this type of botnets, and design a rapid and effective detection system.

Acknowledgements. This work is supported by the Key Laboratory of Network Assessment Technology at Chinese Academy of Sciences, Beijing Key Laboratory of Network Security and Protection Technology, and the National Key Research and Development Program of China (No. 2016YFB0801604, No. 2016QY08D1602, No. 2016QY06X1204).

References

1. Xiang, C., Binxing, F., Jinqiao, S., Chaoge, L.: Botnet triple-channel model: towards resilient and efficient bidirectional communication botnets. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds.) *SecureComm 2013*. LNICST, vol. 127, pp. 53–68. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-04283-1_4
2. Li, C., Jiang, W., Zou, X.: Botnet: survey and case study. In: 2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC), pp. 1184–1187. IEEE (2009)
3. Bailey, M., Cooke, E., Jahanian, F., et al.: A survey of botnet technology and defenses. In: *Conference for Homeland Security, CATCH 2009*. Cybersecurity Applications & Technology, pp. 299–304. IEEE (2009)
4. Amini, P., Pierce, C.: Kraken Botnet Infiltration [EB]. Blog on DV Labs, 2008 (2011). <http://dvlabs.tippingpoint.com>. Accessed 10 June 2011
5. Williams, J.: Operation b107 - Rustock Botnet Takedown (2011). <http://blogs.technet.com/b/mmpc/archive/2011/03/18/operation-b107-rustock-botnet-takedown.aspx>
6. Sharifnya, R., Abadi, M.: DFBotKiller: domain-flux botnet detection based on the history of group activities and failures in DNS traffic. *Digit. Investig.* **12**, 15–26 (2015)

7. Nazario, J., Holz, T.: As the net churns: fast-flux botnet observations. In: 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, pp. 24–31. IEEE (2008)
8. Stone-Gross, B., Cova, M., Cavallaro, L., et al.: Your Botnet is my Botnet: analysis of a Botnet takeover. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 635–647. ACM (2009)
9. Holz, T., Steiner, M., Dahl, F., et al.: Measurements and mitigation of peer-to-peer-based Botnets: a case study on storm worm. *LEET* **8**(1), 1–9 (2008)
10. Davis, C.R., Fernandez, J.M., Neville, S., et al.: Sybil attacks as a mitigation strategy against the storm Botnet. In: 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, pp. 32–40. IEEE (2008)
11. Thomas, K., Nicol, D.M.: The Koobface Botnet and the rise of social malware. In: 2010 5th International Conference on Malicious and Unwanted Software (MALWARE), pp. 63–70. IEEE (2010)
12. Nagaraja, S., Houmansadr, A., Piyawongwisal, P., Singh, V., Agarwal, P., Borisov, N.: Stegobot: A Covert Social Network Botnet. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 299–313. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24178-9_21
13. Cui, X., Fang, B.X., Yin, L.H., Liu, X.Y.: AndBot: towards advanced mobile Botnets. In: Proceedings of the 4th Usenix Workshop on Large-scale Exploits and Emergent Threats, LEET (2011)
14. Yadav, S., Reddy, A.K.K., Reddy, A.L., et al.: Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 48–61. ACM (2010)
15. Gu, G., Perdisci, R., Zhang, J., et al.: BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: USENIX Security Symposium, vol. 5, no. 2, pp. 139–154 (2008)
16. Silva, S.S.C., Silva, R.M.P., Pinto, R.C.G., et al.: Botnets: a survey. *Comput. Netw.* **57**(2), 378–403 (2013)
17. Sanatinia, A., Guevara N.: OnionBots: subverting privacy infrastructure for cyber attacks. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 69–80. IEEE (2015)
18. Ali, S.T., McCorry, P., Lee, P.H.-J., Hao, F.: ZombieCoin 2.0: managing next-generation botnets using Bitcoin. *Int. J. Inf. Secur.* 1–12 (2017)
19. Yan, G., Ha, D.T., Eidenbenz, S.: AntBot: anti-pollution peer-to-peer Botnets. *Comput. Netw.* **55**(8), 1941–1956 (2011)
20. Lehtiö, A.: C&C-as-a-service: abusing third-party web services as C&C channels (2015)
21. Lee, S., Kim, J.: Fluxing Botnet command and control channels with URL shortening services. *Comput. Commun.* **36**(3), 320–332 (2013)
22. Chen, Z., Subramanian, D.: An unsupervised approach to detect spam campaigns that use Botnets on twitter. arXiv preprint [arXiv:1804.05232](https://arxiv.org/abs/1804.05232) (2018)
23. Guo, X., Cheng, G., Hu, Y., et al.: Progress in command and control server finding schemes of Botnet. In: Trustcom/BigDataSE/I SPA, pp. 1723–1727. IEEE (2016)



Deep Packet Inspection with Delayed Signature Matching in Network Auditing

Yingpei Zeng^{1,2}(✉) and Shanqing Guo³

¹ School of Cyberspace, Hangzhou Dianzi University, HangZhou, China
cszyingp@yahoo.com

² China Mobile (HangZhou) Information Technology Co., Ltd., Hangzhou, China

³ School of Computer Science and Technology, Shandong University, Jinan, China
guoshanqing@sdu.edu.cn

Abstract. Deep Packet Inspection (DPI) is widely used in network systems and the processing speed of DPI is very critical. The core part of existing DPI is signature matching, and many researchers focus on improving the signature matching algorithms. In this paper, we work from a different angle: the scheduling of signature matching. We propose a method called Delayed Signature Matching (DSM), which could greatly reduce the number of matching attempts. In the method we do not always immediately match received packets to the signatures, but instead we predefine some protocol specific rules, and evaluate the packets against these rules first to decide when to start signature matching and which signatures to match, thus eliminate lots of useless matching attempts. The proposed DSM method is very suitable for the network auditing scenario since recognizing a flow at the earliest possible time is not required, and the potential seconds of delay brought in by DSM is acceptable. We also find that in the DSM method the number of matching attempts for a flow is unrelated to the number of supported protocols, which is a good property since the number of supported protocols keeps growing. Finally, we implement a prototype of the DSM method in the open source DPI library *nDPI*, and find that it can reduce the signature matching time 27%–40%.

Keywords: DPI · Deep packet inspection
Delayed Signature Matching · DSM · Fast path

1 Introduction

Deep Packet Inspection (DPI) is integrated into many network system today [1, 6–8, 11, 22]. For example, DPI is used in Firewalls [22], network security monitors [13], and intrusion detection systems (IDS) to recognize the protocols of packets for checking further threats in the application layer. Network auditing systems, which may be required by government regulations (e.g., for monitoring public Wi-Fi services), or by the companies (e.g., for monitoring employees’

Internet accessing), also use DPI to recognize which websites users are visiting and which applications users are using.

The processing speed of DPI is quite critical, since one DPI instance usually needs to process traffics from many different terminals, and the volume of the data usually is very big. Thus, the performance is an important consideration for DPI products. For example, commercial products like Qosmos¹ and PACE² claim to handle up to 9–10 Gbps per core, and open source solution like nDPI could handle up to 8.85 Gbps per core as well [1]. Improving the DPI performance could enable the same cores to support more traffics³, or reduce the number of needed CPU cores. Since the core work of existing DPI is to match received packets against known signatures (or called patterns), many researchers tried to improve the signature matching algorithms, e.g., by modifying the construction of deterministic finite automaton (DFA) [7, 11].

In this paper, we propose a new method focusing on the scheduling of signature matching. By looking closely at existing signature matching process, we find some signature matching attempts are wasted since the needed packets have not been received yet. So we propose a method called Delayed Signature Matching (DSM). In the method, we predefine some rules for targeted protocols, and evaluate the received packets against these rules. If the packets pass the rules, we could start signature matching with the signatures defined by the rules. If the packets do not pass the rules eventually, we will use the original processing method as usual. Intuitively, these rules produce *fast paths* in the signature matching (quickly find and match against the proper signatures). We analyze the correctness and performance of the DSM method. We also find an interesting property of DSM: the number of signature matching attempts needed for a flow is constant, and does not grow with the number of supported protocols as the original method does. The delay with DSM is only several seconds at most, which is certainly acceptable in the network auditing scenario. We implement a DSM prototype supporting HTTPS, HTTP, FTP, and POP3 protocols in the open source DPI library nDPI [1, 2], and evaluate it with different datasets. We find that with DSM support for only 4 protocols, the prototype has 27%–40% performance boost in the signature matching.

2 Related Work

There are many existing DPI systems [1, 3, 4, 13, 17]. The performance comparisons among some of them can be found in [6]. The DPI systems can be roughly classified [17] into regex-only [3, 4, 15] and hybrid [1, 13, 17] (i.e., combining regex and code) types. L7 filter [3] is a typical regex-only DPI system, which contains many regexes defined for different protocols, and a protocol’s detection mainly relies on its regex (unfortunately, the regexes have not been updated since 2009).

¹ <https://qosmos.com/>.

² <https://ipoque.com/products/dpi-engine-rsrpace-2>.

³ For example, the mobile subscribers of China Mobile Inc. consumed 23% more traffics in 2018 Q1, comparing with 2017 Q1.

nDPI [1,2] is a hybrid DPI system, and is forked from OpenDPI, which is an open source classifier derived from early versions of PACE (PACE is a commercial DPI product mentioned before) according to [6]. nDPI mainly uses code to match different protocols; however, it also supports automaton and Hyperscan [4] in some steps like host name match. nDPI is open source in Github and under active development by the ntop company. It can detect 240+ protocols now, and is used in another ntop product nProbe as well. nDPI's `guess_protocol_id` also acts as a *fast path* to find the correct protocol parser; however, it is based on the ports and the protocol field of IP header of *one packet* only, and can not fully eliminate useless matching attempts as well (we will show that in Sect. 3).

Many researchers focus on developing new algorithm to speed up the matching of patterns. Kumar et al. proposed *Delayed Input DFA* [11] which substantially reduces space requirements as compared to a DFA. Dharmapurikar et al. proposed to store signatures in bloom filters to implement matching in hardware [12]. Bremler-Barr et al. proposed to use repetitions in flows to skip repeated data by modifying the Aho-Corasick Algorithm [7]. On the other hand, recently, the privacy of deep packet inspection is gaining more attentions [8–10, 16], since detecting patterns in encrypted traffics like HTTPS is demanded [14], and at the same time DPI is increasingly running as a service in the public cloud platforms. These performance or privacy improvement researches usually are orthogonal to the DSM method proposed in the paper, since they focus on the exact matching algorithms, and the DSM method focuses on the *scheduling* of matching.

3 A Motivating Example

We here use an FTP example to describe how the DPI process works and why there are rooms to improve. We use the process of nDPI [1,2] for example, and other DPI engines like [3] are similar in general.

We show the first 9 packets of a typical FTP connection in Fig. 1. It contains a 3-way TCP handshake, and later USER and PASS commands to authenticate the client “demo”.

We then show how the packets of the FTP connection in Fig. 2 are processed by the nDPI engine. Packets are processed in flows in nDPI and all the packets of the FTP connection belong to the same flow. The first 3 packets are processed by 10–12 protocol parsers for signature matching, since only a few parsers are interested in and register for the *TCP and no payload* type packets, and some parsers later find the flow does not match them at all so they exclude themselves for the flow early. For packet #4, the engine first tries the guessed FTP protocol parser based on port 21; however, the FTP parser still cannot confirm that it is an FTP flow at that time (it needs more packets to confirm). The following packets are still no match for detection, and only more parsers exclude themselves for the flow. Finally, the packet #7 with reply code 331 makes the FTP parser believe it is an FTP flow and complete the detection. We can calculate that these protocol parsers are called 175 times in total to complete the FTP detection.

We can see that there are some matching attempts wasted in the processing. For example, matching the packet #4 to the 103 protocol parsers is doomed to

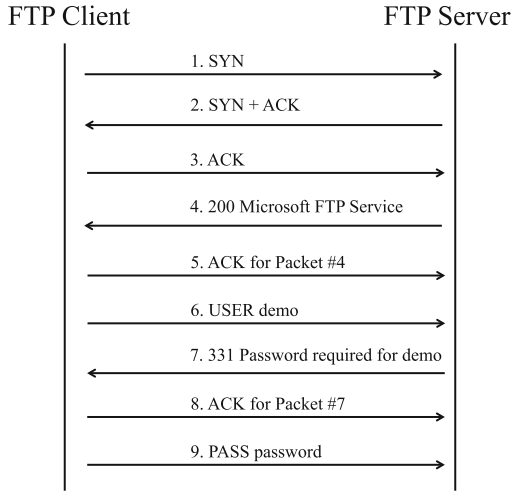


Fig. 1. The initial packets transferred during a typical FTP connection.

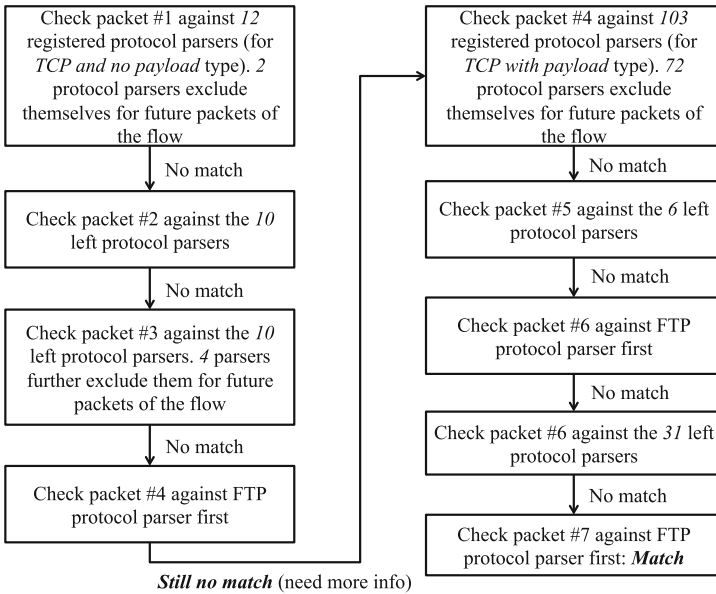


Fig. 2. The exact steps of how the packets in previous FTP connection example are processed. It shows that the protocol parsers are called 175 times (12 + 10 + 10 + 1 + 103 + 6 + 1 + 31 + 1) in total for signature matching to finish the detection. We could reduce the number to only 7 with DSM.

be useless. Though nDPI uses code-based match for FTP protocol, the situation is similar for regex-based DPI engines like L7 Filter and Hyperscan. In L7 Filter the engine keeps appending new packet's content to its flow's received content buffer (2048 bytes at most) and matching the buffer against all protocols' patterns [3]. L7 Filter may use the pattern `220[\x09-\x0d -~]*ftp|331[\x09-\x0d -~]*password` [5] for accurate FTP detection, then it also needs to keep matching the buffer to all patterns until it gets the packet #7 that contains the 331 reply code. Even for more efficient regex matching library like Hyperscan [4] which only needs to feed newly received packet into the library, the whole signature database of all protocols needs to be matched to the newly received packets again and again. Hyperscan also officially advises to avoid big union database if possible for performance⁴, which makes sense in the regex matching theory [15]. In this paper, we would like to reduce the match searching between the contents and patterns *in essence*. With the DSM method proposed in the paper, the DPI engine only needs to match *each packet against one protocol pattern* 7 times (in contrast to the original 175 times) in total for the FTP connection example.

4 Delayed Signature Matching

The basic idea of Delayed Signature Matching (DSM) is that instead of immediately matching received packets to the signatures, we wait for enough packets first. Then the problem is how to determine that currently received packets of a flow are enough. Our solution is to predefine sequences of rules for different protocols, and then evaluate the received packets against the rules first. When the packets pass the sequence of rules of a protocol, we could start signature matching with the protocol's signature. Any failures during the process (e.g., failed to pass the rules) lead to using the original processing method as fallback. Note the rules should be fast to evaluate, and the passing of a sequence of rules should indicate the flow has high probability to match corresponding protocol (otherwise the matching will be wasted). The sequences of rules in effect create *fast paths* in the signature matching, since the packets could directly match against proper signatures so fewer matching attempts are needed.

We show the framework of the delayed signature matching (DSM) method in Algorithm 1. The first flow maintaining step includes finding or creating a flow for the input packet, and updating the variable values used in the *primitives* (introduce later) for the flow. Then for a flow that is not detection completed, the engine checks whether the flow is marked to use the original processing method, or marked to call selected protocol parsers (introduce later) only. In the two cases the engine processes the packet accordingly. For other cases it checks whether the packet could match any DSM rule. If so it will use the instruction of the rule to determine what to do next, like calling the protocol parsers selected by the rule, and waiting for the next packet, etc. At last, if the packet could not match any rule, the engine uses the original processing method as fallback.

⁴ <http://intel.github.io/hyperscan/dev-reference/performance.html#unnecessary-databases>.

We can see the rules are the core part of DSM and in the left part of the section we will focus on introducing them, and use the rules of several protocols as examples.

Algorithm 1. Packet processing with delayed signature matching (DSM).

Input: a packet, DSM rules of protocols.

```

1: Do flow maintaining: find or create the flow for the packet
2: if The flow is not detection completed then
3:   if The flow is marked to use the original processing method then
4:     Use the original processing method
5:   else if The flow is marked to call the selected protocol parsers only then
6:     Call the selected protocol parsers
7:   else if Match any DSM rule then
8:     Follow the rule's instruction, which could be:
9:     - evaluate next rule and follow its instruction
10:    - call the rule's selected protocol parsers
11:    - mark the flow to use the original processing method, and process the buffered
    packets and current packet
12:    - buffer the packet and wait for next packet
13:    - mark the flow to use currently selected protocol parsers only
14:   else
15:     Use the original processing method
16:   end if
17: end if

```

First, we introduce a few types of *primitives* that we use in the DSM rules. These primitives are all simple and can be implemented efficiently, which make sure the rules can be checked very quickly.

- *protocol* type comparisons. For example, *protocol* ==TCP.
- *server_port* arithmetic comparisons, including ==, ≥, etc. For example, *server_port* == 21.
- *pkt_num* arithmetic comparisons, including ==, ≥ etc. *pkt_num* stands for the number of total packets have been received in the flow.
- *payload_pkt_num* arithmetic comparisons. *payload_pkt_num* represents the number of packets that have payload have been received in the flow.
- *tls_pkt_num* arithmetic comparisons. *tls_pkt_num* means the number of total packets that are of Transport Layer Security (TLS) record layer type [19] have been received in the flow. We implement the primitive by simply checking the first byte for 20 (change_cipher_spec), 21 (alert), 22 (handshake), and 23 (application_data) values.
- *payload[i]* ==A equation check. It means to check whether the byte at index *i* of the packet payload is equal to *char* A.
- *payload(i, len)* ==ABC equation check. It means to check whether the *len* bytes start from index *i* of the packet payload are equal to *string* ABC.

Then, we show the processing of FTP and POP3 protocols with DSM rules in Fig. 3. In the flowchart, *the DSM rules are represented by diamonds, and the instructions of rules are represented by rectangular boxes following the rules.* For FTP, after the common flow maintaining process, and assuming the engine needs to check the packet with DSM rules (e.g., the flow is not detection completed etc., as mentioned in the DSM framework previously), the engine checks the rule *protocol ==TCP* first. If the packet passes the rule, it checks another rule on whether the port is the common FTP port 21. If not it will further check other port rules if any. If port matches 21 and it will check whether the *payload.pkt_num* ≥ 4 , which means the packet #7 in Fig. 1 should have been received for the flow. If the rule is not satisfied, another rule *pkt_num* ≥ 10 will be checked, which means whether a threshold number of packets (here we set to 10, same as nDPI) have been received. If *pkt_num* exceeds the threshold, DSM will fall back to the original method, otherwise it will wait for the next packet. If the rule *payload.pkt_num* ≥ 4 satisfies, it will check the *payload[0]* against , i.e., the first character of the PASS command. If the rule satisfies again, then there are enough evidences that it is an FTP flow, so the rule selects the FTP protocol parser to parse all packets received so far. If the result of the parser is matched then we mark the flow as detection completed; the flow successfully goes through the fast path created with DSM rules. Otherwise, it is not an FTP flow and we exclude FTP from the possible protocol list and use the original packet processing method.

The DSM rules of the POP3 protocol are quite similar to the rules of FTP, except that we now match the USER command of POP3 [21] in the rule, i.e., *payload[0] ==U*, and only 2 packets with payload would be enough for the POP3 parser to recognize the flow, i.e., *payload_pkt_num* ≥ 2 .

At last, we show the DSM rules we created for HTTPS in Fig. 4. HTTPS is becoming prevalent and more than 30% top 1,000,000 websites use HTTPS by default now (i.e., redirecting HTTP pages to URLs with HTTPS)⁵. Similarly, the *protocol* rule and *server_port* rule are checked first. We then check whether there is at least one TLS packet received before (e.g., the Client_Hello message), and *payload_pkt_num* is greater than or equal to 3 (e.g., the Client_Hello, Server_Hello, and Certificate messages) [19]. If both rules are satisfied then the flow is highly likely to be an SSL connection now, so the HTTPS (TLS) protocol parser is selected. Similarly, if the parser does confirm the application protocol is matched, we mark the flow as detection completed. Otherwise, however, we cannot simply exclude SSL protocol from the flow, because the flow may be of TLS type, but the TLS protocol parser needs more packets to further detect its application protocol. So we add a rule to check whether it is a TLS flow by checking whether the server name has been gotten (either from the Server Name Indication (SNI) extension [20] of TLS, or from the server's certificate), or whether the TLS process has passed some stages (represented by *ssl_stage*). If we confirm that it is a TLS flow, we will mark the flow to always use the

⁵ <https://statoperator.com/research/https-usage-statistics-on-top-websites/>.

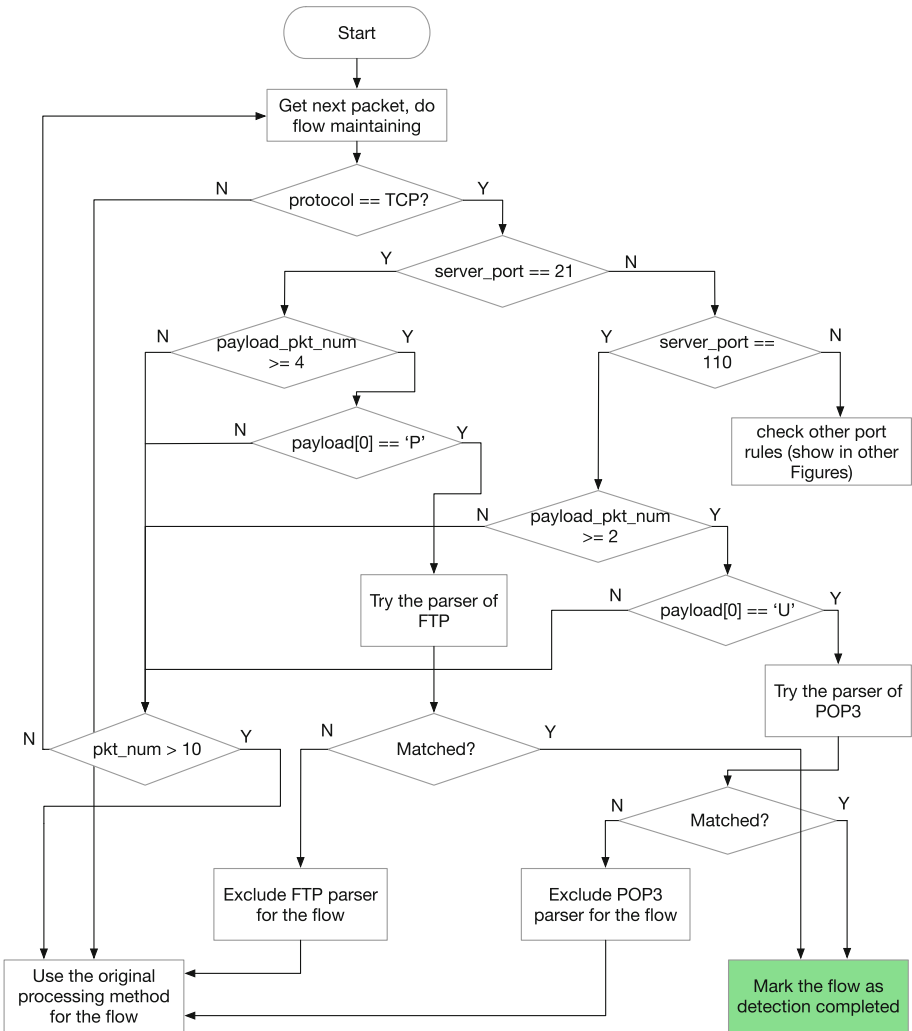


Fig. 3. The DSM rules for FTP and POP3. The rules are represented by diamonds, and the instructions of rules are represented by rectangular boxes following the rules. Flows that could pass through the DSM rules to the downright green box of the flowchart are favored, since they pass through the fast path and have very few matching attempts (Color figure online).

selected TLS protocol parser only. Otherwise, we exclude the TLS protocol from consideration and use the original processing method as well.

Note that we need to periodically check for flows that applied DSM (i.e., having buffered packets) but got stuck for some time, and use the original processing method to process them. This is because the flows may satisfy some rules of a protocol but cannot pass through them. For example, the *server_port*

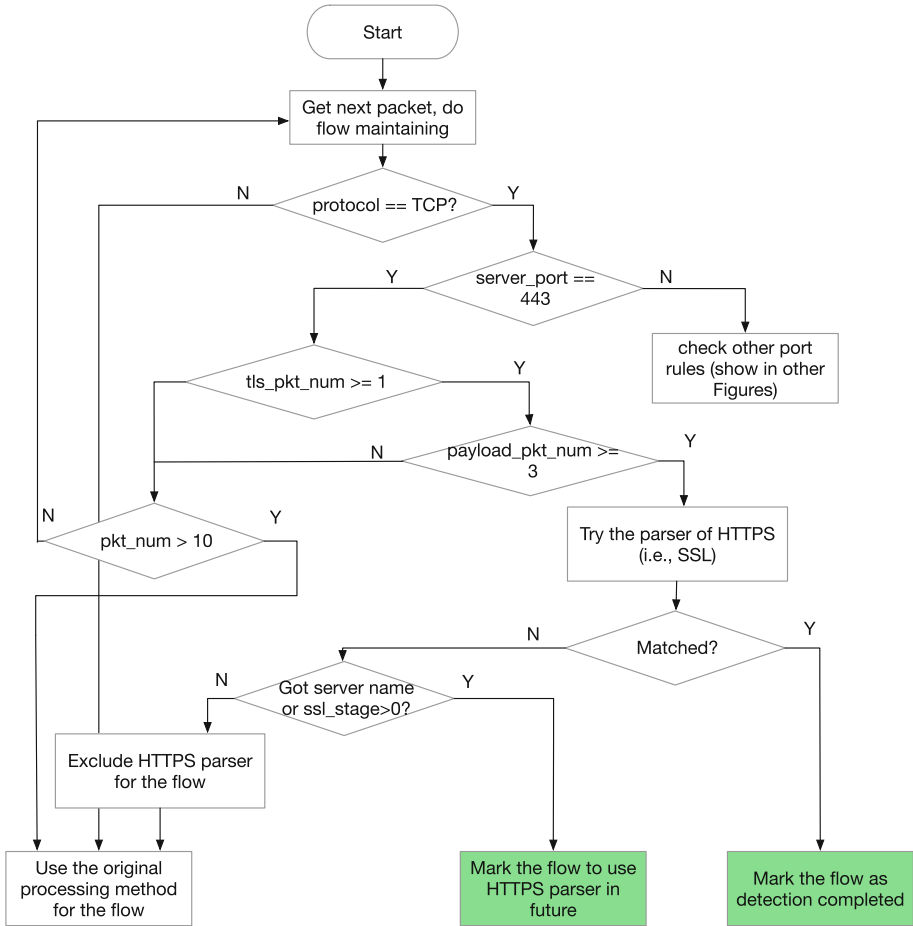


Fig. 4. The DSM rules for HTTPS. Similarly, flows that could pass DSM rules to the two downright green boxes of the flowchart are favored (Color figure online).

rule is matched but the *payload_pkt_num* is not enough. On the other side the flows may not have enough packets to eventually satisfy the threshold rule, i.e., the $pkt_num \geq 10$ rule. If we do not use the original processing method to process them, they will not be processed by any parsers even if some parsers can recognize them. We set the stuck time threshold to 2 seconds, which should be sufficient for existing protocols. Also, for efficiency such periodical check for stuck flows could be done together with existing idle flow cleaning process.

Another thing needs to be noted is how we define DSM rules. First, we prefer to wait for a bit more-than-enough number of packets before we begin the signature matching (e.g., for FTP we require that $payload_pkt_num \geq 4$ but not 3). This is because more packets usually imply more evidences on the type of the flow, and thus ensure higher successful probability of the selected

protocol parsers. This may not be suitable for the scenarios that require the earliest possible flow recognition like Firewall or IDS [22], but is suitable for network auditing. Second, on the other side, we may intentionally make the rules a bit relaxed (e.g., *tls_pkt_num* ≥ 1 for the TLS protocol) to suit different auditing scenarios and packet transmission strategies, because we want to make the rules more portable (written once and suitable for all). For example, in one of our network auditing scenarios, we mainly capture packets of one direction (upstream), and we also find TLS may transmit different record layer packets in one TCP packet. Both the first and second considerations make us prefer to define DSM rules waiting for more packets than the corresponding parser actually needs.

5 Analysis

We give analysis on the correctness and performance of the DSM method below.

5.1 The Correctness of the DSM Method

The delayed signature matching method essentially is to use the delayed packets to determine which protocol parsers to try for a flow, and there are four cases after trying the selected protocol parsers. The first case is that the selected protocol parsers fully detect the flow's protocol and we mark the flow as detection completed. In this case, the correctness of our method relies on the self-containing property of the protocol parsers, which means they should not need other protocol parsers to process the packets first before they can correctly recognize a flow. The self-containing property should be a reasonable assumption in reality, otherwise the codebase would be very fragile since users may change the protocol set to match for their environment. We also confirm that all the protocol parsers we checked in nDPI do have the property.

The second case is that the selected protocol parsers do not recognize the flow's protocol at all, and then we exclude the corresponding protocol and use the original processing method. The correctness of the DSM method in this case relies on the correctness of the decision that the selected protocol parsers cannot recognize the flow now and in the future. When the protocols are simple it is easy to make the decision based the rules.

The third case is that the selected protocol parsers recognize the flow as their types for sure, but they need more packet to act as detection completed, and we now simply mark the flow to always use the selected protocol parsers in future. The correctness of the DSM method in this case relies on correctly determining that the protocol parsers have recognized the flow and just need more packets for changing states. Like in HTTPS, we use the server name and *ssl_stage* as the indicators of TLS detection.

The fourth case is the selected protocol parsers are still unsure on the protocol of the flow. We didn't show any example on the case previously, since we intentionally avoid the case when creating the rules. However, there may have

complicated protocols that it is hard to bypass this situation when creating their rules. If so, we could not apply DSM to these protocols in the first place, or if we do use DSM, we could simply switch to use the original processing method then (but do not exclude the selected protocol parsers), and the correctness is also guaranteed.

We note that the DSM rules could be misled by protocol masquerading mechanisms like FTE [17], since they intentionally change the signatures of protocols to mimic other protocols. Defeating them is out of the scope of DSM and should be the responsibility of the protocol parsers. For example, if the mimicked protocol's parser is augmented to use new detection mechanisms like entropy-based detection [18], DSM could still successfully detect the original protocol.

5.2 Performance Analysis

We analyze the computation cost first. In order to simplify the analysis, we assume a protocol has only one payload packet type (UDP or TCP payload). We assume there are n protocol parsers interested in the payload type, and the corresponding protocol parser needs m packets to mark a flow of the protocol as detection completed. Then, for the original processing method, the number of calls to signature matching S_o can be defined as below, where a_i represents the ratio of the number of remaining parsers to the number of original parsers after processing the i th packet (i starts from 1):

$$S_o = n + a_1n + a_1a_2n + \dots + a_1a_2\dots a_{m-1}n, \quad \text{when } m > 1. \quad (1)$$

Now we use p to represent the possibility that the DSM method successfully matches a flow of the protocol, i.e., in either case #1 or case #3 as described in Sect. 5.1, and we assume the rules select k protocol parsers. Then we only need at most km calls to signature matching in the two cases (since protocol parsers may even exclude themselves from the detection of the flow later). For other two cases, we at most call signature matching $km + S_o$ times (i.e., DSM fails and the engine uses the original method as fallback). So we could represent the number of calls to signature matching of DSM method as below:

$$S_d = pkm + (1 - p)(km + S_o). \quad (2)$$

Usually we create rules that have high probability to successfully match the flows of the protocol, so p is approximate to 1 (we will later confirm that in Sect. 6) and then S_d is approximate to km :

$$S_d \approx km, \quad \text{when } p \approx 1. \quad (3)$$

The value of km usually is much smaller than n (k usually is 1–2, $m < 10$). Also we can see a very promising property from the equation: S_d is constant and not related to n now, which means adding new protocol parsers will not increase the computation cost, in contrast to the original processing method.

On the other hand, the evaluation of DSM rules does add some costs to the whole processing. However, we intentionally make the rules simple, and usually

only several increment operations and several comparisons are needed. So the added cost is very small, which is also confirmed in our experiments.

We also analyze the extra delay when using DSM. Note that if we create rules that strictly fit to the requirements of the protocol parser (i.e., do not waiting for more packets), and a flow’s packets match the rules as expected, then there is no extra delay for the flow comparing with the original processing method. This is because, even using the original processing method, the same set of packets needs to be received before the flow becomes detection completed. However, there are some cases that DSM has extra delay. First, if we create rules that need more packets than the parser actually needs (for the reasons we described before), we will have delay by waiting for extra packets. Depending on the number of extra packets, DSM may have several RTT (Round Trip Time) delay. One RTT usually is at most several hundred milliseconds in the Internet, so such delay is in the order of second at most. Second, for the flows that enter DSM process but get stuck there because of no matching rules and not enough packets, the extra delay is related to the stuck threshold time we set (at most 2 times of the threshold). Thus for the 2-second threshold we set, corresponding delay is 4 seconds at most.

6 Evaluation

We implement our DSM method prototype in the popular open source DPI library nDPI [1,2], with about 600 lines of code. The prototype contains the DSM rules of 4 protocols: HTTPS, FTP, POP3, and HTTP. It is open source for research usage at <https://github.com/zyingp/nDPI/tree/fastpath>⁶. We do all experiments on a PC installing Ubuntu 16.04 LTS, with Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz×4 and 16 GB memory.

We prepare 6 different datasets for the experiments, as shown in Table 1. The datasets come from two testbeds and a web crawler. We have two testbeds (named testbed A and B) for our network auditing tests, and in both testbeds *we mainly capture one side of the traffics* (i.e., upstream only, except for several special TCP ports) for auditing. One testbed contains 2 China Mobile enterprise WiFi gateways (i.e., hotspots), with 9 computers and phones connected. Another testbed contains 1 WiFi gateway with 2 computers and 1 phone connected (mainly computer traffics). TB_A36, TB_A79, and TB_A304 datasets are from the same testbed A but captured at different times and have different sizes. TB_B326 is from the testbed B. In order to test DSM with other traffic patterns, we also used a crawler running on an iPad to browse Alexa Top 100 and Top 500 websites⁷ and got two full traffic traces (i.e., both upstream and downstream traffics) named Top100 and Top500.

We first ensure that the DSM method (i.e., the engine only processes the aforementioned four protocols with DSM rules, and processes other protocols with the original method) detects exact the same protocols as the original

⁶ We also put our raw experiment results there.

⁷ <https://www.alexa.com/topsites>.

Table 1. The properties of the datasets.

Name	Size (MB)	Time (hour)	Num. of flows	Num. of pkts	Traffic description
TB_A36	36.2	30.7	12895	133610	upstream, computer & phone
TB_A79	79.6	23.9	30735	422966	upstream, computer & phone
TB_A304	304.6	103.4	110174	1599281	upstream, computer & phone
TB_B326	326.4	93.2	24970	1383572	upstream, mainly computer
Top100	135.2	0.6	3701	222924	both directions, tablet
Top500	685.3	3.1	37473	1035565	both directions, tablet

method alone for different datasets (e.g., the same 240+ protocols nDPI currently supports). During that we fix several implementation bugs. We also discover a bug of nDPI at that time (in the *ndpi_detection_giveup* function, the *protos* union in *ndpi_flow_struct* is always used as *ssl* type while it sometimes is of *http* type).

We then check the numbers that protocol parsers are called for these datasets, and show the result in Fig. 5. The result is very promising; the DSM method reduces about 42% (for TB_B326 dataset) to 76% (for Top100 dataset) of the calls needed in the original method, with DSM rules only for the four protocols. It is reasonable considering the great reduction of parser called times for protocols like FTP.

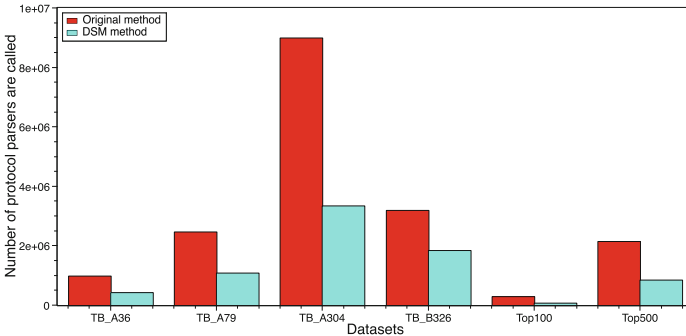


Fig. 5. Comparison on the numbers of the protocol parsers are called. The DSM method omits about 42%–76% of the calls needed in the original method.

Next we test the processing throughput of the whole DPI process. We run the built-in *ndpiReader* program 15 times for each dataset, discard the first 5 runs whose results may not be stable yet (since the program has large file I/O), and calculate the average of the left 10 runs (we use the same strategy for later experiments as well). We show the result in Fig. 6. When DSM is used, the throughputs all are higher than the original method without DSM: speed up about 20% for the TB_A* datasets, 11% for TB_B326, and about 7% for Top100 and Top500 datasets. The improvement is not as much as the previous

experiment. This is because here the program needs to do more work. The whole process includes loading packets from dataset files, packet unpacking, flow maintaining, and signature matching. We only improve the signature matching part. The other parts amortize the improvement on signature matching, especially for datasets like Top100 and Top500 which have more packets/traffics in a flow so signature matching consumes relatively less time in the whole process.

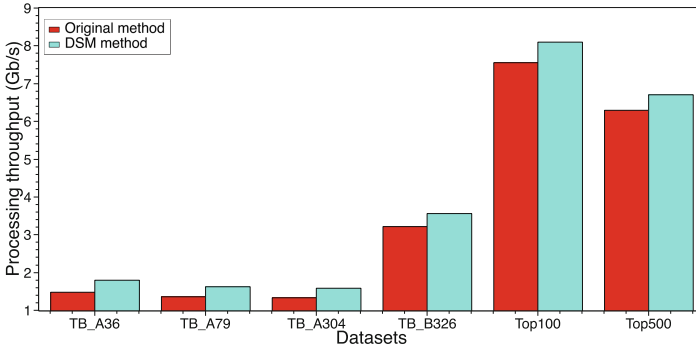


Fig. 6. Processing throughputs of the DSM method and the original method (loading packets from files needs to be done at the same time). DSM improves over the original method about 20% for the TB_A* datasets, 11% for TB_B326, and about 7% for the Top100 and Top500 datasets.

In order to get more accurate results on how the DSM method improves the DPI process, we evaluate it in refined scopes. First, we compare only the time spent on signature matching, which includes basic packet analysis (e.g., processing tcp flags), calling protocol parsers, and also rule evaluation for DSM. The result is shown in Fig. 7. We can see DSM could improve over the original

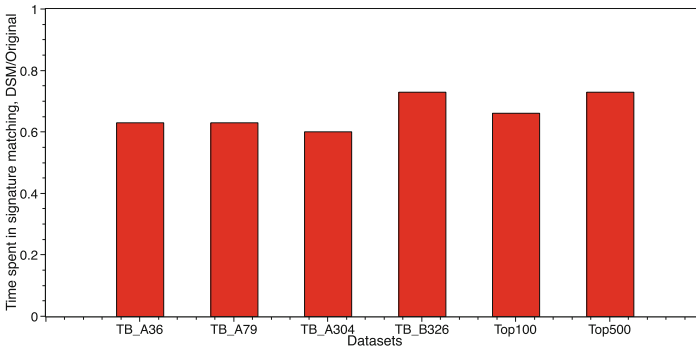


Fig. 7. The ratio of the DSM method to the original method on the time spent only on signature matching (and including rule evaluation for DSM). DSM improves over the original method about 27%–40%.

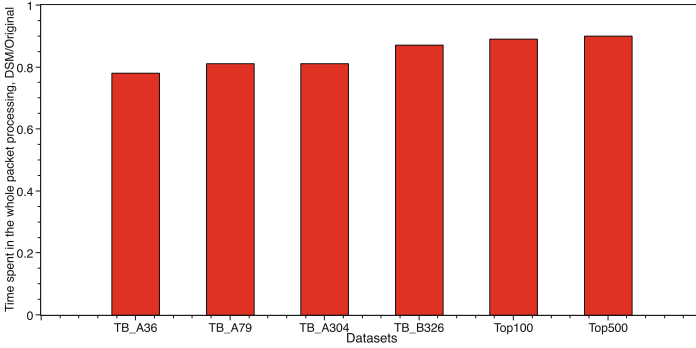


Fig. 8. The ratio of the DSM method to the original method on the time spent on whole packet processing (excluding the time spent on loading packet from files, but including the time on all other work like packet unpacking, flow maintaining, and signature matching). DSM improves over the original method about 10%–22%.

method about 27%–40%. Then, we check the DSM’s improvement on the whole packet processing (including all the work the program does in Fig. 6 except loading packets from files). This is more interesting since it contains actually all the work after we get a packet, either from pcap file or live capture (via libpcap, DPDK, or PF_RING). We show the result in Fig. 8. We can see that the DSM method consumes only 78%–90% of the time of the original method (improving 10%–22%).

Finally, we would like to check the effectiveness of current DSM rules. We calculate the number of flows that try the DSM selected protocol parsers (i.e., for the coverage ratio), and we also calculate the number of flows that try the DSM selected protocol parsers but fail to complete detection and turn to the original method as fallback (i.e., for the fail ratio/successful ratio). We show the result in Fig. 9. We can see that the coverage of our DSM rules is 30%–60% for these

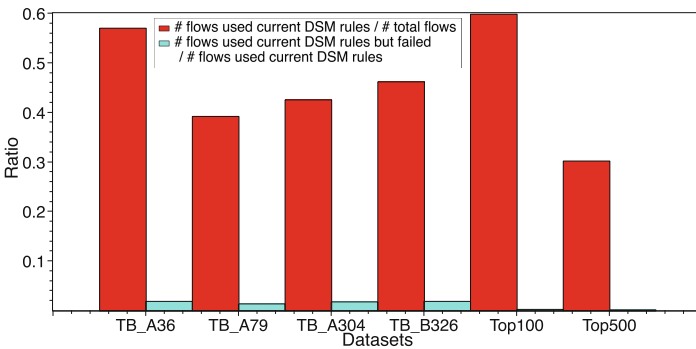


Fig. 9. The flow coverage and successful ratio of the DSM rules in our prototype. The flow coverage of current DSM rules is 30%–60%. The ratio of flows tried DSM but failed (still fell back to the original method) to all flows tried DSM is only 0.2% to 2%.

datasets, which is promising since we only implement rules for four protocols. Most of the coverage is due to the DSM rules of two protocols: HTTPS and HTTP. Also, the flows tried DSM but failed have a very small proportion, only 0.2%–2%. We also look into the fail cases and find the majority is due to the wrong protocol exclusion implementation of the nDPI HTTP protocol parser (e.g., a retransmission of a partial HTTP request may not have a line structure and the HTTP protocol parser will wrongly conclude that it is not HTTP).

7 Conclusions

In this paper we propose delayed signature matching (DSM) for reducing useless signature matching attempts. We achieve that by defining rules to tell when the signature matching could start for a flow and which protocol parsers to use. If a flow does not match any rules then the original method is used. We analyze the DSM method to show its correctness and efficiency. We also show the delay caused by DSM is at most several seconds which is affordable in network auditing, and may also be affordable in other scenarios that do not need real-time actions to packets. We implement DSM rules for four protocols including HTTPS in the open source DPI library nDPI, and evaluate them with different datasets. The result shows that the DSM method accelerates signature matching about 27%–40%, and accelerates the whole process 7%–20% (the more the signature matching time accounts for the total time, the more that whole process is accelerated).

References

1. Deri, L., Martinelli, M., Bujlow, T., Cardigliano, A.: nDPI: open-source high-speed deep packet inspection. In: Proceedings of International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 617–622. Nicosia (2014)
2. nDPI. <https://github.com/ntop/nDPI>. Accessed 5 June 2018
3. L7-filter. <http://l7-filter.sourceforge.net>. Accessed 5 June 2018
4. Hyperscan. <http://intel.github.io/hyperscan>. Accessed 5 June 2018
5. Bober, A.: Introduction to Layer 7-filter. https://mum.mikrotik.com//presentations/PL10/l7_interprojekt.pdf
6. Bujlow, T., Carela-Español, V., Barlet-Ros, P.: Independent comparison of popular DPI tools for traffic classification. *Comput. Netw.* **76**, 75–89 (2015)
7. Anat, B., Shimrit, T., Yotam, H., Hay, D.: Leveraging traffic repetitions for high-speed deep packet inspection. In: Proceedings of INFOCOM, pp. 2578–2586 (2015)
8. Sherry, J., Lan, C., Ada Popa, R., Ratnasamy, S.: BlindBox: deep packet inspection over encrypted traffic. In: Proceedings of SIGCOMM, pp. 213–226 (2015)
9. Yuan, X., Wang, X., Lin, J., Wang, C.: Privacy-preserving deep packet inspection in outsourced middleboxes. In: Proceedings of INFOCOM 2016, pp. 1–9. San Francisco (2016)
10. Schiff, L., Schmid, S.: PRI: privacy preserving inspection of encrypted network traffic. In: Proceedings of IEEE Security and Privacy Workshops (SPW), pp. 296–303. San Jose (2016)

11. Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., Turner, J.: Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In: Proceedings of SIGCOMM, pp. 339–350, Pisa (2006)
12. Dharmapurikar, S., Krishnamurthy, P., Sproull, T.S., Lockwood, J.W.: Deep packet inspection using parallel bloom filters. *IEEE Micro* **24**(1), 52–61 (2004)
13. Paxson, V.: Bro: a system for detecting network intruders in real-time. In: Proceedings of the 7th Conference on USENIX Security Symposium (1998)
14. Durumeric, Z., et al.: The security impact of HTTPS interception. In: Proceedings of NDSS (2017)
15. Backurs, A., Indyk, P.: Which regular expression patterns are hard to match? In: Proceedings of FOCS (2016)
16. Poddar, R., Lan, C., Ada Popa, R., Ratnasamy, S.: SafeBricks: shielding network functions in the cloud. In: Proceedings of NSDI (2018)
17. Dyer, K., Coull, S., Ristenpart, T., Shrimpton, T.: Protocol misidentification made easy with format-transforming encryption. In: Proceedings of CCS (2013)
18. Wang, L., Dyer, K.P., Akella, A., Ristenpart, T., Shrimpton, T.: Seeing through network-protocol obfuscation. In: Proceedings of CCS (2015)
19. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol version 1.2, RFC 5246, August 2008
20. Eastlake, D.: Transport layer security (TLS) extensions: extension definitions, RFC 6066, January 2011
21. Myers, J., Rose, M.: Post office protocol - version 3, RFC 1939, May 1996
22. Porter, T.: The perils of deep packet inspection, security focus (2005)



Low-Rate DoS Attack Detection Based on Two-Step Cluster Analysis

Dan Tang, Rui Dai^(✉), Liu Tang, Sijia Zhan, and Jianping Man

Hunan University, Changsha, Hunan Province, China
dr_506@hnu.edu.cn

Abstract. The low-rate denial of service (LDoS) attacks reduce the throughput of TCP traffic by sending high rate and short duration bursts periodically to the victim. Although many LDoS attack detection methods have been proposed, LDoS attacks are still difficult to accurately detect due to their low rate and good concealment. In this paper, we propose a novel method to detect LDoS attacks based on the fact that TCP traffic under LDoS attacks is more discrete than normal traffic and traffic under DDoS attacks. Two-step cluster analysis is adopted to cluster the network traffic based on the discrete characteristics of TCP traffic, and then the suspected cluster is detected by abnormal pieces analysis. The two-step cluster analysis method is proved to be effective for detecting LDoS attacks based on NS2 simulation. Experiments on public dataset LBNL and dataset WIDE also show that the method has a low rate of false positive.

Keywords: LDoS attack · Attack detection
Two-step cluster analysis · Abnormal pieces analysis

1 Introduction

Denial of service (DoS) attacks [5] are one of the most threatening attacks of Internet. Traditional DoS attacks consume the limited network resources by pouring a mass of attack flows into network continually that will make servers can't deal with requests of legitimate users in time. Launching a DoS attack requires the attacker to send attack flows keeping high rate and high frequency to the victim server, and we can distinguish DoS attack traffic from normal network traffic by the abnormal statistic characteristic [14]. The LDoS attacks [11] take advantage of adaptive mechanism of network protocol like TCP congestion control mechanism to slow down or limit TCP flows. The attacker sends high rate and short duration bursts periodically to the victim, that will trigger the congestion control mechanism repeatedly and cut down the TCP flows, then throughput of the whole network will decline.

This work was supported by China National Natural Science Foundation under Grant No.61772189 and No.61702173.

In this paper, two-step cluster analysis method is proposed to detect LDoS attacks. In the part of data acquisition, the network topology is designed on network simulator NS2 and network traffic is collected from the key router. Public dataset LBNL [13] is downloaded from Lawrence Berkeley National Laboratory and public dataset WIDE [7] is downloaded from WIDE Project. They are both analyzed by Wireshark. In the part of data preprocess, the discrete characteristics of TCP traffic are calculated as the input records of the two-step cluster analysis method. In the part of LDoS attack detection, two-step cluster analysis is adopted to cluster the network traffic, and then the suspected cluster is detected by analyzing pieces of TCP data. The detection results show that the method we proposed can accurately detect LDoS attacks.

2 Related Work

Currently, a lot of methods have been proposed to detect LDoS attacks, most of the detection methods can be divided into two types, the first detection type is based on the features of LDoS attack flows and the other is based on the abnormal characteristics of network traffic. As for the first detection type, Sun [19, 20] proposed the DTW detection method based on the feature of periodic time of attack flows. The DTW method allocated bandwidth on routers by deficit round robin algorithm for protecting the transmission resources of TCP flows, and the DTW method can obtain the cycle time and the bursts length of the LDoS attack flows. Kuzmanovic et al. [11] and Sarat et al. [18] proposed defense methods by taking advantage of AQM mechanism, they are based on the feature of high rate and short duration bursts of attack flows. The methods based on the AQM mechanism can filter out attack flows effectively. Guo [8] proposed the classification model CRF to detect BGP-LDoS attacks, and the classification features are chosen based on the three features of attack flows. However, it is generally recognized that these methods have a high false positive rate, because the features are not obviously enough when the network has lots of links of background traffic. The second detection type, for example, Luo [15] proposed the PLDoS attack detection method based on the wavelet analysis, Chen et al. [23] suggested to detect LDoS attacks by spectrum analysis, Tang [21] came up with the AEWMA detection method to analyze the ACK traffic, Wu [22] proposed the detection method by exploiting and estimating the changes in multifractal characteristics of network traffic, and Yue [24] adopted the neural network to identify the LDoS attack traffic based on the wavelet energy spectrum. These methods have a high detection rate, but we still have to make efforts to reduce the false positive rate of detection.

3 The Abnormal Characteristics of TCP Traffic Caused by LDoS Attacks

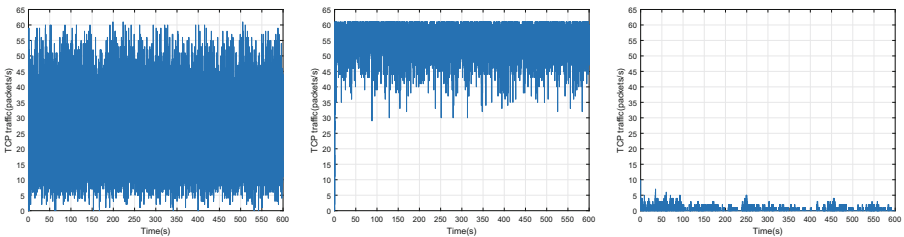
The LDoS attacks are aimed at cutting down TCP flows [12]. From the large time scale perspective, the TCP traffic is in the cycle of “steady decline and

then steady increase” based on the periodicity of LDoS attack flows. However, the TCP traffic without any attacks is centred around the high frequency, and the TCP traffic under other attacks like DDoS attacks is concentrated in the low frequency because there is not enough time for the TCP congestion control mechanism to recover TCP traffic. Figure 1 describes the TCP traffic with different network environments in large time scale(0s–600s). The samples in the figure include all TCP flows in the network. As is shown in the figure, the TCP traffic under LDoS attacks is more discrete than the TCP traffic in normal network environment and other attacks like DDoS attacks. The variance, the mean deviation and the coefficient of variation are adopted to measure the dispersion of TCP traffic. The variance indicates the discreteness of the total samples, the formula is shown as (1). The mean deviation inflects the deviation and dispersion of the sample distribution to its central value, the value of mean deviation is calculated as (2), and the coefficient of variation is a standardized measure of dispersion of a probability or frequency distribution, it is defined as the ratio of the standard deviation to the mean of the samples, like (3) shows. Where N is the total number of samples, x_i is the i_{th} sample value, m is the mean of total samples.

$$V = \frac{1}{N} \sum_{i=1}^N (x_i - m)^2 \quad (1)$$

$$MD = \frac{1}{N} \sum_{i=1}^N |x_i - m| \quad (2)$$

$$CV = \frac{\sqrt{V}}{m} \quad (3)$$



TCP traffic under LDoS attacks TCP traffic without any attacks TCP traffic under DDoS attacks

Fig. 1. TCP traffic with different network environments in large time scale.

From the small time scale perspective, in a LDoS attack period, the TCP congestion control mechanism would be triggered when the attacker sends high rate bursts to the victim, then the network would set limits on TCP traffic, and the TCP traffic starts to recover when the attack stops. The TCP traffic would

undergo the process of “steady decline and then steady increase”, and the range of TCP traffic is much larger than the normal network and DDoS attacks in a short duration of time. Figure 2 shows the TCP traffic with different network environments in small time scale (100s–110s). The samples in the figure include all TCP flows in the network. As is shown in the figure, the range of TCP traffic under a LDoS attack is much larger than the range of TCP traffic without any attack or under a DDoS attack.

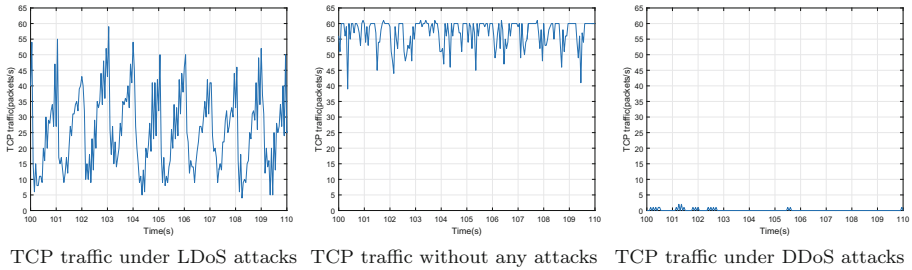


Fig. 2. TCP traffic with different network environments in small time scale.

4 The Proposed Detection Methods

The two-step cluster analysis method is proposed to cluster the network traffic based on the discrete characteristics of TCP traffic from the large time scale. For reducing the false positives, the abnormal pieces analysis is adopted to identify the suspected cluster which may suffer LDoS attacks from the small time scale.

4.1 The Two-Step Cluster Analysis

Clustering [9] is the unsupervised classification of input records into clusters based on the similarity, it is widely used in the data mining [1, 6], search engines [2], image segmentation [4, 16] and intrusion detection [10, 17] in network. The two-step cluster analysis [3] method is a scalable cluster analysis algorithm designed to handle very large data sets, so it is entirely applicable for analyzing complicated network traffic.

According to the discrete characteristics of TCP traffic, the two-step cluster analysis method is proposed to identify the network traffic. Two-step cluster analysis can gather the similar records into one cluster and divide the different records into other clusters. The variance, the mean deviation and the coefficient of variation of TCP traffic are calculated as the input records of the two-step cluster analysis method. The TCP traffic that is suffered LDoS attacks could assembled into a cluster because their values of discrete characteristics are similar.

Two-step cluster analysis method for detection of LDoS attacks has four steps: (1) Divide the TCP traffic into a number of fixed length detection units

(DU), and calculate three discrete characteristics of each DU as input records. (2) Pre-cluster the input records into many small sub-clusters. (3) Cluster the sub-clusters resulting from pre-cluster step. (4) Assign the cluster membership to identify which one is the suspected cluster. The process of two-step cluster analysis is given as Fig. 3, and more details for two-step cluster analysis algorithm, please visit the IBM Knowledge Center [3].

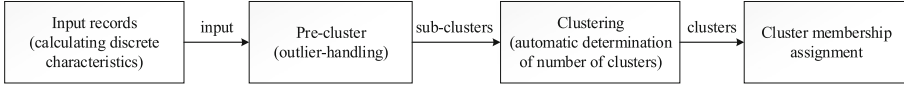


Fig. 3. The process of two-step cluster analysis method.

4.2 The Abnormal Pieces Analysis

The two-step cluster analysis method would cluster input records into several clusters, the cluster with the largest value of the input records would be suspected to be suffered LDoS attacks. The two-step cluster analysis method analyzes TCP traffic on the large time scale, which may lead to high false positive rate. So another method based on analyzing pieces of TCP traffic is proposed for the sake of reducing false positives. From the perspective of small time scale, TCP traffic dropped rapidly and then recovered under LDoS attacks, the range of TCP traffic is much larger than the normal network in a short duration of time attack, so we divide every DU in the suspected cluster into many test pieces (TP), each TP has k samples. The TP must be larger than the attack period T so that we can get a complete changing process of TCP traffic. The steps of LDoS attack detection in a DU are shown as follows:

- Find the maximum sample $Smax$ and the minimum sample $Smin$ in TP_i ($i = 1, 2, \dots, k$), let the $range(i) = Smax - Smin$;
- If $range(i) > \Omega_1$, the TP_i is defined as the abnormal TP(ATP);
- Let $TPR = \frac{count(ATP)}{k}$, if $TPR > \Omega_2$, the DU might suffer from LDoS attacks.

The threshold value Ω_1 is gained from the training data, which is composed of many TPs from normal network traffic. Calculating the range in each TP, and the Ω_1 is calculated as (4), the $Mean(range)$ is the average value of ranges and the $Std(range)$ is the standard deviation of ranges, z is the constant associated with detection accuracy. The train data is divided into many DUs, calculating the TPR in each DU, and the threshold value Ω_2 is calculated as (5), $Mean(TPR)$ is based on the average value of TPRs, $Std(TPR)$ is based on the standard deviation of TPRs, the value of z is the same as (4).

$$\Omega_1 = Mean(range) + z \times Std(range) \quad (4)$$

$$\Omega_2 = Mean(TPR) + z \times Std(TPR) \quad (5)$$

The flow-process diagram of LDoS attack detection is shown as Fig. 4.

5 Experiments and Results Analysis

To prove the effectiveness and performance of the detection method we have put forward, the experiments are constructed in the network simulator NS2, the public dataset LBNL [13] and the public dataset WIDE [7] respectively.

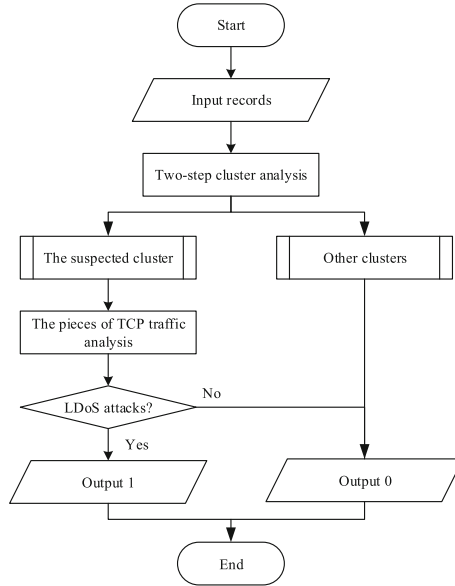


Fig. 4. The flow-process diagram of LDoS attack detection.

5.1 Experiments on NS2

The network topology is designed in NS2, which is shown as Fig. 5. There are three routers in network topology, and the link between router2 and router3 is the bottleneck link, whose bandwidth is 10 Mbps and network time-delay is 30 ms, apart from this, all links' bandwidths are 100 Mbps and network time-delay is 15 ms. Meanwhile, there are 25 legal TCP links in topology and 10 of them are set to generate background traffic. The attack flows are periodically launched by the link that connected router1, which target the bottleneck link.

We set three experiments in NS2 platform, and the parameters of attacks are set as Table 1. It is proved by Kuzmanovic [12] that the attack period $T \approx 1s$ can achieve the perfect attack performance, the attack bursts length $L \approx 0.1s$ and the attack bursts rate $R = 20$ Mbps based on the RTTs and the bottleneck bandwidth of network respectively. Meanwhile, we fix two attack parameters and vary one attack parameter to evaluate the precision of the detection method.

The simulation starts from 0 to 320s, and the attack is launched from 100s to 220s. We set sampling time $st = 0.05s$ and detection unit $DU = 20s$, so we

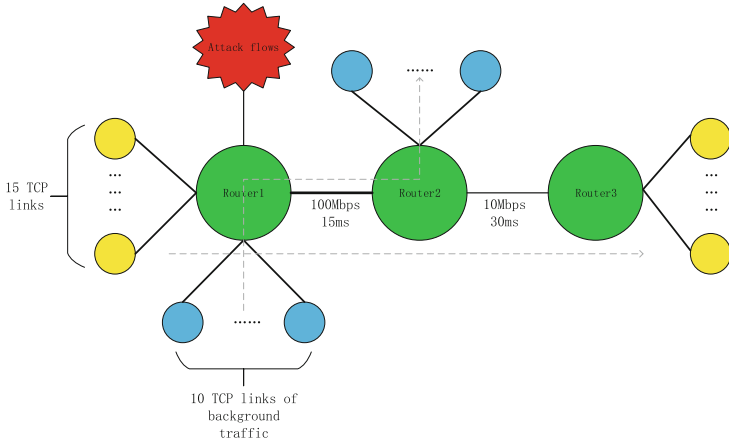


Fig. 5. Network topology in NS2.

Table 1. Attack parameters in NS2.

Experiments	Attack period (T)	Attack burst length (L)	Attack burst rate (R)
1	0.1s–2s	0.1s	20Mbps
2	1s	0s–0.3s	20Mbps
3	1s	0.1s	5Mbps–30Mbps

got 16 DUs after a simulation, among which 6 DUs (6_{th} to 11_{th}) were suffered LDoS attacks. Calculating the variance, the mean deviation and the coefficient of variation of TCP traffic for every DU, by which the method we proposed could cluster network traffic. The programming language AWK is used to extract the data of TCP traffic from the bottleneck link.

Table 2 shows the cluster results corresponding to the attack parameters we set in Table 1. In experiment 1, it’s almost DDoS attack when the attack period T is as short as the attack burst length L, the TCP traffic approaches to zero when suffered from DDoS attacks and it is as steady as the normal network. In experiment 2, it’s a normal network when L equals to zero. As can be seen from Table 2, input records are clustered into two groups, the TCP traffic under LDoS attacks can be clustered from normal network traffic easily by two-step cluster analysis method. However, the normal DUs like DU_1 and DU_{12} are wrongly attributed to the attacked DUs. In the DU_1 , the source-end of TCP tries to establish connections to the destination-end, TCP traffic is in a process of growth, it’s more unstable for DU_1 than other DUs, so does the DU_{12} , which is in the process of recovering TCP traffic from attack.

It’s a common occurrence that many TCP connections start to link up with the destination-end in every second. For reducing false positives, it’s necessary to detect the DUs again which are suspected to be suffered LDoS attacks.

Table 2. The cluster results.

Experiments	T	L	R	Cluster 1	Cluster 2
1	0.1s–0.2s	0.1s	20 Mbps	$DU_1-DU_{11}, DU_{13}-DU_{16}$	DU_{12}
	0.3s–0.5s			$DU_2-DU_5, DU_{13}-DU_{16}$	DU_1, DU_6-DU_{12}
	0.6s–2s			$DU_1-DU_5, DU_{12}-DU_{16}$	DU_6-DU_{11}
2	1s	0s	20 Mbps	DU_2-DU_{16}	DU_1
		0.1s		$DU_1-DU_5, DU_{12}-DU_{16}$	DU_6-DU_{11}
		0.2s–0.3s		$DU_2-DU_5, DU_{12}-DU_{16}$	DU_1, DU_6-DU_{11}
3	1s	0.1s	5 Mbps–10 Mbps	$DU_2-DU_5, DU_{12}-DU_{16}$	DU_1, DU_6-DU_{11}
			15 Mbps–30 Mbps	$DU_1-DU_5, DU_{12}-DU_{16}$	DU_6-DU_{11}

The pieces of TCP traffic analysis method is proposed to detect LDoS attacks in small time scale. The train data is gained from the normal network environment based on NS2 platform, and it is lasted for 600s. The test piece (TP) is set as 2s, and we get the threshold $\Omega_1 = 15.40$, $\Omega_2 = 0.46$, the constant value $z = 2.58$ by the train data. We test the DUs of the suspected cluster (cluster 2) by the abnormal pieces analysis method, the Table 3 shows the results based on the pieces of TCP traffic analysis method. We can see from Table 3, the DU_1 is removed from the suspected cluster and other DUs that suffered from LDoS attacks are remained.

Table 3. The results based on the pieces of TCP traffic analysis method.

Experiments	T	L	R	$APR < \Omega_2$	$APR > \Omega_2$
1	0.1s	0.1s	20 Mbps	—	DU_{12}
	0.2s			—	DU_{12}
	0.3s			DU_1, DU_7	DU_6, DU_8-DU_{12}
	0.4s–0.5s			DU_1, DU_{12}	DU_6-DU_{11}
	0.6s–2s			—	DU_6-DU_{11}
2	1s	0s	20 Mbps	DU_1	—
		0.1s	—	DU_6-DU_{11}	
		0.2s	DU_1	DU_6-DU_{11}	
		0.3s	DU_1	DU_6-DU_{11}	
3	1s	0.1s	5 Mbps–10 Mbps	DU_1	DU_6-DU_{11}
			15 Mbps–30 Mbps	—	DU_6-DU_{11}

5.2 Experiments on Public Dataset LBNL

The dataset LBNL [13] comes from the Lawrence Berkeley National Laboratory, we conduct experiment on the public dataset LBNL to prove the method has a low false positive rate. The test data from LBNL is lasted more than 18 hours, in which no LDoS attacks occurred. We set the sampling time $st = 0.1s$,

the detection unit $DU = 100s$, and the test piece $TP = 2s$, so we got 649 DUs in total, each DU has 50 TPs. The Fig. 6 shows the results of experiment based on the two-step cluster analysis method. There are 94 DUs (cluster 1) which are suspected to be suffered LDoS attacks. Then we analyze the TCP traffic pieces of the 94 DUs. The train data is extracted from another LBNL dataset that is in a normal network environment, and the train data is lasted for 600s. We get the parameters by the train data as: $z = 2.58$, $\Omega_1 = 105.95$, $\Omega_2 = 0.88$. The results based on the analyzing pieces of TCP traffic method is shown as Fig. 7. Finally we get 16 false positives after the two-step cluster analysis method and the abnormal pieces analysis method, the false positive rate is 2.46%.

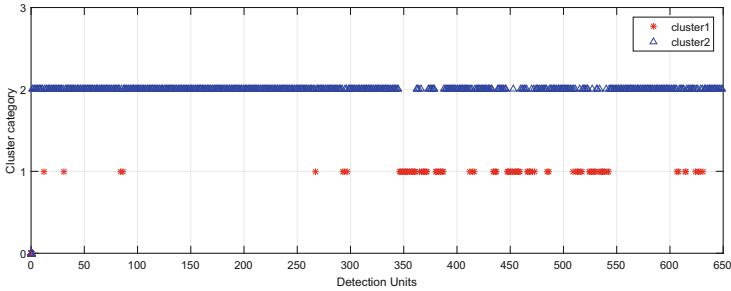


Fig. 6. Cluster results of LBNL dataset.

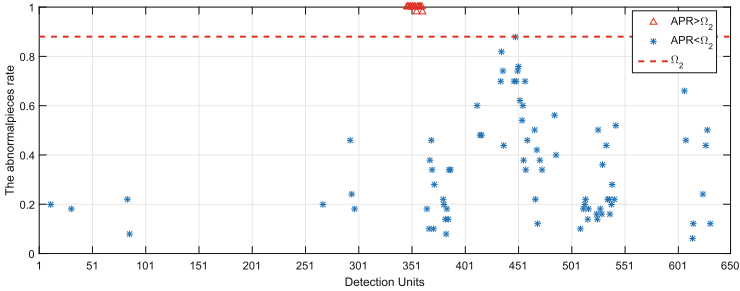


Fig. 7. The results based on the abnormal pieces analysis method.

The TCP traffic of the DUs that are identified to be attacked is shown as Fig. 8, we take following two DUs as examples. In a small time scale, the TCP traffic changes rapidly, the amount of fluctuation of TCP traffic is similar to the traffic that is attacked by LDoS attacks.

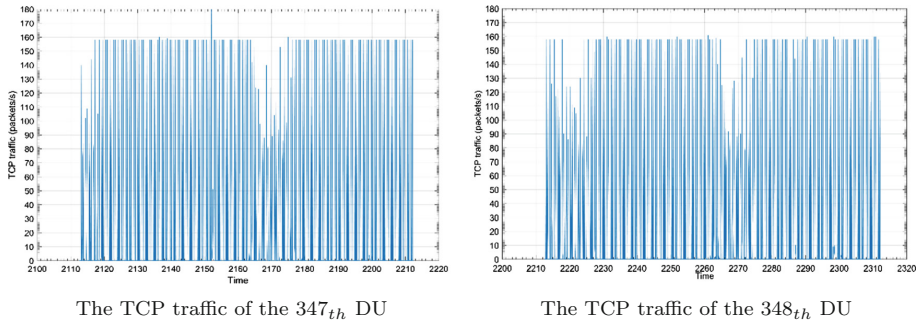


Fig. 8. The TCP traffic of the false positives detection units.

As for the LBNL dataset, the two-step cluster analysis for detecting LDoS attacks has a lower false positive rate than the AEWMA [21] method, the results are shown as Table 4.

Table 4. The comparison results.

Detection method	Number of DUs	The false positives	False positive rate
Two-step cluster analysis	649	16	2.46%
AEWMA	649	22	3.38%

5.3 Experiments on Public Dataset WIDE

The dataset WIDE [7] is a traffic data repository maintained by the MAWI Working Group of the WIDE Project, we conduct experiment on the latest dataset which collected 10 weeks of network traffic in 2018. Monday's network traffic was used as test data and test data was lasted for 150 min. We set the sampling time $st = 0.1s$, the detection unit $DU = 100s$, and the test piece $TP = 2s$, so we got 90 DUs in total, each DU has 50 TPs. The Fig. 9 shows the results of experiment based on the two-step cluster analysis method. There are 14 DUs (cluster 1) which are suspected to be suffered LDoS attacks. Then we analyze the TCP traffic pieces of the 14 DUs. Tuesday's network traffic was used as train data, in which no LDoS attacks occurred. We get the parameters by the train data as: $z = 2.58$, $\Omega_1 = 4540.29$, $\Omega_2 = 0.38234$. The results based on the analyzing pieces of TCP traffic method is shown as Fig. 10. Finally we get 5 false positives and the false positive rate is 5.56%.

The TCP traffic of the false positive unit 21 and the false positive unit 22 is shown as Fig. 11. 21_{th} DU contains TCP traffic of 2000s to 2100s, and 22_{th} DU contains TCP traffic of 2100s to 2200s. Since the TCP traffic in 21_{th} DU

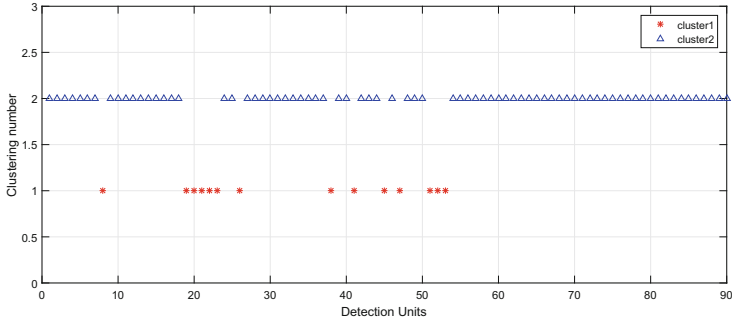


Fig. 9. Cluster results of WIDE dataset.

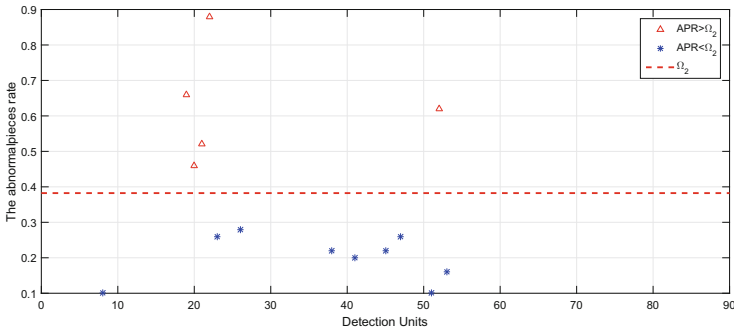
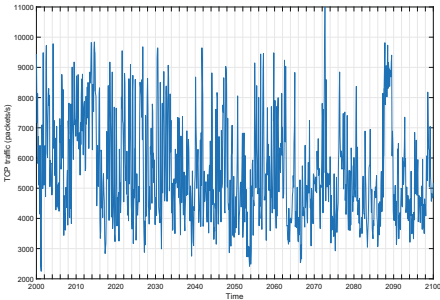
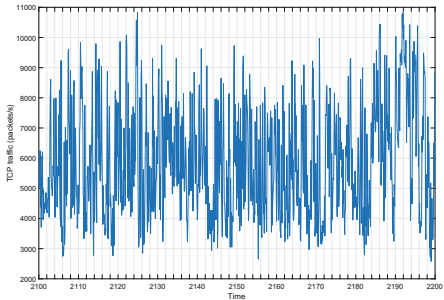


Fig. 10. The results based on the abnormal pieces analysis method.



The TCP traffic of the 21th DU



The TCP traffic of the 22th DU

Fig. 11. The TCP traffic of the false positives detection units.

keeps fluctuating and the TCP traffic is discrete, it was divided into suspicious clusters (cluster 1). The TCP traffic in 21th DU changes rapidly, the amount of fluctuation of TCP traffic is similar to the traffic that is attacked by LDoS attacks. 22th DU is similar to 21th DU, they are both detected as suffered LDoS attacks.

6 Conclusions

We have proposed the two-step cluster analysis method to detect LDoS attacks. According to the discrete distribution of TCP traffic caused by LDoS attacks, three values for measuring dispersion of TCP traffic are calculated to be the input records for the detection method. Then the cluster which is suspected to be suffered LDoS attacks would be detected again by the abnormal pieces analysis method for reducing false positives. The results of experiment on NS2 are proved the effectiveness of the method we proposed, and experiments on public dataset LBNL and dataset WIDE show that the method has a low rate of false positive.

It's hard to find the public dataset that contains LDoS attacks. In the future work, we're going to build the test-bed including LDoS attacks in real network environment for further verification of the methods we proposed.

References

1. Berkhin, P.: A survey of clustering data mining techniques. Group. Multidimension. Data **43**(1), 25–71 (2006)
2. Bijuraj, L.V.: Clustering and its applications. In: Proceedings of National Conference on New Horizons in IT-NCNHIT, pp. 169–172 (2013)
3. Center, I.K.: Two-step cluster analysis. https://www.ibm.com/support/knowledgecenter/SSLVMB_25.0.0/statistics_mainhelp_ddita/spss/base/idh_twostep_main.html. Accessed 10 Aug 2018
4. Chuang, K.S., Tzeng, H.L., Chen, S., Wu, J., Chen, T.J.: Fuzzy c-means clustering with spatial information for image segmentation. Comput. Med. Imaging Graph. **30**(1), 9–15 (2006)
5. Gligor, V.D.: A note on the denial-of-service problem. In: IEEE Symposium on Security and Privacy, p. 139 (1983)
6. Grabmeier, J., Rudolph, A.: Techniques of cluster algorithms in data mining. Data Min. Knowl. Disc. **6**(4), 303–360 (2002)
7. Group, M.W.: Packet traces from wide backbone. <http://mawi.wide.ad.jp/mawi/>. Accessed 10 Aug 2018
8. Guo, Y., Yan, J., Qiu, H., Zhang, L.: A CRF-theory-based method for BGP-LDOS attack detection. In: IEEE International Conference on Computer and Communications, pp. 1071–1075 (2017)
9. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
10. Khan, L., Awad, M., Thuraisingham, B.: A new intrusion detection system using support vector machines and hierarchical clustering. VLDB J. **16**(4), 507–521 (2007)
11. Kuzmanovic, A., Knightly, E.W.: Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2003), pp. 75–86 (2003)
12. Kuzmanovic, A., Knightly, E.W.: Low-rate TCP-targeted denial of service attacks and counter strategies. IEEE/ACM Trans. Netw. **14**(4), 683–696 (2006)

13. LBNL: ICSI enterprise tracing project. <http://www.icir.org/enterprise-tracing>. Accessed 10 Aug 2018
14. Loukas, G., ke, G.: Protection against denial of service attacks: a survey. *Comput. J.* **53**(7), 1020–1037 (2010)
15. Luo, X., Chang, R.K.C.: On a new class of pulsing denial-of-service attacks and the defense. In: *Network and Distributed System Security Symposium, NDSS 2005, San Diego*, pp. 61–79 (2005)
16. Ray, S., Turi, R.H.: Determination of number of clusters in k-means clustering and application in colour image segmentation. In: *Proceeding 4th ICAPRDT*, pp. 137–143 (1999)
17. Ray, S., Turi, R.H.: Intrusion detection with unlabeled data using clustering. In: *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA)*, pp. 1–14 (2001)
18. Sarat, S., Terzis, A.: On the effect of router buffer sizes on low-rate denial of service attacks. In: *Proceedings International Conference on Computer Communications and Networks, 2005 ICCCN 2005*, pp. 281–286 (2005)
19. Sun, H., Lui, J., Yau, D.: Distributed mechanism in detecting and defending against the low-rate TCP attack. *Comput. Netw.* **50**(13), 2312–2330 (2006)
20. Sun, H., Lui, J.C.S., Yau, D.K.Y.: Defending against low-rate TCP attacks: Dynamic detection and protection. In: *IEEE International Conference on Network Protocols*, pp. 196–205 (2004)
21. Tang, D., Chen, K., Chen, X.S., Liu, H.Y., Li, X.: Adaptive EWMA method based on abnormal network traffic for ldos attacks. *Math. Probl. Eng.* **2014**(3), 166–183 (2014)
22. Wu, Z., Zhang, L., Yue, M.: Low-rate Dos attacks detection based on network multifractal. *IEEE Trans. Dependable Secure Comput.* **13**(5), 559–567 (2016)
23. Yu, C., Kai, H.: Collaborative detection and filtering of shrew DDoS attacks using spectral analysis. *J. Parallel Distrib. Comput.* **66**(9), 1137–1151 (2006)
24. Yue, M., Liu, L., Wu, Z., Wang, M.: Identifying LDoS attack traffic based on wavelet energy spectrum and combined neural network. *Int. J. Commun. Syst.* **31**(2), e3449 (2018)

Full Paper Session III: Real-World Cryptography



On the Weakness of Constant Blinding PRNG in Flash Player

Chenyu Wang, Tao Huang^(✉), and Hongjun Wu

Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore, Singapore
cwang014@e.ntu.edu.sg, {huangtao,wuhj}@ntu.edu.sg

Abstract. Constant blinding is considered an effective mitigation against JIT spray attacks. In this paper, we study the design and implementation of constant blinding mechanism in Flash Player and analyse the weakness in its pseudo random number generator (PRNG). We demonstrate how such weakness can be exploited to recover the seed value in PRNG, thus bypass the constant blinding in Flash Player.

We propose two methods to circumvent constant blinding in Flash Player. The first method aims at recovering the seed value using cryptanalysis on the PRNG algorithm, which turns out to provide only 21-bit entropy. The second method focuses on ill-considered implementation of PRNG, which puts obvious signature value next to the seed value and makes it easy for attacker to search. To demonstrate the two methods are both practical, we present proof-of-concept attacks based on existing vulnerability. We have reported the issue to Adobe Flash security team and CVE-2017-3000 is assigned to us. To the best of our knowledge, we are the first to analyse the randomness in constant blinding and integrate cryptanalysis in constant blinding bypass. Furthermore, we implement a prototype tool Constant Blinding Enhancement (ConBE) based on dynamic instrumentation framework to defend against our proposed attacks. In ConBE, we provide a stronger defence than the official patch of Flash Player.

1 Introduction

JIT-spray attack was first proposed by Blazakis [8] in 2010. Through JIT compilation, large constant in client-side script, e.g. JavaScript and ActionScript, will be dynamically emitted into code heap. More specifically, JIT compiler loads the script during runtime and compiles the script code into executable code. Constant number in the script will be emitted as immediate value in generated assembly code. It means that a sequence of multiple bytes, which is under the control of malicious attacker, is embedded into code heap. If the attacker hijacks the control flow to the address where the sequence of multiple bytes is located, these misaligned bytes could be served as short gadgets for attackers to build a complete exploit. In the case of Return Oriented Programming (ROP) attack, those constant numbers can be used as ROP gadgets to construct ROP chain.

Constant blinding mechanism is adopted in JIT compilers to defend against JIT spray attacks. It aims at preventing the constant controlled by attacker appearing in code heap. For this purpose, constant blinding relies on a PRNG to generate a random number. The number serves as a secret cookie to scramble the value of the constant that will be emitted in memory. Therefore, the PRNG plays a significant role in constant blinding. If an attacker can recover the seed value of the PRNG under reasonable time cost, the scrambled value will be predictable and the attacker will gain the ability to put ROP gadgets into code heap via JIT-spray attacks at his own will.

Since the critical role of PRNG in modern security systems, it has been analysed in several previous works. A commonly used method in attacking a PRNG is to lower the entropy used in the PRNG. This can be done by exploiting the weakness in the external entropy source. Kim et al. [14] analysed the OpenSSL PRNG on the Android system. They demonstrated that the lack of entropy at the time of seed initialization will make it vulnerable for attackers to predicate the state of PRNG and recover the secret key of SSL session. Similar work [13] was also done on Linux PRNG. It demonstrated that the low entropy provided by the PRNG will make the device vulnerable to IPv6 fragmentation attack and stack canary bypass. Constant blinding PRNG in Flash Player tries to design an efficient PRNG algorithm itself for providing secret cookie while reducing the performance overhead. Our research shows that the flawed design and ill-considered implementation still make attacker completely recover the seed status and launch JIT-spray attacks.

In this paper, we analyse the design and implementation of the PRNG for constant blinding in Flash Player. We propose a novel attack to lower the entropy of the PRNG. Instead of analysing the external entropy source like the previous work [13, 14], we make cryptanalysis on the PRNG algorithm itself. Due to insufficient confusion and diffusion of the PRNG, we can reduce the entropy in the secret generated cookie to only 21-bit. We can recover the seed value in less than one second. Moreover, we propose another method to recover the seed value in the PRNG in Flash Player. We find that the seed value is stored in heap in the implementation of the PRNG and it is always located next to some constant values. We can use those constant values as signatures to locate the seed value in memory. The search for the seed value only requires $\mathcal{O}(1)$ time, which is fast and stable. Compared with previous work [16, 17] that only searches for some corner cases in constant blinding, we are the first to analyse the PRNG in constant blinding and predict the secret cookie by recovering seed value.

We further propose and implement a dynamic binary instrumentation framework ConBE (Constant Blinding Enhancement) based on PIN to mitigate our attacks on PRNG. We introduce extra entropy in the generation of pseudo number, making cryptanalysis on PRNG impractical. We also move the seed value to an isolated heap, making it hard for attackers to search in memory. Compared with the latest patch of Flash Player, we provide a stronger protection that an attacker cannot recover the seed value even if the attacker gains the ability to arbitrarily read or write in memory. We evaluate our mitigation for Flash Player

on Windows 7 platform and compare the performance overhead with the patched version of Flash Player.

In summary, we make the following three contributions:

- Analyse the design and implementation of PRNG for constant blinding in Flash Player. Propose two feasible methods to circumvent constant blinding based on cryptanalysis and information disclosure respectively.
- Present the proof-of-concept exploit based on existing vulnerability and evaluate its performance overhead.
- Propose a prototype mitigation against our attack based on dynamic binary instrumentation framework.

2 Technical Background

In this section, we first introduce the attacking model including the attacker abilities and the mitigation employed in our target. Then we explain the basic concepts of Return-Oriented-Programming and how attackers chain gadgets together for a successful exploit. Finally we give a general idea of the JIT-spray attack and how constant blinding is supposed to mitigate JIT-spray attack.

2.1 Attacking Model

In our attacking model, we assume that the attacker gains the following abilities.

- The target is under the protection of DEP [3] and ASLR [5], which is enabled at the hardware level and OS level.
- The target is assumed to be protected by code diversification technique, such as G-free, that there exists no available gadget in static code or dynamically generated code.
- The target is vulnerable to memory corruption. Hence, the attacker can read arbitrary value in memory and modify value at given address.

We take Flash Player as our target with DEP and ASLR in place. DEP is a defence against shellcode injection attack. DEP prevents attacker from directly writing shellcode into executable code heap and hijacking the control flow to the shellcode. ASLR is another defence that raises the bar of successful exploitation. Through randomizing the base address of loaded module and mapped memory, attacker has to exploit information exposure vulnerability to disclose critical information before hijacking control flow. We will discuss more academic work on protection for JIT code in Sect. 6.2.

The second one is a common practice for developing stable exploits today. Attackers usually exploit various kinds of vulnerabilities in target [21], including use-after-free, type confusion, etc. Recent exploits [1, 2] on Flash Player show that ActionScript provides attacker chances to trigger vulnerability, leak critical information in memory. Therefore, ASLR is not a big concern in our attacking model.

```

script code:
fun() {
    a = 0xc3585a59;
    a = 0xdeadbeef;
}

JIT code without constant blinding:
mov     ecx, 0xc3585a59
mov     dword ptr [edx+0x14], ecx
mov     ecx, 0xdeadbeef
mov     dword ptr [edx+0x14], ecx

JIT code with constant blinding:
mov     ecx, 0xccf1d312
xor     ecx, 0xfa9894b
mov     dword ptr [edx+0x14], ecx
mov     ecx, 0xd10437a4
xor     ecx, 0xfa9894b
mov     dword ptr [edx+0x14], ecx

```

Listing 1.1. Constant Blinding in JIT Code of Flash Player

2.2 Return Oriented Programming

Since executing shellcode in non-executable memory is forbidden due to the existence of DEP, attackers turned their attention to code reuse attacks, i.e. using the existing code in static libraries or dynamically generated code to launch attacks. ROP attack is a typical kind of code reuse attacks.

ROP gadget refers to a short sequence of assembly instructions that end with *ret*. In ROP attack [20,24], attacker usually has to prepare addresses of gadgets in stack and chains those ROP gadgets via *ret* instruction. The ROP gadgets are supposed to achieve various kinds of operations, e.g. (1) load value from stack into register, (2) arithmetic operation, (3) store value into memory, (4) load value from memory into register, etc. Attackers can combine different types of the gadgets to launch the attack.

2.3 JIT Spray Attacks and Constant Blinding

In 2005, Abadi et.al. proposed Control Flow Integrity (CFI) [6] enforcement to defend against ROP attack. Now there are various implementations of CFI introduced in academia and industry. For example, G-Free [19] is designed to eliminate available gadgets in binary. All those mitigations try to reduce the available gadgets for code reuse attacks in static code. However, JIT spray attack uses the dynamically generated code to provide ROP gadgets. In another word, JIT spray attack is out of the protection scope of those mitigations. In the work of Blazakis [8] and Sintsov [22], constant numbers in JIT-generated code provides a 4-byte long gadget for attacker in executable code heap.

In JIT Spray attacks, the constant in script can be used as a short gadget for further exploitation. Constant number `0xc3585a59` in Listing 1.1 will be emitted into code heap by JIT compiler as an immediate value in instruction `mov ecx, 0xc3585a59`. The byte sequence of this instruction will be `0xb9 0x59 0x5a 0x58 0xc3` in memory on a little-endian system. Suppose the sequence of bytes is located at `0x602010`. If attacker hijacks the control flow to `0x602011`. The misaligned byte sequence will be interpreted as `pop ecx (0x59)`, `pop edx (0x5a)`, `pop eax (0x58)` and `ret (0xc3)` from the view of assembly code and used as an ROP gadget.

Constant blinding was proposed to mitigate JIT-spray attack in JITSafe [9] and has already been deployed in the script engine that supports JIT compilation. To be more specific, it prevents the value of a constant number appearing in memory. The most common solution is to generate a secret cookie and XOR the constant number with the secret cookie to get a new number. At the moment of emitting JIT code, the new number will be moved into a register first. The register will be then XORed with the secret cookie to restore the original number. We give an example in Listing 1.1 to demonstrate how constant blinding works. The constant number, `0xc3585a59` is blinded by a secret cookie `0xfa9894b`. In the following of the paper, we call the generated random number as secret cookie and call the value, which results from XOR operation on original value and secret cookie, as blinded constant.

For a user-defined function, the JIT code of the function is generated at the first time the function is being called [4]. The JIT compiler will check if JIT code of this function exists. If code has not been generated yet, the JIT compiler will generate the code first and execute the generated code. Otherwise, the JIT code will be executed directly. Before JIT compilation, constant values in script are stored in a symbol table located in memory region with readable and writeable permission. We will discuss how to exploit the JIT compilation process to launch our attack later in Sect. 3.4.

3 Attacks on Constant Blinding in Flash Player

As described above, security of constant blinding heavily relies on the secret cookie generated. If the secret cookie is predictable for attacker, the defence against JIT spray attack will be weakened. In Sect. 3.1, we will discuss the implementation details of constant blinding PRNG in Flash Player. Next, we will demonstrate how we recover the seed value with two different methods in less than 1s. The first method is to apply cryptanalysis on the hash function. We will show that the seed value of PRNG algorithm can be recovered at $\mathcal{O}(2^{21})$ time complexity in Sect. 3.2. The second method is to search the seed value directly. Different from searching the whole memory space to locate the seed value, we find the seed value in memory at $\mathcal{O}(1)$ time cost based on pointer redirection in Sect. 3.3.

3.1 Constant Blinding in Flash Player

The PRNG contains three components as shown in Listing 1.2: a seed initialization function (*RandomFastInit*), a hash function (*RandomPureHasher*) and a generator function (*GenerateRandomNumber*) to generate the final secret cookie for constant blinding. Another function (*TandomFastNext*) generates a number based on the seed value and updates its value, but it does not change the entropy in the seed value.

```

void RandomFastInit(pTRandomFast pRandomFast) {
    pRandomFast->uValue = (int)(getPerformanceCounter());
    pRandomFast->uSequenceLength = 0x7fffffff;
    pRandomFast->uXorMask = 0x14000000;
}

int RandomPureHasher(int iSeed) {
    int iResult;
    iSeed = ((iSeed<<13)⊕iSeed)-(iSeed>>21);
    iResult = (iSeed*(iSeed*iSeed*c3 +c2) + c1);
    iResult = iResult & kRandomPureMax;
    iResult = iResult+iSeed;
    iResult = ((iResult<<13)⊕iResult)-(iResult>>21);
    return iResult;
}

int GenerateRandomNumber(pTRandomFast pRandomFast) {
    if(pRandomFast->uValue == 0){
        RandomFastInit(pRandomFast);
    }
    long aNum = RandomFastNext(pRandomFast);
    aNum = RandomPureHasher(aNum * 71L);
    return aNum & kRandomPureMax;
}

```

Listing 1.2. Constant Number Generation Process

In initialization function, the seed value (*uValue*) is initialized by *QueryPerformanceCounter* (Windows API). The hash function takes the seed value as input and generates a hash value. The new value will be ANDed with *kRandomPureMax* (0x7fffffff) in the generator function to produce the final cookie.

In hash function, the variables *c1, c2, c3* are three constant numbers. This hash function adds no extra entropy into the generated number but aims to make it hard for attacker to reverse the seed value. Attacker's goal is to retrieve the seed value, predicate the secret cookie generated in next round and embed the desired value in the executable code heap. Though reversing the seed value via brute force seems feasible, it is impossible in practice because the default running timeout in Flash Player is 15s. Attacker must recover the seed value in less than 15s while brute forcing requires a few minutes on average according to our test.

Algorithm 1. Reverse Polynomial Function

- 1: **Function name:** Reverse
 - 2: **Input:** V : the value to reverse
 - 3: **Output:** S : the set of candidate value
 - 4: $S = \emptyset$
 - 5: target_bit = 0
 - 6: curSolution = 0
 - 7: reverseBit(V , target_bit, curSolution, S)
 - 8: **return** S
-

$$\begin{aligned}
 & a_{01}a_{02}a_{03}a_{04}a_{05}a_{06}a_{07}a_{08}a_{09}a_{10}a_{11}a_{12}a_{13}a_{14}a_{15}a_{16}a_{17}a_{18}a_{19}a_{20}a_{21}a_{22}a_{23}a_{24}a_{25}a_{26}a_{27}a_{28}a_{29}a_{30}a_{31}a_{32} \\
 \oplus & a_{14}a_{15}a_{16}a_{17}a_{18}a_{19}a_{20}a_{21}a_{22}a_{23}a_{24}a_{25}a_{26}a_{27}a_{28}a_{29}a_{30}a_{31}a_{32} 0 0 0 0 0 0 0 0 0 0 0 0 \\
 = & t_{01}t_{02}t_{03}t_{04}t_{05}t_{06}t_{07}t_{08}t_{09}t_{10}t_{11}t_{12}t_{13}t_{14}t_{15}t_{16}t_{17}t_{18}t_{19}t_{20}t_{21}t_{22}t_{23}t_{24}t_{25}t_{26}t_{27}t_{28}t_{29}t_{30}t_{31}t_{32} \\
 + & 0 a_{01}a_{02}a_{03}a_{04}a_{05}a_{06}a_{07}a_{08}a_{09}a_{10}a_{11}
 \end{aligned}$$

Fig. 1. Reorganized hash function bit by bit

3.2 Attack Based On Cryptanalysis

Hash function is supposed to make attacker unable to reverse the seed value under reasonable time cost. However, we find that the hash function used for constant blinding in Flash Player is insufficient in confusion and diffusion, such that the attacker still can get the seed value in short time. The hash function in Listing 1.2 can be simplified into two functions. The first one is a bit manipulation function $f(n)$, and the other is a third-degree polynomial function $g(n)$. These two functions can be generalised as following equations:

$$\begin{aligned}
 f(n) &= ((n \ll 13) \oplus n) - (n \gg 21) && (1) \\
 g(n) &= (c_3 * n^3 + c_2 * n + c_1) \& 0x7fffffff + n && (2)
 \end{aligned}$$

For the polynomial function, we can easily reverse the input value. The simplified algorithm is given in Algorithm 1. The reverse algorithm starts to scan the target value from the least significant bit (Line 7). Then, we apply a backtracking Algorithm 2 to reverse the input value bit by bit. We guess one bit and use the polynomial function to verify (Line 7 and 14) if the result matches the first i bits of target value.

For the bit manipulation function, we design Algorithm 3 to reduce the time complexity to $\mathcal{O}(2^{21})$, which is more efficient compared with $\mathcal{O}(2^{32})$ of brute-force method. To recover the seed value of bit manipulation function, we express the seed value in a 32-digit string as $a_1a_2a_3\dots a_{31}a_{32}$ and express target value as $t_1t_2t_3\dots t_{31}t_{32}$. We reorganize the bit manipulation function $f(n)$ and present the process of bit manipulation function in Fig. 1. According to the generalized equation, the seed value is divided into three parts. The higher part is $a_{01}\dots a_{11}$, the middle part is $a_{12}\dots a_{21}$ and the lower part is $a_{22}\dots a_{32}$. Since target value is known to us, attack on PRNG can be divided into two steps. We can recover the value of higher part and lower part first and then reverse the possible value

Algorithm 2. Backtrack Algorithm to Reverse Polynomial Function bit by bit

```

1: Function name: reverseBit
2: Input:  $V$ : the value to reverse,  $i$ : the  $i$ -th bit to check,  $sol$ : current solution,  $S$ : the
   set to store candidate value
3: if  $i == 32$  then
4:    $S.add(sol)$ ;
5:   return
6: end if
7: if  $verify\_ith\_bit(V, i, 0)$  then
8:    $sol = sol + 0 \ll i$ 
9:    $i = i + 1$ 
10:   $reverseBit(V, i, S, sol)$ 
11:   $i = i - 1$ 
12:   $sol = sol - 0 \ll i$ 
13: end if
14: if  $verify\_ith\_bit(V, i, 1)$  then
15:    $sol = sol + 1 \ll i$ 
16:    $i = i + 1$ 
17:    $reverseBit(V, i, S, sol)$ 
18:    $i = i - 1$ 
19:    $sol = sol - 1 \ll i$ 
20: end if
21: return

```

of middle part. For higher part and lower part, we iterate over possible values of the lower part (Line 5) and reverse the value of higher part through value in $t_{22} \dots t_{32}$ (Line 6–11). For middle part, we iterate over possible values of the middle part (Line 12) and use the known value of $t_1 \dots t_{21}$ to verify if the value is valid (Line 14–17). Therefore, the time complexity of the whole process will be $\mathcal{O}(2^{11}) * \mathcal{O}(2^{10}) = \mathcal{O}(2^{21})$.

3.3 Attack Based on Memory Disclosure

Besides the attack based on cryptanalysis, the implementation of the PRNG ignores the possibility that the attacker can locate the seed value in memory through searching memory space via info disclosure vulnerability.

To find the seed value quickly and stably, we have to solve two challenges during our search in memory. The first challenge is how to assure that the accessed value is the seed value we are searching for. The second challenge is to get the memory address of the seed value in the heap.

For the first challenge, the solution is straightforward. In initialization function of the PRNG, the seed value is initialized together with `sequenceLength` and `XorMask`. However, the values of those two variables are constant numbers and these two values are located in memory adjacent to the seed value. These two variables can be used as signature values for attacker to locate the seed value.

For the second challenge, the difficulty comes from the fact that the seed value and the victim *Vector* object are located in two different heaps. Usually,

Algorithm 3. Reverse algorithm for the polynomial function

```

1: Function name: reverseHash
2: Input:  $V$ : the value to reverse
3: Output:  $S$ : the set of candidate value
4:  $S = \emptyset$ 
5: for  $lowPart \in [0, 0x7ff]$  do
6:    $tmpPart = V \& 0x7ff$ 
7:   if  $tmpPart > lowPart$  then
8:      $highPart = (1 \ll 11) + lowPart - tmpPart$ 
9:   else if  $tmpPart \leq lowPart$  then
10:     $highPart = lowPart - tmpPart$ 
11:   end if
12:   for  $midPart \in [0, 0x3ff]$  do
13:      $value = highPart \ll 21 + midPart \ll 11 + lowPart$ 
14:      $t = (value \ll 13) \oplus value - (value \gg 21)$ 
15:     if  $t == V$  then
16:        $S.add(value)$ ;
17:     end if
18:   end for
19: end for

```

```

function fun1 () { a = 0x41414141; }
function fun2 () { c = 0x43434343; }
fun1 ();
fun2 ();

```

Listing 1.3. Benign code for demonstrate how blinded constant are generated

a successful exploit on Flash Player relies on corrupting the metadata *length* in a *Vector* object, which enables attacker to arbitrarily read memory out of the bound of original buffer. Attempt to read value at given address, which is not readable or unmapped, will cause access violation. To avoid unnecessary crash in our exploit, we have to figure out a reliable and quick way to locate the heap where the seed value is located. Our solution is to create an anchor object and save the reference to anchor object in the same heap as victim vector. The anchor object contains a reference to another object which is located in the same heap as the seed value. Through chains of pointer indirection, we can locate the heap storing the seed value. Since size of the heap is a constant number, time complexity to search for the seed value is $\mathcal{O}(1)$.

3.4 Full Exploit Generation

In our exploit, we call a function as constant releasing function if its generated code contains a blinded constant. In Listing 1.3, we list two simple constant releasing functions and invoke them one after another in script. From the view of constant blinding, the steps to emit code can be separated into a few steps.

```

function fun1 () { a = 0x41414141; }
function fun2 () { c = 0x42424242; }
function fun3 () { c = 0x43434343; }
fun1 ();
fun2 ();
evilCode ();
fun3 ();

```

Listing 1.4. Malicious code to retrieve seed value and embed desired value in code heap

1. Generate JIT code for fun1
 - Call *GenerateRandomNumber* and get Key1
 - Retrieve 0x41414141 from symbol table
 - Generate JIT code: `mov ecx, 0x41414141 \oplus Key1`
 - Generate JIT code: `xor ecx, Key1`
2. Generate JIT code for fun2
 - Call *GenerateRandomNumber* and get Key2
 - Retrieve 0x43434343 from symbol table
 - Generate JIT code: `mov ecx, 0x43434343 \oplus Key2`
 - Generate JIT code: `xor ecx, Key2`

As discussed in Sect. 2.3, constants used in the script are stored in a symbol table. Since the constant values in this table are not blinded. We can locate the address of the table by inserting some magic constant in script. If we can predicate the secret cookie in next round, we can modify the value in the table and embed desired 4-byte long value in code heap in the end.

In our final exploit, we insert malicious code before generating JIT code of *fun3*. In malicious code, we trigger the vulnerability, search in memory and recover the seed value via given methods. To embed blinded constants in memory, we need to call multiple constant releasing functions before running our malicious code in Listing 1.4. Since the hash function is a many-to-one function, we may get multiple possible seed values if we are given only one secret cookie in the process of reversing the hash function. Leaking multiple secret cookies can help us find the real seed value. The necessary steps can be generalized:

1. Generate JIT code for fun1
 - Call *GenerateRandomNumber* and get Key1
 - Retrieve 0x41414141 from symbol table
 - Emit binary code: `mov ecx, 0x41414141 \oplus Key1`
 - Emit binary code: `xor ecx, Key1`
2. Generate JIT code for fun2
 - Call *GenerateRandomNumber* and get Key2
 - Retrieve 0x42424242 from symbol table
 - Emit binary code: `mov ecx, 0x42424242 \oplus Key2`
 - Emit binary code: `xor ecx, Key2`

3. evilCode: search and modify memory under attacker's control
 - Search memory to find Key1 and Key2
 - Recover seed value, and predict the value of Key3
 - Locate 0x43434343 in symbol table
 - Modify 0x43434343 to be (TargetValue \oplus Key3)
4. Generate JIT code for fun3
 - Call *GenerateRandomNumber* and get Key3
 - Retrieve (TargetValue \oplus Key3) from symbol table
 - Emit binary code: mov ecx, TargetValue (TargetValue \oplus Key3 \oplus Key3)
 - Emit binary code: xor ecx, Key3

4 Design and Implementation of ConBE

To mitigate our attacks, we propose Constant Blinding Enhancement (ConBE) based on dynamic binary instrumentation as a prototype to demonstrate our mitigation strategy. Adobe has patched this vulnerabilities in the latest version of Flash Player (25.0.0.127). We build this tool to provide stronger protection for all versions of Flash Player. In this section, we will discuss our mitigation strategy and implementation in detail.

4.1 Mitigation Principles

To mitigate the attack based on cryptanalysis, we make it harder for attacker to reverse the hash function. At present, we rely on well-documented hash function to achieve this. To protect seed value from information disclosure, our solution is to separate the seed value from those signature values and store the seed value in an isolated heap. Similar to partitioned heap, we invoke *VirtualAlloc* function to create a new data heap to store the seed value. It means that except for one global pointer referencing the seed value in the heap there will be no other data pointer in memory referencing any values in the data heap. At present, our solution relies on ASLR provided by OS to randomize the heap location.

4.2 Implementation Details

We build ConBE based on PIN tool [15], a dynamic binary instrumentation framework. As a prototype, we use ConBE to test the effectiveness of our mitigation strategy and provide protection for older versions of Flash Player.

To mitigate cryptanalysis, we insert an MD5 hash function at the end of identified hash function. In particular, we take the value in *eax* as input value and calculate its MD5 hash value. The output of the MD5 hash function is a 128-bit value. We pick the least significant 32 bits as its return value.

To thwart information disclosure attack, we have to achieve two goals. The first one is to separate the seed value from those signature values. The second one is to hide the seed value in memory. Different from X64 platform, X86 system does not have a free segment register to save the value in an isolated memory

segment. To implement both goals, our solution is to newly allocate a heap and save the seed value in the heap. We rely on ASLR provided by system to randomize address of the memory. Under our protection, attacker has to search through the whole memory. It takes too much time and exceeds the default timeout of ActionScript.

4.3 Advantages of ConBE

The latest PRNG in Flash Player relies on Windows API *CryptGenRandom* to generate the random number. To be more specific, the PRNG maintains a secret buffer of 256 bytes long and invokes the API once to fill the secret buffer with random values. As a result, $256/4=64$ secret cookies will be available and stored in the secret buffer for future use. For each user-defined function, JIT compiler traverses the secret buffer and pick an unused 4-byte value as secret cookie.

Once all the 64 secret cookies have been used once, *CryptGenRandom* will be called again to generate a new secret buffer. Partitioned heap in Flash Player is responsible for assuring that the secret buffer will not be located in the same heap as victim object is located. Allocating a 256-byte long buffer for storing secret cookie is a trade-off between performance and security. In Sect. 5.2, we demonstrate the performance overhead of putting the seed value in a separate heap is pretty high. Generating 64 secret cookies in a row rather than invoke *CryptGenRandom* for each user-defined function can improve performance overhead. Since the seed values are still stored in data segment, dedicated attackers may still locate the seed value with generic information disclosure technique. On the contrast, ConBE could be customized on X64 system to store the only pointer that references the seed value in extra segment rather than data segment to make it harder for attacker to search in memory space.

5 Evaluation

In Sect. 5.1, we will give a full exploit to recover the seed value of PRNG based on one existing CVE in Flash Player. Then we evaluate the execution time of our attack. Then we evaluate the performance overhead of our proposed prototype mitigation. In Sect. 5.2, we evaluate the performance overhead of ConBE and compared that with the patched Flash Player. The performance evaluation is tested in a Windows 7 running on Intel Xeon E5-2630 processor with 4GB RAM.

5.1 Evaluation on Full Exploit

To show the effectiveness of our attack, we present a proof-of-concept exploit based on CVE-2015-5119 [1], a use-after-free vulnerability in Flash Player 18.0.0.206¹.

In the exploit of CVE-2015-5119, an attacker can corrupt the metadata length of a *Vector* object and utilizes the extended vector to gain arbitrary

¹ The weak PRNG exists from version 2 to version 24.0.0.221 of Flash Player.

read/write ability in memory. As discussed in Sect. 3.3, we use a *ByteArray* object as an anchor to leak the address of the heap where the seed value is stored. More specifically, there exists a reference to our target heap at the offset 0×30 in *ByteArray* object. We first discover the address of a *ByteArray* object, read the reference to locate the target heap and then search through memory for the signature value to locate the seed value.

Table 1 demonstrates the time needed to retrieve the secret value compared to a normal execution. We use internal function of ActionScript *Date.getMilliseconds* and *Date.getSeconds* to get the execution time. In both normal execution and poc exploit, we pick the time when we corrupt the meta-data of victim vector as starting time. For normal execution, we take the time when all blinded constants are released as ending time. For poc exploit, we take the time when the exploit finds all necessary ROP gadgets as ending time. From the table, we can see that the Information Disclosure method and Cryptanalysis method induce 0.033s and 0.266s overhead respectively comparing to the normal execution. From the view of an exploit, such performance overhead is tolerable and will not influence much on the exploit performance.

5.2 Evaluation on ConBE

We have successfully applied our tool in Flash Player 18.0.0.203. To evaluate the effectiveness of ConBE, we have to answer two questions: (Q1) What is the performance overhead induced by ConBE? (Q2) What is the difference on performance between ConBE and the officially patched version of Flash Player?

To answer Q1, we have to compare the performance overhead between ConBE patched one and the original unpatched one. To answer Q2, we need to compare the performance overhead between ConBE patched one and the latest officially patched one. For the ConBE patched one and the original unpatched one, we calculate the time between fetching seed value from memory and generating final secret cookie. Because the patched version of Flash Player uses the Windows API *CryptGenRandom* to generate secret value, we calculate the execution time of the API call.

Since internal function *Date.getMilliseconds* of ActionScript only provides millisecond resolution, we can tell no difference in the time latency using those internal functions. Instead, we use PIN to insert instructions to log the CPU cycles used to generate the secret value. To be more specific, we instrument *rdtsc* instruction to log the CPU cycles.

Table 1. Time required to retrieve the secret value

Tested mode	Required time
Normal	1.732
Information disclosure	1.765
Cryptanalysis	1.998

Table 2. CPU cycles required to generate secret number

Flash player version	CPU cycles
18.0.0.203	1113660
18.0.0.203 (ConBE)	4151108
25.0.0.127	3155080

Table 2 demonstrates the CPU cycles to generate secret number. Time to generate the secret cookie in officially patched one is about 2.8 times that in the original one. On the contrary, ConBE is about 3.7 times that of the original one. We think the time delay is resulted from the following two reasons. First, we use an MD5 hash function and allocate a separate heap to save seed value in our implementation. But for the patched version, it relies on existing partitioned heap to store the generated secret cookie. The secret cookie is stored in a heap together with some security-unrelated data. This process in ConBE is more complicated than the patched one. Second, our security enforcement is built on PIN, a dynamic instrumentation framework. Similar works [10, 11, 25] built on PIN have shown that this framework usually induced a high performance overhead.

6 Related Work

6.1 Other Techniques Bypassing Constant Blinding

Athanasakis et al. [7] focuses on the fact that constant blinding does not blind constant whose length is less than 2 bytes. Therefore the attacker is still able to construct enough useful ROP gadgets for a successful exploit under 64-bit system. However, our work demonstrates that the attacker is able to allocate 4-byte long ROP gadget in memory, which provides more flexibility and availability of ROP gadgets. Moreover, our attack works on both 32-bit and 64-bit system.

In the work of Maisuradze et al. [16], the author proposed that the relative jump offset can also be used to allocate desired 2-byte or 3-byte ROP gadget in memory. However, due to existence of NOP sledding adopted by most JIT rendering engine, this attack needs to validate the emitted gadget before chaining them as ROP gadgets. On the contrary, our attack on constant blinding does not require the process of validation and ensure that the desired ROP gadgets will always be emitted in memory.

Dachshund [17] finds some corner cases, which was not covered by constant blinding, to emit desired target value in JIT code. In Chakra, the script engine of Microsoft Edge, it is discovered that the constant number in function argument is not blinded by secret value. In Chrome, There also exist a few corner cases where constant blinding does not take place. To the best of our knowledge, we are the first to analyse the PRNG in constant blinding and predict the secret cookie for bypassing constant blinding. Moreover, flash player is still supported on all main stream browsers nowadays, our work provides a more general and stable attacking vector to emit executable gadget in memory.

6.2 Protection on JIT Code

INSert [26], RIM [27] and JITSafe [9] all propose techniques to defend against JIT-Spray attack. To prevent immediate value from being emitted in code heap, they all rely on a secret value to obfuscate the immediate value. The security of these mitigation is based on the assumption that the secret value is not predictable to attacker. However, our work shows that the assumption is not guaranteed in its implementation of Flash Player. Rock-JIT [18] implements a CFI enforcement on JIT compilation. For JIT compiler, Rock-JIT builds the Control Flow Graph (CFG) based on existing code and enforces a fine-grained CFI. For JITted code, Rock-JIT adopts a coarse-grained CFI. Rock-JIT maintains a set to remember all starting instruction of all JITted code and insert check to verify the validity of target before indirect jump. Rock-JIT is built on source code of Chrome V8 script engine and is hard to evaluate its effectiveness in Flash Player, which does not make its source code public. DSCG [23] aims at preventing possible attacks that exploits the code cache in Chrome V8 engine is readable, writable and executable. However, Flash Player has taken such attack into consideration. The permission of code heap is writable and readable at the time of emitting JIT code, while the permission of code heap is executable and readable at the time of executing JIT code. The short attacking window makes it less likely for attacker to launch such exploit. However, our work shows that even if the code heap is enforced with $W \oplus E$, attacker is still able to emit some desired gadgets in code heap.

7 Conclusion

In this paper, we demonstrate that the secret cookie for constant blinding in Flash Player is predictable due to its weak design and implementation of PRNG. We proposed two different methods to recover the seed value used in the PRNG. One is cryptanalysis based on the insufficient confusion and diffusion in hash function. Another one is the ill-considered implementation to store the seed value in memory. It enables attacker to find the seed value without much effort. Different from previous works that concentrate on some corner cases in constant blinding mechanism, we are the first to analyse the constant blinding PRNG and recover the seed value for defeating it. We further propose ConBE and demonstrate its effectiveness against our attack.

References

1. CVE-2015-5119. ByteArray Use-After-Free. <https://www.exploit-db.com/exploits/37523/>. Accessed 06 July 2018
2. CVE-2015-5122. opaqueBackground Use-After-Free. <https://www.exploit-db.com/exploits/37599/>. Accessed 06 July 2018
3. Data execution prevention. [https://msdn.microsoft.com/en-us/library/windows/desktop/aa366553\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa366553(v=vs.85).aspx). Accessed 06 July 2018

4. Inside AVM. https://recon.cx/2012/schedule/attachments/43_Inside_AVM_REcon2012.pdf. Accessed 06 July 2018
5. Windows software security defense. <https://msdn.microsoft.com/en-us/library/bb430720.aspx>. Accessed 06 July 2018
6. Abadi, M., Budi, M., Erlingsson, U., and Ligatti, J.: Control-flow integrity. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 340–353. ACM (2005)
7. Athanasakis, M., Athanasopoulos, E., Polychronakis, M., Portokalidis, G., Ioannidis, S.: The devil is in the constants: Bypassing defenses in browser JIT engines. In: 13th Conference on Network and Distributed System Security Symposium (NDSS) (2015)
8. Blazakis, D.: Interpreter exploitation. In: 4th USENIX Workshop on Offensive Technologies (WOOT) (2010)
9. Chen, P., Wu, R., Mao, B.: JITSafe: a framework against just-in-time spraying attacks. *IET Inf. Secur.* **7**(4), 283–292 (2013)
10. Davi, L., Sadeghi, A.-R., Winandy, M.: ROPdefender: a detection tool to defend against return-oriented programming attacks. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 40–51. ACM (2011)
11. Follner, A., Bodden, E.: ROPocodynamic mitigation of code-reuse attacks. *J. Inf. Secur. Appl.* **29**, 16–26 (2016)
12. Gawlik, R., Kollenda, B., Koppe, P., Garmany, B., Holz, T.: Enabling client-side crash-resistance to overcome diversification and information hiding. In: 14th Conference on Network and Distributed System Security Symposium (NDSS) (2016)
13. Kaplan, D., Kedmi, S., Hay, R., Dayan, A.: Attacking the Linux PRNG on android: weaknesses in seeding of entropic pools and low boot-time entropy. In: WOOT (2014)
14. Kim, S.H., Han, D., Lee, D.H.: Predictability of android openssl’s pseudo random number generator. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, pp. 659–668. ACM (2013)
15. Luk, C.-K., et al.: Pin: building customized program analysis tools with dynamic instrumentation. In: ACM SIGPLAN Notices, vol. 40, pp. 190–200. ACM (2005)
16. Maisuradze, G., Backes, M., Rossow, C.: What Cannot be Read, cannot be leveraged? Revisiting assumptions of JIT-ROP defenses. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 139–156 (2016)
17. Maisuradze, G., Backes, M., Rossow, C.: Dachshund: digging for and securing against (non-) blinded constants in JIT code. In: 15th Conference on Network and Distributed System Security Symposium (NDSS) (2017)
18. Niu, B., Tan, G.: RockJIT: securing just-in-time compilation using modular control-flow integrity. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1317–1328. ACM (2014)
19. Onarlioglu, K., Bilge, L., Lanzi, A., Balzarotti, D., Kirida, E.: G-free: defeating return-oriented programming through gadget-less binaries. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 49–58. ACM (2010)
20. Prandini, M., Ramilli, M.: Return-oriented programming. *IEEE Symp. Secur. Priv.* **10**(6), 84–87 (2012)
21. Serna, F.J.: The info leak era on software exploitation. https://media.blackhat.com/bh-us-12/Briefings/Serna/BH_US_12_Serna_Leak_Era_Slides.pdf. Accessed 6 Oct 2018

22. Sintsov, A.: JIT-spray attacks & advanced shellcode. <https://conference.hitb.org/hitbsecconf2010ams/materials/D1T2%20-%20Alexey%20Sintsov%20-%20JIT%20Spray%20Attacks%20and%20Advanced%20Shellcode.pdf>. Accessed 6 Oct 2018
23. Song, C., Zhang, C., Wang, T., Lee, W., Melski, D.: Exploiting and protecting dynamic code generation. In: 13th Conference on Network and Distributed System Security Symposium (NDSS) (2015)
24. Tran, M., Etheridge, M., Bletsch, T., Jiang, X., Freeh, V., Ning, P.: On the expressiveness of return-into-LIBC attacks. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 121–141. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23644-0_7
25. Uh, G.-R., Cohn, R., Yadavalli, B., Peri, R., Ayyagari, R.: Analyzing dynamic binary instrumentation overhead. In: WBIA Workshop at ASPLOS, Citeseer (2006)
26. Wei, T., Wang, T., Duan, L., Luo, J.: Insert: protect dynamic code generation against spraying. In: 2011 International Conference on Information Science and Technology (ICIST), pp. 323–328. IEEE (2011)
27. Wu, R., Chen, P., Mao, B., Xie, L. RIM: a method to defend from JIT spraying attack. In: 2012 Seventh International Conference on Availability, Reliability and Security (ARES), pp. 143–148. IEEE (2012)



Random Number Generators Can Be Fooled to Behave Badly

George Teșeleanu^{1,2}(✉)

¹ Advanced Technologies Institute, 10 Dinu Vintilă, Bucharest, Romania
ati@dcti.ro

² Department of Computer Science, A.I.Cuza University of Iași,
700506 Iași, Romania
george.teseleanu@info.uaic.ro

Abstract. In this paper, we extend the work on purely mathematical Trojan horses initially presented in [15]. This kind of mechanism affects the statistical properties of an infected random number generator (RNG) by making it very sensitive to input entropy. Thereby, when inputs have the correct distribution the Trojan has no effect, but when the distribution becomes biased the Trojan worsens it. Besides its obvious malicious usage, this mechanism can also be applied to devise lightweight health tests for RNGs. Currently, RNG designs are required to implement an early detection mechanism for entropy failure, and this class of Trojan horses is perfect for this job.

1 Introduction

In [15] the authors propose an interesting mechanism that blurs the line between what constitutes a Trojan horse and what does not. To detect their mechanism, a program needs to somehow differentiate between a naturally unstable random number generator (RNG) and artificially unstable one (obtained by means of certain mathematical transformations). To our knowledge, [15] is the only previous work that discusses this topic.

More precisely, in [15] a digital filter is described. Usually, digital filters are applied to RNGs to correct biases¹, but this filter has an opposite purpose. When applied to a stream of unbiased bits the filter is benign. On the other hand, if applied to a stream of biased bits the filter amplifies their bias. Thereby, making the RNG worse.

In this paper we extend the filter from [15]², provide a new class of filters and discuss some new possible applications. The main application we propose for these filters is RNG testing (*e.g.*, boosting health tests implemented in a RNG). Recent standards [11, 13] require a RNG to detect failures and one such method for early detection can be to apply an amplifier and then do some lightweight

¹ They are called randomness extractors [8].

² The filter presented in [15] corresponds to the greedy amplifier with parameter $n = 3$ described in Sect. 3.

testing³. Based on the results obtained in our paper, we provide concrete examples of how to detect such failures in Sect. 5.1 and Appendix A.

Due to recent events [4, 6, 7, 12] RNGs have been under a lot of scrutiny. Thus, wondering what kind of mechanisms can be implemented by a malicious third party in order to weaken or destabilize a system becomes natural. Amplifying filters provide a novel example of how one can achieve this. Based on the failure detection mechanisms proposed in Sect. 5.1, we show, for example, how a manufacturer can manipulate the architecture to become malicious.

Structure of the paper. Notations and definitions are presented in Sect. 2. The core of the paper consists of Sects. 3 and 4 and contains two classes of filters. Applications are given in Sect. 5. We conclude in Sect. 6. Experimental results are presented in Appendix A.

2 Preliminaries

Throughout the paper, we consider binary strings of length n composed of *independent and identically distributed* (i.i.d.) bits generated by a RNG. By 0^n and 1^n we understand the all zero and the all one strings. Also, for figures we use the standard representation of the x-axis and y-axis.

Let $0 \leq \varepsilon \leq \frac{1}{2}$ be a real number and b a random bit. Then, without loss of generality, we denote the probability of $b = 0$ by $P_0 = \frac{1}{2} - \varepsilon$ and of $b = 1$ by $P_1 = \frac{1}{2} + \varepsilon$. We will refer to ε as bias. The complement rule states that $P_1 = 1 - P_0$. Let $P(a)$ be the probability of a random string being a . Then for any $A \subseteq \mathbb{Z}_2^n$ we denote by $P(A) = \sum_{a \in A} P(a)$.

Let u be a binary string and $A \subseteq \mathbb{Z}_2^n$. Then $w(u)$ denotes the hamming weight of u and $w(A)$ the set $\{w(a) \mid a \in A\}$. Note that since we are working with i.i.d. bits, for any $u, v \in \mathbb{Z}_2^n$ such that $w(u) = w(v)$, the equality $P(u) = P(v)$ holds. Thus, from a probabilistic point of view, it does not matter which element of the set $\{u \in A \mid w(u) = k\}$ we choose to work with.

The element $\min(A)$ ($\max(A)$) is the smallest (biggest) integer of the set A , while $\min_w(A)$ ($\max_w(A)$) is an element from A that has the smallest (biggest) hamming weight. We say that a pair of sets (S_0, S_1) is an equal partition of the set S if the following hold: $S = S_1 \cup S_2$, $S_1 \cap S_2 = \emptyset$ and $|S_1| = |S_2|$.

To ease description, we use the notation C_k^n to denote binomial coefficients. Pascal's identity states that $C_k^n = C_k^{n-1} + C_{k-1}^{n-1}$, where $1 \leq k \leq n$. Note that $|\{u \in \mathbb{Z}_2^n \mid w(u) = k\}| = C_k^n$.

In this paper, we consider a digital filter to be a mapping from \mathbb{Z}_2^n to \mathbb{Z}_2 . If we continuously apply a filter to data generated by a RNG⁴, then three types of filters arise:

- **bias amplifier** - the output data has a bigger bias than the input data;

³ For example the tests described in [10].

⁴ Note that except for $n = 1$ the bit rate of the RNG will drop.

- **neutral filter** - the output data has the the same bias as the input data;
- **bias corrector**⁵ - the output data has a smaller bias than the input data.

Let (S_0, S_1) be an equal partition of a set S . Let D be a digital filter such that it maps S_0 and S_1 to 0 and 1, respectively (see Table 1). Also, let ε_D be the output bias of D . We say that a **bias amplifier** is maximal if ε_D is maximal over all the equal partitions of \mathbb{Z}_2^n . To compare **bias amplifiers** we measure the distance between $P(S_1)$ and $P(S_0)$.

Table 1. Conversion table.

Bit 0	Bit 1
S_0	S_1

Before stating our results, some restrictions are needed. If the input bits are unbiased (*i.e.* $P_0 = \frac{1}{2}$) or have a maximum bias (*i.e.* $P_0 = 0$ or $P_1 = 0$) we require the filter to maintain the original bias. If one replaces a **bias corrector** with a **bias amplifier**, the amplifier must behave as the corrector when the RNG has bias 0 or $\frac{1}{2}$. The last requirement is that the filter amplifies the bias in the direction that it already is. Without loss of generality, we assume that the bias is towards 1.

3 Greedy Bias Amplifiers

In this section we generalize and improve the **bias amplifier** described in [15]. We first present a **neutral filter** and based on it we develop a maximal **bias amplifier**. We can easily transform one into the other by changing the conversion table.

Lemma 1. *Let $S_0 = \{u \in \mathbb{Z}_2^n \mid u = 0\|v, v \in \mathbb{Z}_2^{n-1}\}$ and $S_1 = \{u \in \mathbb{Z}_2^n \mid u = 1\|v, v \in \mathbb{Z}_2^{n-1}\}$. Then $P(S_0) = P_0$ and $P(S_1) = P_1$.*

Proof. Since we are working with i.i.d. random bits the following holds

$$P(S_0) = \sum_{v \in \mathbb{Z}_2^{n-1}} P(0\|v) = \sum_{v \in \mathbb{Z}_2^{n-1}} P_0 P(v) = P_0 \sum_{v \in \mathbb{Z}_2^{n-1}} P(v) = P_0.$$

Similarly, we obtain $P(S_1) = P_1$. □

Using Lemma 1 we can devise a **neutral filter** N by mapping all the elements of S_0 and S_1 to 0 and 1, respectively. Starting from the equal partition

⁵ We prefer to use this notion instead of randomness extractor, because it simplifies our framework.

(S_0, S_1) (Lemma 1), using a greedy algorithm (Algorithm 1), we devise a new equal partition that serves as the core of a maximal bias amplifier.

Algorithm 1.

Input: An integer n

Output: An equal partition of \mathbb{Z}_2^n

- 1 Set $S_0 = \{u \in \mathbb{Z}_2^n \mid u = 0\} \parallel v, v \in \mathbb{Z}_2^{n-1}$ and $S_1 = \{u \in \mathbb{Z}_2^n \mid u = 1\} \parallel v, v \in \mathbb{Z}_2^{n-1}$
 - 2 Set $\alpha = \max_w(S_0)$ and $\beta = \min_w(S_1)$
 - 3 **while** $w(\alpha) < w(\beta)$ **do**
 - 4 Set $S_0 = (S_0 \setminus \{\alpha\}) \cup \{\beta\}$ and $S_1 = (S_1 \setminus \{\beta\}) \cup \{\alpha\}$
 - 5 Update $\alpha = \max_w(S_0)$ and $\beta = \min_w(S_1)$
 - 6 **end**
 - 7 **return** (S_0, S_1)
-

Lemma 2. *Let k be a positive integer and let (S_0, S_1) be the output of Algorithm 1. Then the following properties hold*

1. *If $n = 2k + 1$ then $S_0 = \{u \mid 0 \leq w(u) \leq k\}$ and $S_1 = \{u \mid k + 1 \leq w(u) \leq n\}$. Also, $P(S_0) = \sum_{i=0}^k C_i^n (P_0)^{n-i} (P_1)^i$ and $P(S_1) = \sum_{i=0}^k C_i^n (P_0)^i (P_1)^{n-i}$.*
2. *If $n = 2k$ then $S_0 = \{u \mid 0 \leq w(u) \leq k - 1\} \cup T_0$ and $S_1 = \{u \mid k + 1 \leq w(u) \leq n\} \cup T_1$, where (T_0, T_1) is an equal partition of $\{u \in \mathbb{Z}_2^n \mid w(u) = k\}$. Also, $P(S_0) = \sum_{i=0}^{k-1} C_i^n (P_0)^{n-i} (P_1)^i + \frac{C_k^n}{2} (P_0 P_1)^k$ and $P(S_1) = \sum_{i=0}^{k-1} C_i^n (P_0)^i (P_1)^{n-i} + \frac{C_k^n}{2} (P_0 P_1)^k$.*
3. *If $\varepsilon = 0$ then $P(S_0) = P(S_1) = \frac{1}{2}$ and if $\varepsilon = \frac{1}{2}$ then $P(S_0) = 0$ and $P(S_1) = 1$.*

Proof. During the while loop Algorithm 1 swaps the elements whose weight is written in Column 2, Table 2 with the elements that have their weight written in Column 3, Table 2.

Table 2. Operations performed during the while loop.

Number of switches	Weight of S_0 elements	Weight of S_1 elements
C_0^{n-1}	$n - 1$	1
C_1^{n-1}	$n - 2$	2
...
C_{i-1}^{n-1}	$n - i$	i
...

The while loop ends when $w(\alpha) \geq w(\beta)$. According to Table 2, this is equivalent with $n - i \geq i$. When $n = 2k + 1$ we obtain that the while loop stops when $i \leq k + 1$. When $n = 2k$ the loop stops when $i \leq k$. Thus, we obtain the sets S_0 and S_1 . The probabilities $P(S_0)$ and $P(S_1)$ are a direct consequence of the

structure of the sets and the fact that $C_k^n = C_{n-i}^n$. The last item is simply a matter of computation. \square

Lemma 3. *Let (S_0, S_1) be the output of Algorithm 1. If we map all the elements of S_0 and S_1 to 0 and 1, respectively, then we obtain a maximal bias amplifier G .*

Proof. According to Lemma 2 all the lowest and highest probability elements are in S_0 and S_1 , respectively. Thus, the statement is true. \square

Lemma 4. *Let (S_0^n, S_1^n) be the output of Algorithm 1 for $n = 2k + 1$. Then the following hold*

1. $P(S_0^n) = P(S_0^{n+1})$ and $P(S_1^n) = P(S_1^{n+1})$.
2. $P(S_0^n) - P(S_0^{n+2}) = P(S_1^{n+2}) - P(S_1^n) = 2\varepsilon C_k^n (P_0 P_1)^{k+1}$.
3. $P(S_0^n) > P(S_0^{n+2})$ and $P(S_1^n) < P(S_1^{n+2})$.
4. $P(S_1^n) - P(S_0^n) < P(S_1^{n+2}) - P(S_0^{n+2})$.

Proof. We prove the first statement using induction. When $k = 1$ we have $S_0^1 = \{0\}$, $S_1^1 = \{1\}$, $S_0^2 = \{00, 01\}$ and $S_1^2 = \{10, 11\}$. Using Lemma 1, we obtain $P(S_0^1) = P_0 = P(S_0^2)$ and $P(S_1^1) = P_1 = P(S_1^2)$. Thus, proving the statement for the case $k = 1$.

We now assume that the statement is true for k (i.e. $P(S_0^n) = P(S_0^{n+1})$ and $P(S_1^n) = P(S_1^{n+1})$) and we do it for $k + 1$. Applying Pascal's identity twice to $P(S_0^{n+2})$ we obtain

$$\begin{aligned}
 P(S_0^{n+2}) &= \sum_{i=0}^{k+1} C_i^{n+2} (P_0)^{n+2-i} (P_1)^i = (P_0)^{n+2} + (n+2)(P_0)^{n+1} P_1 \\
 &\quad + \sum_{i=2}^{k+1} (C_i^n + 2C_{i-1}^n + C_{i-2}^n) (P_0)^{n+2-i} (P_1)^i.
 \end{aligned} \tag{1}$$

We rewrite Eqs. (1) as a sum of S^1, S^2, S^3 (described next):

$$\begin{aligned}
 S^1 &= (P_0)^{n+2} + n(P_0)^{n+1} P_1 + \sum_{i=2}^{k+1} C_i^n (P_0)^{n+2-i} (P_1)^i \\
 &= (P_0)^2 P(S_0^n) + C_{k+1}^n (P_0)^{n+1-k} (P_1)^{k+1},
 \end{aligned} \tag{2}$$

$$\begin{aligned}
 S^2 &= 2(P_0)^{n+1} P_1 + 2 \sum_{i=2}^{k+1} C_{i-1}^n (P_0)^{n+2-i} (P_1)^i \\
 &= 2 \sum_{i=0}^k C_i^n (P_0)^{n+1-i} (P_1)^{i+1} = 2P_0 P_1 P(S_0^n),
 \end{aligned} \tag{3}$$

$$\begin{aligned}
 S^3 &= \sum_{i=2}^{k+1} C_{i-2}^n (P_0)^{n+2-i} (P_1)^i = \sum_{i=0}^{k-1} C_i^n (P_0)^{n-i} (P_1)^{i+2} \\
 &= (P_1)^2 P(S_0^n) - C_k^n (P_0)^{n-k} (P_1)^{k+2}.
 \end{aligned} \tag{4}$$

Reassembling Eqs. (2) to (4) we obtain

$$\begin{aligned} P(S_0^{n+2}) &= P(S_0^n) + C_{k+1}^n(P_0)^{n+1-k}(P_1)^{k+1} - C_k^n(P_0)^{n-k}(P_1)^{k+2} \\ &= P(S_0^n) - 2\varepsilon C_k^n(P_0P_1)^{k+1}. \end{aligned} \quad (5)$$

Applying Pascal's identity twice to $P(S_0^{n+3})$ we obtain

$$\begin{aligned} P(S_0^{n+3}) &= \sum_{i=0}^{k+1} C_i^{n+3}(P_0)^{n+3-i}(P_1)^i + \frac{C_{k+2}^{n+3}}{2}(P_0P_1)^{k+2} \\ &= (P_0)^{n+3} + (n+3)(P_0)^{n+2}P_1 \\ &\quad + \sum_{i=2}^{k+1} (C_i^{n+1} + 2C_{i-1}^{n+1} + C_{i-2}^{n+1})(P_0)^{n+3-i}(P_1)^i + \frac{C_{k+2}^{n+3}}{2}(P_0P_1)^{k+2}. \end{aligned} \quad (6)$$

Let $\alpha = \sum_{i=0}^k C_i^{n+1}(P_0)^{n+1-i}(P_1)^i$. We rewrite Eq. (6) as a sum of S^4 , S^5 , S^6 (described next):

$$\begin{aligned} S^4 &= (P_0)^{n+3} + (n+1)(P_0)^{n+2}P_1 + \sum_{i=2}^{k+1} C_i^{n+1}(P_0)^{n+3-i}(P_1)^i \\ &= (P_0)^2\alpha + C_{k+1}^{n+1}(P_0)^{n+2-k}(P_1)^{k+1}, \end{aligned} \quad (7)$$

$$\begin{aligned} S^5 &= 2(P_0)^{n+2}P_1 + 2 \sum_{i=2}^{k+1} C_{i-1}^{n+1}(P_0)^{n+3-i}(P_1)^i \\ &= 2 \sum_{i=0}^k C_i^{n+1}(P_0)^{n+2-i}(P_1)^{i+1} = 2P_0P_1\alpha, \end{aligned} \quad (8)$$

$$\begin{aligned} S^6 &= \sum_{i=2}^{k+1} C_{i-2}^{n+1}(P_0)^{n+3-i}(P_1)^i = \sum_{i=0}^{k-1} C_i^{n+1}(P_0)^{n+1-i}(P_1)^{i+2} \\ &= (P_1)^2\alpha - C_k^{n+1}(P_0)^{n+1-k}(P_1)^{k+2}. \end{aligned} \quad (9)$$

Reassembling Eqs. (7) and (9) we obtain

$$\begin{aligned} P(S_0^{n+3}) &= P(S_0^{n+1}) + C_{k+1}^{n+1}(P_0)^{n+2-k}(P_1)^{k+1} - C_k^{n+1}(P_0)^{n+1-k}(P_1)^{k+2} \\ &\quad - \frac{C_{k+1}^{n+1}}{2}(P_0P_1)^{k+1} + \frac{C_{k+2}^{n+3}}{2}(P_0P_1)^{k+2} \\ &= P(S_0^{n+1}) - C_k^n(P_0P_1)^{k+1} \left\{ \frac{n+1}{k+1} \left[(P_0)^2 - \frac{1}{2} \right] \right. \\ &\quad \left. + P_0P_1 \left[-\frac{n+1}{k+2} + \frac{(n+1)(n+2)(n+3)}{2(k+1)(k+2)(k+2)} \right] \right\} \\ &= P(S_0^{n+1}) - C_k^n(P_0P_1)^{k+1} \left\{ 2 \left[(P_0)^2 - \frac{1}{2} \right] + 2P_0P_1 \right\} \\ &= P(S_0^{n+1}) - 2\varepsilon C_k^n(P_0P_1)^{k+1}. \end{aligned} \quad (10)$$

Applying the induction step to Eqs. (5) and (10) we obtain that $P(S_0^{n+2}) = P(S_0^{n+3})$. The following equality is a consequence of the complement rule

$$P(S_1^{n+2}) = 1 - P(S_0^{n+2}) = 1 - P(S_0^{n+3}) = P(S_1^{n+3}).$$

This completes the proof the first statement. The remaining statements are a direct consequence of Eq. (5) and the complement rule. \square

Corollary 1. *Let (S_0^n, S_1^n) be the output of Algorithm 1 for $n = 2k + 1$. Then $P(S_0^n) - P(S_0^{n+2}) > P(S_0^{n+2}) - P(S_0^{n+4})$ and $P(S_1^{n+2}) - P(S_1^n) > P(S_1^{n+4}) - P(S_1^{n+2})$.*

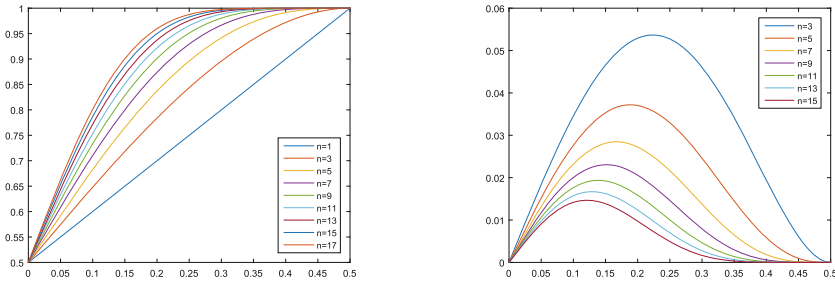
Proof. Using Lemma 4 we obtain that $P(S_0^n) - P(S_0^{n+2}) > P(S_0^{n+2}) - P(S_0^{n+4})$ is equivalent with $2\varepsilon C_k^n (P_0 P_1)^{k+1} > 2\varepsilon C_{k+1}^{n+2} (P_0 P_1)^{k+2}$. Rewriting the inequality we obtain

$$1 > \frac{(2k + 2)(2k + 3)}{(k + 1)(k + 2)} P_0 P_1.$$

The proof is concluded by observing that

$$\frac{(2k + 2)(2k + 3)}{(k + 1)(k + 2)} P_0 P_1 < 4 \left(\frac{1}{4} - \varepsilon^2 \right) = 1 - 4\varepsilon^2 \leq 1. \quad \square$$

Figure 1(a) and (b) are a graphical representation of Lemma 4 ($n \leq 17$) and Corollary 1 ($n \leq 15$), respectively. The x-axis represents the original bias ε , while the y-axis represents $P(S_1^n)$ (Fig. 1(a)) and $P(S_1^{n+2}) - P(S_1^n)$ (Fig. 1(b)).



(a) Probability of S_1^n .

(b) The distance $P(S_1^{n+2}) - P(S_1^n)$.

Fig. 1. Greedy amplifier.

Using the properties stated in Lemmas 2 and 4, we will next describe an equivalent and simplified version of Algorithm 1. Note that devising a greedy **bias amplifier** only makes sense when n is odd.

Algorithm 2.

Input: An odd integer n

Output: An equal partition of \mathbb{Z}_2^n

```

1 Set  $S_0 = S_1 = \emptyset$ 
2 for  $i = 0, \dots, 2^n - 1$  do
3   if  $w(i) \leq k$  then
4     |  $S_0 = S_0 \cup \{i\}$ 
5   end
6   else
7     |  $S_1 = S_1 \cup \{i\}$ 
8   end
9 end
10 return  $(S_0, S_1)$ 

```

4 Von Neumann Bias Amplifier

Von Neumann introduced in [14] a simple, yet effective method for correcting the bias of a RNG. Each time the RNG generates two random bits b_1 and b_2 , the filter outputs b_1 if and only if $b_1 \neq b_2$. It is easy to see that $P(b_1 b_2 = 01) = P(b_1 b_2 = 10) = P_0 P_1$. Thus, the bias of the output data is 0. We further generalize Von Neumann's method and explain how to replace it's conversion table in order to obtain a maximal **bias amplifier**. Through this section we will restrict n to be of the form $2k$, where k is a positive integer.

Lemma 5. *Let $V = \{u \in \mathbb{Z}_2^n \mid w(u) = k\}$ and let (V_0, V_1) be an equal partition of V . Then $P(V_0) = P(V_1) = \frac{C_k^n}{2} (P_0 P_1)^k$.*

Proof. Since (V_0, V_1) is an equal partition of V , we obtain that $|V_0| = |V_1| = \frac{|V|}{2} = \frac{C_k^n}{2}$. Note that $P(u) = (P_0 P_1)^k$, for any $u \in V$. Combining these two facts we obtain the statement of the lemma. \square

Using Lemma 5 we can devise a **corrector filter**⁶ V_c by mapping all the elements of V_0 and V_1 to 0 and 1, respectively. In Algorithm 3 we provide an example of how to generate a pair (V_0, V_1) .

⁶ with the bias of the output data 0.

Algorithm 3.

Input: An integer n
Output: An equal partition of V
1 Set $V_0 = V_1 = \emptyset$ and $V = \{u \in \mathbb{Z}_2^n \mid w(u) = k\}$
2 Set $\alpha = \max(V)$ and $\beta = \min(V)$
3 **for** $i = 1, \dots, C_k^n/2$ **do**
4 Set $V_0 = V_0 \cup \{\beta\}$ and $V_1 = V_1 \cup \{\alpha\}$
5 Update $V = V \setminus \{\alpha, \beta\}$
6 Set $\alpha = \max(V)$ and $\beta = \min(V)$
7 **end**
8 **return** (V_0, V_1)

We further show that the probabilities V_0 and V_1 get smaller if we increase n . This translates in a lower bit rate if we apply V_c . Note that increasing n does not change the bias of the output data, thus making V_c ⁷ useless in practice if used only for correcting biases.

Lemma 6. *Let (V_0^n, V_1^n) be the output of Algorithm 3 for $n = 2k$. Then $P(V_0^n) > P(V_0^{n+2})$.*

Proof. We remark that $P(V_0^n) > P(V_0^{n+2})$ is equivalent with

$$1 > \frac{(2k + 1)(2k + 2)}{(k + 1)(k + 1)} P_0 P_1.$$

The proof is now similar to Corollary 1 and thus is omitted. □

Figure 2 is a graphical representation of Lemma 6 ($n \leq 18$). The x-axis represents the original bias ϵ , while the y-axis represents $P(V_0^n)$.

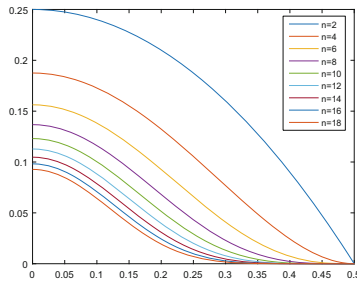


Fig. 2. Probability of V_0^n and V_1^n .

Note that when $P_0 = 0$ or $P_1 = 0$ we obtain $P(V_0) = P(V_1) = 0$. When constructing a **bias amplifier** V_a we must have the same behavior. Thus, the

⁷ for $n \geq 4$.

strings we use to construct V_a need to contain at least a 0 and an 1. When $n = 2$ the only strings that contain 0 and 1 are 01 and 10, but these are the basis for the Von Neumann **bias corrector**. Hence, when $n = 2$ there are no **bias amplifiers**. This leads to the restriction $n \geq 4$. We again use a greedy approach (Algorithm 4) and devise a core for V_a .

Algorithm 4.

Input: An integer n
Output: Two sets V_0 and V_1
1 Set $V_0 = V_1 = \emptyset$ and $W = \mathbb{Z}_2^n \setminus \{0^n, 1^n\}$
2 Set $\alpha = \min_w(W)$ and $\beta = \max_w(W)$
3 **for** $i = 1, \dots, C_k^n/2$ **do**
4 Set $V_0 = V_0 \cup \{\alpha\}$ and $V_1 = V_1 \cup \{\beta\}$
5 Update $W = W \setminus \{\alpha, \beta\}$
6 Set $\alpha = \min_w(W)$ and $\beta = \max_w(W)$
7 **end**
8 **return** (V_0, V_1)

Lemma 7. *Let x be an integer such that $\sum_{i=1}^x C_i^n < C_k^n/2 < \sum_{i=1}^{x+1} C_i^n$. Define $y = C_k^n/2 - \sum_i C_i^n$, $W_0 \subset \{u \in \mathbb{Z}_2^n \mid w(u) = x + 1\}$, $W_1 \subset \{u \in \mathbb{Z}_2^n \mid w(u) = n - x - 1\}$, such that $|W_0| = |W_1| = y$. Also, let (V_0, V_1) be the output of Algorithm 4. Then the following properties hold*

1. $V_0 = \{u \mid 1 \leq w(u) \leq x\} \cup W_0$ and $V_1 = \{u \mid n - x \leq w(u) \leq n - 1\} \cup W_1$.
2. $P(V_0) = \sum_{i=1}^x C_i^n (P_0)^{n-i} (P_1)^i + y (P_0)^{n-x-1} (P_1)^{x+1}$ and $P(V_1) = \sum_{i=1}^x C_i^n (P_0)^i (P_1)^{n-i} + y (P_0)^{x+1} (P_1)^{n-x-1}$.
3. If $\varepsilon = 0$ then $P(S_0) = P(S_1) = \frac{1}{2}$ and if $\varepsilon = \frac{1}{2}$ then $P_0 = 0$ and $P_1 = 1$.

Proof. The proof is a direct consequence of Algorithm 4 and thus is omitted. \square

Lemma 8. *Let (V_0, V_1) be the output of Algorithm 4. If we map all the elements of V_0 and V_1 to 0 and 1, respectively, then we obtain a maximal **bias amplifier** V_a .*

Proof. According to Lemma 7 all the lowest and highest probability elements are in V_0 and V_1 , respectively. Thus, the statement is true. \square

Figure 3 is a graphical representation of Lemma 7 ($n \leq 18$). The x-axis represents the original bias ϵ , while the y-axis in Fig. 3(a) and (b) represents $P(V_0^n)$ and $P(V_1^n)$, respectively.

Unfortunately, due to the nature of x and y , the best we could do is to heuristically provide a graphical representation of Conjecture 1 (Fig. 4). We could not theoretically prove it in general.

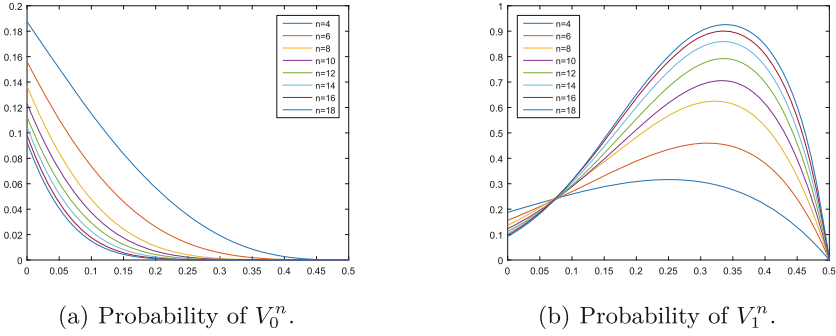


Fig. 3. Von Neumann amplifier.

Conjecture 1. Let n be even, (S_0^{n-1}, S_1^{n-1}) be the output of Algorithm 1 for $n-1$ and (V_0^n, V_1^n) be the output of Algorithm 4 for n . Denote by $M^n = [P(V_1^n) - P(V_0^n)] \cdot [P(V_1^n) + P(V_0^n)]^{-1}$. Then $M^n < M^{n+2}$ and $P(S_1^{n-1}) - P(S_0^{n-1}) < M^n$.

Note that in the case of greedy amplifiers the metric $[P(S_1^{n-1}) - P(S_0^{n-1})] \cdot [P(S_1^{n-1}) + P(S_0^{n-1})]^{-1}$ is equal to $P(S_1^{n-1}) - P(S_0^{n-1})$. Thus, Conjecture 1 states that the Von Neumann amplifier for a given n is better at amplifying ϵ than its greedy counterpart. We chose to state the conjecture such that it is true for all $n \geq 4$, but, from Fig. 4, we can observe that as n grows the Von Neumann amplifier becomes better at amplifying ϵ^8 . Note that in Fig. 4 the x-axis represents the original bias ϵ , while the y-axis represents the values $P(S_1^{n-1}) - P(S_0^{n-1})$ (interrupted line) and M^n (continuous line).

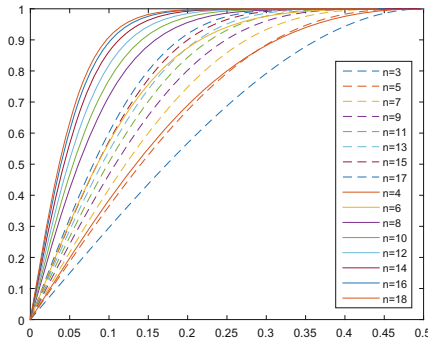


Fig. 4. Comparing greedy amplifiers (interrupted line) with Von Neumann amplifiers (continuous line).

⁸ e.g the Von Neumann amplifier for $n = 8$ is better than the greedy amplifiers for $n = 3, \dots, 17$.

5 Applications

5.1 The Good

RNG standards [11,13] require manufactures to implement some early detection mechanism for entropy failure. Health tests represent one such method for detecting major failures. There are two categories of health tests: startup tests and continuous tests. The former are one time tests conducted before the RNG starts producing outputs, while the latter are tests performed in the background during normal operation.

We propose a generic architecture for implementing health tests (Fig. 5). We first store data D (obtained from the noise source) in a buffer, then we apply a **bias amplifier** to it and obtain data D_a . Next, we apply some lightweight tests on D_a . If the tests are passed, the RNG outputs D , otherwise D is discarded. Note that the **bias amplifier** can be implemented as a lookup table, thus obtaining no processing overhead at the expense of $\mathcal{O}(2^n)$ memory.

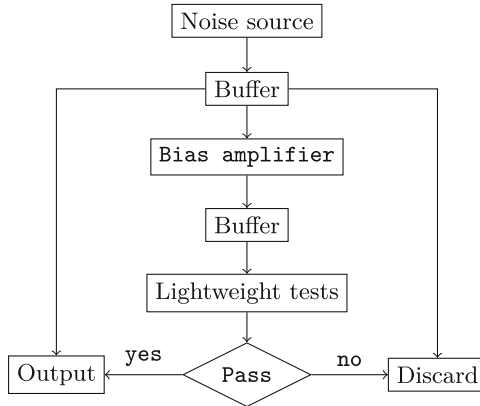


Fig. 5. Generic architecture for implementing health tests.

In our instantiations we used the health tests implemented in Intel’s processors [10]. Intel’s health tests H_i use a sliding window and count how many times each of the six different bit patterns (Column 1, Table 3) appear in a 256 bit sample. An example of allowable margins for the six patterns can be found in Column 2, Table 3. The thresholds mentioned in Tables 3 and 4 were computed using 10^6 256 bit samples generated using the default RNG from the GMP library [3].

We first propose a continuous test using the greedy amplifiers described in Sect. 3. Depending on the available memory we can use one of the greedy amplifiers and then apply H_i . Note that n should be odd due to Lemma 4. If the health test are implemented in a processor it is much easier to use $n = 4, 8, 16$. From the health bounds presented in Table 3, we can observe that the differences between

data without amplification and data with amplification are not significant. Thus, one can easily update an existing good RNG⁹ by adding an extra buffer and an amplification module, while leaving the health bounds intact. Note that due to the unpredictable number of output bits produced by a Von Neumann amplifier, greedy amplifiers are better suited for continuous testing.

Table 3. Health bounds for greedy amplifiers (amp.).

Bit pattern	Allowable number of occurrences per sample			
	without amp.	n = 3 amp.	n = 5 amp.	n = 7 amp.
1	$90 < m < 165$	$88 < m < 165$	$89 < m < 167$	$90 < m < 165$
01	$45 < m < 83$	$45 < m < 82$	$46 < m < 83$	$45 < m < 83$
010	$8 < m < 59$	$9 < m < 62$	$10 < m < 58$	$7 < m < 60$
0110	$1 < m < 38$	$2 < m < 34$	$2 < m < 35$	$2 < m < 34$
101	$10 < m < 59$	$10 < m < 61$	$10 < m < 60$	$9 < m < 63$
1001	$1 < m < 35$	$2 < m < 36$	$0 < m < 35$	$1 < m < 35$

If the design of the RNG has a Von Neumann module, then Von Neumann amplifiers can be used to devise a startup test. Before entering normal operation, the Von Neumann module can be instantiated using the conversion table of the corresponding amplifier. For example, when $n = 4$ one would use $V_0 = \{0001, 0010, 0100\}$ and $V_1 = \{0111, 1011, 1101\}$ ¹⁰ instead of $V_0 = \{0011, 0101, 0110\}$ and $V_1 = \{1001, 1010, 1100\}$ ¹¹. The resulting data can then be tested using H_i and if the test pass the RNG will discard the data and enter normal operation. Note that the first buffer from Fig. 5 is not necessary in this case. Note that Von Neumann amplifiers require $n > 2$, thus the speed of the RNG will drop. This can be acceptable if the data speed needed for raw data permits it, the RNG generates data much faster than the connecting cables are able to transmit or the raw data is further used by a pseudo-random number generator (PRNG).

5.2 The Bad

One can easily turn the benign architecture presented in Fig. 5 into a malicious architecture (Fig. 6). In the new proposed configuration, health tests always output **pass** and instead of outputting D the system outputs D_a .

The malicious configuration can be justified as a bug and can be obtained from the original architecture either by commenting some code lines (similarly to [6]) or by manipulating data buffers (similarly to [7]). Note that code inspection

⁹ that already has H_i implemented.

¹⁰ the sets used to define the maximal Von Neumann amplifier.

¹¹ the sets used to define the Von Neumann corrector.

Table 4. Health bounds for Von Neumann correctors (corr.) and amplifiers (amp.).

Bit pattern	Allowable number of occurrences per sample			
	n = 4 corr.	n = 4 amp.	n = 6 corr.	n = 6 amp.
1	$88 < m < 166$	$91 < m < 167$	$89 < m < 167$	$90 < m < 168$
01	$43 < m < 83$	$44 < m < 83$	$44 < m < 85$	$45 < m < 82$
010	$9 < m < 59$	$10 < m < 60$	$7 < m < 58$	$9 < m < 60$
0110	$1 < m < 33$	$1 < m < 36$	$2 < m < 35$	$2 < m < 33$
101	$10 < m < 58$	$11 < m < 61$	$10 < m < 57$	$8 < m < 60$
1001	$0 < m < 34$	$2 < m < 35$	$1 < m < 34$	$1 < m < 34$

or reverse engineering will reveal these so called bugs. A partial solution to detection can be implementing the architecture in a tamper proof device and deleting the code if someone tinkers with the device. Another partial solution is embedding the architecture as a submodule in a more complex architecture (similarly to [6]). This solution is plausible due to the sheer complexity of open-source software and the small number of experts who review them [5].

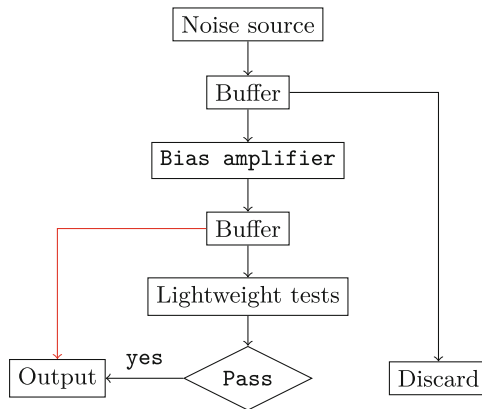


Fig. 6. Generic architecture for infecting RNGs.

Another problem is that the RNG will output $D_a s$ instead of $D s$ and this translates to lower data rates. A possible solution to this problem is to use D_a as a seed for a PRNG and then output the data produced by the PRNG. Thus, raw data is never exposed. A problem with this approach is that in most cases the PRNG will also mask the bias. The only case that is compatible with this approach is when the bias is large. Therefore one can simply use an intelligent brute force to find the seed. Hence, breaking the system.

A more suitable approach to the aforementioned problem is to use a pace regulator [9]. This method uses an intermediary buffer to store data and

supplies the data consumer with a constant stream of bits. Unfortunately, if data requirements are high, then the regulator will require a lot of memory and in some cases the intermediary buffer will be depleted. Thus, failing to provide data.

A solution specific to greedy amplifiers is to implement in all devices a **neutral filter** after D and output the resulting data D_n . Thus, when a malicious version of the RNG is required, one can simply replace the conversion table of the **neutral filter** with the conversion table of the corresponding **bias amplifier**. For example, when $n = 3$ one would change $S_0 = \{000, 001, 010, 100\}$ and $S_1 = \{111, 110, 101, 100\}$ ¹² with $S_0 = \{000, 001, 010, 100\}$ and $S_1 = \{111, 110, 101, 011\}$ ¹³. It is easy to see that in this case both D_n and D_a have the same frequency.

Since we are modifying the statistical properties of the raw data, a simple method for detecting changes is black box statistical testing (for example using [2]). Thus, if a user is getting suspicious he can detect the “bugs”. Again, a partial solution is to implement the malicious architecture as a submodule inside a more complex architecture either in tamper proof devices, either in complex software. Thus, eliminating the user’s access to raw data.

6 Conclusions and Future Work

In our paper we studied and extended **bias amplifiers**, compared their performance and provided some possible applications for them. Even thou in its infancy, this area of research provides insight into what can go wrong with a RNG.

A possible future direction would be to extended our results to other randomness extractors. Of particular interest, is finding a method to turn a block cipher or a hash function¹⁴ into an amplifier.

Acknowledgments. The author would like to thank Diana Maimuț and the anonymous reviewers for their helpful comments.

A Experimental Results

To test the configuration proposed in Sect. 5.1, Fig. 5 and obtain some metrics (Table 5) we conducted a series of experiments. More precisely, we generated 105000 256 bit samples using the Bernoulli distribution instantiated with the Mersenne Twister engine (mt19937) found in the C++ random library [1]. Then, we applied the bias amplifying filters from Table 3 and counted how many samples are marked **pass**. In the case of raw data, a sample is marked **pass**¹⁵ if it

¹² the sets used to define the **neutral filter**.

¹³ the sets used to define the maximal greedy amplifier.

¹⁴ For a formal treatment of how one can use a block cipher or a hash function to extract randomness we refer the reader to [8].

¹⁵ The terminology used by Intel is that the sample is “healthy”.

passes the H_i test from Column 1, Table 3. In the case of bias amplification, if a 256 bit buffer b_a from D_a passes H_i , all the input buffers that were used to produce b_a are marked **pass**. Note that to implement our filters we used lookup tables and thus we had no performance overhead.

From Table 5 we can easily see that when the bias is increased, the number of samples that are marked **pass** is lower than H_i in the case of greedy amplifiers. Also, note that the rejection rate is higher as n increases. Thus, enabling us to have an early detection mechanism for RNG failure.

Table 5. Greedy amplifiers (amp.) metrics.

ϵ	Number of samples marked pass			
	without amp.	n = 3 amp.	n = 5 amp.	n = 7 amp.
0.00	104999	104997	105000	105000
0.01	104999	104991	104990	105000
0.02	104996	104979	104945	104965
0.03	104988	104925	104685	104384
0.04	104949	104631	103545	101661
0.05	104856	103620	99370	91413
0.06	104598	100668	88845	69832
0.07	104002	93840	69810	41286
0.08	102763	81660	46110	17724
0.09	100411	64332	23460	5404
0.10	96381	44262	9005	1043
0.11	89967	26142	2625	105
0.12	80849	12882	570	0
0.13	69164	5253	65	0
0.14	55856	1704	0	0
0.15	41777	420	0	0
0.16	29039	87	0	0
0.17	18410	21	0	0
0.18	10470	6	0	0
0.19	5331	0	0	0
0.20	2393	0	0	0

We also conducted a series of experiments to test the performance of the startup test proposed in Sect. 5.1. This time, we generated data until we obtained 1000 256-bit samples, applied the bias correcting/amplifying filters from Table 4 and counted how many of these samples pass the H_i test from Column 1, Table 3. Another metric that we computed is the number of input bits required to generate one output bit.

Table 6. Von Neumann correctors (corr.) and amplifiers (amp.) metrics.

ϵ	Number of samples that pass H_i		
	n = 2 corr.	n = 4 amp.	n = 6 amp.
0.00	1000	1000	1000
0.01	1000	1000	1000
0.02	1000	1000	995
0.03	1000	998	940
0.04	1000	981	721
0.05	1000	919	322
0.06	1000	806	79
0.07	1000	567	7
0.08	1000	310	0
0.09	1000	134	0
0.10	1000	53	0

Note that in Table 6 we only wrote the $n = 2$ corrector, since for $n = 4, 6$ the results are almost identical. From Table 6 we can easily observe that when the bias is increased the number of samples that pass H_i is lower than the corrector in the case of Von Neumann amplifiers. As in the case of greedy amplifiers, we can observe that the rejection rate is higher as n increases. The experimental data also shows that Von Neumann amplifiers perform better than the greedy amplifiers when rejecting bad samples.

In Table 7 we can see that more data is required to generate one bit as n grows. When the bias increases, we can observe that compared to Von Neumann

Table 7. Von Neumann correctors (corr.) and amplifiers (amp.) throughput.

ϵ	Number of input bits per number of output bits				
	n = 2 corr.	n = 4 corr.	n = 4 amp.	n = 6 corr.	n = 6 amp.
0.00	3.9958	10.6646	10.6751	19.1374	19.2776
0.01	3.9978	10.6690	10.6817	19.1873	19.2548
0.02	4.0044	10.6852	10.6885	19.2513	19.2017
0.03	4.0106	10.7272	10.6873	19.3623	19.0892
0.04	4.0202	10.7956	10.6900	19.5129	18.9534
0.05	4.0352	10.8755	10.6952	19.7228	18.7933
0.06	4.0531	10.9713	10.6980	20.0087	18.5889
0.07	4.0755	11.1025	10.6876	20.3259	18.3405
0.08	4.1013	11.2489	10.6709	20.7180	18.0855
0.09	4.1264	11.3916	10.6841	21.1418	17.8104
0.10	4.1594	11.5733	10.6823	21.6591	17.5187

correctors the throughput of the corresponding amplifiers is better. Thus, besides having an early detection mechanism, it also takes less time to detect if an RNG is broken if we use a Von Neumann amplifier.

References

1. C++ Random Library. www.cplusplus.com/reference/random/
2. NIST SP 800–22: download documentation and software. <https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software>
3. The GNU multiple precision arithmetic library. <https://gmplib.org/>
4. Ball, J., Borger, J., Greenwald, G.: Revealed: How US and UK spy agencies defeat internet privacy and security. *The Guardian*, vol. 6 (2013). <https://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security>
5. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) *CRYPTO 2014*. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_1
6. Bello, L.: DSA-1571-1 OpenSSL—Predictable Random Number Generator (2008). <https://www.debian.org/security/2008/dsa-1571>
7. Checkoway, S.: A systematic analysis of the juniper dual EC incident. In: *ACM CCS 2016*, pp. 468–479. ACM (2016)
8. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and key derivation using the CBC, cascade and HMAC modes. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_30
9. Ferradi, H., Géraud, R., Maimuț, D., Naccache, D., de Wargny, A.: Regulating the pace of von neumann correctors. *J. Crypt. Eng.* **8**(1), 1–7 (2017)
10. Hamburg, M., Kocher, P., Marson, M.E.: Analysis of Intel’s Ivy bridge digital random number generator (2012) http://www.rambus.com/wp-content/uploads/2015/08/Intel_TRNG_Report_20120312.pdf
11. Killmann, W., Schindler, W.: A proposal for: functionality classes for random number generators, version 2.0 (2011). https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31.Functionality_classes_for_random_number_generators.e.pdf?__blob=publicationFile
12. Perlroth, N., Larson, J., Shane, S.: NSA able to foil basic safeguards of privacy on web. *New York Times* **5** (2013). <https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html>
13. Turan, M.S., Barker, E., Kelsey, J., McKay, K., Baish, M., Boyle, M.: NIST DRAFT special publication 800–90B: recommendation for the entropy sources used for random bit generation (2012). <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf>
14. Von Neumann, J.: Various techniques used in connection with random digits. *Appl. Math Ser.* **12**, 36–38 (1951)
15. Young, A., Yung, M.: *Malicious Cryptography: Exposing Cryptovirology*. John Wiley and Sons, Hoboken (2004)



Utilizing GPU Virtualization to Protect the Private Keys of GPU Cryptographic Computation

Ziyang Wang^{1,2,3}, Fangyu Zheng^{1,2(✉)}, Jingqiang Lin^{1,2,3},
and Jiankuo Dong^{1,2,3}

¹ Data Assurance and Communication Security Research Center,
CAS, Beijing 100093, China

{wangziyang,zhengfangyu,linjingqiang,dongjiankuo}@iie.ac.cn

² State Key Laboratory of Information Security, Institute of Information
Engineering, CAS, Beijing 100093, China

³ School of Cyber Security, University of Chinese Academy of Sciences,
Beijing 100049, China

Abstract. Nowadays graphics processing units (GPUs) have become popular parallel computing platforms known as General-Purpose GPU (GPGPU) computing. GPUs thereby are chosen by some security researchers as cryptographic accelerators to secure massive volumes of transactions. However, their security issues are ignored in spite of their popularity and performance. There are some possible information leakages faced with malicious attacks or even in the normal GPU computing. Our objective is to secure the confidentiality of cryptographic keys in GPU computing environments and provide easy-to-use programming with few constraints. In this paper, we propose a prototype in Linux, a system of GPGPU computing solution empowered by GPU virtualization technology, which keeps encrypted keys in guest machine to protect secret keys from leakage even in the event of full system compromise. With the API interception and redirection of CUDA, applications in Virtual Machines (VMs) can access the GPU device in a transparent way. Besides, we use `virtio`, a dedicated virtual I/O device, to transfer data between virtual and host machines in high performance. In our current study, we evaluate our prototype with the GPU implementation of ECC. We show that it can protect private keys of GPU cryptographic computation and it also incurs low performance penalty compared with the native environment, therefore demonstrating the prototype is secure and requires reasonable overhead.

Keywords: GPU · GPGPU · GPU virtualization
Information leakage · Isolation

This work was supported by National Natural Science Foundation of China under Award no. 61772518.

1 Introduction

Over the last decade, graphics processing units (GPUs) have been increasingly used both as accelerating graphics rendering engines and parallel programmable processors due to their high computing power and low price.

With hundreds to thousands of streaming processing cores, modern GPUs are used to speed up computations in the single-instruction-multiple-data (SIMD) fashion, providing ample computation cycles and high memory bandwidth to massively parallel applications. As a result, GPUs have quickly been applied in a broad spectrum of applications.

Meanwhile, the expanding demand for cryptographic operations for secure communication and authentication requires high-performance implementations. In fact, the GPU-based implementations of cryptographic operations (e.g., RSA, AES, ECDSA, SHA-1) achieve significantly higher throughput and efficiency than CPU implementations [9, 11, 12, 23]. GPUs are leveraged to offload cryptographic workloads from CPUs. For example, the GPU implementation of AES achieves up to 28x higher throughput.

Although the GPU-based implementations of cryptographic operations aim at security, a thorough analysis of the GPU environment has not been well studied. As regards security and isolation, they are not considered as important as performance. In fact, GPU and CPU architecture are quite different, therefore they face different threats.

For discrete GPUs' architecture, their independent memory and computational resources are physically partitioned from CPU, which seems to make it plausible that GPUs could be used as *secure co-processors*. In CCS '14, Vasiladis et al. [29] propose PixelVault, which is a system implementing AES and RSA for keeping sensitive information (including cryptographic keys) and performing cryptographic operations exclusively on GPU. PixelVault chooses GPU registers, which are reported to be automatically reset to zero when the kernel is loaded, as the secret and private keys storage. Intermediate sensitive data is kept encrypted by master key in GPU global memory. No doubt the master key is stored in GPU registers. Any computation with the secret keys is exclusively limited to those registers. As a result, PixelVault prevents even privileged host code from accessing any sensitive code or data on GPU. By exposing private keys in plaintext only in GPU registers, and keeping PixelVault's critical code exclusively in the GPU instruction cache, PixelVault seems to be a promising approach to prevent even privileged host code from accessing any sensitive code or data on GPU.

With a different technology roadmap, in 2016, Kim et al. [13] propose OBMI which is a SMM-based (System Management Mode) framework for bootstrapping secure cryptographic operations on GPGPUs while preserving robustness, efficiency and programmability. Unlike PixelVault, they store keys in GPU cache which also cannot be accessed by CPU processes and cannot be accessed after termination of GPU kernel. However, data in GPU cache is beyond control of programmer, it is critical to ensure data remain in the cache and can not be evicted. To subvert this issue, they store the key in the constant cache using

SMM before system booting. They also utilize SMM for isolating authenticated GPU kernel in instruction cache. Thus, only SMI (System Management Interrupt) and trusted kernel can access the key. By handling sensitive data only in SMM, OBMI can secure cryptographic operations.

Unfortunately, while some characteristics of GPU architecture and execution model are officially confirmed, some are poorly documented and not validated experimentally. Indeed, Zhu et al. [32] demonstrate how unpublished or recently introduced features of GPUs may bypass the protection mechanisms of PixelVault and compromise the whole system. They refute the following security assumptions of PixelVault in details. Exploiting memory mapped input output (MMIO) registers, they can invalidate the GPU instruction caches and replace them with their own malicious code from running kernels. Using recent changes in debugging support, privileged users are able to attach any running kernel and read the GPU registers, effectively extracting the secret keys. What's worse, it is unclear how to disable this capability. And for OBMI, because However as mentioned before [32], unpublished MMIO registers are able to flush the instruction cache, allowing to inject malicious code. Consequently, it breaks the security assumption of both PixelVault and OBMI.

As a summary, any kind of information leakage from security-sensitive applications (e.g., those runs security protocols or cryptographic algorithms) would severely undermine the trustworthiness of GPU computing. Thus in order to protect the secret keys under a range of memory leakages and threats to the underlying system, we propose a key-protection isolation mechanism on GPGPUs with system-level virtualization technology. The contributions of this paper are threefold:

1. We propose a secure GPU computing model for the GPU-based cryptographic service. We suggest a GPU virtualization approach for cryptographic computations by API remoting method, which isolates master key and keeps accelerating operations safe. To the best of our knowledge, we are the first to utilize GPU virtualization technology to solve the security issues of GPU.
2. Based on the proposed model, we implement a prototype on the commodity GPU with QEMU-KVM with various kinds of optimization. The extra virtualization layer also allows us to introduce mechanisms for checking the integrity of the accelerated GPU code, which is previously very complex to implement in GPUs.
3. By comparing the native throughput of ECC with GPU virtualization through various experiments, we evaluate the performance overhead of our solutions. The evaluations show that our approach incurred only a negligible performance degradation with preventing private keys from leakage of GPU cryptographic computation.

The rest of the paper is organized as follows: we begin by providing necessary background for the current work. Then we describe our design in Sect. 3. In Sect. 4, the detailed implementation is revealed and we evaluate the performance. We discuss security analysis in Sect. 5. Finally, we conclude the paper in Sect. 6.

2 Preliminaries and Related Work

This section gives some basic introduction to GPU, GPU virtualization and some attacks and defenses on cryptographic keys in modern processors.

2.1 GPU Basis

The computational capabilities of GPUs for executing parallel applications are based on hundreds or thousands of processing cores and a high bandwidth memory architecture. A GPU has several Streaming Multiprocessors (SM) which are in turn composed of Streaming Processor cores (SP, or CUDA cores), registers, caches.

A GPU application consists the host code running on the CPU and kernels which runs on the GPU. GPU kernels are special functions executing n times in parallel by n different threads. The number of threads can be specified at kernel launch time.

Running a task on GPU follows three steps:

- The DMA controller transfers the input data from host memory to GPU memory.
- The host application launches the kernel which runs on GPUs.
- The DMA controller transfers the output data from GPU memory to host memory.

As CUDA is becoming a prevalent programming model of GPGPU, we focus on CUDA runtime API while virtualizing GPUs.

2.2 GPU Virtualization

Although virtualization provides a wide range of benefits, such as system security, ease of management, isolation, and live migration, virtualizing GPUs is a relatively new area of study and remains a challenging endeavor which is due to undisclosed details of GPU implementation and unstandardized GPU architectures.

API remoting approach is a kind of protocol redirection, which virtualizes GPUs in a simple way and without significant performance penalty by providing a GPU wrapper library to a guest OS to intercept GPU runtime calls. This approach does not adopt custom GPU driver in the guest [10]. vCUDA [25] and rCUDA [3] are recent projects using API remoting in GPU virtualization.

vCUDA provides a CUDA wrapper library and virtual GPUs (vGPUs) in the guest and the vCUDA stub in the host. vGPUs are created per application by the wrapper library and give a complete view of the underlying GPUs to applications. Instead of emulation, rCUDA creates virtual CUDA-compatible devices on machines without GPU by adopting remote GPU-based acceleration.

However, these frameworks either rely on the scheduling mechanisms provided by the CUDA runtime, or allow applications to execute on GPU in sequence, possibly leading to low resource utilization and consequent suboptimal performance.

2.3 Attack and Defense on Cryptographic Keys in CPU

As long as the secrecy of cryptographic keys involved in cryptographic operations is guaranteed, the cryptosystem is trustworthy even if it has been compromised. Nevertheless keeping cryptographic keys safe is still a great challenge in any cryptosystem [16]. During cryptographic operations, private keys are always loaded into main memory as plaintext, therefore private keys are prone to memory disclosure attacks.

A malicious program can exploit Meltdown [17] and Spectre [14] in modern processors to steal data from the main memory, dumping passwords, personal photo, emails, instant messages and so on.

Although various mechanisms have been proposed in memory protection [28,30], the main memory is still vulnerable to physical attacks, such as cold-boot [8], DMA attack [1,27], which could bypass the protections of OS. To prevent cold-boot attack, AESSE [21], TRESOR [22], and Amnesia [26] store AES keys exclusively in CPU registers. PRIME [5] and Copker [6] implements the RSA algorithm in AVX registers and cache respectively. While Mimosa [7] utilizes hardware transactional memory to protect the RSA cryptographic operations from cold-boot and memory disclosure attacks.

However the CPU-bound encryption approach requires the integrity of the OS kernel. Any compromised OS kernel can easily leak the register or the cache within the CPU. TRESOR-HUNT [1] exploits DMA to inject malicious code into the OS kernel memory and then access the keys in registers.

2.4 Attack and Defense on Cryptographic Keys in GPU

Not only for CPUs, recent works have started investigating the security vulnerabilities of GPUs. Some works notice the GPU driver does not erase memory after kernel termination, indicating that they can leak sensitive data [24,31].

Memory isolation policies enforced by a CPU cannot be applied to GPU memory automatically, so any discrepancy between CPUs and GPUs can lead to unexpected information leakages. By exploiting such vulnerabilities, Pietro et al. [24] recover both plaintext and encryption key of AES from GPU global memory. For thwarting information leakage in GPUs, it is believed that the best solution is memory isolation enhancements performed at the driver/hardware level. Nevertheless, it would be better that CUDA should allow OS to monitor usage and control access to GPU resources.

As modern GPUs share virtual and even physical memory with CPUs, buffer overflows becomes possible in GPU and can lead to remote code execution, corruption on sensitive data and security problems as CPU-based overflows [2,20]. Erb et al. [4] present a tool that utilizes canaries to detect buffer overflows caused by GPGPUs kernel in OpenCL GPU applications.

A recent study shows that remanent data in GPU memory can be retrieved since GPU does not automatically zero its memory after termination. [15,24,31] Even implementing an appropriate erasing operations for the GPU memory, attackers with GPU driver privileges can also access GPU memory with MMIO

registers [19]. Some researchers treat discrete GPUs as secure co-processors storing private keys in GPU registers [29] or GPU cache [13] while offloading cryptographic operations onto GPUs. Unfortunately, due to the widespread commercial strategy to hide implementation details from competitors, manufacturers of GPUs are reluctant on publishing the internals of their solutions [18], which implies the discrete GPUs cannot be trusted as secure co-processors, rather, they may host stealthy malware [32].

3 System Design

3.1 Threat Model and Design Goals

Threat Model. We intend to provide a isolated cryptographic computation environment using GPU in virtual machine from the OS in commodity platform. In this situation, the objective of the adversary is to leak the sensitive information of cryptographic operations from victim’s system. We assume that the adversary has the ability to fully control over the VM and obtain root privilege through intrusion attacks. We consider the malicious user has no physical access to the computer. Otherwise the victim machine is venerable to hardware-based attacks such as cold-boot attack or DMA attack.

The underlying VMM is mostly safe so that even if the OS of VM is compromised, the hacker cannot escape the guest virtual machine and execute code on hypervisor or host operating system. Moreover, we ignore denial-of-service attacks.

Design Goals. Our most primary goal is to design a safe environment for GPU accelerating cryptographic computation without leaking secret keys. This implies that no keys or sensitive information should get into memory. Considering this policy, we can isolate the GPU and OS with a master key from a virtual machine. To restrain cryptographic operations from dealing with secret keys, they are transferred from VM to host using secure channel. The actual secret keys and related sensitive information should never be exposed to the memory of VM. In that case even if the VM is compromised, no sensitive information in VM memory can be leaked.

Meanwhile, we need to guarantee the throughput of cryptographic computations. The performance of cryptographic operations should be not effected obviously.

3.2 System Architecture

The framework we propose is organized in three main architectural features. By using CUDA API Remoting, we implement GPU virtualization in guest OS. CUDA cryptographic applications can utilize GPU with original API functions in the same manner as a typical GPGPU program. However, secret keys must

not be transferred into memory of guest OS in case they are disclosed via malicious attacks. All we suggest to manage keys securely is to isolate the master key in VMM, while secret keys from any application need to encrypt/decrypt via the master key. Moreover, to prevent leakage of sensitive information from GPU memory, we also verify the validity of the CUDA fat binary before application launches a request. If there is no information of CUDA fat binary in our white list, the system denies the request and records this abnormal behavior. In this way, users can launch secure cryptographic operations on virtualized environment. The architecture will be discussed in depth in the following sections.

Isolation Using GPU Virtualization. The first step of our isolation mechanism for securing accelerating cryptographic computation is the ability to utilize GPU in virtual machine. As we all know, GPU vendors somehow are not willing to provide general purpose GPU virtualization solution. They tend not to publish the source code and implementation details of their GPU drivers, which are essential for virtualizing GPUs at the driver level, for commercial reasons. Even when driver implementations are unveiled, for example, by reverse engineering methods, significant changes are introduced with every new generation of GPUs to improve performance. As a result, specifications revealed by reverse engineering become useless.

In a word, there are no standard interfaces for virtualizing GPU devices in driver level.

Fortunately the API remoting approach overcomes aforementioned limitations, because it can emulate a GPU execution environment without exposing physical GPU devices in the guest OS. The premise of API remoting is to provide guest OS with a custom library which contains the same API as the original library. However the library intercepts GPU calls from the application and redirects the request with proper parameters to the host OS for execution as normal calls through shared memory or sockets. Only results are delivered to the application through wrapper library in reverse.

It is flexible that the wrapper library can be dynamically linked to existing applications at runtime. What is more important is that API remoting approach incurs negligible performance overhead. In addition, this approach can be monitored by underlying hypervisors as the virtualization layer is implemented in user space.

Key-Protection Mechanism. The main focus of our work is the key-protection mechanism during the GPU accelerating cryptographic operations. Instead of storing the master key in registers or cache of GPU like PixelVault and OBMI do, we decide to keep master key in VMM. All keys residing in VMs are encrypted by the master key. Thus only during cryptographic operations, VMM manages to decrypt cryptographic keys with the master key and uploads the actual keys into GPU global memory where kernel can retrieve. Therefore, even if adversaries manage to acquire the secret keys from the memory space of VMs, they would get encrypted content which is unuseful.

GPU Binary Verification. Although our key-protection mechanism can securely upload the secret key into the GPU global memory, it is useless if the GPU kernel is compromised. Thus adversary can launch a malicious GPU code-injection attack and patiently leak the key from the GPU global memory. In order to prevent such attack, verifying the integrity of GPU kernel is a prerequisite for protecting secret keys. Before launching the accelerating cryptographic operation in VM, VMM checks the integrity of the GPU kernel. After validating authenticated GPU code, VMM allows the GPU binary to execute then. Otherwise, the cryptographic operations are rejected and recorded in abnormal logs. With the proposed method, we can securely execute arbitrary, authenticated GPU code without any tampering.

4 Implementation and Performance Evaluation

Based on the design principles, we describe various aspects of our prototype in detail, and propose a general system architecture and some more details in the implementation of our prototype.

4.1 Architecture Overview

The overall design and process details of the proposed solution are shown in Fig. 1. In this architecture, we choose `virtio`, a standard for para-virtualization I/O devices, as the transfer channel, which we discuss in following sections. By using API remoting approach, CUDA driver is not essential for guest OS.

First, when a CUDA application demands a GPU service, it can dynamically link to wrapper library and invoke CUDA runtime API as the typical GPGPU program. Wrapper library intercepts calls before the calls reach the GPU driver in the guest OS and redirects them to `virtio` front-end driver through `ioctl`. Then guest driver forwards API requests with proper parameters via `virtio` buffer to back-end driver in VMM. If it is the first time running for application, VMM verifies the fat binary file using HMAC and then checks it whether in our white list. Only passing the validation, API requests can be executed as the same

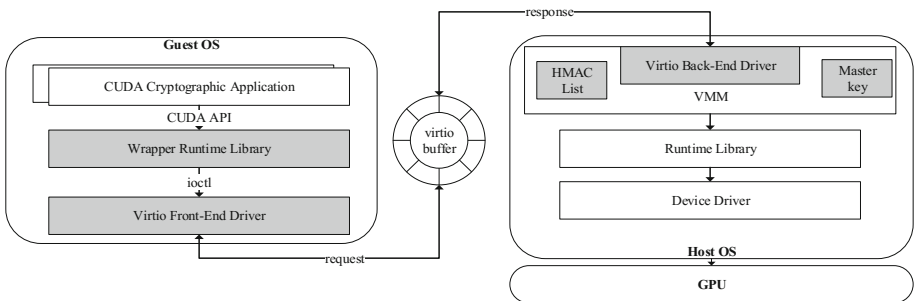


Fig. 1. The architecture of the proposed prototype.

real runtime APIs do. Otherwise, VMM denies of execution. Finally, the results should be sent back to the application in reverse path.

4.2 Implementation Details

API Remoting GPU Virtualization. In general, the typical phase for execution of a kernel requires several steps, which we illustrate using the vector addition as an example:

1. **Initialization.** The process obtains the GPU module from the CUDA binary, which comprises the CUDA fat binary code and other related data such as statically allocated variables.
2. **Memory Allocation.** The process requests memory allocation on the GPU for the data used by kernel execution.
3. **Input Data Transfer.** All the data used by kernel execution must be copied from RAM to GPU global memory allocated in second step.
4. **Kernel Execution.** The GPU code is executed with the parameters and configurations, such as block size and thread size.
5. **Output Data Transfer.** Once the kernel execution is completed, the results in GPU should be transferred to RAM.
6. **Memory Release.** GPU memory which is allocated before is released.
7. **Finalization.** The process releases all the associated resources and quits.

According to our experiments, all the cardinal running APIs we need to complete a typical cuda application is shown in Table 1.

Table 1. The functionality of primary runtime API

Operation	Functionality	Stage
<code>__cudaRegisterFunction</code>	Get the handle to kernel called with binary code and function name	Initialization
<code>__cudaUnregisterFatBinary</code>	Release the fat binary	Finalization
<code>__cudaRegisterFatBinary</code>	Get the handle to the fat binary	Initialization
<code>cudaMalloc</code>	Allocate memory on the device	Memory allocation
<code>cudaConfigureCall</code>	Configure a device launch	Kernel execution
<code>cudaSetupArgument</code>	Configure setup arguments	Kernel execution
<code>cudaLaunch</code>	Launch a kernel	Kernel execution
<code>cudaFree</code>	Free memory on the device	Memory release
<code>cudaMemcpy</code>	Copy data between host and device	Input&Output data transfer

The first three functions with “__” prefix are not meant to be called directly by user code but they are so important that `nvcc` compiler injects them into the source code. Their declaration in `/usr/local/cuda/include/crt/host_runtime.h`

shows us the interface. While the other functions without underline prefix are directly called by user code and they are declared in */usr/local/cuda/include/cuda_runtime_api.h*.

Each wrapper runtime API invokes `ioctl()` system call for sending requests and getting responses from virtual GPU. In this case, `ioctl()` takes the file descriptor of virtual character device as first argument. The second argument is a dedicated device-dependent request code for each runtime API. The third argument is an untyped pointer to the request meta structure which we fill with proper parameters.

Data Transfer Between VMs and VMM. Virtio is a de-facto standard for para-virtualization I/O devices and aims to improve performance of accessing devices on guest OSes over the traditional emulated devices. Virtio abstracts a common set of emulated devices which the VMM exports to the VM via normal PCI devices. To boost the I/O performance, a custom device driver in guest OS communicates with the associated back-end service in VMM. Guest OS writes “guest-physical” addresses to the configure space to inform the VMM of buffer addresses. By simply adding an offset the actual “host-virtual” addresses can be calculated in VMM.

Our proposed implementation, `virtio-vgpu`, consists of a virtual PCI device (`virtio-vgpu-pci`) and a token (`virtio-vgpu-token`) that is logically attached to it. To support `virtio-vgpu`, “`virtio-vgpu-pci`” and “`virtio-vgpu-token`” options should be appended to the QEMU command line when the VM launches. `virtio-vgpu-pci` is interpreted into virtual device for guest OS, while `virtio-vgpu-token` also requires a PEM formatted private key file which stores master key as back-end with the “`keypath`” argument.

`virtio-vgpu` provides a front-end driver in guest OS for forwarding requests and returning the response of CUDA runtime APIs. In more detail, the driver specifies `ioctl` commands for wrapper runtime API calling. All the parameters should be rearranged in buffer with the meta structure and auxiliary data. As the driver is complemented in kernel space, transferred data from user space should be copied to the kernel space. New kernel memory is allocated, filled with content from the user space and then concatenated to the end of buffer. The final buffer is sent to VMM via virtio buffers. The returned buffer is also arranged like this, which contains of meta structure and complementary data. Finally, all essential results are copied to user space.

Besides the front-end driver, the other back-end driver in VMM is responsible for receiving requests, keeping states of objects, executing operations and sending back the responses. The transferred data is analyzed via the information of meta structure located in the front of buffer. As we mention before, VMM responses differently according to dedicated request code from the meta structure. By different request code, VMM calls actual runtime API and forwards back the buffer with the returned value and essential parameters. In addition, VMM also initializes the virtual objects list after validating the authentication of GPU code when dealing with `__cudaRegisterFatBinary` requests. The virtual objects list

manages all pointers to allocated memory, kernel configuration and parameters, events, streams. Still, the device information is collected as well in order to response quickly for the device management, such as `cudaGetDevice` request.

GPU Kernel Authentication. In order to prevent the maliciously modified GPU code compromising the device, our system only allows authenticated GPU code to execute on device. To validate the integrity of GPU kernel, the mechanism is divided into two parts. Firstly, the administrator should obtain the signature of fat binary file from CUDA binary using HMAC-SHA256 signatures in advance. Now the secret key of HMAC is generated from master key. Then those signatures are needed to be converted to base64 for displaying. The fat binary file can be generated from the source code by `nvcc` compiler with the option “`--fatbin`”. Meanwhile VMM maintains a white list to dynamically manage base64 strings of HMAC signatures of GPU code. The white list is implemented as a file appended to **virtio-vgpu-token**.

```
typedef struct {
    int magic;
    int version;
    const unsigned long long* data;
    void *filename_or_fatbins; /* version 1: offline filename,
                               * version 2: array of prelinked fatbins */
} __fatBinC_Wrapper_t;
```

Fig. 2. Fatbin control structure.

Secondly, what confronts us is how to extract the fatbin file from the CUDA binary file using runtime APIs. Luckily, we can utilize some control structure information from CUDA include directory. `__cudaRegisterFatBinary` takes the pointer to fatbin control structure as input, the structure defined in *fatBinaryCtl.h* is shown in Fig. 2. The address of fat binary can be followed by field **data**. Furthermore, the size of fat binary file is controlled by fat binary header structure defined in *fatbinary.h*. Finally, with the address and size, the fat binary file surely is able to be extracted and be transferred to VMM via virtio.

VMM checks the integrity of fat binary file once receiving it from guest OS. By checking whether base64 string converted from HMAC-SHA256 signature of fat binary file exists in while list or not, VMM determines that the fat binary file is authenticated. If the fat binary file is valid, VMM initializes the virtual GPU environment and deal with following requests. Otherwise, VMM rejects the request and reports this abnormal behavior to administrators.

4.3 Performance Evaluation

We assess the performance of the GPU virtualization prototype in comparison to the native machine running on commodity CPU and GPU. Our host OS consists

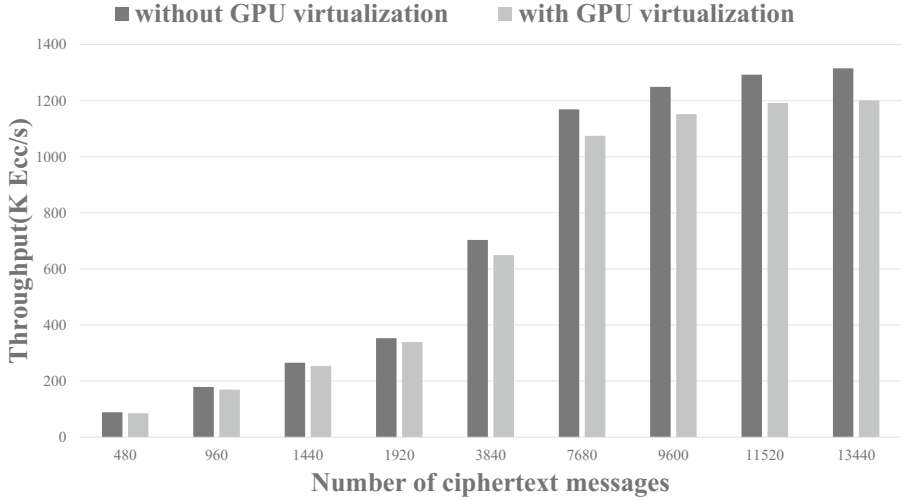


Fig. 3. Performance comparison for Curve25519 with different numbers of requests.

Ubuntu 16.04 x86-64 (kernel v4.4.0), QEMU v1.7.1 and NVIDIA Geforce GTX TITAN BLACK, and the guest OS is CentOS 6.6 (kernel v3.13.7). We implement the wrapper library of CUDA v8.0. To adopt our mechanism for common GPU cryptographic operations, we develop auxiliary runtime API in wrapper library.

The throughput is evaluated via Curve25519 and AES-128 algorithms. Curve25519 is an elliptic curve which is intended to operate at the 128-bit security level. We implement scalar multiplication on Curve25519 by using Montgomery ladder in a constant time. By making full use of the PTX ISA instruction supported by NVIDIA GPUs and making optimization from two aspects, the finite field arithmetic and the curve algorithm, the performance of Curve25519 scalar multiplication has been promoted. For AES, we modify the OpenSSL AES with a 128-bit symmetric key to GPU implementation.

We conduct the experiment to measure the latency incurred by the API remoting approach. In order to measure the impact of our GPU virtualization mechanism on real scene of cryptographic computing, the time imposed by data copying is included from the total time. Figure 3 shows the throughput changes of Curve25519 with and without our proposed mechanism via various request sizes.

As the figure shows, the overhead incurred by GPU virtualization is insignificant for Curve25519. For Curve25519, the average degraded throughput is up to 92% of the original throughput. This implies that our mechanism brings negligible performance degradation for asymmetrical cryptographic operations.

However, there is much lower performance for AES-128, which is up to 70% throughput degradation with 3 MB input data. We tested the performance penalty of kernel execution excluding the time of data copying. The degraded throughput is nearly 98% of native throughput. From our point of view, the

time of data copying takes up most of the time of virtual cryptographic computing. This implies that our solution might be ineffective to frequently request for symmetrical cryptographic operations with large input data.

Note that we have not optimize implementation of our prototype yet. The efficiency of data copying via virtio needs to be optimized in our future work.

5 Security Analysis

In this section, our experiment shows our mechanism is able to protect sensitive information from memory disclosure attacks. To this end, we need to make sure there is no occurrence of keys of Curve25519 in memory space of VM. By using **dump-guest-memory** and **info registers** command in QEMU console, the whole memory image and states of registers of VM can be obtained respectively. Since we have already know the plaintext of secret keys in cryptographic computation, we search the secret key strings and the master key string inside the dump file. Fortunately, it turns out that no binary sequence of any key exists. Hence, our GPU virtualization prototype can effectively prevent private keys from leakage of GPU cryptographic computation.

In order to maintain all loaded secrets on GPU and make sure exclusive control of the GPU, PixelVault forces a CUDA kernel to run indefinitely and consumes all available device resources. As a result, PixelVault is dedicated to a single cryptographic operation. Not to say consuming considerable power, this not only degrades performance but also significantly reduces flexibility of computations. Since GPU registers can not be shared between different threads, PixelVault also increases complexity of GPGPU programming.

Compared with the prior works, our solution has the following advantages:

1. We isolate the master key in host. The secret keys used by GPU-accelerated cryptographic operations are not exposed to attackers in plaintext, but encrypted by master key beforehand. However the encrypted secret keys are only decrypted in host OS. Any accelerating cryptographic operation in VM can not reveal the sensitive information.
2. Any authenticated cryptographic computation application can be executed in the VM with a insignificant degradation of throughput. Experiment results show that performance penalty of API-remoting-based GPU computation for ECC is within 8% of native GPU computation.
3. We provide wrapped CUDA runtime library which keeps the same interface as the original library so that developers do not need to modify applications greatly.
4. Our solution does not depend on the characteristics of GPU hardware. Ignoring the architecture of underlying hardware, it is compatible for multiple products.

6 Conclusion

In this paper, we have proposed the design and implementation of a framework for preventing private keys from leakage in accelerating cryptographic computations utilizing GPU virtualization. By isolating the master key in VMM and establishing authenticated GPU binaries, the real keys are never exposed to the guest OS, so that the compromise of the guest OS will not threaten the secrecy of keys. Moreover, The API-remoting-based GPU virtualization with `virtio` is proved as a high performance computing solutions which allows cryptographic applications within VM to leverage GPU acceleration. Our evaluations show that GPU virtualization incurs a insignificant performance degradation for asymmetric cryptographic algorithm ECC. We also prove that secret keys are never leaked into memory space of VMs. However the major reason of performance degradation incurred by GPU virtualization is data transmission which is a unavoidable problem. Our work currently may not be proper for symmetrical cryptographic operations like AES with large input data.

In the future work, we continue to optimize our prototype with data transmission, lazy calling and implement more runtime APIs. Now our prototype does not support multiplexing and live migration that we intend to extend the prototype with.

References

1. Blass, E.-O., Robertson, W.: Tresor-hunt: attacking CPU-bound encryption. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 71–78. ACM (2012)
2. Di, B., Sun, J., Chen, H.: A study of overflow vulnerabilities on GPUs. In: Gao, G.R., Qian, D., Gao, X., Chapman, B., Chen, W. (eds.) NPC 2016. LNCS, vol. 9966, pp. 103–115. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47099-3_9
3. Duato, J., Pena, A.J., Silla, F., Mayo, R., Quintana-Orti, E.S.: Performance of CUDA virtualized remote GPUs in high performance clusters. In: 2011 International Conference on Parallel Processing, pp. 365–374, September 2011
4. Erb, C., Collins, M., Greathouse, J.L.: Dynamic buffer overflow detection for GPGPUs. In: Proceedings of the 2017 International Symposium on Code Generation and Optimization, pp. 61–73. IEEE Press (2017)
5. Garmany, B., Müller, T.: Prime: private RSA infrastructure for memory-less encryption. In: Proceedings of the 29th Annual Computer Security Applications Conference, ACSAC 2013, pp. 149–158. ACM, New York (2013)
6. Guan, L., Lin, J., Luo, B., Jing, J.: Copker: computing with private keys without RAM. In: NDSS, pp. 23–26 (2014)
7. Guan, L., Lin, J. 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), pp. 3–19. IEEE (2015)
8. Halderman, J.A., et al.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM **52**(5), 91–98 (2009)


9. Harrison, O., Waldron, J.: Efficient acceleration of asymmetric cryptography on graphics hardware. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 350–367. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02384-2_22
10. Hong, C.-H., Spence, I., Nikolopoulos, D.S.: GPU virtualization and scheduling methods: a comprehensive survey. *ACM Comput. Surv. (CSUR)* **50**(3), 35 (2017)
11. Iwai, K., Kurokawa, T., Nisikawa, N.: AES encryption implementation on CUDA GPU and its analysis. In: 2010 First International Conference on Networking and Computing (ICNC), pp. 209–214. IEEE (2010)
12. Jang, K., Han, S., Han, S., Moon, S.B., Park, K.: SSLShader: Cheap SSL acceleration with commodity processors. In: NSDI (2011)
13. Kim, Y., et al.: On-demand bootstrapping mechanism for isolated cryptographic operations on commodity accelerators. *Comput. Secur.* **62**, 33–48 (2016)
14. Kocher, P., et al.: Spectre attacks: exploiting speculative execution. ArXiv e-prints, January 2018
15. Lee, S., Kim, Y., Kim, J., Kim, J.: Stealing webpages rendered on your browser by exploiting GPU vulnerabilities. In: 2014 IEEE Symposium on Security and Privacy (SP), pp. 19–33. IEEE (2014)
16. Lin, J., Luo, B., Guan, L., Jing, J.: Secure computing using registers and caches: the problem, challenges, and solutions. *IEEE Secur. Priv.* **14**(6), 63–70 (2016)
17. Lipp, M., et al.: Meltdown. ArXiv e-prints, January 2018
18. Lombardi, F., Pietro, R.D.: Towards a GPU cloud: benefits and security issues. In: Mahmood, Z. (ed.) *Continued Rise of the Cloud*. CCN, pp. 3–22. Springer, London (2014). https://doi.org/10.1007/978-1-4471-6452-4_1
19. Maurice, C., Neumann, C., Heen, O., Francillon, A.: Confidentiality issues on a GPU in a virtualized environment. In: Christin, N., Safavi-Naini, R. (eds.) *FC 2014*. LNCS, vol. 8437, pp. 119–135. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_9
20. Miele, A.: Buffer overflow vulnerabilities in cuda: a preliminary analysis. *J. Comput. Virol. Hacking Tech.* **12**(2), 113–120 (2016)
21. Müller, T., Dewald, A., Freiling, F.C.: AESSE: a cold-boot resistant implementation of AES. In: *Proceedings of the Third European Workshop on System Security*, pp. 42–47. ACM (2010)
22. Müller, T., Freiling, F.C., Dewald, A.: Tresor runs encryption securely outside RAM. In: *USENIX Security Symposium*, vol. 17 (2011)
23. Pan, W., Zheng, F., Zhao, Y., Zhu, W.-T., Jing, J.: An efficient elliptic curve cryptography signature server with GPU acceleration. *IEEE Trans. Inf. Forensics Secur.* **12**(1), 111–122 (2017)
24. Di Pietro, R., Lombardi, F., Villani, A.: CUDA leaks: a detailed hack for CUDA and a (partial) fix. *ACM Trans. Embedded Comput. Syst. (TECS)* **15**(1), 15 (2016)
25. Shi, L., Chen, H., Sun, J., Li, K.: vCUDA: GPU-accelerated high-performance computing in virtual machines. *IEEE Trans. Comput.* **61**(6), 804–816 (2012)
26. Simmons, P.: Security through amnesia: a software-based solution to the cold boot attack on disk encryption. In: *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 73–82. ACM (2011)
27. Stewin, P., Bystrov, I.: Understanding DMA malware. In: Flegel, U., Markatos, E., Robertson, W. (eds.) *DIMVA 2012*. LNCS, vol. 7591, pp. 21–41. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37300-8_2
28. Szekeres, L., Payer, M., Wei, T., Song, D.: Sok: eternal war in memory. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 48–62. IEEE (2013)

29. Vasiliadis, G., Athanasopoulos, E., Polychronakis, M., Ioannidis, S.: PixelVault: using GPUs for securing cryptographic operations. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1131–1142. ACM (2014)
30. Wartell, R., Mohan, V., Hamlen, K.W., Lin, Z.: Binary stirring: self-randomizing instruction addresses of legacy x86 binary code. In: Proceedings of the 2012 ACM Conference on Computer and Communications security, pp. 157–168. ACM (2012)
31. Zhou, Z., Diao, W., Liu, X., Li, Z., Zhang, K., Liu, R.: Vulnerable GPU memory management: towards recovering raw data from GPU. *Proc. Priv. Enhancing Technol.* **2017**(2), 57–73 (2017)
32. Zhu, Z., Kim, S., Rozhanski, Y., Hu, Y., Witchel, E., Silberstein, M.: Understanding the security of discrete GPUs. In: Proceedings of the General Purpose GPUs, pp. 1–11. ACM (2017)

Full Paper Session IV: Encrypted Computing



Accelerating Integer Based Fully Homomorphic Encryption Using Frequency Domain Multiplication

Shakirah Hashim^(✉)  and Mohammed Benaissa

University of Sheffield, Sheffield, UK
shashiml@sheffield.ac.uk

Abstract. In this paper, the hardware implementation of Integer based Fully Homomorphic Encryption (FHE) is investigated. A new methodology is proposed to speed up the encryption process by optimizing the very large asymmetric multiplications required. A frequency domain approach is adopted for the multiplication using the Number Theoretic Transforms (NTTs) where the strict relationship between the NTT parameters is relaxed to allow for more optimized hardware implementations on FPGA. This is achieved specifically by relaxing the traditional requirement for a simple transform kernel in favour of optimal transform lengths and moduli in terms of the number of overall iterations, suitable data path, and FPGA architecture. It is shown both analytically and via implementation results that the proposed approach yields faster FHE over the integers implementations. Based on the methodology, a proposed hardware architecture with optimized NTT parameters synthesized on Xilinx Kintex-7 FPGA shows 55% and 76% speed improvement for Medium and Large key sizes respectively.

Keywords: Fully Homomorphic Encryption · Number Theoretic Transform
Hardware implementation

1 Introduction

Fully Homomorphic Encryption (FHE) allows a computation to be done on encrypted data (ciphertext) and no decryption is needed prior to any computation, offering thus better privacy [1]. FHE has emerged as a powerful cryptographic tool in recent years as it has been shown to possess both additive and multiplicative homomorphic properties. However, it is still far from practical deployment due to their complexity, mainly due to the huge key size involved. Three variants of FHE: Lattice-Based, Ring Learning with Error (RLWE) and Integer-Based have been an area of active research in recent years to investigate the potentials and limitations of FHE by investigating software [2–5] and hardware [6–9] implementations.

Implementing Lattice-Based FHE in software was initially proposed in [2]; it requires huge key sizes between 17 Megabytes (MB) to 2.3 Gigabytes (GB) with key generation taking from 2.5 s to 2.2 h. Van Dijk *et al.* revised the original FHE scheme and proposed Integer Based FHE [10] where both homomorphic properties are

computed over the integers with the objective of promoting simplicity in its scheme. Later, Coron *et al.* improved this scheme with smaller key sizes of 0.95 Mb to 802 Mb and key generation time between 4.38 s to 43 min [4].

A modulus switching technique was introduced in [5] which allows leveled multiplication on smaller moduli, hence results in smaller public key sizes. In [5], the authors worked on RLWE based FHE, managed to reduce noise growth from quadratic to linear complexity even without modulus switching. Cousins *et al.* introduced the Chinese Remainder Transform (CRT) on Lattice-Based FHE which splits a larger modulus into multiple moduli so that parallelization can be employed on Field-Programmable Gate Array (FPGA) Virtex 6, however extra time is needed for re-conversion from the Montgomery domain to regular integers [11, 12]. Later, Gentry in [13] presented an encryption of 150-bit Advanced Encryption Standard (AES) homomorphically which takes 73.03 s for key generation and 3 Gb memory usage without bootstrapping.

Apart from FHE, recent research also focused on Somewhat Homomorphic Encryption (SHE) [14, 15]. Smart *et al.* in [3] suggested multiple stages of encryption (known as re-encrypt) on larger message sizes rather than single bit proposed originally in [1]; however, key generation still requires more than an hour even for small key size. An improvised version of [3] is done by introducing a Single Instruction Multiple Data (SIMD) implementation in [16], which performs 4.13 times faster re-encryption and 12 times smaller ciphertext than one without SIMD. Also working on SHE, Poppelmann *et al.* [17] showed that Lattice-Based SHE is possible to be deployed on FPGA Spartan-6 with 9063 Number Theoretic Transform (NTT) coefficients multiplication per second, provided NTT parameters are selected appropriately. The recent SHE work is based on Ring-LWE variants and aimed at accelerating the encryption for cloud computing at the FPGA level and also enlarge the NTT coefficients by introducing a 1228-bit modulus [18]. However, the resulting multiplication process was relatively slower than the software implementation with the same NTT size; 26.67 s and 2.98 s respectively. The bottleneck being the memory access.

To accelerate the FHE performance, the authors in [6] exploited the speed of Graphical Processing Units (GPUs) and encrypted 7.68 times faster than standard Central Processing Unit (CPUs). Then, the authors in [19] introduced Integer-Based FHE by batch to reduce the bottleneck on AES encryption. Later, Doroz *et al.* proposed pre-computation of Schönhage Strassen multiplier parameters which allowed FHE encryption to perform better with only 18.1 ms (ms) [20]. Recently, the concept of a re-encryption box was proposed by Roy *et al.* [21] at the hardware level to reduce the effects of growing noise on the ciphertexts. The re-encryption box is also exploited to accelerate the search operation on the encrypted data.

The first hardware implementation for Integer Based FHE was proposed by Cao *et al.* in [8] with two building blocks of a large NTT multiplier and Barrett reduction to speed up FHE on high-end FPGA technology Virtex 7. Their encryption time is 44.72 times faster than software implementation for ‘Large’ key size. Comba scheduling is proposed in [22], by utilizing Digital Signal Processing (DSP) slices for uneven operands to shorten the delays during multiplications while reducing ‘Write to Memory’ operation. Meanwhile, recent research by Cao *et al.* [9] proposed Low Hamming Weight (LHW) design on Virtex 7 to allow simpler multiplications while reducing

hardware usage at the same time. The encryption time of this work outperforming benchmark software implementations by 131 times for ‘Large’ key size, while the encryption time showed by this scheme is between 0.0006 s to 3.317 s, resulting in the best FHE achievement by far with a reasonable speed and small footprint.

Inspired by the significant performance reported with strong potential for improvements, we focused our work on the Integer-Based FHE scheme by Van Dijk *et al.* [10]. The central theme of this scheme is about simplicity. It is easier in terms of parameter selection compared to Lattice-Based while its hardness is based on Greatest Common Divisor (GCD) approximate problem. Furthermore, the sizes of the parameters in Integer-Based FHE are defined clearly in [9], unlike the other variants where only the matrix size is defined rather than bit size.

We propose to accelerate FHE over the integers by adopting frequency domain multiplication using the NTT specifically targeted for FPGAs. The FPGA platform is chosen over custom hardware Application-Specific Integrated Circuits (ASICs) due to the high availability of resources such as DSPs which have dedicated mathematical functions on modern FPGAs.

We followed the seminal work in [2, 5], pronouncing the operands size in four different groups: Toy, Small, Medium and Large as shown in Table 1. At least 150 k to 19 m bits operands are required for the encryption steps which is a large number, hence normal Schoolbook multiplication is no longer efficient. In recent years, there have been many reported ideas by researchers to optimize large number multiplications especially in cryptography; such as Comba [22, 23], Karatsuba [24, 25] and frequency domain conversion methods [14, 26]. The idea of adopting a frequency domain approach on hardware such as in [8, 14, 15, 27, 28] has increasingly gained acceptance as an efficient method to accelerate the multiplication process given its computational complexity being in the order $O(n\log(n))$ for n -bits operand. Researchers in [9, 29, 30] have also shown that NTT hardware implementations outperformed software implementations at certain magnitudes.

Table 1. Test instances for encryption process

Test instances	Bit-length, X_i	Bit-length, B_i	τ
Toy	150 k	936	158
Small	830 k	1476	572
Medium	4.2 m	2016	2110
Large	19.0 m	2556	7659

In this paper, we further advance research in this area by relaxing the strict relationship between the NTT parameters to allow for more optimized hardware implementations on FPGA. To speed up the large integer multiplications required in FHE schemes such as the one proposed in [5], previous research has sought to optimize the multiplication steps within the NTT transform computations by fixing the kernel α to be simply a two or a power of two value [9]. However, such approach tends to impose

restrictions on the possible transform lengths to be deployed, thereby affecting potential optimizations in the overall multiplication process.

In this paper, we propose a different methodology whereby we relax the requirement for a simple kernel in favour of optimal transform lengths and moduli in terms of the number of overall iterations, suitable data path, and FPGA architecture. The kernel multiplications by α 's required for the optimal word lengths and moduli can be easily implemented in the form of Look-Up Tables (LUTs) integral to any FPGA fabrics.

The specific contributions of this paper are summarised as follows:

- A set of NTT parameters that supports large operands for NTT multiplication is proposed.
- Analysis of important hardware design trade-offs; such as the butterfly costs of the NTT building blocks against multiplication iteration for each key group in FHE (Toy, Small, Medium and Large).
- An iterative multiplication method is incorporated to support a small footprint design on hardware while at the same time maximizing the multiplier size to speed-up the overall multiplication process.
- Hardware implementation is validated with results showing improved performance.

The rest of the paper is organised as follow. Section 2 recaps the introduction and mathematical background of FHE over the integers. Our proposed methodology is illustrated in Sect. 3. Section 4 covers the implementation aspects with results given in Sect. 5. The paper concludes with a Conclusion section.

2 Integer Based Fully Homomorphic Encryption

Integer Based FHE needs to perform key generation, encryption and decryption with the additional step of evaluation. Our work in this paper, in line with previously reported implementations [9], is focused solely on the encryption step defined in (1). The work in [9] is workable for binary messages only with message space $\mathcal{Q} = 2\{0, 1\}$; [31] proposed a larger space $\mathcal{Q} > 2$, which means the message can be non-binary with an extended circuit. Their key size is also reduced, although no specific size is reported.

$$c \leftarrow m + 2r + 2 \sum_{i=1}^{\tau} X_i \cdot B_i \text{ mod } X_0 \quad (1)$$

Noted, c is ciphertext; m is a single bit of plaintext binary message with only bit 0 or 1; r is a random signed integer; X_0 is a part of the public key; B_i is a random integer sequence, and X_i is a τ -bits public key sequence with $1 \leq i \leq \theta$. We direct the interested reader to refer to the original work in [5, 10] for details on the parameter selection in (1) and Table 1.

As seen from (1), the FHE encryption step needs two core operational building blocks: (1) Multiplication; and (2) Reduction. These can be designed as individual building block and combined later as a complete process of FHE encryption. Meanwhile, as can be seen from Table 1, both multiplicands X_i and B_i are not symmetrical in size. Multiplicands are also known as operands after this point. Thus, we exploit this

unsymmetrical property to propose a hybrid multiplication approach of Schoolbook and NTT based multiplication. Schoolbook multiplier is employed for the outer iterations whereas the NTT multipliers will be used for the inner multiplications. In fact, employing symmetric multiplication methods for non-symmetric operands leads to significant waste of computational time as well as hardware resources.

2.1 Number Theoretic Transform (NTT) Multiplication

The NTT has been used widely in signal processing for implementing convolution and correlation operations because of its error-free advantages (no rounding or truncation errors) and efficient implementation. Recently there has been a revival of interest in NTTs to be deployed in frequency domain approaches to implementing large operand multiplications required in new offerings in Cryptography. Dai *et al.* in [32] proposed large NTTs of 2^{15} coefficient integrated with CRT in order to accelerate NTRU-based FHE. Meanwhile, diminished-1 NTTs is used for performing SWIFFT hash function in [33] to simplify modular NTTs but is limited to certain modular form such as Fermat primes only. Promising more parallelization, NTT is also widely used in hardware implementation with good performances [8, 34]. The Mathematical representation of an NTT is given in (2). Where $k = 0, \dots, N - 1$ and α is twiddle-factor with the condition of $\alpha^N \equiv 1 \pmod{m}$.

$$X(t) = \sum_{n=0}^{N-1} x(n)\alpha^{nk} \pmod{m} \quad (2)$$

From (2), parameters α, m and N are interdependent. The desirable choice of NTT parameters traditionally involved [35]:

- α to be selected as two or a power of two so that the exponentiation operations required can be implemented as shift operation;
- N to be highly composite, a power of two if possible so that efficient NTT type algorithms can be employed
- m has a special form so that reduction can be a simple operation.

In this paper, we use Classical Modular NTT, with each operation is bounded by ring Z_m where m is moduli. Algorithm 1 describes NTT multiplication steps with 4 underlying steps; Forward Transform, Pointwise Multiplication, Inverse Transform and Carry Accumulation.

Algorithm 1 Number Theoretic Transform (NTT)

```

1: Let  $\alpha$  be a primitive  $n$ -th root of unity in  $m$  and  $b$  is word size
2: Let  $\mathbf{x} = (x_0, \dots, x_{(n/2)-1}, 0 \dots 0)$ ,  $\mathbf{y} = (y_0, \dots, y_{(n/2)-1}, 0 \dots 0)$  and  $\mathbf{z} = (z_0, \dots, z_{n-1})$ 
3: Input:  $x, y, \alpha$ 
4: Output:  $z = x * y$ 
5: Precompute:  $\alpha^i$  where  $i = 0, 1, \dots, n-1$ 
6:   for  $i$  from 0 to  $n-1$ ;
7:      $X = \text{NTT}_{\alpha_i}^m(x_i)$            //Forward Transform
8:      $Y = \text{NTT}_{\alpha_i}^m(y_i)$            //Forward Transform
9:   end for
10:  $Z = X (*) Y \text{ mod } m$            //Pointwise Multiplication
11: for  $i$  from 0 to  $n-1$ ;
12:    $z = \text{INTT}_{\alpha_i^{-1}}^m(Z_i)$      //Inverse Transform
13: end for
14: for  $i$  from 0 to  $n-1$ ;
15:    $z = \sum_{i=0}^{N-1} (z_i \ll (i \cdot b))$  //Carry Accumulation
16: end for
17: Return  $z$ 

```

3 Proposed Methodology

The efficiency of NTT designs as explained before is related closely to the trade-off of its three key inter-related parameters, namely the kernel α , the transform length N and the modulus m . In this paper we stipulate that in the context of FHE where very large multiplications of asymmetrical operands are required, a methodology that allows more flexibility in terms of transform length, offers better scope for improving overall FHE performance on modern FPGA platforms. The proposed methodology is more efficient than traditional methodologies driven by overcoming the complexity of the multiplications by the kernel of the transform at the detriment of the transform length. In this case, the impact of the transform length on overall performance is far more significant than that of the kernel multiplication within the NTT. This is because, the long multiplier unit will be able to cater for larger operand size, thus minimize the number of partial product iterations. As a result, multiplication complexity can be reduced specifically for asymmetric operands. A study of NTT parameters and its optimization is discussed in the next section.

3.1 NTT Parameters Optimization

The central parameter to be optimized is the NTT length as large NTT length can facilitate larger operands, by relaxing the kernel α restriction. The choice of modulus needs a specific consideration, as explained later so that every operation during the NTT over the defined ring is optimized for the targeted hardware. Importantly, the NTT

coefficient must be within the dynamic range b , as expressed in (3) to ensure no overflow error. More details of dynamic range is in [36].

$$\frac{N}{2}(b - 1)^2 < m \tag{3}$$

To illustrate the improvements in operand sizes achieved by the proposed approach we report in Table 2 the comparison between two types of moduli, Solinas and Fermat (F_6); they are 64 bits and 65 bits moduli respectively. Solinas 1 and F_6 1 show the NTT parameter set without optimization, whereas Solinas 2 and F_6 2 show these parameters with our proposed optimization. The optimization is done by enlarging the NTT length as well as relaxing the kernel restriction. As a result, both Solinas 2 and F_6 2 result in much larger multiplier sizes of 1792 bits and 3072 bits which correspond to almost double the length.

Table 2. Comparison between Solinas and Fermat moduli NTT parameters

	Solinas		Fermat	
	Solinas 1 [37]	Solinas 2	Fermat F_6 1 [38]	Fermat F_6 2
N -point	64	128	128	256
Twiddle-factor α	8	$2^{49} - 2^1$	2	$2^{33} - 2^1$
Dynamic range b	28	28	24	24
Multiplier size	896 bits	1792 bits	1536 bits	3072 bits
Modulus m	$2^{64} - 2^{32} + 1$		$2^{64} + 1$	
Reduction cost	1 shift, 2 addition, 2 subtraction		2 addition, 1 subtraction	

Let $y \text{ mod } p$ where $y = 2^{96}a + 2^{64}b + 2^{32}c + d$, a 128 bits integer. The Solinas reduction can be simplified as (4).

$$2^{32}(b + c) - a - b + d \tag{4}$$

Algorithm 2 is used for Special form modulus, of $2^{n-1} \pm 1$ as proposed in [39]. We used this Algorithm for Fermat F_6 1 and Fermat F_6 2 reduction.

Algorithm 2 Special form modulus reduction [39]

- 1: Let $p = 2^m \pm 1$
 - 2: **Input:** x, p
 - 3: **Output:** $y = x \pmod{p}$
 - 4: $y_1 = x[m - 2: 0] + x[2m - 1: 2m - 2] - x[2m - 3: m - 1]$
 - 5: **if** $y_1 < 0$
 - 6: $y = y_1 + p;$
 - 7: **else**
 - 8: $y = y_1$
 - 9: **end if**
 - 10: **Return** y
-

In terms of reduction's complexity cost, Solinas just needs shift, addition and subtraction. Also, the Solinas form lends itself to efficient FPGA implementation. As the goal of this work is to design a large multiplier on a targeted FPGA, then, Solinas 2 was chosen as the optimal modulus as it covers an acceptable number of operands; 1792 bits and the 64 bits modulus is an optimal fit in terms of a single word. Although $F_6 2$ can cover larger operands of 3072 bits, its 65 bits modulus needs more than a single word operand, which is not optimal for hardware implementation. Even if the diminished-1 number system can be adopted to handle 65 bits modular operation as suggested in [40], the conversion to and from this number system is costly and can become a performance bottleneck in particular for the special case of the zero detection.

Cost Analysis

We first analyzed the operational cost of the NTT block for Solinas 1 and Solinas 2 individually and later we analyzed the cost for overall multiplication during the encryption. For a fair comparison, we presume α for Solinas 1 and Solinas 2 are pre-computed over the Solinas modulus beforehand and stored in LUTs as 64-bits Read-only Memories (ROMs). This was also done before in [27, 41] with the same purpose of speeding-up the kernel multiplication process.

In our work, $64 \times \frac{N}{2}$, pre-computed operands are needed to be stored in the LUTs which is relatively small compared to the available LUTs of the targeted hardware, Kintex 7. Exponentiation by α during the Butterfly operation in (5) can be replaced with a 'Read' operation which is obviously faster than computing exponential α by using an algorithm.

Meanwhile, as NTTs over the ring has a symmetrical root of unity, then it benefits the NTT implementation because the same table also can be used for retrieving α^{-1} for Inverse NTT (INTT) [36]. This way, only N multiplications are required for each transform. As the overall NTT multiplication building block has 2 forward and 1 inverse transforms, then $3N$ multiplications are required. The same goes for the 'Read' operations during the NTT multiplication which is $3 \lfloor \frac{N}{2} \log_2 N \rfloor$.

$$X_i = A_i + \alpha^i B_i, \quad X_{i+\frac{N}{2}} = A_i - \alpha^i B_i \quad (5)$$

The NTT multiplier size n_c can be determined from (6). Division by two is because we use Zero-padded convolution, means only $\frac{N}{2}$ coefficients are employable, and the other $\frac{N}{2}$ appended as zeros.

$$n_c = \frac{N \times b}{2} \quad (6)$$

Table 3 shows the comparison between Solinas 1 and Solinas 2, specifically in terms of operations during the NTT and the space required to store the precomputed operands. As illustrated in Table 3, the Butterfly, 'Read' operation and Addition/Subtraction dominate the cost in Solinas 2, which as expected are higher than Solinas 1, as Solinas 2 caters for larger NTT points. Solinas 2 also requires more LUTs space to store pre-computed α . Crucially though Solinas 2 has the largest NTT points among the similar work done previously in [28, 42].

Table 3. Solinas 1 vs Solinas 2

	Solinas 1 [38]	Solinas 2
Butterfly B_u	576	1344
'Read' operation	576	1344
Point multiplication	64	128
Addition/Subtraction	1152	2688
Precomputed operands in LUTs	32 of 64 bits	64 of 64 bits
NTT multiplier size n_c	896 bits	1792 bits

Next, we analyze the entire multiplication, but first we explain how the multiplication building block works during the FHE encryption. As discussed earlier, the NTT multiplier blocks are used for computing the partial products whereas accumulation is completed using a Schoolbook method. In symmetric operands (n -bits) of the Schoolbook method, n^2 multiplication and $2n - 1$ accumulations are needed. However, as in our case asymmetric multiplication is required and the partial products are completed by the NTT multiplier block; then assumption is made that a partial product iteration P_i represents the number of multiplications as determined in (7). Meanwhile, accumulation A_i in (8) represents the number of additions required for accumulating the partial products. Given two operands of asymmetrical size a (n_a bits) and b (n_b bits) with the multiplier size of n_c -bit.

$$P_i = \left\lceil \frac{n_a}{n_c} \right\rceil \times \left\lceil \frac{n_b}{n_c} \right\rceil \quad (7)$$

$$A_i = \left\lceil \frac{n_a}{n_c} \right\rceil + \left\lceil \frac{n_b}{n_c} \right\rceil \quad (8)$$

Figure 1 explains graphically the impact of multiplier size towards partial product iteration and accumulation. Let a and b , the asymmetric operands of 32-bits and 16-bits respectively. Two different multipliers 8-bit and 16-bit are used to show the relationship between the multiplier size and the complexity of multiplication. 32 bits operand is chunked into the multiple blocks depending on multiplier size. The accumulation chain relies on the partial product iteration. For example, an 8-bit multiplier requires 8 partial product iterations and 5 accumulation chains whereas a 16-bit multiplier only consumes 2 partial product iterations and 2 accumulation chains. Essentially, fewer iterations are needed for larger multipliers while long carry accumulation chains also can be minimized.

We analyze the complexity of the multiplication building block, during the FHE encryption with different key sizes Toy, Small, Medium and Large as illustrated in Table 4. P_i and A_i are obtained from Eqs. 7, and 8 respectively. We also include the Butterfly cost B_i in Table 4 which corresponds to the number of butterflies involved during the NTT multiplication to perform FHE encryption as shown in (9). The values of P_i , A_i and B_i in Table 4 represent the overall costs and complexity of the multiplication during the FHE encryption.

$$B_i = B_u \times P_i \tag{9}$$

As can be seen from Table 4, if the multiplier is large enough to cover the operands b_i in a minimum NTT block, then the partial product iterations and accumulation are reduced significantly. For example, Toy operand b_i can fit in a single NTT block of Solinas 2. However, for Solinas 1, operand b_i does not fit a single NTT block, instead 2 NTT block iterations are needed, thus, complicates the multiplication process quadratically.

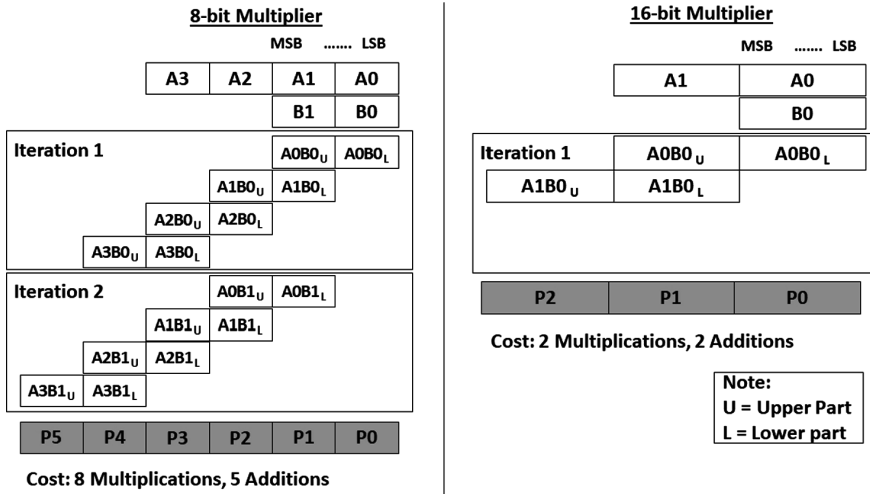


Fig. 1. 8-bit multiplier vs 16-bit multiplier

Overall, the number of partial product iterations (P_i) and accumulations (A_i) in Solinas 2 is reduced drastically compared with Solinas 1. In fact, the butterfly cost in Solinas 2 is also much lower than Solinas 1 despite Solinas 2 incurring a larger butterfly cost than Solinas 1 in a single multiplier block.

Based on this analysis, we confirm that choosing appropriate multiplier size can significantly reduce the multiplication building block complexity and therefore by relaxing the kernel restriction to enable longer length NTT, the overall complexity cost of the multiplication building blocks is reduced significantly.

Also, from the complexity analysis in Table 4, our parameter optimization using Solinas 2 shows a significant improvement compared to Solinas 1. For that reason, we conclude that Solinas 2 is more efficient for large asymmetric operands. This is due to the large size of the multiplier which leads to small partial product iterations and short carry chain. In fact, Solinas 2 also costs fewer butterflies, hence reduce entire multiplication complexity.

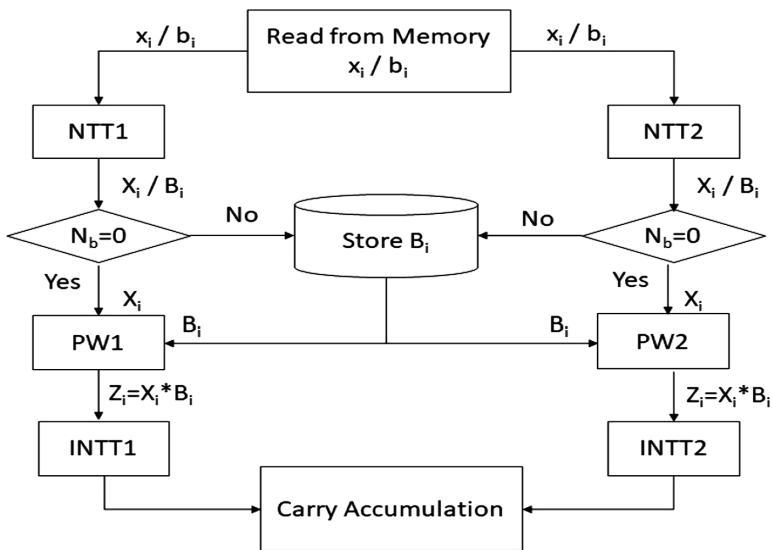
Table 4. Complexity costs of Solinas 1 and Solinas 2

Key size	P_i		A_i		B_i	
	Solinas 1	Solinas 2	Solinas 1	Solinas 2	Solinas 1	Solinas 2
Toy	336	84	170	85	193536	112896
Small	1854	464	929	465	1067904	623616
Medium	14064	4688	4691	2346	8100864	6300672
Large	63618	21206	21209	10605	36643968	28500864

4 The Architecture of NTT Multiplier

Labview FPGA 15 is being used for this hardware implementation, targeted to Xilinx Kintex-7 XC7K160T FPGA device and Xilinx Vivado 2014.4 compiler. Given the size of the operands needed, it is assumed that Block Random Access Memory (BRAMs) is used and sufficient to store X_i and B_i as multiple data chunks where each chunk is b bits size.

The architecture of the NTT Multiplier is depicted in Fig. 2. Initially, both NTT1 and NTT2 are used to transform the B_i operands. After the B_i operands are completely transformed into frequency domain, they are stored in a BRAM B_i . Next, X_i are transformed into frequency domain using NTT1 and NTT2. This also means for each iteration; the NTT block can cover $2n_c$ bits. Then, pointwise (PW) multiplication takes place in parallel by 2 PW units; PW1 and PW2 have 128 points each. During pointwise multiplication, X_i is fed on the fly from both NTT1 and NTT2 outputs, whereas B_i is read from BRAM Y . The output of PW1 and PW2 then are loaded into INTT1 and

**Fig. 2.** The proposed NTT multiplier architecture

INTT2 respectively. The proposed design is pipelined, so after the INTT takes place, then the following output of INTT is generated at the following clock cycle. The product is then loaded into the accumulation unit for addition and carry management. This unit merely involves shifting and addition.

In the case where B_i does not fit into a single NTT unit, then pointwise multiplication should be done iteratively. For example, operands B_i for Medium and Large exceed the multiplier size as they need two NTT blocks; so pointwise multiplication must undergo 2 iterations to complete the multiplication for both blocks, hence more clock cycles required for this case.

5 Results and Discussion

The synthesis result for our proposed NTT Multiplier is within the available resources of the targeted hardware Kintex-7 as seen in Table 5. As can be seen, registers and LUTs are same for all key sizes Toy, Small, Medium and Large. This has happened because the same NTTs unit is being used for each group. The latency is different due to the number of iteration for each group is different. Meanwhile, BRAMs represent an amount to store the operands X_i and B_i as well as the final results after the reduction.

Table 5. Synthesis results for proposed NTT multiplier

	Toy	Small	Medium	Large
Registers	18462	18462	18462	18462
LUTs	26328	26328	26328	26328
BRAMs	41	209	526	702
Freq (MHz)	165.21	164.69	161.00	154.44

The latency in Table 6 is calculated using the clock cycles count and the synthesis design frequencies which is generated by the tools. As the timing for both the multiplication and reduction building block are obtained, then the encryption time Enc_t can be computed as (10).

$$Enc_t = (Group\ 1\ timing \times \tau) + (2 \times Group\ 2\ timing) \quad (10)$$

From (10), the first bracket refers to multiplication timing whereas the second bracket refers to reduction timing. Note that we used Barrett reduction which also

Table 6. Latency and timing for proposed NTT multiplier of each group

Key size	Latency	Timing (ms)	Group2	Latency	Timing (ms)
Toys	4542	0.027	Toys2	4542	0.027
Small	4922	0.030	Small2	4922	0.030
Medium	6802	0.042	Medium2	12246	0.076
Large	15256	0.100	Large2	29154	0.0189

utilized the same NTT building blocks with different operands [9]. Multiplication by two for the reduction building block is because the Barrett reduction needs two large multiplications [43]. The Encryption time of each group is presented in Table 6.

We also compared our result with previous research [9] in Table 7. As can be seen, our design outperforms [9] for the Medium and Large groups. This proves that our design manages to reduce multiplication complexity specifically for large operands such as Medium and Large. Although [9] performs better in Toy and Small, but the encryption time of our design shows that it does not increase gradually from Toy to Large. We notice that our design is not efficient for Toy and Small because the operand B_i just utilized 20% and 41% out of full NTT blocks respectively. This can be improved in the future by designing a scalable design which can be flexible depending on size of operands.

Table 7. Encryption Time of our proposed design and previous research [9]

Key size	Encryption time (s)		
	Proposed design	LHW [9]	Low latency [9]
Toy	0.004	0.001	0.003
Small	0.017	0.011	0.056
Medium	0.089	0.198	1.000
Large	0.770	3.317	16.595

6 Conclusion

In this paper, we proposed a new methodology to speed up the large modular multiplications required in FHE schemes in frequency domain using NTTs. The methodology is based on relaxing the strict relationship between the NTT parameters imposed by having a simple transform kernel. In our approach, more emphasis is put on the transform length as it was shown that this parameter has more effect on overall hardware performance. Both Analytical and implementation presented in this paper show that the proposed methodology leads to improved large NTT multiplication. In fact, our Optimized NTT Multiplier is 55% and 76% faster than [9] for Medium and Large group respectively. The results attained illustrate that FHE encryption time is improved. Further enhancements can be carried out by deploying several NTT blocks in parallel.

References

1. Gentry, C.: A fully homomorphic encryption scheme. Ph.D thesis, Stanford University (2009)
2. Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_9

3. 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
4. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_28
5. Coron, J.-S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_27
6. Wang, W., Hu, Y., Chen, L., Huang, X., Sunar, B.: Accelerating fully homomorphic encryption using GPU. In: High Performance Extreme Computing (HPEC), pp. 1–5. IEEE (2012)
7. Wang, W., Huang, X.: FPGA implementation of a large-number multiplier for fully homomorphic encryption. In: International Symposium on Circuits and Systems, pp. 2589–2592. IEEE (2013)
8. Cao, X., Moore, C., O’Neill, M., O’Sullivan, E., Hanley, N.: Accelerating fully homomorphic encryption over the integers with super-size hardware multiplier and modular reduction. <http://eprint.iacr.org/2013/616>
9. Cao, X., Moore, C., Oneill, M., Osullivan, E., Hanley, N.: Optimised multiplication architectures for accelerating fully homomorphic encryption. IEEE Trans. Comput. **65**, 2794–2806 (2016)
10. 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
11. Cousins, D.B., Rohloff, K., Peikert, C., Schantz, R.: SIPHER: scalable implementation of primitives for homomorphic encryption—FPGA implementation using Simulink. In: High Performance Extreme Computing Conference (2011)
12. Cousins, D.B., Rohloff, K., Peikert, C., Schantz, R.: SIPHER: an update on SIPHER (Scalable Implementation of Primitives for Homomorphic EncRyption) - FPGA implementation using Simulink. In: High Performance Extreme Computing Conference (2012)
13. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_49
14. Öztürk, E., Doröz, Y., Sunar, B., Savaş, E.: Accelerating somewhat Homomorphic Evaluation using FPGAs. IACR Cryptology EPrint Archive, 1–15, <https://eprint.iacr.org/2015/294>
15. Doröz, Y., Öztürk, E., Savaş, E., Sunar, B.: Accelerating LTV based homomorphic encryption in reconfigurable hardware. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 185–204. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_10
16. Smart, N.P., Vercauteren, F.: Fully homomorphic simd operations. Cryptology ePrint Archive, Report 2011/133 (2011), <http://eprint.iacr.org/2011/133>
17. Doröz, Y., Öztürk, E., Savaş, E., Sunar, B.: Accelerating LTV based homomorphic encryption in reconfigurable hardware. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 185–204. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_10

18. Roy, S.S., Jarvinen, K., Vliegen, J., Vercauteren, F., Verbauwhede, I.: HEPCloud: an FPGA-based multicore processor for FV somewhat function evaluation. *IEEE Trans. Comput.* (2018)
19. Cheon, J.H., et al.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, Phong Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_20
20. Doröz, Y., Öztürk, E., Sunar, B.: Accelerating fully homomorphic encryption in hardware. *IEEE Trans. Comput.* **64**(6), 1509–1521 (2015)
21. Roy, S.S., Vercauteren, F., Vliegen, J., Verbauwhede, I.: Hardware assisted fully homomorphic function evaluation and encrypted search. *IEEE Trans. Comput.* **66**(9), 1562–1572 (2017)
22. Moore, C., O’Neill, M., Hanley, N., O’Sullivan, E.: Accelerating integer-based fully homomorphic encryption using Comba multiplication: In: *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation*. IEEE (2014)
23. Großschädl, J., Avanzi, R.M., Savas, E., Tillich, S.: Energy-efficient software implementation of long integer modular arithmetic. *Cryptograph. Hardw. Embedded Syst.* **04**(104), 75–90 (2005)
24. Bos, J.W.: High-performance modular multiplication on the cell processor. In: Hasan, M.A., Hellesteth, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 7–24. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13797-6_2
25. Basu Roy, D., Mukhopadhyay, D.: An efficient high speed implementation of flexible characteristic-2 multipliers on FPGAs. In: Rahaman, H., Chattopadhyay, S., Chattopadhyay, S. (eds.) VDAT 2012. LNCS, vol. 7373, pp. 99–110. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31494-0_12
26. Longa, P., Naehrig, M.: Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 124–139. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_8
27. Liu, Z., Seo, H., Sinha Roy, S., Großschädl, J., Kim, H., Verbauwhede, I.: Efficient ring-LWE encryption on 8-bit AVR processors. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 663–682. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_33
28. Emmart, N., Weems, C.: High precision integer multiplication with a GPU. In: *IEEE International Symposium on Parallel and Distributed Processing Workshops and Ph.D Forum*, pp. 1781–1787 (2011)
29. Jayet-Griffon, C., Cornélie, M. A., Maistri, P., Elbaz-Vincent, P., Leveugle, R.: Polynomial multipliers for fully homomorphic encryption on FPGA. In: *2015 International Conference on ReConfigurable Computing and FPGAs*. IEEE (2016)
30. Chen, D.D., Yao, G.X., Cheung, R.C.C., Pao, D., Koç, Ç.K.: Parameter space for the architecture of FFT-Based montgomery modular multiplication. *IEEE Trans. Comput.* **65**(1), 147–160 (2016)
31. Nuida, K., Kurosawa, K.: (Batch) fully homomorphic encryption over integers for non-binary message spaces. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 537–555. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_21
32. Dai, W., Doroz, Y., Sunar, B.: Accelerating NTRU based homomorphic encryption using GPUs. In: *2014 IEEE High Performance Extreme Computing Conference, HPEC 2014*. IEEE (2014)
33. Györfi, T., Creț, O., Borsos, Z.: Implementing modular FFTs in FPGAs - a basic block for lattice-based cryptography. In: *16th Euromicro Conference on Digital System Design*, pp. 305–308 (2013)

34. Reddy, N., Amarnath, D., Srinivasa, Rao, J., Suman, V.: Design and simulation of FFT processor using radix-4 algorithm using FPGA. *Int. J. Adv. Sci. Technol.* **61**, 53–62 (2013)
35. Burrus, C., Eschenbacher, P.: An in-place, in-order prime factor FFT algorithm. *IEEE Trans. Acoust. Speech Sign. Process.* **29**(4), 806–817 (1981)
36. Brassard, G., Paul, B.: *Algorithmics: Theory and Practice*. Prentice Hall, Upper Saddle River (1988)
37. Solinas, J.A.: *Generalized mersenne numbers*. Faculty of Mathematics, University of Waterloo (1999)
38. Kalach, K., David, J.P.: Hardware implementation of large number multiplication by FFT with modular arithmetic. In: *The 3rd International Conference on IEEE-NEWCAS 2005*, pp. 267–270. IEEE (2005)
39. Zimmermann, R.: Efficient VLSI implementation of modulo $(2^{\sup n} / \text{spl} + \text{plusmn} / 1)$ addition and multiplication. In: *Computer Arithmetic Proceedings 14th IEEE Symposium*, pp. 158–167. IEEE (1999)
40. Leibowitz, L.: A simplified binary arithmetic for the fermat number transform. *IEEE Trans. Acoust. Speech Sign. Process.* **24**(5), 356–359 (1976)
41. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange—a new hope. In: *USENIX Security Symposium*, vol. 2016 (2016)
42. Cao, X., Moore, C.: New integer-FFT multiplication architectures and implementations for accelerating fully homomorphic encryption. *IACR Cryptology EPrint Archive* 2013/624 (2013)
43. Barrett, P.: Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko, A.M. (ed.) *CRYPTO 1986*. LNCS, vol. 263, pp. 311–323. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_24



Comparison-Based Attacks Against Noise-Free Fully Homomorphic Encryption Schemes

Alessandro Barenghi , Nicholas Mainardi  , and Gerardo Pelosi 

Department of Electronics, Information and Bioengineering – DEIB,
Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{alessandro.barenghi,nicholas.mainardi,gerardo.pelosi}@polimi.it

Abstract. Homomorphic Encryption provides one of the most promising means to delegate computation to the cloud while retaining data confidentiality. We present a plaintext recovery attack against fully homomorphic schemes which have a polynomial time distinguisher for a given fixed plaintext, and rely on the capability of homomorphically compare a pair of encrypted integer values. We improve by a constant factor the computational complexity of an exhaustive search strategy, which is linear in the recovered plaintext value, and show that it significantly increases the number of recoverable plaintexts. We successfully validate our attack against two noise-free fully homomorphic encryption schemes, which fulfill the mentioned requisite and were claimed to be secure against plaintext recovery attacks.

Keywords: FHE · Noise-free schemes · Plaintext recovery attack

1 Introduction

Fully Homomorphic Encryption (FHE) is a powerful cryptographic primitive, which allows to perform computation on encrypted data, retaining the correctness of the computation once the result is decrypted. The idea of FHE was first proposed by Rivest in 1978 with the name of *Privacy Homomorphisms* [20]. Designing a FHE scheme remained an open problem for three decades, during which only partially homomorphic encryption (PHE) schemes, which allow to perform only a set of operations (e.g., additions), or SomeWhat Homomorphic Encryption (SWHE) schemes, which allow to perform only a limited number of additions and multiplications, were proposed. In 2009 Gentry [10] proposed the first FHE scheme, allowing to perform an unbounded number of additions and multiplications on encrypted data. Despite the low computational efficiency, FHE has gained attention as it provides a way to outsource computation on private data to a third party such as a cloud-hosted service without revealing any information neither about the data involved in the computation nor about its result, since it is decrypted by the client alone, differently from other primitives

such as Secure Multi-Party Computation [24] or Functional Encryption [1]. Since Gentry’s seminal work, several schemes achieving better performances were proposed [6, 7, 9, 12], as well as new techniques to speed up homomorphic computations, such as *batching* [11, 21]. Nevertheless, FHE schemes still have two practical concerns to be solved before wide adoption is possible: (i) ciphertext expansion and (ii) the computational overhead imposed on homomorphic operations to preserve the correctness of the decrypted result. Indeed, the ciphertext space of SWHE/FHE schemes is consistently larger than the plaintext one, therefore even a single operation on ciphertexts is quite time consuming. The preservation of the correctness of the decrypted result needs to cope with a certain amount of randomness, typically called *noise*, that is added to the ciphertext when processing it. The amount of noise cannot be too high, lest a decryption failure occurs. Unfortunately, each homomorphic operation, especially multiplication, increases the amount of noise in the ciphertexts. Therefore, after a while, the computation must be halted (as in SWHE schemes), or a procedure to refresh the ciphertext, i.e., decrease the noise without decrypting, must be run. Such a procedure, introduced by Gentry in his original scheme [10], is called *bootstrapping* and it allows to transform a SWHE scheme, satisfying certain constraints, in a FHE one. However, this procedure is quite cumbersome, and needs to be periodically performed, slowing down the overall computation. To overcome this burden, alternative noise management techniques have been proposed, such as modulus switching [5] and scale-invariant schemes [2, 3]. Acknowledging the difficulties in noise handling, some noise-free schemes have also been proposed: their ciphertexts have no noise, thus an unbounded number of operations can be performed without any costly noise management technique being involved. Nevertheless, while common noisy SWHE/FHE schemes are based on well-known and scrutinized mathematical problems, such as the Learning With Errors problem [16], noise-free schemes usually rely on less common algebraic trapdoors, which typically do not have widely scrutinized reductions to hard problems. Indeed, Liu in [15] proposed a noise-free FHE scheme, based on the *approximate greatest common divisor* problem, that was subsequently broken in [23]. Kipnis in [14] proposed a FHE scheme based on commutative rings, provably secure against ciphertext-only attacks; however knowing two plaintext-ciphertext pairs is sufficient to break the scheme [22]. Li in [13] proposed to employ non commutative rings to build FHE schemes, while Nuida [18] introduced a framework to construct FHE schemes based on group presentations obfuscated by Tietze transformations. The open challenge with schemes in [13, 18] resides in the definition of a mapping between integer plaintext values and the elements of the mentioned algebraic structures, without losing neither security guarantees nor homomorphic capabilities. Lastly, Wang in [23] introduced two noise-free octonion-based FHE schemes (called *OctoM* and *JordanM*) with trapdoors based on solving quadratic modular equations (with a composite modulus) and proved their security in a ciphertext-only scenario. Due to this property, they are, to the best of our knowledge, the only noise-free FHE schemes suitable for practical usage.

While in general the homomorphic capabilities of a cryptosystem do not weaken the security guarantees per se, they may increase the adversarial power, if combined with other vulnerabilities. The advantages provided by homomorphic capabilities to the attacker were discussed in [4], focusing on the so-called *linearly decryptable* schemes, i.e., cryptosystems whose decryption function can be expressed as a *dot product* between key and ciphertext values represented in a multi-dimensional vector space. Linearly decryptable schemes usually employ a significant amount of noise to hinder Known Plaintext Attacks (KPAs). Nevertheless, in [4] the authors shown that if the scheme can homomorphically evaluate the majority function, then a KPA becomes practically viable. Moreover, in [23] the authors introduced, for linearly decryptable schemes, an algorithm to determine if the plaintext corresponding to a given ciphertext is equal to the integer value 1. We remark that noise free *OctoM* and *JordanM* FHE schemes are linearly decryptable, and thus affected by the aforementioned issues.

Contributions. We present a plaintext recovery attack, against FHE schemes having plaintexts in \mathbb{Z}_n , with $n > 2$, and where it is possible to devise an efficient algorithm able to determine if a generic ciphertext under a given key k is the encryption of a fixed plaintext m , which we denote as *m-distinguisher*. Although, to the best of our knowledge, such a distinguisher has been proposed for linearly decryptable schemes only, our attack will be applicable to any FHE scheme for which such a distinguisher can be found. Our attack, which is performed in a ciphertext-only scenario, leverages the capability to homomorphically compare two encrypted integer values, obtaining a computational complexity which is linear in the plaintext integer value being recovered and improving over an exhaustive search strategy by a significant constant factor. We successfully validate the proposed attack against two linearly decryptable noise free octonion-based cryptosystems [23] (*OctoM* and *JordanM*), which were claimed to be computationally secure in a ciphertext-only attack scenario. Furthermore, we apply our attack to retrieve enough plaintexts from the said schemes so that mounting a KPA to recover the key becomes viable.

2 Preliminaries

Definition 1 (Negligible Function). *A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if, for every univariate positive polynomial, $\text{poly}(x) \in \mathbb{R}[x]$, there exists an integer $c > 0$ such that $\forall x > c$, $|\epsilon(x)| \leq \frac{1}{\text{poly}(x)}$.*

Definition 2 (Indicator Function). *Given a set S and a subset $A \subseteq S$, the indicator function of the elements of A over the ones included in S is defined as: $\mathbb{1}_A : S \rightarrow \{0, 1\}$, where $\mathbb{1}_A(x) = 1$ if $x \in A$, 0 otherwise.*

2.1 Homomorphic Encryption Algorithms

Our definition of Fully Homomorphic Encryption follows [7], without constraining the encryption function to deal with a single bit at time.

An homomorphic encryption (HE) scheme specifies three sets: \mathcal{M} , \mathcal{C} and \mathcal{F} . The set of plaintexts \mathcal{M} usually coincides with the set of integer values ranging from 0 to $n - 1$, with $n > 2$, assumed to be the representatives of the residue classes modulo n , i.e., $\mathbb{Z}_n \equiv \mathbb{Z}/n\mathbb{Z}$. The ciphertext space \mathcal{C} includes elements with an algebraic representation that depends on the specific HE scheme at hand. The set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_n[x_1, x_2, \dots, x_a]$, with $a \geq 1$ and degree greater or equal to zero, defines the functions that the HE scheme at hand allows to be evaluated. That is, each of these polynomials computes a function $f : \mathcal{M}^a \rightarrow \mathcal{M}$, $a \geq 1$ over the plaintexts, and is also referred to as an *arithmetic circuit* composed by gates performing multiplications and additions in \mathbb{Z}_n . We provide the definition of an HE scheme starting from an asymmetric HE scheme, and describe a symmetric one by difference.

Definition 3 (Public-key Homomorphic Encryption Scheme). *A public-key Homomorphic Encryption scheme is defined as a tuple of four polynomial time algorithms $\langle \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval} \rangle$:*

- **Key Generation.** $\langle sk, pk, evk \rangle \leftarrow \text{KeyGen}(1^\lambda)$ is a probabilistic algorithm that, given the security parameter λ , generates the secret key sk , the public key pk and the public evaluation key evk .
- **Encryption.** $c \leftarrow \text{Enc}(pk, m)$ is a probabilistic algorithm that, given a message $m \in \mathcal{M}$ and the public key pk , computes a ciphertext $c \in \mathcal{C}$
- **Decryption.** $m \leftarrow \text{Dec}(sk, c)$ is a deterministic algorithm that, given a ciphertext $c \in \mathcal{C}$ and the secret key sk , outputs a message $m \in \mathcal{M}$
- **Evaluation.** $c \leftarrow \text{Eval}(evk, f, c_1, c_2, \dots, c_a)$ is a probabilistic algorithm computing a ciphertext $c \in \mathcal{C}$, using an arithmetic circuit $f \in \mathcal{F}$ with $a \geq 1$ inputs, the ciphertexts c_1, c_2, \dots, c_a , and the evaluation key.

The following properties must hold:

- **Decryption Correctness.** $\forall m \in \mathcal{M} : \text{Dec}(sk, \text{Enc}(pk, m)) = m$.
- **Evaluation Correctness.** $\forall f \in \mathcal{F}, m_1, \dots, m_a \in \mathcal{M}$:
 $\Pr(\text{Dec}(sk, \text{Eval}(evk, f, c_1, \dots, c_a)) = f(m_1, \dots, m_a)) = 1 - \epsilon(\lambda)$,
 where $c_1 = \text{Enc}(pk, m_1) \wedge \dots \wedge c_a = \text{Enc}(pk, m_a)$ and $\epsilon(\lambda)$ is a negligible function of the security parameter of the scheme.
- **Compactness.** $\forall f \in \mathcal{F}, c_1, \dots, c_k \in \mathcal{C}$:
 $|\text{Eval}(evk, f, c_1, \dots, c_k)| \leq \text{poly}(\lambda)$, where $|\cdot|$ denotes the bit length of a ciphertext, while $\text{poly}(\cdot)$ denotes a positive univariate polynomial.

The requirement on the evaluation correctness trivially states that by decrypting the output of the `Eval` algorithm we obtain the correct result of the computation homomorphically performed by `Eval` on the ciphertexts. In particular, the `Eval` algorithm evaluates a polynomial, defined over the plaintext space, in the sequence of input ciphertexts by replacing the modular additions and multiplications with the homomorphic operations `Add` and `Mul`, respectively, that are, in

turn, two probabilistic polynomial time algorithms defined over the ciphertext space \mathcal{C} :

- **Homomorphic Addition.** $c \leftarrow \text{Add}(evk, c_1, c_2)$ computes a ciphertext $c \in \mathcal{C}$ such that $\text{Dec}(sk, c) = \text{Dec}(sk, c_1) + \text{Dec}(sk, c_2)$
- **Homomorphic Multiplication.** $c \leftarrow \text{Mul}(evk, c_1, c_2)$ computes a ciphertext $c \in \mathcal{C}$ such that $\text{Dec}(sk, c) = \text{Dec}(sk, c_1) \cdot \text{Dec}(sk, c_2)$.

When defining a symmetric-key homomorphic encryption scheme, the only difference is the key generation algorithm $\text{KeyGen}(1^\lambda)$ outputting a tuple $\mathbf{k} = \langle sk, pk, evk \rangle$ with $sk = pk$. Lastly, We recall the categorization of HE schemes depending on the specific choice of the set of functions \mathcal{F} which can be evaluated. Specifically, a PHE scheme exhibits a function $f \in \mathcal{F}$ defined via an arithmetic circuit including a single type of gate (an additive one or a multiplicative one). A SWHE scheme exhibits a function $f \in \mathcal{F}$ defined via an arithmetic circuit with a depth no higher than a fixed (scheme-dependent) threshold. Finally, a FHE scheme exhibits a function $f \in \mathcal{F}$ defined via an unconstrained arithmetic circuit.

2.2 Homomorphic Comparisons

One of the requirement to apply our attack is the existence of an algorithm able to determine if a generic ciphertext is an encryption of a fixed plaintext m . Therefore, we now provide a formal definition for this algorithm, which we refer to as m -distinguisher.

Definition 4 (m -distinguisher). *Let $\langle \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval} \rangle$ be a homomorphic encryption scheme with security margin λ , and let \mathcal{M}, \mathcal{C} be the plaintext and ciphertext spaces, related by the generated key $\mathbf{k} = \langle sk, pk, evk \rangle$. Let $A_{\mathbf{k}}^m \subset \mathcal{C}$, be the set of ciphertexts corresponding to the encryption of a plaintext $m \in \mathcal{M}$, i.e.: $A_{\mathbf{k}}^m = \{c \in \mathcal{C} \text{ s.t. } \text{Dec}(sk, c) = m\}$.*

Given a plaintext $m \in \mathcal{M}$, an m -distinguisher is a deterministic polynomial time algorithm A_m taking as input a ciphertext $c \in \mathcal{C}$ and the public portion of \mathbf{k} (i.e., $\mathbf{k}_{\text{pub}} = \langle pk, evk \rangle$ for public-key systems and $\mathbf{k}_{\text{pub}} = \langle evk \rangle$ for symmetric ones), and computes the indicator function of the elements of $A_{\mathbf{k}}^m$ over the set of ciphertexts, $\mathbb{1}_{A_{\mathbf{k}}^m} : \mathcal{C} \rightarrow \{0, 1\}$, in such a way that

$$\frac{|\{c \in \mathcal{C} \text{ s.t. } A_m(c, \mathbf{k}_{\text{pub}}) = \mathbb{1}_{A_{\mathbf{k}}^m}(c)\}|}{|\mathcal{C}|} \geq 1 - \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is a negligible function of the security margin of the system.

Given the existence of this m -distinguisher, our attack leverages the capability to homomorphically compare two encrypted integers. Therefore, we now present the main methods proposed in the literature to compute this functionality, including the one used in our attack. First of all, performing comparisons requires to homomorphically evaluate the *greater-than* function on a chosen integer interval.

Definition 5 (Greater-than Function). Given a positive integer b and an interval of integers $D_t = \{0, 1, \dots, t-1\}$, with $t \geq 2$, the greater-than function $GT_{t,b} : D_t \times D_t \rightarrow \{b-1, b\}$ is defined as:

$$GT_{t,b}(x, y) = \begin{cases} b & \text{if } x \geq y, \\ b-1 & \text{otherwise} \end{cases}$$

To the extent of evaluating this function with an HE scheme, we need to find a polynomial $f_{gt} \in \mathcal{F} \subseteq \mathbb{Z}_n[x, y]$, such that $f_{gt}(x, y) = GT_{t,b}(x, y)$, with $2 \leq t \leq n$, $1 \leq b < n$, and x, y being the representatives of residue classes modulo n , (i.e., $x, y \in \mathbb{Z}_n$) considered as integers less than t . Such a polynomial can be easily found if the plaintext space is \mathbb{Z}_2 : indeed, additions and multiplications become **xor** and **and** gates, while the input variables are the single-bit values in the binary encodings of x and y , and thus there are many logical circuits computing $GT_{t,b}(\cdot, \cdot)$ function.

Considering a plaintext ring $\mathcal{M} = \mathbb{Z}_n$, with $n > 2$, which is the case targeted in our work, finding an efficiently computable polynomial for the $GT_{t,b}(\cdot, \cdot)$ function is a challenging task. Çetin in [8] reports two methods to compute the $GT_{t,b}(\cdot, \cdot)$ function which do not require interaction between the secret key owner and the party who performs homomorphic evaluation. However, both of these methods are not suitable for our attack: indeed, the first one is not applicable to a composite module n ; the second method computes an approximation of the $GT_{t,b}(\cdot, \cdot)$, while our attack needs an exact computation of this function.

A more effective solution is proposed in [17]: the greater-than function is computed as $GT_{t,b}(x, y) = SIGN_{t,b}(x - y)$, where $SIGN_{t,b}(z)$ is a function defined over $\overline{D}_t \subseteq \mathbb{Z} = \{-t+1, \dots, 0, \dots, t-1\}$ such that $SIGN_{t,b}(z) = b$ if $z \geq 0$, $b-1$ otherwise. The homomorphic evaluation of the function $SIGN_{t,b}(\cdot)$ requires a polynomial $f_{sign} \in \mathcal{F} \subseteq \mathbb{Z}_n[z]$ fulfilling $f_{sign}(z \bmod n) = SIGN_{t,b}(z)$, with $2 \leq t \leq \frac{n}{2}$, $1 \leq b < n$ and $z \in \overline{D}_t$. In [17], the polynomial f_{sign} is computed applying the Lagrange interpolation formula to $2t-1$ points having coordinates $(z, SIGN_{t,b}(z))$, with $z \in \overline{D}_t$, and considering a prime modulus, i.e., $n = p$.

As we are considering as a plaintext space the ring \mathbb{Z}_n with a generic modulus $n > 2$, we introduce an additional constraint on the integer t , formalized in Lemma 1, to extend the applicability of the aforementioned method to a generic ring \mathbb{Z}_n :

Lemma 1. Given an integer $t \geq 2$, and a set $\overline{D}_t = \{-t+1, \dots, 0, \dots, t-1\}$, the polynomial $f(z) \in \mathbb{Z}_n[z]$, with $n > 2$, interpolating $2t-1$ points having the x -coordinate ranging over all values in $z \in \overline{D}_t$ exists if $t \leq \frac{q}{2}$, where q is the smallest prime factor of n .

Proof. Considering $2t-1$ points $\{(x_1, y_1), \dots, (x_{2t-1}, y_{2t-1})\}$ in $\mathbb{Z}_n \times \mathbb{Z}_n$, the interpolating polynomial $f \in \mathbb{Z}_n[x]$, with degree at most $2t-2$, can be computed by the Lagrange interpolation formula:

$$f(x) = \sum_{i=1}^{2t-1} y_i \prod_{j=1, j \neq i}^{2t-1} (x - x_j)(x_i - x_j)^{-1}$$

The existence of the multiplicative inverses (in \mathbb{Z}_n) required in this formula is ensured if all the values $x_i - x_j$ are co-prime with n . Assuming the x -coordinates to be mutually distinct and in \overline{D}_t , the constraint $t \leq \frac{q}{2}$ implies that $-q < -2t + 2 \leq x_i - x_j \leq 2t - 2 < q$. Since q is the smallest prime factor of n , then all the elements in $\mathbb{Z}_n \setminus \{0\} \cap \{-q + 1, \dots, q - 1\}$ are co-prime with n , therefore all the values $x_i - x_j$ are co-prime with n , and thus invertible, allowing $f(x)$ to be interpolated by the Lagrange formula. \square

In conclusion, by Lagrange interpolation we can obtain a polynomial $f_{sign} \in \mathcal{F} \subseteq \mathbb{Z}_n[z]$ which computes the function $SIGN_{t,b}(z), \forall z \in \overline{D}_t$, and then a polynomial $f_{gt} \in \mathcal{F} \subseteq \mathbb{Z}_n[x, y]$, computing the function $GT_{t,b}(x, y), \forall x, y \in D_t$, as $f_{gt}(x, y) = f_{sign}(x - y)$. Since $f_{gt} \in \mathcal{F}$, it can be homomorphically evaluated by the `Eval` algorithm of the HE scheme, by replacing addition and multiplications of the polynomial with corresponding homomorphic operations (`Add` and `Mul`) whose inputs are ciphertexts in \mathcal{C} . In the following, we denote the algorithm `Eval`(evk, c_1, c_2, f_{gt}) by $HGT_{t,b}(c_1, c_2)$, which takes as input two ciphertexts with corresponding plaintexts $m_1, m_2 \in D_t$, and outputs an encryption of $GT_{t,b}(m_1, m_2)$. In particular, since $GT_{t,b}$ is defined on the interval $D_t = \{0, \dots, t - 1\}$, $t \leq \frac{q}{2}$, with q being the smallest prime factor of n , then $c_1, c_2 \in \mathcal{C}_t = \{c \in \mathcal{C} \text{ s.t. } \text{Dec}(sk, c) < t\}$ is a sufficient condition for $\text{Dec}(sk, HGT_{t,b}(c_1, c_2)) = GT_{t,b}(m_1, m_2)$. The computational complexity required to interpolate $2t - 1$ points by applying the Lagrange formula is $O(t^2)$ operations in \mathbb{Z}_n ; while the evaluation of the polynomial $f_{sign} \in \mathbb{Z}_n[z]$, whose degree is at most $2t - 2$, has a computational complexity $O(t)$. From this, it is easy to note that the computational cost of the $HGT_{t,b}(\cdot, \cdot)$ algorithm is $O(t)$. We note that, while there are no current algorithms to compute $HGT_{t,b}(\cdot, \cdot)$ in less than $O(t)$, research efforts driven by the usefulness of a homomorphic comparison may lead to an improvement in this sense. Since our methodology relies on the computation of $HGT_{t,b}(\cdot, \cdot)$ as an atomic component, such improvements will positively affect the efficiency of our attack.

3 Attack Strategy

In the following we detail a plaintext recovery attack which takes as input a ciphertext and the publicly available evaluation key evk of the HE scheme at hand (which can be either a FHE, or a SWHE capable of computing $HGT_{t,b}$). Since a FHE scheme must allow the evaluation of arbitrary polynomials, then it must provide to the evaluator a method to obtain encryptions of known values, preferably avoiding interaction with the key owner. In case the HE scheme is an asymmetric one, such ciphertexts can be directly obtained employing the public key encryption algorithm, while for a symmetric scheme, the encryption of any value can be obtained from a single encryption of $\hat{m} = 1$ leveraging homomorphic operations (we can obtain encryptions of all powers of 2 by iteratively summing \hat{m} by itself, and then compute the encryption of any integer value leveraging its binary representation). Thus, we assume, in case of a symmetric FHE scheme, that an encryption of $\hat{m} = 1$ is embodied in the evaluation key evk to allow the

computation of encryptions of known values by the evaluator. Such encryption of \hat{m} can be used also by the attacker to obtain the ciphertexts required to perform the attack.

Comparison-Based Attack. The core idea is to perform a homomorphic binary search over the possible candidates for the value of the plaintext corresponding to the ciphertext at hand. To this end, a comparison function CMP , taking two ciphertexts as inputs and yielding an outcome in cleartext, is computed leveraging the homomorphic greater-than function $HGT_{t,b}$ (see Sect. 2). Since the result of the $HGT_{t,b}$ function is still encrypted, the m -distinguisher is employed to determine its actual (plaintext) value. The attacker can compute the comparison function CMP employing the aforementioned strategy as follows:

Definition 6 (Comparison Function). *Given the ciphertexts $c_1, c_2 \in \mathcal{C}_t$ and A_m the algorithm computing the m -distinguisher, where m is a fixed plaintext value, the function $CMP : \mathcal{C}_t \times \mathcal{C}_t \rightarrow \{1, 0, -1\}$ is computed as:*

$$CMP(c_1, c_2) = \begin{cases} 1 & \text{if } v_1 = 1 \wedge v_2 \neq 1, \\ 0 & \text{if } v_1 = 1 \wedge v_2 = 1, \\ -1 & \text{otherwise} \end{cases}$$

with $v_1 = A_m(HGT_{t,m}(c_1, c_2), \mathbf{k}_{\text{pub}})$, $v_2 = A_m(c_1 - c_2 + c_m, \mathbf{k}_{\text{pub}})$ and c_m being an encryption of m computed by the attacker.

Denoting with T_d the computational complexity of the m -distinguisher, we have that the time complexity of CMP is $T_{CMP} = O(t + 2T_d)$ as its execution involves at most two computations of the m -distinguisher plus one computation of the $HGT_{t,m}$ function, which has complexity $O(t)$. Leveraging the function CMP , the binary search strategy locates the value of the actual plaintext in the range D_t , which is t elements wide, with a computational cost of $O(T_{CMP} \cdot \log(t)) = O((t + 2T_d) \log(t))$.

Starting from the strategy which has just been described, we improve its effectiveness extending the range of the recoverable plaintexts. To this end, we split the set of recoverable plaintexts into $|D_t| = t$ sized chunks, find into which chunk the plaintext is likely to be contained, and proceed to retrieve it employing the binary search approach. We denote with D_s the set of recoverable plaintexts ($D_s = \{0, 1, \dots, s - 1\}$, $s \leq n$), and with \mathcal{C}_s the set of ciphertexts obtained encrypting plaintexts in D_s , i.e.: $\mathcal{C}_s = \{c \in \mathcal{C} \text{ s.t. } \text{Dec}(sk, c) < s\}$. The recoverable message space D_s is split into σ chunks containing numerically consecutive plaintexts, with $\sigma = \lceil \frac{s}{t} \rceil$: for instance, the i -th chunk contains plaintexts values $\{(i - 1)t, \dots, it - 1\}$.

Algorithm 1 shows how our improved attack is performed. It iterates over all the σ chunks, testing, for each one of it, if the plaintext m_c , corresponding to the input ciphertext c , may be contained in the chunk being scanned (lines 2–9). To this end, the algorithm starts by testing if m_c may be in a chunk $\{(i - 1)t, \dots, it - 1\}$ by verifying if $GT_{t,m}(m_c, (i - 1)t) = m$ (lines 3–4). In case

Algorithm 1. Plaintext Recovery Attack

Input: ciphertext $c \in \mathcal{C}_s$, $\mathcal{C}_s = \{c \in \mathcal{C} \text{ s.t. } \text{Dec}(sk, c) < s\}$
Output: plaintext $m_c = \text{Dec}(sk, c)$, $m_c \in \mathbb{Z}_n$

- 1 **begin**
- 2 **for** $i \leftarrow 1$ **to** σ **do**
- 3 $c_{gt} \leftarrow HGT_{t,m}(c, \text{Enc}(pk, (i-1)t))$
- 4 **if** $A_{(m)}(c_{gt}, k_{\text{pub}}) = 1$ **then**
- 5 $c_{gt} \leftarrow HGT_{t,m}(c, \text{Enc}(pk, it-1)) + \text{Enc}(pk, 1)$
- 6 **if** $A_{(m)}(c_{gt}, k_{\text{pub}}) = 1$ **then**
- 7 $m_c \leftarrow \text{BinarySearch}(c - \text{Enc}(pk, (i-1)t))$
- 8 **if** $m_c \neq \perp$ **then**
- 9 **return** $m_c + (i-1)t$

this test succeeds (line 4, case of the **if** being taken), Algorithm 1 proceeds to test also if m_c is smaller than the upper bound $it-1$ of the chunk at hand, by verifying that $GT_{t,m}(m_c, it-1) = m-1$ with an analogous approach (lines 5–6). If the tests at lines 3–6 succeed, then the current chunk may contain the plaintext m_c , and so Algorithm 1 attempts a plaintext recovery employing the binary search approach described in precedence over the current chunk (line 7). We note that the answer of these tests are subject to potential false positives. Indeed, if $m_c \notin \{(i-1)t, \dots, it-1\}$, then $m_c - (i-1)t \notin \overline{D}_t$ or $m_c - (it-1) \notin \overline{D}_t$; thus, it means that the polynomial $f_{\text{sign}}(z) \in \mathbb{Z}_n[z]$, obtained by interpolating points whose x -coordinates range over \overline{D}_t , is evaluated on a point $z \notin \overline{D}_t$, hence yielding an outcome which is either outside the set $\{m-1, m\}$ or (by coincidence) inside it. Therefore, it may happen that $f_{gt}(m_c, (i-1)t) = f_{\text{sign}}(m_c - (i-1)t) = m$ and $f_{gt}(m_c, it-1) = f_{\text{sign}}(m_c - (it-1)) = m-1$ even if $m_c \notin \{(i-1)t, \dots, it-1\}$. In this case, the interval $\{(i-1)t, \dots, it-1\}$ is identified as a false positive. However, these false positive are filtered out later in the algorithm. Indeed, since the binary search is effective only under the assumption that the sought plaintext is in D_t , Algorithm 1 (line 7) exploits the homomorphic operations to subtract the value of the lower bound of the current chunk from m_c , working on its corresponding ciphertext c , to retrieve the value of $m_c \bmod t$, provided that the chunk detection was not reporting a false positive. If a result is returned (line 8), the actual value of m_c is reconstructed adding back the lower bound of the current chunk to the value retrieved by the binary search (line 9), otherwise the current chunk is a false positive.

We now consider the time complexity of Algorithm 1 as a function of the value of the plaintext to be retrieved m_c . Algorithm 1 is expected to perform $\lceil \frac{m_c}{t} \rceil$ iterations of the outer loop. Each one of the iterations, save for the last one, will fail the membership tests with very high probability (false positives are unlikely), thus resulting in a computational effort which is $O(t + T_d)$ at each iteration. However, we now consider the overall worst-case complexity $T_a(m_c)$ of the improved plaintext recovery attack:

$$O\left(\lceil \frac{m_c}{t} \rceil (t + T_d + T_{\text{BinarySearch}})\right) = O\left(\log(t)(m_c + \lceil \frac{m_c}{t} \rceil T_d)\right) \quad (1)$$

Therefore, our attack has a linear time complexity, which is the main reason why it is able to practically recover only ciphertexts whose corresponding plaintext is not too big. However, by setting $t = 2^{20}$ (an upper bound imposed by the $O(t^2)$ computational cost of Lagrange interpolation), we see that, unless $T_d > 2^{23}$, recovering plaintexts as big as 2^{32} still retains a computational complexity $T_a(m_c) < 2^{40}$. Since many typical FHE scenarios involve computations on relatively small values (e.g. power consumption statistics from smart meters), we deem this plaintext recover capability effective enough to be worth considering.

To conclude the description of our attack, we now show the speed-up obtained by Algorithm 1 over an exhaustive search strategy leveraging only the m -distinguisher. This latter attack tries all plaintext values $x \in \mathbb{Z}_n$ in increasing order, with the recovered plaintext being the first x such that $\mathbf{A}_m(\mathbf{Enc}(pk, x) - c + \mathbf{Enc}(pk, m)) = 1$. Denoting the value of the recovered plaintext as m_c , with this strategy we employ the m -distinguisher m_c times, therefore the complexity of this approach is $O(m_c T_d)$. The speed-up of our attack over this simple strategy can be computed as follows:

$$\frac{m_c T_d}{T_a(m_c)} = \frac{m_c T_d}{\log(t)(m_c + \lceil \frac{m_c}{t} \rceil T_d)} = \frac{m_c t T_d}{\log(t)(m_c t + m_c T_d)} = \frac{t T_d}{\log(t)(t + T_d)}$$

This calculation shows that our attack improves the exhaustive search strategy by a constant factor, thus without changing its asymptotic complexity. In particular, the speed-up depends on the values of t , chosen by the attacker, and T_d , given by the target scheme. Although this improvement may seem negligible, we will show, for the FHE schemes targeted by our attack, that the magnitude of the speed-up may be significant in practice, as it largely increases the number of recoverable plaintexts.

4 Two Case Studies

In this section, we evaluate our attack against two symmetric noise free FHE schemes [23], `OctoM` and `JordanM`. Although there is an efficient 1-distinguisher for these schemes, they were claimed to be secure against ciphertext-only adversaries aiming to recover either the plaintext or the secret key [23], making them a proper target for our attack.

4.1 Target Fully Homomorphic Encryption Schemes

We report a description of the two target symmetric FHE schemes, `OctoM`¹ and `JordanM`, focusing only on the details which are relevant for our attack: the characterization of the ciphertext space and the description of homomorphic operations. We refer the reader to [23] for further details. The plaintext space

¹ We find out that, to make `OctoM` multiplicatively homomorphic, some additional constraints are needed: they will be shown in the full version of the paper.

is the ring of integers \mathbb{Z}_n , with a composite $n > 2$, for both schemes. The ciphertext space of **OctoM** is the set of 8×8 matrices with entries in \mathbb{Z}_n , i.e., $\mathbb{Z}_n^{8 \times 8}$, while the one of **JordanM** is the set of 3×3 matrices with entries in $\mathbb{O}(\mathbb{Z}_n)$, where $\mathbb{O}(\mathbb{Z}_n)$ is the non commutative, non associative algebra of octonions whose support is the vector space \mathbb{Z}_n^8 . Both schemes perform a single matrix addition to homomorphically add two ciphertexts, while to homomorphically multiply two ciphertexts C_1, C_2 the two schemes employ different procedures:

- **OctoM Homomorphic Multiplication.** $C_{mul} = C_2 \cdot C_1 \cdot C_{-1}$, where \cdot denotes the matrix dot product and $C_{-1} \in \mathbb{Z}_n^{8 \times 8}$ is an encryption of the plaintext value $n - 1$, embodied in the evaluation key, evk .
- **JordanM Homomorphic Multiplication.** $C_{mul} = C_1 \star C_2$, where \star denotes the Jordan product.

4.2 Security Analysis

As already acknowledged in [23], the target FHE schemes are linearly decryptable, that is their decryption function can be expressed as a dot product between the key and the ciphertext represented in a d -dimensional vector space defined over the ring \mathbb{Z}_n . For the two target FHE schemes, the ciphertext space dimension d is 64 for **OctoM**, since a ciphertext is an 8×8 matrix, while $d = 9 \cdot 8 = 72$ for **JordanM**, since the ciphertext matrix is a 3×3 matrix whose entries are elements of $\mathbb{O}(\mathbb{Z}_n)$. Linearly decryptable schemes are vulnerable to KPAs: if the attacker has approximately d plaintexts/ciphertexts pairs, then a linear system of equations can be built to recover the key and decrypt any ciphertext. In addition, an efficient 1-distinguisher for any linearly decryptable scheme is proposed in [23]. We now describe the construction of this distinguisher, since we leverage it to perform our attack. Given a ciphertext C , represented as a d dimensional vector, consider the first $d + 1$ powers of C . Since the ciphertext space dimension is d , then these $d + 1$ ciphertexts are linearly dependent. Therefore, by definition, there are non trivial solutions to the system of d equations with $d + 1$ unknowns a_i defined as $\sum_{i=1}^{d+1} a_i C^i = 0$. Since the decryption function is linear and the encryption scheme is multiplicatively homomorphic, the following equality also holds: $\sum_{i=1}^{d+1} a_i m^i = 0$, where $m = \text{Dec}(sk, C)$. If $m = 1$, this equation becomes $\sum_{i=1}^{d+1} a_i 1^i = 0 \Rightarrow \sum_{i=1}^{d+1} a_i = 0$. Therefore, if the additional constraint $\sum_{i=1}^{d+1} a_i \neq 0$ is added to the system of equations $\sum_{i=1}^{d+1} a_i C^i = 0$, a solution is found if and only if $m \neq 1$. In conclusion, by looking at the solution of this system, we can determine if $m = 1$ or not. The computational complexity of this 1-distinguisher is $O(d^3)$, since solving a system of equations has cubic complexity. We remark that the system can be solved directly on ciphertexts, no knowledge about the plaintexts or the key is required. Therefore, this distinguisher is a particular case of Definition 4, since it does not employ k_{pub} , the publicly available portion of the cipher key. While the existence of these vulnerabilities (1-distinguisher and KPA) is acknowledged by designers of **OctoM** and **JordanM** too, their security analysis claims [23, Theorem 7] that the hardness of solving quadratic equation systems in \mathbb{Z}_n (with a composite n) guarantees that

no information about plaintexts can be inferred from ciphertexts. The proof of this claim is based on two reductions: first, the problem of finding the secret key is reduced to the problem of solving a system of multivariate quadratic equations in \mathbb{Z}_n , then the problem of recovering a plaintext is reduced to the problem of solving a univariate quadratic equation in \mathbb{Z}_n . These reductions state that solving quadratic equations in \mathbb{Z}_n is sufficient to break the cryptosystems, but they do not state that recovering the secret key or a plaintext is as hard as solving quadratic equations in \mathbb{Z}_n . Thus, there is no contradiction between the existence of our attack and the hardness of solving quadratic equations in \mathbb{Z}_n (which is as hard as factoring n).

4.3 Breaking Target FHE Schemes with Our Attack

As already discussed in Sect. 3, since a FHE scheme must allow the evaluation of arbitrary polynomials, then it must provide to the evaluator a method to obtain encryptions of known values, preferably avoiding interaction with the key owner. Since no method to provide this capability was proposed for the considered FHE schemes, we assume that an encryption of 1 is embodied in the evaluation key to provide this capability to users of the FHE scheme. It is worth noting that an encryption of 1 is not necessary for the `OctoM` scheme, since it can be computed by squaring the encryption of -1 already provided in the evaluation key. In the instantiation of our attack against `JordanM`, there is a relevant issue related to the fact that, as outlined in Definition 4, the m -distinguisher may have a wrong outcome on a negligible portion of the ciphertexts. However, this portion is not negligible for several ciphertexts being used in our attack. The problem arises because of two random values which are employed to randomize the encryption. We denote these two values for a ciphertext C by r_C, s_C . In particular, we find out two relevant facts about these values²: first, if $s_C = 1 \vee r_C = 1$, then the 1-distinguisher will classify the ciphertext as an encryption of 1 independently from the plaintext value m ; secondly, the evaluation correctness property (see Definition 3) holds not only for the message m , but for r_C, s_C too. The latter fact basically means that, given two ciphertexts C_1, C_2 , encrypted with random values respectively r_{C_1}, s_{C_1} and r_{C_2}, s_{C_2} , these two properties, related to homomorphic operations `Add` and `Mul` of `JordanM` scheme, hold:

$$\begin{aligned} C = \text{Add}(C_1, C_2) &\rightarrow r_C = r_{C_1} + r_{C_2} \wedge s_C = s_{C_1} + s_{C_2} \\ C = \text{Mul}(C_1, C_2) &\rightarrow r_C = r_{C_1} \cdot r_{C_2} \wedge s_C = s_{C_1} \cdot s_{C_2} \end{aligned}$$

As a consequence, a ciphertext C_{gt} obtained through homomorphic evaluation of $GT_{t,1}$ function has only four possible assignments to its random values $r_{C_{gt}}, s_{C_{gt}}$, which are $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$, since the image of $GT_{t,1}$ is $\{0, 1\}$. Therefore, while for a generic ciphertext C , $Pr(r_C = 1 \vee s_C = 1)$ is negligible, and thus this issue is not relevant for the reliability of the 1-distinguisher in general, for

² Proofs are omitted for the sake of brevity. They will be included in the full version of the paper as long as a fully detailed description of `JordanM` and `OctoM` cryptosystems.

a generic ciphertext C_{gt} obtained through homomorphic evaluation of $GT_{t,1}$ the same probability is 0.75, hence the outcome of the 1-distinguisher, when its input is a ciphertext C_{gt} as in our attack, is likely to be erroneous.

To overcome this issue, we devise a ciphertext refreshing procedure, which employs the available encryption of 1 to compute a new ciphertext C' as $C + C - \text{Enc}(pk, 1) * C$, having the same plaintext m , but random values $r_{C'}, s_{C'}$ which are highly likely to be different from 1, since they depend on the random values chosen for the encryption of 1. Therefore, in our attack we employ a slightly tailored version of the distinguisher, whose output is equal to $A_1(C) \cdot A_1(C')$. By using this distinguisher, we can perform our attack on both the target FHE schemes to recover plaintexts. Then, after d plaintexts have been recovered, we can perform the KPA and recover the key, breaking the schemes.

We can now estimate the computational complexity of our attack for the target FHE schemes. For linearly decryptable schemes, T_d , the computational complexity of the 1-distinguisher is $O(d^3)$, with $d = 64$ for `OctoM` and $d = 72$ for `JordanM`, which means that $T_d = O(2^{19})$ for both schemes. However, the distinguisher is always invoked twice in our attack to increase its reliability, therefore the computational complexity we are going to use in place of T_d , in the formulae derived in Sect. 3 to estimate the computational effort of our attack, is $T'_d = 2T_d = O(2^{20})$. Given this estimation, we can see that it is practical to recover plaintext values as big as 2^{32} , which is expected to be enough for a significant number of ciphertexts in FHE applications. The computational effort required to recover a plaintext value $m_c = 2^{32}$, can be computed as follows (see Eq. 1 in Sect. 3), replacing T_d with $T'_d = 2^{20}$ and setting $t = 2^{20}$:

$$T_a(2^{32}) \leq 2^{32} \log(2^{20}) + \left\lceil \frac{2^{32}}{2^{20}} \right\rceil 2^{20} \log(2^{20}) = 2^{32} \cdot 20 + 2^{32} \cdot 20 \leq 2^{38}$$

Conversely, recovering a plaintext as big as $m_c = 2^{32}$ via an exhaustive search strategy has a computational cost of $O(m_c T_d) = 2^{32} \cdot 2^{19} = 2^{51}$ (note that with this strategy we do not need to invoke twice the 1-distinguisher, thus we can use T_d instead of T'_d). Indeed, the speed-up of our attack is: $\frac{t T_d}{\log(t)(t+T'_d)} \geq \frac{2^{39}}{2^5 \cdot 2^{21}} = 2^{13}$. This result shows that the improvement of our attack is not negligible: considering a computational effort fixed a-priori, the number of plaintexts recoverable by our attack is 2^{13} times bigger than the number of plaintexts recoverable by the exhaustive search strategy (when $t = 2^{20}$). For instance, with a computational cost bounded by 2^{38} , our attack can recover plaintexts up to 2^{32} , while the exhaustive search can recover plaintexts up to 2^{19} . We successfully implemented the `OctoM` and `JordanM` cryptosystems as well as the described plaintext recovery attack in Python 2.7, with the intent to verify the effectiveness of the proposed attack. In practice, the security level of the target schemes affects the computational effort to perform the homomorphic operations as well as the modular arithmetic operations needed to evaluate a m -distinguisher. Therefore, such a dependency from the security level and/or the parameter sizes of the cryptoscheme is included in the computational complexity formulae of both our attack and the exhaustive search as the same multiplicative factor (which has

been omitted in the previous treatment). Hence, independently from the security margin, when the plaintext values are bounded (e.g. less than 2^{32}) our method largely improves the practicality of their derivation employing only ciphertext material.

5 Conclusions

We present a new type of plaintext recovery attack based on the capability of homomorphically evaluating the comparison between two encrypted integers and assuming the existence of an efficient algorithm to determine if a generic ciphertext is an encryption of a fixed value m . Although the computational cost of our attack is linear in the value of the plaintext being recovered, it significantly improves the number of recoverable plaintexts w.r.t. an exhaustive search strategy, which, in turn, might mean recovering a vast portion of ciphertexts in a FHE application scenario.

Acknowledgements. This work was supported in part by the EU Commission grant: “M2DC” (H2020 RIA) Grant agreement no. 688201.

References

1. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
2. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45239-0_4
3. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_50
4. Brakerski, Z.: When homomorphism becomes a liability. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 143–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_9
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012, pp. 309–325. ACM (2012). <https://doi.org/10.1145/2090236.2090262>
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
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**(2), 831–871 (2014). <https://doi.org/10.1137/120868669>
8. Çetin, G.S., Doröz, Y., Sunar, B., Martin, W.J.: An investigation of complex operations with word-size homomorphic encryption. ePrint Archive (1195) (2015). <https://eprint.iacr.org/2015/1195.pdf>

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
10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169–178. ACM (2009). <https://doi.org/10.1145/1536414.1536440>
11. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval and Johansson [19], pp. 465–482. https://doi.org/10.1007/978-3-642-29011-4_28
12. 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
13. Li, J., Wang, L.: Noise-free symmetric fully homomorphic encryption based on non-commutative rings. IACR ePrint Archive, Report 2015/641 (2015). <https://eprint.iacr.org/2015/641>
14. Kipnis, A., Hibshoosh, E.: Efficient methods for practical fully homomorphic symmetric-key encryption, randomization and verification. IACR ePrint Archive 2012, 637 (2012). <http://eprint.iacr.org/2012/637>
15. Liu, D.: Practical fully homomorphic encryption without noise reduction. IACR ePrint Archive 2015, 468 (2015). <http://eprint.iacr.org/2015/468>
16. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval and Johansson [19], pp. 700–718. https://doi.org/10.1007/978-3-642-29011-4_41
17. Narumanchi, H., Goyal, D., Emmadi, N., Gauravaram, P.: Performance analysis of sorting of FHE data: integer-wise comparison vs bit-wise comparison. In: AINA 2017, pp. 902–908. IEEE CS (2017). <https://doi.org/10.1109/AINA.2017.85>
18. Nuida, K.: A simple framework for noise-free construction of fully homomorphic encryption from a special class of non-commutative groups. IACR ePrint Archive 2014, 97 (2014). <http://eprint.iacr.org/2014/097>
19. Pointcheval, D., Johansson, T. (eds.): EUROCRYPT 2012. LNCS, vol. 7237. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-3-642-29011-4>
20. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On Data Banks and Privacy Homomorphisms. Foundations of Secure Computation. Academia Press, Ghent (1978)
21. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2014). <https://doi.org/10.1007/s10623-012-9720-4>
22. Tsaban, B., Lifshitz, N.: Cryptanalysis of the MORE symmetric key fully homomorphic encryption scheme. J. Math. Cryptol. **9**(2), 75–78 (2015). <https://doi.org/10.1515/jmc-2014-0013>
23. Wang, Y., Malluhi, Q.M.: Privacy preserving computation in cloud using noise-free fully homomorphic encryption (FHE) schemes. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 301–323. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_15
24. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS 1982, pp. 160–164. IEEE CS (1982). <https://doi.org/10.1109/SFCS.1982.38>



On Security in Encrypted Computing

Peter T. Breuer¹, Jonathan P. Bowen^{2,4}, Esther Palomar³,
and Zhiming Liu⁴(✉)

¹ Hecusys LLC, Atlanta, GA, USA
ptb@hecusys.com

² London South Bank University, London, UK
jonathan.bowen@lsbu.ac.uk

³ Birmingham City University, Birmingham, UK
esther.palomar@bcu.ac.uk

⁴ RISE, Southwest University, Chongqing, China
zhimingliu88@swu.edu.cn

Abstract. Encrypted computing is an emerging approach to security and privacy of user data on a computing system with respect to the operating system and other powerful insiders as adversaries. It is based on a processor that ‘works encrypted’, taking encrypted inputs to encrypted outputs while data remains in encrypted form throughout processing. An appropriate machine code instruction set is required, plus an ‘obfuscating’ compiler, and then the three part system provably provides cryptographic semantic security for user data, given that the encryption is independently secure. In other words, encrypted computing does not compromise the encryption. This paper presents the developing theory.

Keywords: Encrypted computing · Computer security · Data security

1 Introduction

This paper describes an emerging approach to provable security for user data against the operator, operating system and other powerful insiders in a computing system: *encrypted computing*. By that is meant that the processor takes inputs and produces outputs in encrypted form and observations via the programming interface of its internal states show only encrypted data. Our aim in this document is to project the developing theory. Engineered boundaries have fallen short as security barriers in the past, as recent attacks [14] on Intel’s flagship SGXTM [1] architecture for secure computing attest. The mathematics of encrypted computing shows that, to any adversary who does not know the encryption, the feasible interpretations of a program code and its execution trace are arbitrarily many and any method of attack, whether known or unknown, deterministic or stochastic, must fail to uncover what a given bit of data is with better than the probability from guesswork (see Sect. 8). That is the definition of *cryptographic semantic security* [13], and access rights are not a consideration.

The adversary in this setting is technically the *operator mode* of working of a suitable processor, and attacks are programs composed of the processor's machine code instructions. The operator mode 'works unencrypted' in the conventional way in the supporting processor, while the *user mode* 'works encrypted' as described in the opening to this section. Operator (also called 'supervisor') mode is synonymous with no access restrictions, whereas user mode is restricted to certain registers and areas of memory. Operator mode is the mode in which the operating system runs and a processor starts in operator mode when it is switched on, in order to load the operating system code from disk. Conventional software relies on the processor to change from user mode to operator mode and back to supply system support (e.g., disk I/O) as required, so the operator mode of working of the processor intrinsically presents difficulties as an adversary for the user mode. This document will use 'the operator' for operator mode. A malicious operating system is 'the operator', as is a human with administrative privileges, perhaps obtained by physically interfering with the boot process.

How the user gets an encryption key into the supporting processor is not the subject of this document. Diffie-Hellman hardware [7] may do key-exchange in public view to a write-only internal store, for example, without revealing the key to any observer, the operator included. A simple argument says there is not even a penalty to getting key management wrong: if (a) user B's key is still loaded internally when user A runs, then A's programs do not run correctly because the running encryption is wrong for them and A is as badly off as a spy as the operator but with less privilege, and if (b) B's key is in the machine together with B's program when A runs, then user A cannot supply appropriate encrypted inputs nor interpret the encrypted output, and is in no better position than the operator, against whom encrypted computing should already protect.

A possible scenario for an attack by the operator is where user data consists of scenes from animation cinematography being rendered in a server farm. The computer operators at the server farm have an opportunity to pirate for profit portions of the movie before release and they may be tempted. Another possible scenario is the processing in a specialised facility of satellite photos of a foreign power's military installations to reveal changes since a previous pass. If an operator (or a hacked operating system) can modify the data to show no change where there has been some, then that is an option for espionage. A *successful attack* by the operator is one that discovers the plaintext of user data or alters it to order. That is familiar in everyday situations too – for example, malware can gain operator system access and intercept the plaintext of encrypted user mail.

Note that it is not claimed here that the operator will not be able to interfere with user data at all; they can, say by writing zeros to memory or turning the machine off. What is claimed is that the operator cannot interfere so as to write in user data an intended independently defined value such as $\log \pi$ or the encryption key, or bias the likelihood of that outcome. That theory is explained here.

A medium term practical goal is a server for remote batch ('offline') computations. In that paradigm, the user compiles the program anew for each new set of (encrypted) inputs, submits the input and object code to the remote platform,

and receives back (encrypted) outputs. Theory says there will be no relation between the plaintext values beneath the encryption in the trace of one run versus that in another, so the arrangement is awkward to attack. The encryption key may be changed from run to run. However, there may be no need to change it frequently as the user's program will also offset inputs and outputs (and intermediate values) by different random amounts known only to the user beneath the encryption for each new run. The offset by different numbers each time everywhere among the plaintext values is a generator of maximal entropy for what is effectively an extra one-time coding pad beneath the encryption.

This article is organised as follows. Section 2 gives the historical context and state of the art. Section 3 sets out the components of an encrypted computing system. Section 4 shows by example what encrypted computing looks like. Security problems that generically arise from naive encrypted computing are considered in Sect. 5. Section 6 introduces theory to overcome it, first introduced in [5]. An appropriate machine code instruction set is required, and that is described in Sect. 7. An 'obfuscating' compiler is also required and that is described in Sect. 8. Section 8 shows the combination of processor, instruction set and compiler guarantees semantic security 'relative to the security of the encryption' (the hypothesis that the encryption is independently secure). The meaning of that is 'encrypted computing does not compromise encryption'.

Notation

Encryption of plaintext x is denoted by $\mathcal{E}[x]$ or x' , where \mathcal{E} is a one-to-many 'nondeterministic function', a function of x and extra hidden variables such as padding. Decryption of ciphertext ζ is denoted by $\mathcal{D}[\zeta]$, a function, with $\mathcal{D}[x'] = x$. The key k for encryption/decryption will be implicit when only one is involved, otherwise $\mathcal{E}[x, k]$ and $\mathcal{D}[\zeta, k]$. Equality (not identity) of ciphertexts $\chi = \zeta$ is defined as $\mathcal{D}[\chi] = \mathcal{D}[\zeta]$, so $x' = y'$ iff $x = y$, with $x' \neq y'$ iff $x \neq y$.

Operations on ciphertext will borrow the same names as on plaintext but in square brackets. Thus $\mathcal{E}[x_1][+] \mathcal{E}[x_2] = \mathcal{E}[x_1 + x_2]$, meaning that $\mathcal{E}[x_1][+] \mathcal{E}[x_2]$ may be calculated by decrypting the ciphertexts back to plaintexts x_1, x_2 , adding, then encrypting again. Whether the calculation is like that or not (the encryption may already possess that property), the abstraction is applicable.

2 Background

In 2009 Gentry produced a fully homomorphic encryption (FHE) [10], fulfilling a prediction of Rivest et al. [26] 30 years earlier. That is an encryption in which ciphertexts can be added to add the (1-bit) plaintexts beneath, and multiplied to multiply them. In 2010 one of the present authors realised that homomorphism is a joint function of arithmetic and encryption together, and hardware can be redesigned to provide the arithmetic that makes any given encryption homomorphic with respect to it. Moreover, conditionals (not part of Gentry's

scheme) can also be handled, giving rise to computationally complete hardware-assisted encrypted computing. The proof of that was published in 2013 [4]. It followed experiments that built a model of a pipelined superscalar processor in Java (<http://sf.net/p/jmips>) and replaced its arithmetic logic unit (ALU), generating encrypted working (<http://sf.net/p/kpu>) as predicted.

From 2014 to 2016, the open source *or1ksim* simulator (<http://opencores.org/or1k/Or1ksim>) for the *OpenRISC* (<http://openrisc.io>) processor architecture was modified first to 64-bit and then 128-bit encrypted computing and cycle-accurate simulation of a complete OpenRISC compliant reduced instruction set computer (RISC) [23] ‘running encrypted’. That (a) demonstrated the principle of working in a superscalar model for engineers who may not have accepted mathematical proofs and formally-oriented computer science, and also (b) explored the limits. With respect to (b), it was unknown if conventional instruction sets and processor architectures and organisation would be compatible with the idea, or how interactions with the operating system and processor interrupts would work. It became clearer, for example, that not every kind of code could run encrypted in the context – compilers and programs that arithmetically transform the addresses of program instructions (as distinct from addresses of program data) must run unencrypted because instruction addresses remained unencrypted by design, in order to prevent known plaintext attacks (KPAs) [2] on encrypted but predictable address sequences in a trace.

The existing GNU *gcc* v4.9.1 compiler (<http://github.com/openrisc/or1k-gcc>) and *gas* v2.24.51 assembler (<http://github.com/openrisc/or1k-src/gas>) ports for OpenRISC v1.1 were adapted for an encrypted instruction set (executables are the standard ELF format). Those now twice-ported compiler and utilities are at <http://sf.net/p/or1k64kpu-gcc> and <http://sf.net/p/or1k64kpu-binutils> respectively. It turns out that only the assembler, not the compiler, needs to know the encryption key. The largest application suite¹ ported to encrypted running so far for that project is 22,000 lines of C, and it and every application ported (now about fifty) has worked well. Though the target platform is 32-bit beneath the encryption, 64-bit integer and 32- and 64-bit floating point programs work well, because of code-level translations that *gcc* performs for platforms without 64-bit and floating point hardware support.

In 2015 details of the HEROIC processor for encrypted computing with Paillier-2048 encryption (of 16-bit data) were published [30]. The basic operation is a 16-bit plaintext/2048-bit ciphertext addition in 20 μ s, equivalent in speed to a 25 KHz classic Pentium. The machine has a stack-based architecture. Those are different from conventional von Neumann architectures but there have been hardware prototypes [15, 28] aimed at Java. A difficulty in using Paillier is that, though it is homomorphic with respect to plaintext addition, that is not mod 2^{16} addition, so each addition result has to be renormalised mod 2^{16} beneath the encryption every time, which accounts for half the cycles taken. It is done by subtracting 2^{16} and looking up a ‘table of signs’ for encrypted numbers to see if the result is negative or positive. To facilitate that, HEROIC encryp-

¹ IEEE floating point test suite at <http://jhauser.us/arithmic/TestFloat.html>.

tion is one-to-one, not one-to-many. Significantly, its ISA is a *one instruction set computing* (OISC) design that has the property that the same program code and runtime trace can be interpreted with respect to the plaintext data beneath the encryption at any point in memory and in the control graph in arbitrarily many ways by any observer and experimenter who does not have the key to the encryption. That means the kind of compilation discussed in Sect. 8 would work to provably secure it, but it is not clear if HEROIC’s authors have a compiler.

In 2018 the 10× faster CryptoBlaze architecture for encrypted computing, also using Paillier but with a nondeterministic component, was published [18].

At the other end of the scale a pathfinding earlier machine for encrypted computing, Ascend [9] (2012), did all its computation in unencrypted form, but with no access for the operator or operating machine while a program is running. Only the inputs and outputs were encrypted (including memory I/O) but the processor ran in ‘Fort-Knox’-like isolation, matching pre-defined statistics on observables such as cache and power drain. Ascend ran RISC MIPS [24] instructions and slowed down by 12–13.5× in encrypted mode with AES-128 (rightly, only relative figures are quoted in [9]), as compared to 10–50% slowdown for the authors own recent processor models for encrypted computing, which have been measured at 104 MIPS (equivalent to a 433 MHz classic Pentium) [6] when clocked at 1 GHz, on the standard Dhrystone benchmark [32].

Physical isolation of processes plus encrypted memory has emerged several times as an idea for secure computing (e.g., [17] for secure entertainment media platforms) and success means doing it as well as Ascend. Otherwise side-channels such as cache-hit statistics [31] and power drain [19] leak information.

In that line, Intel’s SGXTM (‘Software Guard eXtensions’) processor technology [1] is often cited, because it enforces separations between users. The mechanism is key management to restrict users to memory ‘enclaves’. While the enclaves may be encrypted (encryption/decryption units lie on the memory path), that is encrypted storage, a venerable idea [16], not encrypted computing.

SGX machines are used [29] by cloud service providers where assurance of safety is a marketing point. That is founded in customers’ belief in electronics designers ‘getting it right’ rather than mathematical analysis and proof. Engineering may leak secrets via timing variations and power use and SGX has recently fallen victim [14]. Use of SGX secure enclaves has to be written-in by the software author so it is a voluntary security device, whereas encrypted computing is an obligate security device. However, running code entirely inside an SGX enclave is running it in Ascend-style ‘splendid isolation’, but without Ascend’s protection against statistically-based deductions from the observables. SGX does hide explicit timing information, for example, but a code can count its own instructions to retrieve an estimate.

IBM’s efforts at making practical encrypted computation using very long integer lattice-based *fully homomorphic encryptions* (FHEs) based on Gentry’s 2009 cipher deserve mention. The 1-bit logic operations take of the order of 1 s [11]

on customised vector mainframes with a million-bit word, about equivalent to a 0.003 Hz Pentium, but it may be that newer FHEs based on matrix addition and multiplication [12] will be faster. The obstacle to computational completeness is that which HEROIC overcomes with its ‘table of signs’: encrypted comparison with plain 1/0 output is needed, as well as the encrypted addition (and multiplication), but HEROIC’s solution is not feasible for a million-bit encryption.

In principle, applications that require a fixed small number of multiplications can be carried out without overflowing using an FHE without the normalisations that are their hallmark. Such schemes are called *somewhat homomorphic* encryption (SHE). Hardware assistance for a SHE based on the YASHE scheme [3] with ciphertext blocks $2^{15} \times 1228$ bits long is reported in [27]. Their 2048-core parallel hardware does ciphertext addition in 83 ns and multiplication in 59.413 ms, but that is for one bit in plaintext. That would be 32-bit plaintext addition at the speed of a 0.166 Hz classic Pentium (counting 3 exclusive or gates and 3 and gates per 1-bit full adder, taking 6 s for one 32 bit addition).

The slow speeds of even hardware-assisted SHE schemes emphasise how relatively fast the recent general purpose processors for encrypted computing are. The price is a secret embedded in the hardware, as with a Smartcard.

3 Encrypted Computing Systems: Overview

This section briefly recapitulates encrypted computing systems. They consist of:

- (i) **A processor that ‘works encrypted’** in user mode, with encrypted inputs, encrypted outputs, and encrypted intermediate states, but which ‘works unencrypted’ in operator mode.

If user data were not in encrypted form throughout, then the operator, having full access, could read it and write it to order, so this kind of processor is needed.

- (ii) **A machine code instruction set** that prevents ‘algebraic’ attacks via certain ‘chosen instructions’ that conventional instruction sets contain.

For example, the conventional instruction that performs $x' [/] x'$, produces an encrypted 1 from any encrypted user datum x' the operator cares to copy.

- (iii) **An ‘obfuscating’ compiler** that smooths out statistical biases that may be present in machine code.

Else a program would contain human biases such as low numbers like 0,1,... in loop counts, which could be used in a statistically-based dictionary attack.

The following axioms from [5] refine the hardware requirements (i), (ii):

Axioms

- (1) Each instruction's action is a black box. (i)
- (2) Each instruction is observed to read and write data in encrypted form. (i)
- (3) Arithmetic instructions embed encrypted constants, adjustments to which may be made to accommodate any planned offsets in inputs to and output from the instruction (see Sect. 7). (ii)
- (4) There are no collisions possible between the encrypted constants embedded in some instructions and the ciphertext that the processor writes to and reads from registers and memory. (ii)

How (1) and (2) are achieved is a question of the hardware. It is failure of (1) that the Intel (and likely other manufacturers') vulnerability exploits in the recent Meltdown [21] and Spectre [20] attacks. There, speculative execution brings data into cache that remains visible even though the instructions are aborted, so 'nothing' leaves a trace. One of the conceptually simplest ways of achieving (2) is to use an encryption that permits arithmetic to be done without decrypting and re-encrypting: a *homomorphic* encryption. Some processors for encrypted computing have used homomorphic encryptions, as described in Sect. 2.

Certain classical processor features contradict (2) when 'observe' is understood to mean testing the processor state by any programmatic means, not only reading a register, and are to be avoided in designs. A machine code 'set if equal' instruction that compares two ciphertext inputs and sets a status flag if the plaintexts are equal would be mistaken instruction set design. The instruction output (the status flag) in that case would not be in encrypted form, as required by (2). It is also not in any register, but it could be tested with a following 'branch if set' instruction, because the branch is seen to be taken or not taken as the (inaccessible) status flag is set or not set. That makes a classical 'set/test-flag' style of processor instruction set design inappropriate. Yet the OpenRISC standard specifies that style of instruction set and so our own prototype processors step back to an earlier MIPS style of RISC design for branch instructions. Our own design's branch instructions do not test a status flag but compare (less than, equal to, etc.) register contents and branch or do not branch on the result as determined in conjunction with extra encrypted instruction bits (see Sect. 7). That cannot be used for binary search to determine a value by virtue of (3,4).

The axiom (3) is a feature of an appropriate instruction architecture (Sect. 7), while (4) may be achieved via disjoint paddings beneath the encryption.

A fifth axiom is sometimes needed. It extends (2) to allow testing by means of externally known facts, not only the processor's programming interface:

- (5) There are no sources of ciphertext whose plaintext is known independently.

That avoids known plaintext attacks. It would contradict (5) to design-in a read-only register that holds the known processor stepping number (encrypted).

The axiom carries over to statistics too: all registers should feasibly contain anything at start-up, with equal probability. A classical RISC read-only **zer** register that contains zero (encrypted) for user mode would contradict that, so cannot be.

4 What Does Encrypted Computing Look Like?

Encrypted running is illustrated in Table 1, where the same program has been compiled twice, and the resulting machine codes have been run (the two instances are top, vs. bottom in the table). Each time, the compiler has embedded different (encrypted) constants in the machine code (disassembly at left). As a result different encrypted values appear throughout the execution traces (right), but the decrypted result (boxed) is nevertheless the same.

Table 1. Program codes and execution traces of exactly the same form may have encrypted data whose plaintext is arbitrarily different at any point yet get the same result.

1st code fragment	1st fragment's trace
<i>addr. instruction disassembly</i>	<i>addr. update</i>
96C sub sp sp zer $\mathcal{E}[-471185111]$	96C sp $\leftarrow \mathcal{E}[-3412890104]$
980 jal A	980 ra \leftarrow 984
	...
984 add t0 v0 zer $\mathcal{E}[-236230946]$	984 t0 $\leftarrow \mathcal{E}[-236230942]$
998 sub sp sp zer $\mathcal{E}[-1219116768]$	998 sp $\leftarrow \mathcal{E}[-1219116896]$
9AC lw a0 $\mathcal{E}[1219116768]$ (sp)	9AC a0 $\leftarrow \mathcal{E}[\boxed{1}]$
2nd code fragment	2nd fragment's trace
<i>addr. instruction disassembly</i>	<i>addr. update</i>
96C sub sp sp zer $\mathcal{E}[1528657211]$	96C sp $\leftarrow \mathcal{E}[-178928721]$
980 jal A	980 ra \leftarrow 984
	...
984 add t0 v0 zer $\mathcal{E}[-1112987554]$	984 t0 $\leftarrow \mathcal{E}[-1112987550]$
998 sub sp sp zer $\mathcal{E}[-275939886]$	A98 sp $\leftarrow \mathcal{E}[-275940014]$
9AC lw a0 $\mathcal{E}[275939886]$ (sp)	9AC a0 $\leftarrow \mathcal{E}[\boxed{1}]$

LEGEND

Encrypted: $\mathcal{E}[x]$, x' (Same) program point and storage place
 Registers: pc,ra,sp,zer,t0,v0,a0 Content: pc, ra, sp, zer, t0, v0, a0

INSTRUCTION SEMANTICS

sub x y z k' : $x \leftarrow y[-]z[+]k'$ jal a : ra \leftarrow pc + 4; pc \leftarrow a $\mathcal{E}[x][o]\mathcal{E}[y] = \mathcal{E}[x \circ y]$
 add x y z k' : $x \leftarrow y[+]z[+]k'$ lw x k'(y) : $x \leftarrow$ memory[[y[+]k']]

The ‘trick’ is that the compiler creates code that at runtime produces encrypted values whose plaintext values are *offset* from the nominal value all the

way through the calculation. The offsets are different (and randomly generated) for each point in the program control graph per each location in memory. For illustration here, the final offset in register **a0** has been set at 0, but ordinarily the final offset is also randomly generated, albeit known to the user.

5 Vulnerabilities of Naive Encrypted Computation

Being able to run arbitrary computable functions is dangerous in principle because an adversary might use the encrypted computations to subvert the encryption. For example, 32-bit 2s complement arithmetic is used in all modern computing. In that, repeated doubling of anything gives encrypted zero. I.e.:

$$\mathcal{E}[x][+] \dots [+] \mathcal{E}[x] = \mathcal{E}[x + \dots + x] = \mathcal{E}[2^{32}x \pmod{2^{32}}] = \mathcal{E}[0].$$

That opens the encryption to a known plaintext attack. The adversary can obtain encryptions of 0 by forcing the processor to add any initial datum $\mathcal{E}[x]$ in register r to itself 32 times, using its own machine code addition instruction:

$$\underbrace{\text{add } r \ r \ r; \dots; \text{add } r \ r \ r;}_{r \leftarrow r [+] r} \quad \underbrace{\text{add } r \ r \ r; \dots; \text{add } r \ r \ r;}_{r \leftarrow r [+] r}$$

Using multiplication, choosing a random ciphertext has a 50% chance of picking an odd number plaintext and then repeated self-multiplication gives an encrypted 1, by Fermat's Little Theorem²:

$$\mathcal{E}[x][*] \dots [*] \mathcal{E}[x] = \mathcal{E}[x * \dots * x] = \mathcal{E}[x^{2^{31}} \pmod{2^{32}}] = \mathcal{E}[1]$$

Self-multiplying an even number gives encrypted zero, but half the time an encrypted 1 is obtained, and a 50% success rate beats 1/2³² odds from guessing.

Using division, an adversary can get an encrypted 1 from any datum that is not an encrypted zero, which is a near certainty among all the encrypted data passing through the machine, via

$$\mathcal{E}[x][/] \mathcal{E}[x] = \mathcal{E}[x/x] = \mathcal{E}[1]$$

A subroutine for 64-bit division on a 32-bit platform is a place where one would find an encrypted 1 as a program constant or an extra parameter to the subroutine at each application. In any case, a dictionary attack on all the constants in the code should encounter an encrypted 1 among them. Guess which and, by

² Fermat's Little Theorem is $a^\phi = 1 \pmod n$, where a is coprime to n and ϕ is the size of the multiplicative group of integer residues mod n , being the number of residues that are coprime to n . It is needed here in the form $a^\phi = 1 \pmod{2^n}$, where a is odd, i.e., coprime to 2^n . Exactly half the numbers less than 2^n are odd, i.e., coprime to 2^n , and they form the multiplicative group mod n . So ϕ is 2^{n-1} and the theorem says $a^{2^{n-1}} = 1 \pmod{2^n}$. The better-known special form is $a^p = a \pmod p$, p prime.

Table 2. Code and trace may be interpreted in different ways with respect to the plaintext data by an observer who cannot read the encryption.

1st code fragment	1st trace	2nd code fragment	2nd trace
<i>addr. instruction</i>	<i>addr. update</i>	<i>addr. instruction</i>	<i>addr. update</i>
	($x = \mathcal{E}[0]$)		($x = \mathcal{E}[7]$)
0 A: if $x < \mathcal{E}[1]$ goto B	0 A:	0 A: if $x < \mathcal{E}[8]$ goto B	0 A:
1 $x \leftarrow x[-]\mathcal{E}[1]$	↓	1 $x \leftarrow x[-]\mathcal{E}[1]$	↓
2 goto A		2 goto A	
3 B: $x \leftarrow x[+]\mathcal{E}[1]$	3 B: $x \leftarrow \mathcal{E}[1]$	3 B: $x \leftarrow x[+]\mathcal{E}[1]$	3 B: $x \leftarrow \mathcal{E}[8]$
4 if $x < \mathcal{E}[1]$ goto B	4	4 if $x < \mathcal{E}[8]$ goto B	4
	($x = \mathcal{E}[1]$)		($x = \mathcal{E}[8]$)

LEGEND

Encrypted: $\mathcal{E}[x]$ $\mathcal{E}[x][o]\mathcal{E}[y] = \mathcal{E}[x \circ y]$ $\mathcal{E}[x][R]\mathcal{E}[y] = x R y$

repeated addition of encrypted 1s, an adversary may first build all powers of 2 and then build the encryption of any desired number K from its binary code via

$$\mathcal{E}[2^{k_1}][+]\dots[+]\mathcal{E}[2^{k_j}] = \mathcal{E}[2^{k_1} + \dots + 2^{k_j}] = \mathcal{E}[K].$$

Then, if an arithmetic order comparator instruction is available on the platform, any encrypted number could be decrypted by comparing it with an encryption of each 32-bit integer K in turn³. Decryption goes even faster deducing the binary digits one by one, comparing and subtracting (encrypted) 2^k when $\mathcal{E}[2^k][\leq]\mathcal{E}[x][<]\mathcal{E}[2^{k+1}]$ is detected by a conditional branch instruction in the machine (a machine code conditional branch on $\mathcal{E}[x][\leq]\mathcal{E}[y]$ detects if $x \leq y$ then jumps to a designated instruction, just like a **goto** in a higher level language).

The vulnerabilities above apply to any naive system for *arbitrary* computation. It must have comparator instructions in order to trigger branch jumps. In contrast, finite calculation systems can produce the 1/0 result b of a comparison in encrypted form, and the final ciphertext values $\mathcal{E}[x_1]$ and $\mathcal{E}[x_0]$ of variable x from both branches after the comparison are combined in $\mathcal{E}[x_1 * b + x_0 * (1 - b)]$. That is not an option in a system for unbounded computation, which must report the comparison in 1/0 format so the electronics can execute only one branch.

In summary, to *write* an encrypted number to order on a naively constructed platform, ‘just addition and multiplication’ will do, with 50% certainty. *Reading* requires a comparator too. If the encryption itself is even partially homomorphic (i.e., some encrypted operations can be done without access to the encryption key), the processor is not even needed. So there is a case to answer as to security.

6 Secure Encrypted Computing

There are ways of running arbitrary encrypted computations securely. Consider for the moment that the machine code has only instructions *addition of a constant* $y \leftarrow \mathcal{E}[\mathcal{D}[x] + k]$ and branches based on *comparison with a constant* $\mathcal{D}[x] < K$,

³ Rass in [25] has recently independently called this a ‘chosen instruction’ attack.

for registers x, y ⁴. Those suffice for any computation, encrypted⁵. Consider program C using only those two instructions. By a ‘method of observation’ understand a deterministic process, based on observing what a running user program does from step to step and making deductions from what is observed – its trace T . The trace details the sequence of instructions executed, with their addresses, and what register and memory locations each instruction reads and writes and with what values. Assume *the operator cannot already read the encryption*. Then:

Theorem 1 *No method of observation exists by which the operator (who does not possess the key) may decrypt output from C .*

The argument is illustrated by the program in this language that sets $x = \mathcal{E}[1]$, rendered at left in Table 2. There is no single statement of the language that will suffice. The code first loops until x is ‘not too large,’ then loops until it is ‘not too small.’ Exit at B is with $x = \mathcal{E}[1]$ exactly, no matter the value at entry at A. The trace with $x = \mathcal{E}[0]$ on entry and $x = \mathcal{E}[1]$ on exit is shown alongside. The right half of Table 2 shows the same code in which branch comparison constants (red) have been changed by $+7$ beneath the encryption. That admits a trace of exactly the same form but with plaintext numbers beneath the encryption that are $+7$ more than before. Since it is feasible, it is what happens, as computation is deterministic. So there are two possible interpretations of the codes and traces in Table 2 to an observer who does not already know the encryption. The evidence presented to the observer’s method is the same both times in the observer’s own terms: the codes and the traces ‘look the same’, the only differences lying in encrypted constants that by hypothesis the observer cannot read. One may suppose that the codes and traces are short enough that no ciphertext is repeated twice, so those encrypted values that do appear serve as no more than different labels for the same unknowns and have no more significance than that. If the observer has a method for getting at the plaintext value beneath the encryption then it must give the same answer in both cases. Yet the observer’s method must be wrong in one case, because the numbers beneath the encryption all differ by 7. So the method does not exist. The formal argument is simply that:

Proof (Theorem 1). Change C to D by changing all the constants $\mathcal{E}[K]$ in comparison instructions to $\mathcal{E}[K + 7]$. That permits a trace in which all data takes values not $\mathcal{E}[x]$ but $\mathcal{E}[x + 7]$ at every point. The addition instructions, which are unaltered in D , instead of taking $\mathcal{E}[x]$ to $\mathcal{E}[x + k]$ now take $\mathcal{E}[x + 7]$ to $\mathcal{E}[x + 7 + k]$. The observer’s hypothetical method is not sensitive to the change as the observer cannot read the encryption, so the method must give the wrong answer either in the trace of C or in that of D about a value beneath the encryption. \square

⁴ To help the reader over a ‘notation gap’, $y \leftarrow \mathcal{E}[\mathcal{D}[x] + k]$ is written here for $y \leftarrow x [+] k'$.

⁵ A practitioner’s proof of the computational completeness of the instructions $y \leftarrow x + k$ and if $x < K \dots$ is the mathematician J.H. Conway’s well-known *Fractran* programming language [8], in which those are the only instructions. Attention in the computer hardware community may have been first drawn to the fact by [24].

Remark 1. The argument shows that the same code and trace may differ independently at every point beneath the encryption to the maximum extent possible. An assignment instruction $x \leftarrow \mathcal{E}[\mathcal{D}[y] + k]$ may be changed to account for an arbitrarily chosen offset c (instead of $+7$) in the incoming value y beneath the encryption and generate an arbitrarily chosen offset d (instead of $+7$) in the outgoing value x by rewriting the instruction to $x \leftarrow \mathcal{E}[\mathcal{D}[y] + k - c + d]$. \square

7 Instruction Architecture for Encrypted Computing

What makes Theorem 1’s proof work is the following:

Lemma 1. *Every atomic instruction’s inputs $\mathcal{E}[x_1]$ and outputs $\mathcal{E}[x_0]$ may be shifted by constants to $\mathcal{E}[x_1 + k_1]$ and $\mathcal{E}[x_0 + k_0]$ respectively, by means of constants embedded (encrypted) in the instruction, for arbitrary k_0, k_1 .*

That is merely a formal expression of axiom (3) of Sect. 1.

Designing a complete instruction set to comply with (3) requires careful choices to allow the processor to function with full coverage and to work quickly while compilation remains uncomplicated, also consideration of physical restrictions (instruction length, field sizes, opcode map, etc.). HEROIC’s minimalistic instruction set complies, but is not suited to efficient compilation. We call any compliant instruction set a *fused anything and add* (FxA) instruction set⁶ because the natural form of a compliant arithmetic instruction semantics is

$$\begin{aligned} x &\leftarrow \mathcal{E}[(\mathcal{D}[y] - k_1)\Theta(\mathcal{D}[z] - k_2) + k_0] \\ &= (y [-] k'_1) [\Theta](z [-] k'_2) [+] k'_0 \end{aligned}$$

with binary operator Θ (for example, Θ may be multiplication), registers x, y, z .

Our own processor’s instruction set for encrypted working is shown in Table 3. It bears a likeness to OpenRISC’s instruction set and RISC in general, in that there is one memory load/store instruction and the rest of the instructions use registers, but ‘RISCiness’ stops at the increased instruction lengths. The comparison operations also contain an extra bit beneath the encrypted field that says if branch happens on success or on failure, as per axiom (3). Then:

Theorem 2 *There is no method by which the privileged operator can read runtime data from a program C constructed using instructions in Table 3.*

That is by using Lemma 1 in the proof of Theorem 1 for all arithmetic instructions of Table 3⁷. It also follows that interfering and experimenting with the program to substitute a different value for the returned result does not work:

⁶ ‘Addition of a constant’ is not the only option. Bitwise XOR (exclusive OR) with a constant can be used, or ‘multiplication by a prime and addition of a constant’.

The most general possibility is to replace a conventional instruction $x \leftarrow f(y)$ by $x \leftarrow f(y \cdot k_1^{-1}) \cdot k_2$, where \cdot is the operation of a mathematical group and $^{-1}$ is the group inverse operation. For simplicity, addition is used throughout this paper.

⁷ Proofs of results stated but not proved in the text are supplied in the Appendix.

Table 3. Machine code instruction set for encrypted working.

<i>op.</i>	<i>fields</i>	<i>mnem.</i>	<i>semantics</i>
add	$r_0 r_1 r_2 k'$	add	$r_0 \leftarrow r_1 [+] r_2 [+] k'$
sub	$r_0 r_1 r_2 k'$	subtract	$r_0 \leftarrow r_1 [-] r_2 [+] k'$
mul	$r_0 r_1 r_2 k'_0 k'_1 k'_2$	multiply	$r_0 \leftarrow (r_1 [-] k'_1) [*] (r_2 [-] k'_2) [+] k'_0$
div	$r_0 r_1 r_2 k'_0 k'_1 k'_2$	divide	$r_0 \leftarrow (r_1 [-] k'_1) [/] (r_2 [-] k'_2) [+] k'_0$
...			
mov	$r_0 r_1$	move	$r_0 \leftarrow r_1$
ble	$j r_1 r_2 k'$	branch	if $r_1 \leq r_2 [+] k'$ then $pc \leftarrow pc + j$
bge	$j r_1 r_2 k'$	branch	if $r_1 \geq r_2 [+] k'$ then $pc \leftarrow pc + j$
...			
b	j	branch	$pc \leftarrow pc + j$ unconditionally
sw	$k' (r_1) r_2$	store	$\text{mem}[[r_1 [+] k']] \leftarrow r_2$
lw	$r_1 k' (r_2)$	load	$r_1 \leftarrow \text{mem}[[r_2 [+] k']]$
jr	r	jump	$pc \leftarrow r$
jal	j	jump	$ra \leftarrow pc + 4; pc \leftarrow j$
j	j	jump	$pc \leftarrow j$
nop		no-op	

LEGEND

r - register indices k - 32-bit integers pc - prog. count reg.
 j - program count or incr. ' \leftarrow ' - assignment ra - return addr. reg.
 $\mathcal{E}[x], x'$ - encrypted val. $\mathcal{E}[x] [o] \mathcal{E}[y] = \mathcal{E}[x \circ y]$ $\mathcal{E}[x] [R] \mathcal{E}[y] = x R y$

Corollary 1. *There is no method by which the operator can alter program C using other or the same instructions to get an intended output (encrypted).*

The reason is that the program built by the operator to give the intended output cannot be built, by Theorem 2, because the output is readable, as it is known what it decrypts to (this lawyering stands in for a near repeat of the same proof).

Example (Theorem 2, Corollary 1). Take the encryption in the machine to be AES with key k , so encryption is $x' = \text{AES}(x, k)$ for plaintext x and ciphertext x' . Then there is a program C that decrypts (encrypted) input data, though the whole program runs in the encrypted computing environment. It is the AES decryption routine, compiled encrypted. Suppose x' decrypts to x , and x'' is the encryption of x' , k' the encryption of k . Then $C(x'', k') = x'$ by definition, because the unencrypted program takes x' and k to x . That is $C(y', k') = y$, as claimed, on choosing $y = x'$. □

Fortunately, the theorem prohibits the adversary building a program that outputs the encrypted encryption key k' , because with it and program C of the example the adversary would obtain the encryption key in the clear, via $C(k', k') = k$.

Remark 2. The program C of the example that does decryption cannot be intentionally built by an adversary (this is proved in the Appendix).

Example (Corollary 1). The program that sets $x = \mathcal{E}[1]$ at left in Table 2 cannot be intentionally built by the operator. Trying for '1' the operator may instead get the program at right in the table, which produces '8' (encrypted). □

There is need and potential (see Remark 1) for obfuscation here. Human beings only write certain programs, and an adversary may bet on an encrypted 1 being among the data, enabling the 'chosen instruction' attack of Sect. 5.

8 Obfuscating Compilation

For effective and useful ‘obfuscation’ in this context, plaintext data beneath the encryption should be varied from the nominal value at each of the up to $m + 32$ storage locations accessed by the program (m memory locations and 32 registers) at each of the N instructions of the program. A compiler can do that by varying the encrypted constants embedded in the instructions, by axiom (3). The idea is for the compiler variations to hide any human biases. Maximal noise applied by the compiler across different compilations swamps any other signal.

Let MC be the type of machine code, consisting of a sequential list of ‘FxA-compliant’ instructions, as for example in Table 3, and let Expr be the type of expressions, and let Off be the type of integer ‘offsets’. The approach our own compiler takes is to invent and aim for a particular runtime offset from nominal:

$$\llbracket - \rrbracket^r :: \text{Expr} \rightarrow (\text{MC}, \text{Off})$$

where r is the processor register that the value of the expression is to appear in. That is, the result of compiling an expression e is

$$\llbracket e \rrbracket^r = (mc, \Delta e).$$

The value $e + \Delta e$ beneath the encryption will be produced in register r at runtime by running the code mc , where Δe has been freely chosen at compile time. That is, let $s(r)$ be the content of register r in state s of the processor at runtime. The machine code mc emitted is designed to have operational semantics (Table 3):

$$s_0 \xrightarrow{mc} s_1 \text{ where } s_1(r) = \mathcal{E}[e + \Delta e] \quad (\star)$$

An offset $\Delta e = 0$ means the result will be the nominal value. Compilation for (\star) is described in detail in [5]. The upshot is that independently chosen, arbitrary offsets Δe generated by the compiler are induced at *runtime* in the plaintext values written to every register and memory location, differing per point in the program control flow graph. The following lemma is proved in [5]:

Lemma 2. *The obfuscating compiler creates object codes from the same source code that are identical apart from embedded (encrypted) constants. The runtime traces are also identical apart from the ciphertext data values read and written, such that, for any particular plaintext 32-bit value x , the probability across different compilations that $\mathcal{E}[x]$ is in register or memory location l at any given point in the trace is uniformly $1/2^{32}$, independently to the maximum extent permitted by copy instructions and loops in the code.*

The proviso is because a plain copy (‘mov’) instruction always has precisely the same input and output, and a loop means the variations introduced by the compiler must be the same at the beginning as at the end of the loop.

Source code for the compiler, assembler, linker, virtual machine, etc., may be downloaded from <http://sf.net/projects/obfusc>. The compiler currently covers all of ANSI C, and most GNU extensions except computed gotos.

In order to support arrays, pointers p must be declared together with a fixed ‘memory zone’ into which they point, thus:

```
int A[100];
restrict A int *p;
```

The **restrict** A means that the pointer never points outside the memory zone A. The compiler does not know where an unrestricted pointer will point at runtime and this declaration tells it to use an offset ΔA pertaining to zone A at that point in the program for the pointer. Each write through p should change ΔA , so the compiler accompanies it with writes to the rest of A to reset the other entries too. That is computationally inefficient, but cryptographically necessary. Oblivious RAM (ORAM) [22] does the same, but in hardware. In practice the processor will do the writes to memory asynchronously via the ubiquitous ‘write-back’ cache of contemporary processor technology, so the performance penalty is bandwidth, not latency (but a vector write instruction would be helpful).

Recalling Goldwasser & Micali’s definition [13] that ‘semantic security’ is inability to guess a designated bit with any success above chance, Lemma 2 implies:

Theorem 3 *Runtime user data beneath the encryption is semantically secure against the operator for FxA code compiled by the obfuscating compiler.*

The threat alone that the code has been compiled by an obfuscating compiler might be sufficient for the theorem, as that establishes the domain of possible variations that must be considered. But despite the look of it, the theorem is not a strong statement. It should be understood as saying that computation in an encrypted computing system does not reduce the security from the encryption.

9 Conclusion

This paper intends to bring encrypted computing to the attention of the security community as a technology that potentially safeguards user data against a classically all-powerful operator, operating system and other insiders as adversaries. This paper has dealt with theoretical aspects. With the appropriate instruction set and an ‘obfuscating’ compiler, it is shown here that user data cannot be determined by an adversary via any deterministic or stochastic method with any success above chance, provided the encryption used is independently secure. In other words, encrypted computing does not compromise encryption.

Acknowledgments. Zhiming Liu thanks the Chinese NSF for support from research grant 61672435, and Southwest University for grant SWU116007. Peter Breuer thanks Hecusys LLC for continued support in KPU research and development.

Appendix: Proofs of results

Proof (Corollary 1). Suppose for contradiction that the operator builds a new program $D=f(C)$ that returns $\mathcal{E}[y]$. Then its constants $\mathcal{E}[k]$ are found in C and its constants $\mathcal{E}[K]$ likewise, because f has no way of arithmetically combining them (the disjoint subspaces condition (4) on runtime encrypted data and encrypted

program constants means they cannot be combined arithmetically in the processor and the operator does not have the encryption key). Theorem 1 says the operator cannot read output $\mathcal{E}[y]$ of D , yet knows what it is. Done by contradiction. \square

Proof (Theorem 2). Program C is constructed using arbitrary instructions from Table 3 compliant with (3). One may construct a modified code D (see below) that looks the same as C to the adversary who cannot read the encryption, as well as possessing a runtime trace U that looks the same as the original trace T to the adversary, differing only in the cipherspace values read and written. The argument is the same as for Theorem 1 (and Corollary 1): In the given program C , every binary arithmetic instruction necessarily has semantics of the form (the r_i are registers)

$$r_0 \leftarrow \mathcal{E}[(\mathcal{D}[r_1]-k_1) \Theta(\mathcal{D}[r_2]-k_2)+k_0]$$

in order to comply with (3), and it can be adjusted for D via its embedded constants $\mathcal{E}[k_i]$ to accommodate every data value passing through registers and memory to be +7 more beneath the encryption than it used to be in C , as argued in the proof of Theorem 1 and Corollary 1. The change is from k_i to $k'_i=k_i+7$. Similarly, every branch instruction in C necessarily has a test of the form $(\mathcal{D}[r_1]-k_1)R(\mathcal{D}[r_2]-k_2)$ in order to comply with (3). It is changed in D to $(\mathcal{D}[r_1]-k'_1)R(\mathcal{D}[r_2]-k'_2)$ with $k'_i=k_i+7$. Then the branch goes the same way at runtime in trace U for D as it did originally in trace T for C . Unconditional jump instructions are not altered.

The outcome is a trace U that is the same as T modulo the cipherspace values read and written, which by hypothesis cannot be read by the adversary. Those differ by 7 under the encryption in U from the originals in T . Code D looks the same too, modulo the embedded encrypted constants, which also cannot be read by the adversary. Therefore, as in the proof of Theorem 1, a method $f(C, T)$ for decryption must give the same result as $f(D, U)$, yet the answers are different by 7 in the two cases, so the method f cannot exist. \square

Proof (Lemma 2). Consider the arithmetic instruction I in the program. Suppose that by fiddling with the embedded constants in the other instructions in the program it is already possible for all other locations m other than that written by I and at all other points in the program to vary the value $x_m = x + \Delta x$, where $\mathcal{E}[x_m]$ is stored in m , randomly and uniformly across compilations, taking advantage of the instruction set as the compiler described in the text does. Let I write value $\mathcal{E}[y]$ in location l . By the axiom (3) I has a parameter $\mathcal{E}[k]$ that may be tweaked to offset y from the nominal result $f(x + \Delta x)$ on its input $x + \Delta x$ by an amount Δy . The compiler chooses k with a distribution such that Δy is uniformly distributed across the possible range. The instructions in the program that receive y from I may be adjusted to compensate for the Δy change by changes in their controlling parameters. Then $p(y = Y) = p(f(x + \Delta x) + \Delta y = Y)$ and the latter probability is $p(y = Y) = \sum_{Y'} p(f(x + \Delta x) = Y' \wedge \Delta y = Y - Y')$.

The probabilities are independent (because Δy is newly introduced just now),

so that sum is $p(y = Y) = \sum_{Y'} p(f(x + \Delta x) = Y') p(\Delta y = Y - Y')$. That is

$$p(y=Y) = \frac{1}{2^{32}} \sum_{Y'} p(f(x+\Delta x)=Y').$$

Since the sum is over all possible Y' , the total of the summed probabilities is 1, and $p(y=Y)=1/2^{32}$. The distribution of data $\mathcal{E}[x_m]$ in other locations m is unchanged. Done by a structural induction on the machine code program. \square

Proof (Theorem 3). Consider a probabilistic method f that guesses for a particular runtime value beneath the encryption ‘the top bit b is 1, not 0’, with probability $p_{C,T}$ for program C with trace T . The probability that f is right is

$$p((b_{C,T}=1 \text{ and } f(C,T)=1) \text{ or } (b_{C,T}=0 \text{ and } f(C,T)=0))$$

Splitting the conjunctions, that is

$$\begin{aligned} & p(b_{C,T}=1) p(f(C,T)=1 | b_{C,T}=1) \\ & + p(b_{C,T}=0) p(f(C,T)=0 | b_{C,T}=0) \end{aligned}$$

But the method f cannot distinguish the compilations it is looking at as the codes and traces are the same, modulo the (encrypted) values in them, which the adversary cannot read. The method f applied to C and T has nothing to cause it to give different answers but incidental features of encrypted numbers and its internal spins of a coin. Those are *independent* of if the bit b is 1 or 0 beneath the encryption, supposing the encryption is effective. So

$$\begin{aligned} p(f(C,T) = 1 | b_{C,T} = 1) &= p(f(C,T) = 1) = p_{C,T} \\ p(f(C,T) = 0 | b_{C,T} = 0) &= p(f(C,T) = 0) = 1 - p_{C,T} \end{aligned}$$

By Lemma 2, 1 and 0 are equally likely across all possible compilations C , so the probability f is right reduces to

$$\frac{1}{2} p_{C,T} + \frac{1}{2} (1 - p_{C,T}) = \frac{1}{2}$$

since $p(b_{C,T}=1) = p(b_{C,T}=0) = \frac{1}{2}$. \square

Corollary 2. *There is no method by which the operator can build a program C that gives an output $\mathcal{E}[y]$ where y is confined to an independently defined proper set Y of possibilities, not even stochastically with a probability higher than $|Y|/2^{32}$.*

Proof. The proof of Theorem 2 and Corollary 1 may be repeated, confining y to Y , or use Theorem 3, since its ‘probabilistic method f ’ includes constructing a program. \square

Proof. (Remark 2). The structure of the code of the AES decryption routine is known to the operator. By Corollary 1 the operator cannot construct the (encrypted) constants used in the AES decryption routine, but there may be others that will work (does anybody know?). Corollary 2 prevents the operator constructing a program to emit any one of the tuples of encrypted constants that will do, with any probability above chance. Theorem 3 prevents the operator doing it without programmed help. \square

References

1. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative technology for CPU-based attestation and sealing. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP 2013). ACM, New York (2013)
2. Biryukov, A.: Known plaintext attack. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security, pp. 704–705. Springer, Boston (2011). <https://doi.org/10.1007/978-1-4419-5906-5>
3. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45239-0_4
4. Breuer, P.T., Bowen, J.P.: A fully homomorphic crypto-processor design. In: Jürjens, J., Livshits, B., Scandariato, R. (eds.) ESSoS 2013. LNCS, vol. 7781, pp. 123–138. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36563-8_9
5. Breuer, P.T., Bowen, J.P., Palomar, E., Liu, Z.: On obfuscating compilation for encrypted computing. In: Proceedings of the 14th International Conference Security and Cryptography (SECRYPT 2017), pp. 247–254. SCITEPRESS, Portugal, July 2017. <https://doi.org/10.5220/0006394002470254>
6. Breuer, P.T., Bowen, J.P., Palomar, E., Liu, Z.: Superscalar encrypted RISC: the measure of a secret computer. In: Proceedings of the 17th International Conference on Trust, Security and Privacy in Computer and communications (TrustCom 2018), pp. 1336–1341. IEEE, CA, August 2018. <https://doi.org/10.1109/TrustCom/BigDataSE.2018.00184>
7. Buer, M.: CMOS-based stateless hardware security module. U.S. Patent Application No. 11/159,669, April 2006. <https://patents.google.com/patent/US20060072748>
8. Conway, J.H.: FRACTRAN: a simple universal programming language for arithmetic. In: Cover, T.M., Gopinath, B. (eds.) Open Problems in Communication and Computation, pp. 4–26. Springer, Heidelberg (1987). https://doi.org/10.1007/978-1-4612-4808-8_2
9. Fletcher, C.W., van Dijk, M., Devadas, S.: A secure processor architecture for encrypted computation on untrusted programs. In: Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing (STC 2012), pp. 3–8. ACM, New York (2012). <https://doi.org/10.1145/2382536.2382540>
10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009), NY, pp. 169–178 (2009). <https://doi.org/10.1145/1536414.1536440>
11. Gentry, C., Halevi, S.: Implementing gentry’s fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_9
12. 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
13. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC 1982), pp. 365–377. ACM, New York (1982). <https://doi.org/10.1145/800070.802212>

14. Götzfried, J., Eckert, M., Schinzel, S., Müller, T.: Cache attacks on Intel SGX. In: Proceedings of the 10th European Workshop on Systems Security (EuroSec 2017), pp. 2:1–2:6. ACM (2017). <https://doi.org/10.1145/3065913.3065915>
15. Hardin, D.: Real-time objects on the bare metal: an efficient hardware realization of the JavaTM virtual machine. In: Proceedings of the 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001), pp. 53–59. IEEE Computer Society, Washington (2001). <https://doi.org/10.1109/ISORC.2001.922817>
16. Hartman, R.: System for seamless processing of encrypted and non-encrypted data and instructions. US Patent 5,224,166, 29 June 1993. <https://patents.google.com/patent/US5224166>
17. Hashimoto, M., Teramoto, K., Saito, T., Shirakawa, K., Fujimoto, K.: Tamper resistant microprocessor. US Patent 2001/0018736 (2001). <https://patents.google.com/patent/US20010018736A1>
18. Irena, F., Murphy, D., Parameswaran, S.: CryptoBlaze: A partially homomorphic processor with multiple instructions and non-deterministic encryption support. In: Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 702–708. IEEE (2018). <https://doi.org/10.1109/ASPDAC.2018.8297404>
19. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
20. Kocher, P., et al.: Spectre attacks: exploiting speculative execution. [arXiv:1801.01203](https://arxiv.org/abs/1801.01203) [cs.CR], January 2018. <https://dblp.org/rec/bib/journals/corr/abs-1801-01203>
21. Lipp, M., et al.: Meltdown. [arXiv:1801.01207](https://arxiv.org/abs/1801.01207) [cs.CR], January 2018. <https://dblp.org/rec/bib/journals/corr/abs-1801-01207>
22. Ostrovsky, R., Goldreich, O.: Comprehensive software protection system. US Patent 5,123,045, 16 June 1992. <https://patents.google.com/patent/US5123045>
23. Patterson, D.A.: Reduced instruction set computers. *Commun. ACM* **28**(1), 8–21 (1985). <https://doi.org/10.1145/2465.214917>
24. Patterson, D.A., Hennessy, J.: *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, San Mateo (1994)
25. Rass, S., Schartner, P.: On the security of a universal cryptocomputer: the chosen instruction attack. *IEEE Access* **4**, 7874–7882 (2016). <https://doi.org/10.1109/ACCESS.2016.2622724>
26. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. *Found. Secure Comput.* **4**(11), 169–180 (1978)
27. Sinha Roy, S., Järvinen, K., Vercauteren, F., Dimitrov, V., Verbauwheide, I.: Modular hardware architecture for somewhat homomorphic function evaluation. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 164–184. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_9
28. Schoeberl, M.: Java technology in an FPGA. In: Becker, J., Platzner, M., Vernalde, S. (eds.) FPL 2004. LNCS, vol. 3203, pp. 917–921. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30117-2_99
29. Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., Russinovich, M.: VC3: trustworthy data analytics in the cloud using SGX. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 38–54, May 2015. <https://doi.org/10.1109/SP.2015.10>

30. Tsoutsos, N.G., Maniatakos, M.: The HEROIC framework: encrypted computation without shared keys. *IEEE Trans. CAD IC Syst.* **34**(6), 875–888 (2015). <https://doi.org/10.1109/TCAD.2015.2419619>
31. Wang, Z., Lee, R.B.: Covert and side channels due to processor architecture. In: *Proceedings of the 2nd Annual Computer Security Applications Conference (ACSAC 2006)*, pp. 473–482. IEEE (2006). <https://doi.org/10.1109/ACSAC.2006.20>
32. Weicker, R.: Dhrystone: a synthetic systems programming benchmark. *Commun. ACM* **27**(10), 1013–1030 (1984). <https://doi.org/10.1145/358274.358283>

Full Paper Session V: Privacy Protection



PPOIM: Privacy-Preserving Shape Context Based Image Denoising and Matching with Efficient Outsourcing

Meng Zheng, Jun Zhou^(✉), Zhenfu Cao, and Xiaolei Dong

Shanghai Key Lab for Trustworthy Computing, East China Normal University,
Shanghai 200062, China

zmecdu2016@163.com, {jzhou,zfcao,dongxiaolei}@sei.ecnu.edu.cn

Abstract. With the emerging techniques of wireless communication and cloud computing, large volumes of multimedia data are outsourced from resource constrained users to the cloud with abundant resource for both delegated storage and computation. Unfortunately, there is a risk of users' image privacy leakage in the process of outsourcing to untrusted cloud. Most of the existing work achieved privacy-preserving image feature extraction and matching by using public key (fully) homomorphic encryption (FHE), but the heavy computational overhead and communication overhead cannot adapt to resource-constrained mobile devices. Other works disabled to realize image denoising in the encrypted domain or only focused on the scale-invariant feature transform (SIFT) descriptor that is inappropriate for position-sensitive feature extraction. To address these issues, in this paper, a privacy-preserving shape context based image denoising and matching protocol PPOIM with efficient outsourcing is proposed. Firstly, to improve the accuracy of image matching, a privacy-preserving image denoising scheme PPID is proposed without exploiting public key FHE. Then, based on PPID, a privacy-preserving image matching protocol PPOIM adopting shape context descriptor is devised, where two secure and efficient comparison and counting protocols in the encrypted domain are presented. All the original image privacy, query image privacy and image matching result privacy are well protected. Finally, formal security proof and extensive simulations on real-world data sets demonstrate the efficiency and practicability of our proposed PPOIM.

Keywords: Image matching · Privacy-preserving
Shape context descriptor · Secure outsourced computation

1 Introduction

With the development of big data and social network like Flickr or Facebook, huge amounts of personal users' multimedia data are delegated to the cloud from the resource-constrained mobile devices for both outsourced storage and

outsourced computation with expensive complexity. Among types of image processing, image matching have played an increasingly important role in our everyday life. The widely adopted technique of content-based image match means that the cloud returns the boolean match result between images and the user's queried one with similar features such as color, shape and texture that are extracted by exploiting scale-invariant feature transform (SIFT) descriptor, shape context (SC) descriptor, etc. Taking medical image for example, the physicians can judge the aging degree of the elderly persons, by matching their medical image (i.e. X-ray film) with the pattern images signaling different levels of aging, adopting the extracted features such as the step length and the angle with which the elderly's limbs can be lifted.

Unfortunately, the cloud server either works under the semi-honest model or malicious model, where the cloud either strictly carries out the protocol specifications but intending to extract the private information from the interactions with users, or performs arbitrarily to destruct the protocol execution. Therefore, it would disclose the private health condition of the elderly persons by delegating the medical images in their plaintext to the cloud for feature extraction and matching. How to devise an efficient privacy preserving image feature extraction and matching protocol becomes a critical issue for convincing solutions.

Recently, a series of research has focused on the field of privacy-preserving image feature extraction and matching [1–3, 6–8, 12, 13, 16–19, 21, 23–25]. Hsu et al. [4] studied privacy-preserving outsourced feature extraction in the encrypted domain, by using Paillier's additive homomorphic encryption. Unfortunately, their protocol is either computationally-intensive or risks the privacy leakage of the original image. To address the issues, Hu et al. [5] devised a secure outsourcing computation of feature extraction over encrypted image data, by splitting the original image and designing privacy-preserving multiplication and comparison protocols executed by two non-colluded servers, by exploiting Brakerski et al.'s somewhat homomorphic encryption [15]. However, the level of fully homomorphism respectively proposed in [14] and [15] is restricted and the ciphertext expansion would increase every time a ciphertext multiplication is required. Thus the heavy computational and communication overhead in both [4] and [5] is intolerable by resource-constrained devices. J. Zhou et al. [12] proposed an efficient privacy-preserving image feature extraction protocol, however all the above [4, 5, 12] adopted SIFT descriptor, which is only appropriate for searching images with a transforming rotation, scaling, and translation, but cannot be applied to the scenario of image matching adopting the features as relative positions between pixels, as is suggested in the example for judging the aging level of the elderly. Belongie et al. [8] presented an approach to measure similarity between shapes for object recognition based on shape context based descriptor. However, the issue of image privacy-preserving was not considered. In [6], Wang et al. studied privacy-preserving shape-based feature extraction by exploiting the techniques of homomorphic encryption and the garbled circuit protocol, respectively. The high computational complexity can still not adapt to resource-constrained users.

On the other hand, image noise may be introduced under different conditions from intrinsic sensors or extrinsic environments, which are often difficult to avoid in practice and significantly affect the accuracy of image matching. Zheng et al. [9] proposed a privacy-preserving image denoising protocol from external cloud databases by using secure similarity search, Yao’s garbled circuits and image denoising operations, to ensure that similar patches with high quality are precisely obtained after encrypted similarity search. Unfortunately, the denoising operations were completed in the plaintext domain without considering image privacy protection. To address the issues mentioned above, in this paper, a privacy-preserving shape context based image denoising and matching protocol PPOIM with efficient outsourcing is proposed. The main contributions are summarized as follows.

Firstly, a privacy-preserving image denoising protocol PPID is proposed in the encrypted domain, by devising a lightweight secure outsourced computation without public key fully homomorphic encryption (FHE).

Secondly, based on the proposed PPID, we present an efficient privacy-preserving image matching scheme PPOIM based on shape context descriptor. Especially, two efficient comparison and counting protocols in the encrypted domain are carefully designed. Both the original image privacy and the matching result privacy are well protected, and only the authorized user can successfully decipher the final matching result.

Finally, formal security proof and extensive evaluations demonstrate the efficiency and practicability of our PPOIM. Both the computational cost and communication cost are dramatically reduced, compared to the state-of-the-art using public key FHE.

The remainder of this paper is organized as follows. We present the network architecture and the security model in Sect. 2. Then the privacy-preserving image denoising protocol PPID and the privacy-preserving shape context based image matching protocol are proposed in Sect. 3. Formal security proof and performance evaluations are respectively presented in Sects. 4 and 5. Finally, we conclude our paper in Sect. 6.

2 Network Architecture and Security Model

2.1 Network Architecture

The network model of privacy-preserving shape context based image denoising and matching mainly comprises three entities: the data owner, the user and the cloud, which are demonstrated in Fig. 1. The main procedure of our proposed PPOIM are described as follows, (1) The data owner outsources an encrypted database of image patches to the cloud for generating high quality similar patches; (2) The user sponsors an image search token request to the data owner; (3) The data owner performs the search token authorization to the user if her/his image query is permitted; (4) The user uploads the encrypted

query image together with the search token to the cloud; (5) The cloud performs privacy-preserving image denoising and matching by adopting shape context based descriptor and calculating the matching cost in the encrypted domain; (6) The cloud returns all encrypted matching results to the user for decrypting, if the matching cost is smaller than the cost threshold set by the user, two images are considered to be matched each other.

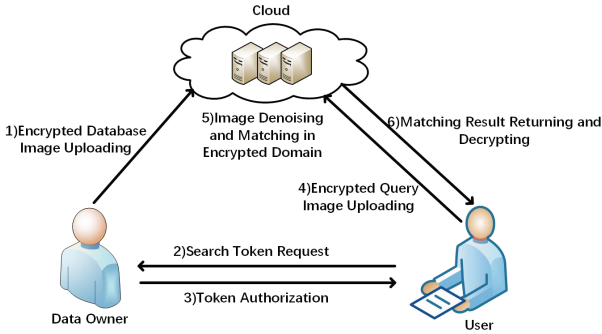


Fig. 1. Network architecture of privacy-preserving image denoising and matching

2.2 Security Model

We formally define the image privacy and the matching result privacy for our proposed PPOIM. The cloud is assumed to be honest-but-curious, which strictly executes the protocol specification but tries its best to extract the private information from the interactions among data owner, user and itself. Image privacy refers to that the data owner’s database images cannot be accessed by the collusion between the cloud and malicious users and the user’s query image cannot be disclosed to the collusion of the cloud and malicious owners. The matching result privacy means that whether the query image matches the database image can only be accessed by the authorized users. The formal security models of these three types of privacy are detailed in the full paper.

3 The Proposed PPOIM

In this section, a privacy-preserving shape context based image denoising and matching protocol PPOIM with efficient outsourcing is proposed, which is composed of three phases, namely the setup phase generating the required parameters, the privacy-preserving image denoising phase PPID, and the privacy-preserving image matching phase PPOIM where the final matching result can be decrypted by the authorized user.

3.1 Setup Phase

On input 1^λ where λ is the security parameter, the system runs a trapdoor permutation generator denoted as a probabilistically polynomial time (PPT) algorithm $\mathcal{G}(1^\lambda)$ and outputs a tuple of permutations (f, f^{-1}) on $\{0, 1\}^{2\lambda}$ with a pair of corresponding keys (PK_f, SK_f) . It also outputs two hash functions $H_0, H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{2\lambda}$ and a cluster of locality-sensitive hash (LSH) functions $h_i : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda (i = 1, 2, \dots, l)$. The public parameters are $PPR = (PK_f, H_0, H_1, h_i (i = 1, 2, \dots, l))$ and the secret key is SK_f assigned to the user. Besides, suppose there is a secure symmetric encryption scheme $SE = (SE.Setup, SE.KGen, SE.E, SE.D)$ with a secret key $\mathbf{K} = (K_g, K_p)$ shared between the data owner and the user, and $F : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ is a pseudorandom function (PRF).

3.2 The Proposed Privacy-Preserving Image Denoising Protocol PPID

In this subsection, an efficient privacy-preserving image denoising protocol PPID is presented, which is composed of four algorithms: **IndexGen** performed on the data owner side, encrypting patch databases with their corresponding secure indexes by exploiting locality-sensitive hashing (LSH) and symmetric encryption (SE), and uploading the encrypted database images to the cloud; **Request** executed on the user side, generating a secure query search token, and transmitting the search token and encrypted query patch to the cloud; **Search** run on the cloud side, ranking all candidate patches and filtering the false positive candidates for denoising operation; and **Denoising** carried out on the cloud side, recovering the clean encrypted patch.

- (1) $\{[\mathbf{P}], \mathcal{D}\} \leftarrow \mathbf{IndexGen}(\mathbf{K}, PK_f, \mathbf{P})$. It takes as input the secret key $\mathbf{K} = (K_g, K_p)$, the public key PK_f for patch encryption and the patch set $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N\}$, where N is the total number of database patches, and returns $\{[\mathbf{P}], \mathcal{D}\}$, where $[\mathbf{P}] = \{[\mathbf{p}_1], [\mathbf{p}_2], \dots, [\mathbf{p}_N]\}$ and \mathcal{D} refer to the ciphertexts of database images and a generic dictionary.

Let $p_{i,t} = (\rho_{i,t}, \theta_{i,t}) (i = 1, 2, \dots, N; t = 1, 2, \dots, n)$ be the polar coordinate of the t -th pixel in database patch \mathbf{p}_i , and $\mathbf{p}_i = \{p_{i,t}\}_{t=1}^n = \{(\rho_{i,t}, \theta_{i,t})\}_{t=1}^n$. The ciphertexts of database images $[\mathbf{P}] = \{[\mathbf{p}_i]\} (i = 1, 2, \dots, N)$ are encrypted as follows. For brief description, we only detailed the process for encrypting $\rho_{i,t}$, and $\theta_{i,t}$ can be encrypted in the same way. The image data owner randomly chooses three big primes p, q, h of $|p| = |q| = |h| = \lambda$ which are kept secret, and computes the publicized $N'' = pq, N' = pqh$. The message space of $\rho_{i,t}$ is on $\mathbb{Z}_{N''}$ as a hidden subgroup of $\mathbb{Z}_{N'}$. Then, the owner computes $\rho_{i,t,p} \equiv \rho_{i,t} \bmod p, \rho_{i,t,q} \equiv \rho_{i,t} \bmod q$. She/he also randomly selects $K_{i,t} \in_R \mathbb{Z}_h$, and computes the additive blinding factor $U_{i,t}^{add} = K_{i,t} N'' \in_R \mathbb{Z}_{N'}$ and the multiplicative blinding factor $U_{i,t}^{mul} = K_{i,t} N'' + 1 \in_R \mathbb{Z}_{N'}$ ($i = 1, 2, \dots, N; t = 1, 2, \dots, n$) such that the final image matching results in our proposed PPOIM can be correctly obtained in the

decryption phase **ImgDec** by calling the algorithm **PPOIM.Dec**(\cdot) where all the additive and multiplicative blinding factors $U_{i,t}^{add}$, $U_{i,t}^{mul}$ can be cancelled out after modular N' . Since we have $1 \equiv q^{-1}q \pmod{p}$, $1 \equiv p^{-1}p \pmod{q}$, the data owner calculates the ciphertexts as follows,

$$\begin{aligned} C_{1,1} &= f_{PK_f}(p \parallel h), \\ C_{2,\rho_{i,t}} &= q^{-1}q\rho_{i,t,p}^p + p^{-1}p\rho_{i,t,q}^q + U_{i,t}^{add} \pmod{N'}, \\ C_{3,\rho_{i,t}} &= (q^{-1}q\rho_{i,t,p}^p + p^{-1}p\rho_{i,t,q}^q)U_{i,t}^{mul} \pmod{N'}. \end{aligned} \quad (1)$$

where \parallel means the concatenation operation, and q^{-1} , p^{-1} respectively denote the inverses of q and p in \mathbb{Z}_p^* and \mathbb{Z}_q^* . Finally, the data owner computes $C_{ram,\rho}^{add} = H_0(p \parallel h \parallel \bigcup_{i=1, t=1}^{N,n} C_{2,\rho_{i,t}})$, $C_{ram,\rho}^{mul} = H_0(p \parallel h \parallel \bigcup_{i=1, t=1}^{N,n} C_{3,\rho_{i,t}})$, and denotes $[\rho_{i,t}] = (C_{2,\rho_{i,t}}, C_{3,\rho_{i,t}})$. Note that $[\theta_{i,t}] = (C_{2,\theta_{i,t}}, C_{3,\theta_{i,t}})$ can be computed in the same way. We have $[\mathbf{p}_{i,t}] = ([\rho_{i,t}], [\theta_{i,t}])$ and the ciphertexts of database images $[\mathbf{P}] = (\{[\mathbf{p}_{i,t}](i = 1, 2, \dots, N; t = 1, 2, \dots, n)\}, C_{ram,\rho}^{add}, C_{ram,\rho}^{mul}, C_{ram,\theta}^{add}, C_{ram,\theta}^{mul})$. We denote the encryption algorithm to generate ciphertexts of database images $[\mathbf{P}]$ as **PPOIM.Enc**(\cdot) which would also be exploited in the following phases of our proposed PPID. Then, the data owner initializes a dictionary \mathcal{D} and the LSH value set \mathbf{G} as two empty sets. For each patch \mathbf{p}_i in patch set \mathbf{P} , the data owner computes LSH values with l LSH functions $h_1(\cdot), h_2(\cdot), \dots, h_l(\cdot)$,

$$\mathbf{g}_i = (h_1(\mathbf{p}_i) \parallel 1, \dots, h_l(\mathbf{p}_i) \parallel l), \quad (2)$$

where vector \mathbf{g}_i is the i -th element in \mathbf{G} , $g_{i,j} = h_j(\mathbf{p}_i) \parallel j (j = 1, 2, \dots, l)$ is the j -th element in vector \mathbf{g}_i . Then, for each $g_{i,j}$ in $\mathbf{g}_i \in \mathbf{G}$, the owner generates

$$K_{1,i,j} = F(K_g, 1 \parallel g_{i,j}), K_{2,i,j} = F(K_g, 2 \parallel g_{i,j}). \quad (3)$$

The data owner initializes a counter $ctr = 0$. For each $g_{i,j}$, if there exists any $g_{k,j} = g_{i,j} (k \in \{1, 2, \dots, N\})$, then it considers \mathbf{p}_k is associated with $g_{i,j}$ and $ctr \leftarrow ctr + 1$. The data owner computes tag $u_{i,j}$ by applying pseudorandom function F and encrypts the corresponding patch sub-identifier $id_{k,j}$ using the symmetric encryption scheme SE as follows,

$$u_{i,j} = F(K_{1,i,j}, ctr), v_{i,j} = SE.E(K_{2,i,j}, id_{k,j}), \quad (4)$$

where $id_k = id_{k,1} \parallel id_{k,2} \parallel \dots \parallel id_{k,l}$ is the unique identifier of a database patch \mathbf{p}_k and $id_{k,j} (j = 1, 2, \dots, l)$ is the sub-identifier of $h_j(\mathbf{p}_k) \parallel j$ in \mathbf{g}_k . Then, the tag-ciphertext pair $(u_{i,j}, v_{i,j})$ is inserted to a generic dictionary \mathcal{D} . Finally, the data owner sends $([\mathbf{P}], \mathcal{D})$ to the cloud server.

(2) $\{Q, [\mathbf{q}], [t''], [T]\} \leftarrow \mathbf{Request}(\mathbf{K}, PK_f, \mathbf{q}, t'', T)$. When a user wants to request the database, she/he firstly need to obtain the token authorization from the data owner by receiving $C_{1,1} = f_{PK_f}(p \parallel h)$. Then, she/he decrypts $p \parallel h = f_{SK_f}^{-1}(C_{1,1})$ by using secret key SK_f and computes $q = N'(ph)^{-1}$. After that, the user generates the ciphertext $[\mathbf{q}]$ and a secure search token Q for the query patch \mathbf{q} as follows. The user firstly hashes \mathbf{q} into a vector of l LSH values

$$\mathbf{g} = \{h_1(\mathbf{q}) \parallel 1, \dots, h_l(\mathbf{q}) \parallel l\}, \quad (5)$$

where $g_j = h_j(\mathbf{q}) \parallel j$ ($j = 1, 2, \dots, l$) is the j -th element of the \mathbf{g} . For each LSH value g_j , a sub-token $Q_j = (K_{1,j}, K_{2,j})$ is generated via

$$K_{1,j} \leftarrow F(K_g, 1 \parallel g_j), K_{2,j} \leftarrow F(K_g, 2 \parallel g_j). \quad (6)$$

The resulting secure search token $Q = \{Q_1, Q_2, \dots, Q_l\}$. On the other hand, the user randomly selects $K_t \in_R \mathbb{Z}_h$, and computes $U_t^{add} = K_t N^n$, $U_t^{mul} = K_t N^n + 1 \in_R \mathbb{Z}_{N'}$ ($t = 1, 2, \dots, n-1$) such that the final matching result would be successfully decrypted after modulo N^n . Then the user encrypts patches $q_t \in \mathbf{q}$ ($t = 1, 2, \dots, n$) with **PPOIM.Enc**(\cdot) to generate the ciphertexts $[q_t] = ([\rho_t], [\theta_t])$. Thus, $[\rho_t] = (C_{2,\rho_t}, C_{3,\rho_t})$, $[\theta_t] = (C_{2,\theta_t}, C_{3,\theta_t})$, $[\mathbf{q}] = \{[\rho_t], [\theta_t], C'_{ram,\rho}, C'_{ram,\rho}, C'_{ram,\theta}, C'_{ram,\theta}\}$. In addition, the user chooses two thresholds t^n, T respectively for obtaining the candidate patches for denoising and for matching cost comparison to derive the final image matching result, encrypts them into $[t^n], [T]$ by exploiting algorithm **PPOIM.Enc**(\cdot). Finally, the user sends $(Q, [\mathbf{q}], [t^n], [T])$ to the cloud.

- (3) $\{S^*, H\} \leftarrow \mathbf{Search}(Q, [\mathbf{q}], [t^n], [\mathbf{P}], \mathcal{D})$. For each sub-token Q_j in Q , the cloud re-computes the pseudorandom tag $u_j = F(K_{1,j}, ctr)$, where ctr is a self-incremental counter and initialized as 0. Let f_{id_i} be an occurrence counter initialized as 0. The cloud searches the generic dictionary \mathcal{D} according to the pseudorandom tag u_j to locate the associated $v_{i,j}$ ($j \in \{1, 2, \dots, l\}$). If $u_j = u_{i,j}$ ($j \in \{1, 2, \dots, l\}$), it decrypts the corresponding patch identifier $id_{k,j} = SE.D(K_{2,j}, v_{i,j})$ via $K_{2,j}$, and increases $f_{id_i} \leftarrow f_{id_i} + 1$. Then, the cloud ranks the candidates \mathbf{p}_i based on the occurrence counter f_{id_i} , and derives an initial set S^* of candidate patches.

However, LSH is an approximation algorithm that trades accuracy for efficiency, which usually locates a large number of candidates with false positives introduced. Thus, to filter the false positive candidates, the cloud computes distance between candidate \mathbf{p}_i in S^* and query image \mathbf{q} . For each encrypted candidates patch $[\mathbf{p}_i] = \{[\mathbf{p}_{i,t}]\} = \{([\rho_{i,t}], [\theta_{i,t}])\} = \{((C_{2,\rho_{i,t}}, C_{3,\rho_{i,t}}), (C_{2,\theta_{i,t}}, C_{3,\theta_{i,t}}))\}$ ($t = 1, 2, \dots, n$) and the encrypted query patch $[\mathbf{q}] = \{[\mathbf{q}_t]\} = \{([\rho_t], [\theta_t])\} = \{((C_{2,\rho_t}, C_{3,\rho_t}), (C_{2,\theta_t}, C_{3,\theta_t}))\}$ ($t = 1, 2, \dots, n$), the cloud computes the squared distance between $[\mathbf{p}_i]$ and $[\mathbf{q}]$ in the encrypted domain to securely refine the ranking for each candidate in S^* .

$$d^2([\mathbf{p}_i], [\mathbf{q}]) = \sum_{t=1}^n (C_{3,\rho_{i,t}}^2 + C_{3,\rho_t}^2) - 2 \sum_{t=1}^n [C_{3,\rho_{i,t}} C_{3,\rho_t} \cos(C_{2,\theta_{i,t}} - C_{2,\theta_t})], \quad (7)$$

where the cosine function is approximated by aggregating the first t' items in its power series expansion as $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + (-1)^{t'} \frac{x^{2t'}}{(2t')!}$ (i.e. In performance evaluation, we would study the impact of different t' on the accuracy of image matching result and the efficiency of our proposed PPOIM.)

Then the cloud compares the squared distance $d^2([\mathbf{p}_i], [\mathbf{q}])$ with threshold $[t^n]$ in the encrypted domain. Let the binary representations of t^n and $d^2([\mathbf{p}_i], [\mathbf{q}])$

be $t^n = \overline{m_{n-1}m_{n-2} \cdots m_0}$ and $d^2(\mathbf{p}_i, \mathbf{q}) = \overline{m'_{n-1}m'_{n-2} \cdots m'_0}$. Owing to the fully homomorphic property of algorithm **PPOIM.Enc**(\cdot), we denote

$$\begin{aligned} [t^n] &= \mathbf{PPOIM.Enc}(m_{n-1})\mathbf{PPOIM.Enc}(2^{n-1}) + \mathbf{PPOIM.Enc}(m_{n-2}) \\ &\mathbf{PPOIM.Enc}(2^{n-2}) + \cdots + \mathbf{PPOIM.Enc}(m_0)\mathbf{PPOIM.Enc}(1), \\ d^2([\mathbf{p}_i], [\mathbf{q}]) &= \mathbf{PPOIM.Enc}(m'_{n-1})\mathbf{PPOIM.Enc}(2^{n-1}) + \mathbf{PPOIM.Enc} \\ &(m'_{n-2})\mathbf{PPOIM.Enc}(2^{n-2}) + \cdots + \mathbf{PPOIM.Enc}(m'_0)\mathbf{PPOIM.Enc}(1). \end{aligned} \tag{8}$$

Then by exploiting the method of successive division with **PPOIM.Enc**(2) that can also be executed and uploaded by the user in the previous **Request** algorithm, the cloud can derive the binary encryption of $[t^n] = \overline{\mathbf{PPOIM.Enc}(m_{n-1})\mathbf{PPOIM.Enc}(m_{n-2}) \cdots \mathbf{PPOIM.Enc}(m_0)} = \overline{m_{n-1}^e m_{n-2}^e \cdots m_0^e}$ and $d^2([\mathbf{p}_i], [\mathbf{q}]) = \overline{\mathbf{PPOIM.Enc}(m'_{n-1})\mathbf{PPOIM.Enc}(m'_{n-2}) \cdots \mathbf{PPOIM.Enc}(m'_0)} = \overline{m'_{n-1}{}^e m'_{n-2}{}^e \cdots m'_0{}^e}$. For binary representations, we have the following observation for $i = 0, 1, \dots, n - 1$,

$$\begin{aligned} m_i &> m'_i \text{ if and only if } m_i m'_i + m_i = 1, \\ m_i &= m'_i \text{ if and only if } m_i + m'_i + 1 = 1, \\ m_i &< m'_i \text{ if and only if } m_i m'_i + m_i + 1 = 1. \end{aligned} \tag{9}$$

Therefore, according to the property of full homomorphism of **PPOIM.Enc**(\cdot), the cloud can evaluate Eq.(9) in the encrypted domain. To compare t^n and $d^2(\mathbf{p}_i, \mathbf{q})$, the binary chop method is adopted. Specifically for $l = \lceil \frac{n}{2} \rceil$, we have

$$\underbrace{\overline{m_{n-1} \cdots m_l}}_{hbs(t^n)} \underbrace{\overline{m_{l-1} \cdots m_0}}_{lbs(t^n)} > \underbrace{\overline{m'_{n-1} \cdots m'_l}}_{hbs(d^2(\mathbf{p}_i, \mathbf{q}))} \underbrace{\overline{m'_{l-1} \cdots m'_0}}_{lbs(d^2(\mathbf{p}_i, \mathbf{q}))} \tag{10}$$

if and only if $(hbs(t^n) > hbs(d^2(\mathbf{p}_i, \mathbf{q}))) \vee (hbs(t^n) = hbs(d^2(\mathbf{p}_i, \mathbf{q})) \wedge (lbs(t^n) > lbs(d^2(\mathbf{p}_i, \mathbf{q}))))$, where $hbs(x)$, $lbs(x)$ respectively refer to the higher binary sequence and the lower binary sequence of x . To recursively performing the comparison until deriving the final output, it is also required to define the following three variations $h_{i,j}, e_{i,j}$ and $l_{i,j}$, respectively referring to the boolean logic values for the conditions $\overline{m_{i+j-1} \cdots m_i} > \overline{m'_{i+j-1} \cdots m'_i}$, $\overline{m_{i+j-1} \cdots m_i} = \overline{m'_{i+j-1} \cdots m'_i}$, $\overline{m_{i+j-1} \cdots m_i} \geq \overline{m'_{i+j-1} \cdots m'_i}$. It is obviously observed that $h_{0,n}, e_{0,n}, l_{0,n}$ will be the final result. For each time, by selecting $l = \lceil \frac{j}{2} \rceil$ and combining Eqs. (9) and (10), we have

$$\begin{aligned} (1) & \text{ If } j = 1, h_{i,j} = m_i m'_i + m_i, \text{ Else } h_{i,j} = h_{i+l,j-1} + e_{i+l,j-1} t_{i,l}; \\ (2) & \text{ If } j = 1, e_{i,j} = m_i + m'_i, \text{ Else } e_{i,j} = e_{i+l,j-1} e_{i,j}; \\ (3) & \text{ If } j = 1, l_{i,j} = m_i m'_i + m_i + 1, \text{ Else } l_{i,j} = t_{i+l,j-1} + e_{i+l,j-1} l_{i,l}. \end{aligned} \tag{11}$$

By comparing the threshold for denoising $[t^n]$ with each $d^2([\mathbf{p}_i], [\mathbf{q}])$ corresponding to each candidate \mathbf{p}_i in S^* , all the encrypted comparing results $H = \{[h_{0,n}]_i\} (i = 1, 2, \dots, N)$ can be computed according to Eq. (11).

- (4) $\hat{\mathbf{q}} \leftarrow \mathbf{Denoising}([\mathbf{q}], S^*, H)$ Collecting the encrypted database patch candidates in S^* , the cloud performs privacy-preserving image denoising by exploiting the classical technique of non-local means (*NLM*) [10], [11], in which a weighted average computation in the encrypted domain is adopted. Given a noisy patch $[\mathbf{q}]$ and a set of ranked patches $S^* = \{[\mathbf{p}_1], [\mathbf{p}_2], \dots, [\mathbf{p}_N]\}$, the clean patch $\hat{\mathbf{q}}$ is estimated as the weighted average of all ranked patches, the detailed process is described as follows.

To compute the normalizing factor $[Z]$, we define h as a filtering parameter depending on the standard deviation σ of the zero-mean Gaussian noise. Next, the cloud calculates $[Z] = \sum_{i=1}^N e'$, where

$$e' = e^{-d^2([\mathbf{p}_i], [\mathbf{q}])h^{-2}} = \sum_{i'=0}^{t'} (-1)^{i'} \frac{(d^2([\mathbf{p}_i], [\mathbf{q}])h^{-2})^{i'}}{i'!} \quad (12)$$

The index function e^x is approximated by aggregation the first t' items in its power series expansion as $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^{t'}}{t'!}$ (i.e. In performance evaluation, we would study the impact of different t' on the accuracy of image matching result and the efficiency of our proposed PPOIM) and the h^{-1} is the inverse of h . Next, the cloud calculates the weight $\omega([\mathbf{q}], [\mathbf{p}_i]) = [Z]^{-1}e'$, where $[Z]^{-1}$ is the inverse of $[Z]$. Finally, the clean patch $\hat{\mathbf{q}}$ is estimated as the weighted average of all encrypted ranked patches,

$$[\hat{\mathbf{q}}] = \sum_{i=1}^N \omega([\mathbf{q}], [\mathbf{p}_i])[\mathbf{p}_i][h_{0,n}]_i = \sum_{i=1}^N [Z]^{-1}e'[\mathbf{p}_i][h_{0,n}]_i. \quad (13)$$

If the full query image \mathbf{I}_q is composed of several patches, then for each patch, the cloud adopts the same denoising method as is explained above to process patch $[\mathbf{q}]$.

3.3 The Proposed Privacy-Preserving Image Matching Protocol PPOIM

After denoising query image in the cloud, an estimate of the original query image $[\hat{\mathbf{I}}_q]$ composed of all $[\hat{\mathbf{q}}]$ can be produced. In this section, we firstly clarify the definition of Shape Context (SC) descriptor. Then, based on our proposed PPID in Sect. 3.2, a privacy-preserving SC-based image matching protocol PPOIM with efficient outsourcing is proposed, which consists of three algorithms **SCGen**, **ImgMatch** and **ImgDec**. We assume that as long as at least one shape in the database image matches the query image $[\hat{\mathbf{I}}_q]$, these two images matches successfully, regardless of the position and rotation angle of the shape in the database image. We also assume that database images and the query image are in the same polar coordinate system, which means that the query image shares the center point with database images. The cloud computes matching cost between the encrypted denoised query image $[\hat{\mathbf{I}}_q]$ and all database images $[\mathbf{I}_i] (i = 1, 2, \dots, N)$, then compares all matching cost with a threshold $[T]$.

Shape Context in Plaintext Domain. Belongie et al. [8] introduced the idea of shape context. In their work, a shape is represented by a set of points sampled from the contours, and shape context describes location information about all other boundary points relative to a specific boundary point in the shape. Here, we prefer to sample the shape with roughly uniform spacing. Each shape context is a coarse log-polar histogram of the coordinates of the remaining points measured using the reference point as the origin and the line joining the reference point and the center as the pole axis. Additionally, the center of mass of any shape is invariant to scaling, rotation or translation. Figure 2 shows the definition of Shape Context.

The shape ‘A’ in Fig. 2 is composed of a set of discrete points $A = \{a_i\} (i = 1, 2, \dots, n)$ sampled from the contour. To compute a shape context of a_i in A , we create a new polar coordinate. Let the referenced point a_i be the new pole and the line joining a_i and the center o of the shape be the new pole axis $\overline{a_i o}$. The set of vectors originating from a_i to the remained $n - 1$ points is generated. To compute the shape context, we firstly divide the full image space into 12 sectors by angle, then draw 5 concentric circles with a_i as center point and the power of 2 as radius. Thus, the full image can be divided into 60 bins. Next, we count the number of boundary points within each bin to form the shape context. All points falling in different bins forms different relative vectors, which becomes the shape context of the point a_i . Then we compute $T_{i,k}$ to indicate the set of points, namely vector $\overline{a_i a_j}$ in $bin(k)$, selecting a_i as the referenced point,

$$T_{i,k} = \{a_j | a_j \neq a_i, (\overline{o a_j} - \overline{o a_i}) \in bin(k)\}. \tag{14}$$

Let $h_i(k) = |T_{i,k}|$ represent the number of points in $T_{i,k}$, thus the shape context $h_i = \{h_i(k)\} (k = 1, 2, \dots, 60)$.

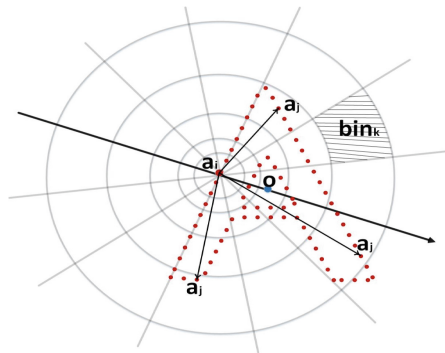


Fig. 2. The description of shape context

Privacy-Preserving Image Matching. In this subsection, a privacy-preserving image matching protocol based on shape context descriptor is proposed, which comprises the following three algorithms **SCGen**, **ImgMatch** and **ImgDec**. The details are presented as follows.

- (1) $\{\{[h_x(k)]\}, \{[h_t(k)]\}\} \leftarrow \mathbf{SCGen}([\mathbf{I}_i], [\hat{\mathbf{I}}_q])$ To generate the encrypted shape context for each sample point in $[\mathbf{I}_i]$ and $[\hat{\mathbf{I}}_q]$. Without loss of generality, we assume that the point $s_{q,t}$ is the pole in shape of image $[\hat{\mathbf{I}}_q]$ and the point $s_{i,x}$ is the pole in shape of image $[\mathbf{I}_i]$, and all $s_{q,t}$, $s_{i,x}(t, x = 1, 2, \dots, n)$ are in the edge of shapes in each image, then we connect the pole $s_{q,t}$ with the center point o_q of shape in image $[\hat{\mathbf{I}}_q]$, the pole $s_{i,x}$ with the same point o_q respectively and divide the full image space into 60 bins, by adopting the method referred in **Shape Context in Plaintext Domain** part. Then the cloud counts how many points are located in $bin(k)$ ($k = 1, 2, \dots, 60$) in each shape respectively. Here, we mainly focus on generating the shape context of point $s_{i,x}$ in image $[\mathbf{I}_i]$. Each $bin(k)$ is fixed by two angles (θ_k, θ_{k_1}) and two polar radius (ρ_k, ρ_{k_1}) , where $\theta_{k_1} > \theta_k$ and $\rho_{k_1} > \rho_k$. To determine whether an encrypted point $[s_{i,x'}] = ([\rho_{x'}], [\theta_{x'}])$ is located in $bin(k)$, the cloud adopts the following modified privacy-preserving comparison operations presented in our proposed privacy-preserving image denoising protocol PPID. If the point $[s_{i,x'}] = ([\rho_{x'}], [\theta_{x'}])$ is in $bin(k)$, it simultaneously satisfies the following four conditions:

- (a) $\rho_{x'} > \rho_k$, returning a final result $[h_{0,n}^{x',k,1}]$; (b) $\rho_{x'} \leq \rho_{k_1}$, returning $[1 - h_{0,n}^{x',k,2}]$;
 (c) $\theta_{x'} > \theta_k$, returning a final result $[h_{0,n}^{x',k,3}]$; (d) $\theta_{x'} \leq \theta_{k_1}$, returning $[1 - h_{0,n}^{x',k,4}]$.

Thus, the computation result $[h^{x'}] = [h_{0,n}^{x',k,1}] \cdot [1 - h_{0,n}^{x',k,2}] \cdot [h_{0,n}^{x',k,3}] \cdot [1 - h_{0,n}^{x',k,4}]$ means whether a point $s_{i,x'}$ is in $bin(k)$, and $[h_x(k)] = \sum_{x'=1}^n [h^{x'}]$ represents the encrypted number of points in $bin(k)$. Thus, all $\{[h_x(k)]\}$ ($k = 1, 2, \dots, 60$) constitutes the shape context of point $s_{i,x}$. Similarly, the shape context of point $s_{q,t}$ in image $[\hat{\mathbf{I}}_q]$ can be calculated as $\{[h_t(k)]\}$ ($k = 1, 2, \dots, 60$).

- (2) $\{\{[h_{0,n}^i]\}, C_3\} \leftarrow \mathbf{ImgMatch}(\{[h_x(k)]\}, \{[h_t(k)]\}, [T])$ After obtaining the shape context for each point, the cloud firstly finds the most matching point among all points $s_{i,x}(x = 1, 2, \dots, n)$ in image $[\mathbf{I}_i]$ for each point $s_{q,t}(t = 1, 2, \dots, n)$ in $[\hat{\mathbf{I}}_q]$. The cloud computes $[cost_{t,x}]$ denoted as the encrypted matching cost between point $s_{q,t}$ and $s_{i,x}$,

$$[cost_{t,x}] = \frac{1}{2} \sum_{k=1}^{60} \frac{([h_t(k)] - [h_x(k)])^2}{[h_t(k)] + [h_x(k)]}, \quad (15)$$

where $[h_t(k)]$ and $[h_x(k)]$ are shape contexts at points $s_{q,t}$ and $s_{i,x}$, respectively.

Given the set of cost $[cost_{t,x}]$ between point $s_{q,t}$ on the query image and all points $s_{i,x}$ on the database images, the cloud need find the minimum matching cost for $s_{q,t}$ in $[\hat{\mathbf{I}}_q]$ in encrypted domain. Thus, the cloud adopts a

modified privacy-preserving comparison operation presented in our proposed privacy-preserving image denoising protocol PPID as follows: A variant $f_{x,x'} = [1 - h_{0,n}^{x,x'}](x, x' = 1, 2 \dots, n)$ is defined as the comparing result between $[cost_{t,x}]$ and $[cost_{t,x'}]$, where $h_{0,n}^{x,x'}$ is the tag showing the whether $cost_{t,x}$ is larger than $cost_{t,x'}$. To find the minimum matching cost with point $s_{q,t}$, the cloud computes $[cost_{t,x}]^{min} = \sum_{x=1}^n (\prod_{x'=1}^n f_{x,x'}) [cost_{(t,x)}]$. Then the cloud can minimize the total encrypted minimum matching cost for each encrypted database image $[I_i]$,

$$[cost_i] = \sum_{t=1}^n [cost_{t,x}]^{min}. \tag{16}$$

The cloud obtains N such encrypted matching cost $\{[cost_i]\}(i = 1, 2, \dots, N)$ and compares them with the threshold $[T]$ by executing the same comparison algorithm mentioned in denoising part, generating the encrypted comparing results $\{[h_{0,n}^i]\}(i = 1, 2, \dots, N)$. Finally, the cloud computes $C_3 = H_1(\bigcup_{i=1}^N [h_{0,n}^i] \parallel C_{ram,\rho}^{add} \parallel C_{ram,\rho}^{mul} \parallel C_{ram,\theta}^{add} \parallel C_{ram,\theta}^{mul} \parallel C'_{ram,\rho}{}^{,add} \parallel C'_{ram,\rho}{}^{,mul} \parallel C'_{ram,\theta}{}^{,add} \parallel C'_{ram,\theta}{}^{,mul})$ and returns it with $\{[h_{0,n}^i]\}(i = 1, 2, \dots, N)$ to the user.

(3) $\{h_T^i\} \leftarrow \mathbf{ImgDec}(\{[h_{0,n}^i]\}, C_3, [\mathbf{P}], [\mathbf{q}], SK_f)$ After receiving the final encrypted comparison results $\{[h_{0,n}^i]\}(i = 1, 2, \dots, N)$, the authorized user performs algorithm **PPOIM.Dec**(\cdot) as follows. The user firstly decrypts $p \parallel h = f_{SK_f}^{-1}(C_{1,1})$ by using the secret key SK_f , and checks whether all of $C_{ram,\rho}^{add} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{2,\rho_{i,t}})$, $C_{ram,\rho}^{mul} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{3,\rho_{i,t}})$, $C_{ram,\theta}^{add} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{2,\theta_{i,t}})$, $C_{ram,\theta}^{mul} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{3,\theta_{i,t}})$, $C'_{ram,\rho}{}^{,add} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{2,\rho_t})$, $C'_{ram,\rho}{}^{,mul} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{3,\rho_t})$, $C'_{ram,\theta}{}^{,add} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{2,\theta_t})$, $C'_{ram,\theta}{}^{,mul} = H_0(p \parallel h \parallel \bigcup_{t=1}^n C_{3,\theta_t})$, $C_3 = H_1(\bigcup_{i=1}^N [h_{0,n}^i] \parallel C_{ram,\rho}^{add} \parallel C_{ram,\rho}^{mul} \parallel C_{ram,\theta}^{add} \parallel C_{ram,\theta}^{mul} \parallel C'_{ram,\rho}{}^{,add} \parallel C'_{ram,\rho}{}^{,mul} \parallel C'_{ram,\theta}{}^{,add} \parallel C'_{ram,\theta}{}^{,mul})$ hold. If not, this algorithm outputs \perp ; otherwise, the user continues to compute $q = N'(ph)^{-1}$, $N^n = pq$ and

$$\begin{aligned} C_{T,p}^i &= ([h_{0,n}^i] \bmod N^n) \bmod p = H_{T,p}^i \bmod p, \\ C_{T,q}^i &= ([h_{0,n}^i] \bmod N^n) \bmod q = H_{T,q}^i \bmod q. \end{aligned} \tag{17}$$

Then the user can decipher the matching results $h_{0,n}^i (i = 1, 2, \dots, N)$ by exploiting the Chinese Remainder Theorem (CRM) as follows,

$$h_{0,n}^i = h'_p q H_{T,p}^i + h'_q p H_{T,q}^i \bmod N^n \tag{18}$$

where h'_p, h'_q respectively satisfies $h'_p q \equiv 1 \bmod p$, $h'_q p \equiv 1 \bmod q$ which can be efficiently computed since the greatest common divisor of p and q namely $gcd(p, q) = 1$. If the final result $h_{0,n}^i = 1 (i = 1, 2, \dots, N)$, the image I_i corresponding to this result matches I_q ; Otherwise, it means that the matching cost is larger than T , and I_i mismatches \hat{I}_q .

It is noted that the algorithms **PPOIM.Enc**(\cdot) and **PPOIM.Dec**(\cdot) preserve the fully homomorphic property, by supporting the mixed operations (i.e. the addition and multiplication operations) on ciphertexts of polar coordinates of both the database images and the query image, namely $[\rho_{i,t}], [\theta_{i,t}], [\rho_t], [\theta_t] (i = 1, 2, \dots, N; t = 1, 2, \dots, n)$, that are required in our PPOIM. All the additive and multiplicative blinding factors $U_{i,t}^{add}, U_{i,t}^{mul}, U_t^{add}, U_t^{mul}$ can be cancelled out after modular N in **PPOIM.Dec**(\cdot) and the original image matching result would be successfully recovered. The correctness of our proposed PPOIM can be straightforwardly derived from the protocol descriptions presented above.

4 Security Proof

In this section, we give the formal security proof of our proposed PPOIM in the aspects of image privacy and matching result privacy.

Theorem 1: (Image Privacy) The database image privacy is unconditionally-secure (information theoretic secure) against the collusion between the cloud and malicious users, namely $H(\rho_{i,t} | [\rho_{i,t}]) = H(\rho_{i,t})$ and $H(\theta_{i,t} | [\theta_{i,t}]) = H(\theta_{i,t})$ where $H(\cdot), H(\cdot | \cdot)$ respectively refer to the entropy function and the conditional entropy function. The unconditional security of query image privacy can be achieved in the same way.

In our PPOIM, the cloud and malicious users not holding secret key SK_f cannot invert the one-way trapdoor permutation f from $C_{1,1}$ generated by **PPOIM.Enc**(\cdot) in Eq. (1) to derive p, q , which are adopted to encrypt each database image $\mathbf{p}_i = (\rho_{i,t}, \theta_{i,t})$. Moreover, the uniformly distributed randomnesses $U_{i,t}^{add}, U_{i,t}^{mul}$ are adopted to further blind \mathbf{p}_i to guarantee the unconditional security of database image privacy. The proof details are referred to the full paper.

Theorem 2: (Matching Result Privacy) Let \mathcal{A} be a malicious adversary defeating the matching result privacy of our proposed PPOIM with a non-negligible advantage defined as $\epsilon', n(\lambda)$, where $n(\lambda)$ refers to the total number of queries made to the oracles and λ is the security parameter. There exists a simulator \mathcal{B} who can use \mathcal{A} to invert the one-way trapdoor permutation with the non-negligible probability $\epsilon \geq \epsilon', n(\lambda) - \frac{n(\lambda)}{2^{\lambda-1}}$. In our proposed PPOIM, the matching result privacy is achieved since only the authorized user possessing the secret key SK_f can decrypt $p \| h = f_{SK_f}^{-1}(C_{1,1})$, compute $q = N'(ph)^{-1}$, and recover the image matching result $h_{0,n}^i$ by Eqs. (17) and (18). in **PPOIM.Dec**(\cdot). The proof details are referred to the full paper.

5 Performance Evaluation

In this section, we evaluate the performance of our proposed PPOIM in the aspects of computational overhead, communication overhead and image matching accuracy. We conduct the extensive evaluation to demonstrate the performance of our proposed PPOIM on the MPEG-7 shape silhouette database [22]

in the aspects of computational cost, communication cost on the data owner, the cloud and the user’s ends, and the image matching accuracy. All our experiments are implemented by exploiting MIRACLE library [20] on a Windows 10 with Intel Core i5-7400 CPU 3.00GHz. The performance is analyzed by an efficiency comparison between our proposed PPOIM and the privacy-preserving shape context based image matching protocol exploiting public key FHE [8], [15]. Let the security parameter be $\lambda = 512$. In our proposed PPOIM, we respectively set $|p| = |q| = |h| = 512$, and the one-way trapdoor permutation implemented by RSA on \mathbb{Z}_N^n where $|N^n| = 1024$ -bit long. Figures 3, 4 and 5 studied the computational cost under the parameters: the number of database images N , the sampled points in each image n and the threshold t' for power series expansion. Figure 3 demonstrates that the computational cost on the data owner’s end of our proposed PPOIM is dramatically lower than [8]. The reason is that [8] requires to execute public key FHE on each sampled point of all database images, namely $O(Nn)$ times in total; while in our PPOIM, the one-way trapdoor permutation, implemented by RSA and the computational cost of which is much less than public key FHE, is required to perform only once to encrypt batch of sampled points. Figure 4 demonstrates the computational cost on the cloud’s end of our PPOIM is considerably less than [8], owing to the fact that Brakerski’s public key FHE adopted in [8] requires to perform $O(N_m^2)$ multiplications for a ciphertext multiplication where N_m denotes the number of ciphertext components. Additionally, N_m would increase by one every time a ciphertext multiplication is needed for image denoising and matching in the encrypted domain. On the contrary, multiplication is required to perform only once every time a ciphertext multiplication is needed in our PPOIM. Figure 5 illustrates that the computational cost on the user’s end is significantly lower than [8], since the decryption of Brakerski’s public key FHE [15] requires the inner product composed of $O(N_m)$ multiplications; while in our PPOIM the multiplication complexity for decryption is $O(1)$.

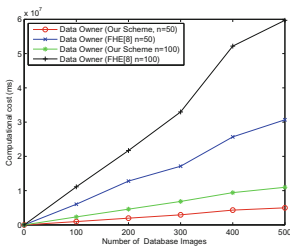


Fig. 3. Computational cost comparison on data owner’s end

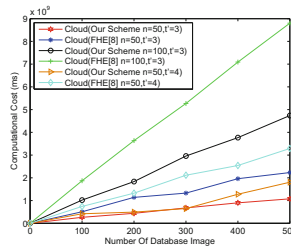


Fig. 4. Computational cost comparison on cloud’s end

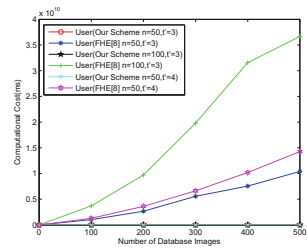


Fig. 5. Computational cost comparison on user’s end

Figures 6, 7 and 8 show that the communication cost of our PPOIM are dramatically reduced no matter at the data owner, the cloud and the user’s ends under the parameters N, n, t' , and the number of LSH functions l , owing to

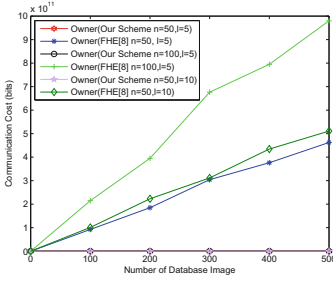


Fig. 6. Communication cost comparison on data owner's end

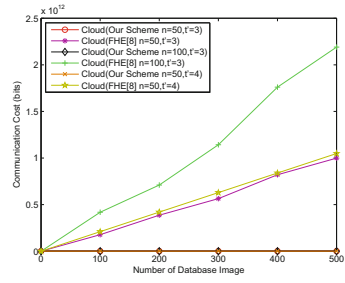


Fig. 7. Communication cost comparison on cloud's end

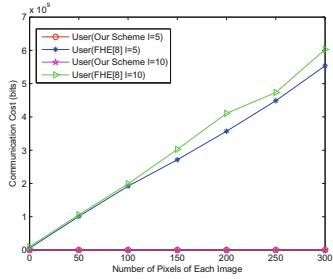


Fig. 8. Communication cost comparison on user's end

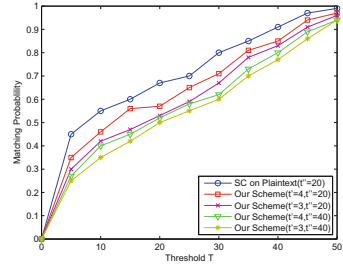


Fig. 9. Image matching accuracy comparison

the same fact that each ciphertext multiplication in [15] would incur an additional ciphertext component, leading to a high communication cost. Figure 9 demonstrates that the image matching accuracy of our PPOIM in the encrypted domain is only slightly lower than the corresponding protocol in plaintext without affecting its availability. It is observed that the matching probability increases as the threshold t' of power series expansion and the threshold of matching cost T increase, since the approximate integers adopted to evaluate the encrypted squared distance $d^2([\mathbf{p}_i], [\mathbf{q}])$ in Eq. (7) and the encrypted normalizing factor $[Z]$ in Eq. (12) would be more accurate, and more database images would match the queried one. The matching probability also increases as the threshold t'' for obtaining the candidate patches for denoising decreases, since more precise patches are found to recover the original clean image more accurately in the encrypted domain.

6 Conclusion

In this paper, a privacy-preserving shape context based image denoising and matching protocol PPOIM with efficient outsourcing is proposed. Firstly, to improve the accuracy of image matching, a privacy-preserving image denoising

scheme PPID is proposed without exploiting public key FHE. Then, based on PPID, a privacy-preserving image matching adopting shape context descriptor is devised. Formal security proof and extensive simulations demonstrate the efficiency and practicability of our proposed PPOIM.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China under Grant 61602180, 61632012, 61672239 and U1636216, and in part by Natural Science Foundation of Shanghai under Grant 16ZR1409200.

References

1. Weng, L., Amsaleg, L., Morton, A., Marchand-Maillet, S.: A privacy-preserving framework for large-scale content-based information retrieval. *IEEE Trans. Inf. Forensics Secur.* **10**(1), 152–167 (2015)
2. Ferreira, B., Rodrigues, J., Leitao, J., Domingos, H.: Practical privacy-preserving content-based retrieval in cloud image repositories. *IEEE Trans. Cloud Comput.* (2017). <https://doi.org/10.1109/TCC.2017.2669999>
3. Zhang, L., Jung, T., Feng, P., Liu, K., Li, X.-Y., Liu, Y.: PIC: enable large-scale privacy preserving content-based image search on cloud. In: *Proceedings of IEEE ICPP*, pp. 949–958, September 2015
4. Hsu, C.Y., Lu, C.S., Pei, S.C.: Image feature extraction in encrypted domain with privacy-preserving SIFT. *IEEE Trans. Image Process.* **21**(11), 4593–4607 (2012)
5. Hu, S., Wang, Q., Wang, J., Qin, Z., Ren, K.: Securing SIFT: privacy-preserving outsourcing computation of feature extractions over encrypted image data. *IEEE Trans. Image Process.* **25**(7), 3411–3425 (2016)
6. Wang, Q., Hu, S., Ren, K., Wang, J., Wang, Z., Du, M.: Catch me in the dark: effective privacy-preserving outsourcing of feature extractions over image data. In: *Proceedings of INFOCOM*, pp. 1170–1178 (2016)
7. Xia, Z., Wang, X., Zhang, L., Qin, Z.: A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Trans. Inf. Forensics Secur.* **11**(11), 2594–2608 (2017)
8. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(4), 509–522 (2002)
9. Zheng, Y., Cui, H., Wang, C., Zhou, J.: Privacy-preserving image denoising from external cloud databases. *IEEE Trans. Inf. Forensics Secur.* **12**(6), 1285–1298 (2017)
10. Chan, S.H., Zickler, T., Lu, Y.M.: Monte Carlo non-local means: random sampling for large-scale image filtering. *IEEE Trans. Image Process.* **23**(8), 3711–3725 (2014)
11. Buades, A., Coll, B., Morel, J.-M.: A non-local algorithm for image denoising. In: *Proceedings of IEEE CVPR*, pp. 60–65, June 2005
12. Zhou, J., Cao, Z., Dong, X., Lin, X.: PPDM: a privacy-preserving protocol for cloud-assisted e-healthcare systems. *IEEE J. Sel. Top. Signal Process.* **9**(7), 1332–1344 (2015)
13. Gentry, C., Halevi, S., Vaikuntanathan, V.: *i*-hop homomorphic encryption and rerandomizable Yao circuits. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 155–172. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_9

14. 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
15. 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
16. Liu, X., Choo, R., Deng, R., Lu, R., Weng, J.: Efficient and privacy-preserving outsourced calculation of rational numbers. *IEEE Trans. Dependable Secur. Comput.* **15**(1), 27–39 (2018)
17. Liu, X., Qin, B., Deng, R., Li, Y.: An efficient privacy-preserving outsourced computation over public data. *IEEE Trans. Serv. Comput.* **10**(5), 756–770 (2017)
18. Taeho, J., Mao, X., Li, X.: Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In: Proceedings of IEEE INFOCOM, pp. 2634–2642 (2013)
19. Damgard, I., Geisler, M., Kroigard, M.: Homomorphic encryption and secure comparison. *Int. J. Appl. Cryptogr.* **1**(1), 22–31 (2008)
20. Multiprecision integer and rational arithmetic C/C++ library. <http://www.shamus.ie/>
21. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of CCSW, pp. 113–124 (2011)
22. Jeannin, S., Bober, M.: Description of core experiments for MPEG-7 motion/shape. RIM 2003. Hrvatska znanstvena bibliografija i MZOS-Svibor (2003)
23. Ghinita, G., Rughinis, R.: An efficient privacy-preserving system for monitoring mobile users: making searchable encryption practical. In: Proceedings of ACM CODASPY, pp. 321–332 (2014)
24. Tang, Q., Wang, J.: Privacy-preserving context-aware recommender systems: analysis and new solutions. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 101–119. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24177-7_6
25. Veugen, T.: Encrypted integer division and secure comparison. *Int. J. Appl. Cryptogr.* **3**(2), 166–180 (2014)



Linking Differential Identifiability with Differential Privacy

Anis Bkakria^(✉), Nora Cuppens-Boulahia, and Frédéric Cuppens

IMT Atlantique, 2 Rue de la Châtaigneraie, 35576 Cesson Sévigné, France
{anis.bkakria,nora.cuppens,frederic.cuppens}@imt-atlantique.fr

Abstract. The problem of preserving privacy while mining data has been studied extensively in recent years because of its importance for enabling sharing data sets. Differential Identifiability, parameterized by the probability of individual identification ρ , was proposed to provide a solution to this problem. Our study of the proposed Differential Identifiability model shows that: First, its usability is based on a very strong requirement. That is, the prior probability of an individual being present in a database is the same for all individuals. Second, there is no formal link between the proposed model and well known privacy models such as Differential Privacy. This paper presents a new differential identifiability model for preventing the disclosure of the presence of an individual in a database while considering an adversary with arbitrary prior knowledge about each individual. We show that the general Laplace noise addition mechanism can be used to satisfy our new differential identifiability definition and that there is a direct link between differential privacy and our proposed model. The evaluation of our model shows that it provides a good privacy/utility trade-off for most aggregate queries.

1 Introduction

Many privacy models have been proposed for protecting individuals' privacy in published data, e.g., k -anonymity [14], l -diversity [13], t -closeness [10], etc. These models suffer from a key limitation: They cannot guarantee that the relationship between individuals' identities and their sensitive information are protected in case in which the adversary has additional knowledge. A privacy notion that is progressively gaining acceptance for overcoming the previously mentioned privacy problem is differential privacy (DP). Informally, DP requires that the impact of the presence of any individual entity in a dataset on the output of the queries to be limited. More specifically, DP ensures that any two databases that differ only in one record will induce output distributions that are close in the manner that the probabilities of each possible query's outputs differ by a bounded multiplicative factor ϵ .

The authors thank the Région Bretagne, CNRS, IMT, FEDER, CD 35 and Rennes Métropole for their support through the CPER Cyber SSI.

Several research have investigated whether DP can provide sufficient protection and how to choose the right value for the parameter ϵ . Lee and Clifton showed in [9] that the DP's parameter ϵ can only limit how much one individual can change the output of a query. It does not limit the amount of information that are revealed about an individual. This limitation makes DP not fully matching the legal definition of privacy that requires the protection of individually identifiable data. Attempting to meet the previous privacy definition, Lee and Clifton proposed in [9] a new privacy model called *differential identifiability* (DI). They assume that a database record can be linked to the identity of an individual, and they provide a model to quantify the leakage of the information on whether an individual participates in the database or not. Informally, if we denote by *possible worlds* the set of all possible databases resulting from removing an (any) individual from the initial database, DI ensures that the identifiability risk of any individual in the universe is less than or equal to a parameter ρ . This parameter can be interpreted as the degree of indistinguishability between possible worlds, where the possible worlds differ by (any) one individual. Unfortunately, the proposed model is based on the assumption that the prior probability of an individual being in the database is the same for all individuals. We believe that this is a very strong requirement since it requires an adversary to know exactly the same amount of information about each individual in the database. Clearly, this assumption is seldom satisfied in the real environments. Moreover, There is no direct translation from the DI parameter ρ to the DP parameter ϵ , and thus, the data utility may become unable to estimate.

In this paper, we try to remedy the previously mentioned drawbacks by proposing a new model called (α, β) -DI. The model aims to limit the leakage of information on whether an individual participates in a database or not when considering an adversary with arbitrary prior knowledge about each individual in the database (the same strong guarantees as DP). We show that the general Laplacian noise addition mechanism for differential privacy can be adapted to provide (α, β) -differentially identifiable outputs and that there is a direct translation between DP and our (α, β) -DI model. In a thorough experimental evaluation on real datasets, we studied the utility that can be provided by our model for several kinds of statistical queries.

The rest of the paper is organized as follows. Section 2 introduces the notations and preliminaries that we are going to use. Section 3 presents the problem we address and the adversary model we consider in our work. In Sect. 4, we first show how to model the belief of an adversary about the individuals present in the database, and second, how the belief of the adversary will change when he/she interacts with the database. Section 5 defines our new Differential Identifiability model. In Sect. 6, we studied whether is it possible to provide privacy and utility without making assumptions about the prior knowledge of the adversary. Then we propose a general Laplacian noise addition mechanism to satisfy (α, β) -DI. Section 7 presents a translation from the two parameters α and β we are using in our model to the DP parameter ϵ . Section 8 evaluates the utility/privacy trade-off provided by our model for different kinds of aggregate queries. We discuss related work in Sect. 9 and conclude the paper in Sect. 10.

Table 1. List of symbols

D	Database to be queried
D^s	Database containing records having the sensitive property s
M	A privacy preserving mechanism
\mathcal{U}	The universe of individuals
ι	An entity in the universe \mathcal{U}

2 Notations and Preliminaries

In our model, we used the set of notations given in Table 1. A dataset D is generated from the data associated with a subset of entities in \mathcal{U} . For all $D', D^s \in \mathcal{U}$, the prior belief that some database D' is equal to D^s is given by $\mathcal{B}_\emptyset(D' = D^s)$. The posterior belief that some database D' is equal to D^s after observing the response τ of a query q is given by $\mathcal{B}_{q,\tau}(D' = D^s)$.

Definition 1 (Adjacent Databases). *Two databases D_1 and D_2 are adjacent ($D_1 \stackrel{L}{\sim} D_2$) if they differ on the data of a single individual ι .*

For sake of simplicity, we will suppose that each individual has only one record in the database. That is, two adjacent databases differ only in one record.

Definition 2 (Global Sensitivity). *Given a query function $q : \mathcal{U} \rightarrow \mathbb{R}$. The global sensitivity of q is defined as following:*

$$\Delta_q = \max_{\forall D, D' \in \mathcal{U}} |q(D) - q(D')|$$

where D and D' are adjacent database and $q(D)$ denotes the result of the execution of the q over the database D .

3 Problem Statement and Adversary Model

We consider a database D containing a set of information about a set of individuals in \mathcal{U} , and D^s the database containing the set of individuals in D having a sensitive property s (e.g., the set of individuals having VIH). As in the DP model, we consider a very strong adversary who knows every single information in D . That is, we suppose that the adversary knows every attribute value in D . In addition, we suppose that the adversary knows that D^s is composed of individuals who have s and that he/she don't know which individuals in D are in D^s . Considering that a privacy-preserving data analysis aims to release analysis results without revealing the identities of the individuals, a privacy breach is then to allow an adversary to figure out individual's presence/absence in D^s .

In our model, we suppose that the adversary has an infinite computational power which will be used to identify the set of individuals in D^s by combining

the knowledge of D and the results of the queries to be executed over D^s . This is identical to finding out the set of missing individuals in D^s from D . In our work, we will consider the worst case in which D and D^s are adjacent databases. That is, the adversary has to find out only the missing individual in D^s to know all individuals in D^s . In the remaining of this paper, we will use D^s to represent a D 's adjacent database where all individual in D^s have a sensitive property s .

4 Adversary Knowledge Modeling

The key to a good privacy model is to correctly quantify how much information an adversary can deduce about the presence of an individual in the published data. This heavily depends on the knowledge the adversary possesses about the individuals in the database. Adversary belief changes each time a result of a query performed over D^s is observed by the adversary. We use the Bayesian inference to model an adversary belief change as defined in the following definition.

Definition 3 (Query observation impact on adversary belief). *For all two pairs of adjacent databases $D \sim D'$ and $D \sim D^s$ where $D, D', D^s \in \mathcal{U}$, given a query function $q : \mathcal{U} \rightarrow \mathbb{R}$, a mechanism M , and $\tau = M(q(D^s))$ the result of the execution of q using M . The adversary belief on $D' = D^s$ after observing $\tau = M(q(D^s))$ is defined as:*

$$\begin{aligned} \mathcal{B}_{q,\tau}(D' = D^s) &= Pr[D' = D^s | M(q(D^s)) = \tau] \\ &= \frac{Pr[M(q(D')) = \tau]}{Pr[(q, \tau)]} \times Pr[D' = D^s] \end{aligned} \tag{1}$$

where $Pr[D' = D^s]$ denotes the prior belief ($\mathcal{B}_\emptyset(D' = D^s)$) of the adversary on $D = D^s$ (before observing $\tau = M(q(D^s))$) and $Pr[(q, \tau)]$ denotes the probability of observing the result τ when the query q is performed.

5 Differential Identifiability: The New Model

Definition 4 ((α, β)-DI). *Given a query function $q : \mathcal{U} \rightarrow \mathbb{R}$, a randomized mechanism M is said to be (α, β)-differentially identifiable if for all two pairs of adjacent databases $D \sim D'$ and $D \sim D^s$ where $D, D', D^s \in \mathcal{U}$:*

$$(1 - \alpha) \times \mathcal{B}_\emptyset(D' = D^s) \leq \mathcal{B}_{q,\tau}(D' = D^s) \leq (1 + \beta) \times \mathcal{B}_\emptyset(D' = D^s) \tag{2}$$

where $0 < \alpha < 1$, $0 < \beta$, and τ denotes the result observed by the adversary for $M(q(D^s))$.

Informally, the randomized mechanism M is (α, β)-differentially identifiable means that the ratio of the adversary belief on $D' = D^s$ before and after observing $M(q(D^s)) = \tau$ is lower and upper bounded respectively by $1 - \alpha$ and $1 + \beta$. The identification risks represented by the lower bound $1 - \alpha$ and the upper

bound $1 + \beta$ are not the same. In the left side of Inequality (2), the value of α bounds the maximum attacker belief change on identifying the presence of the individual ι in the database D^s , where $D' \stackrel{\mathcal{L}}{\sim} D$. More α is bigger, more the adversary belief in $D' = D^s$ will be smaller, and more the adversary belief in $D \stackrel{\mathcal{L}}{\sim} D^s$ will be also smaller. In the right side of Inequality (2), the value of β bounds the maximum belief of an attacker on identifying all the individuals present in the database D^s . More β is bigger more the adversary belief in $D \stackrel{\mathcal{L}}{\sim} D^s$ will be bigger.

Most existing privacy frameworks bound only the adversary’s belief on the presence of one individual in the database. We believe that bounding the adversary’s belief on identifying all individuals in the database is very useful. To illustrate, let us suppose that the database D contains information about 10 individuals and that the prior adversary’s belief that each individual in $D \cap D^s$ is 10^{-1} . Now if we suppose that the data publisher wants to bound the probability of identifying the presence of an individual in D^s to $1/5$, the adversary can end up with the following belief: for 9 individual, the probability that each one of them is in D^s is equal to $(10^{-1} - 10^{-6})/9$. For the last individual, the probability that he/she is in D^s is equal to 10^{-6} . Since D and D^s are neighboring database, the adversary might know all individual in D^s with a probability of $1 - 10^{-6}$.

We studied how our definition of DI composes. Given a data consumer (adversary) who access a database multiple times via differentially identifiable mechanisms each of which having its own DI guarantees, what level of DI is still guaranteed on the union of those outputs? In order to formally define composition, we consider a similar composition scenario as the one proposed in [6]. A composition experiment considers an adversary \mathcal{A} who is trying to break privacy and figure out whether or not a particular individual is in the database by analyzing the hypotheses on the output of a sequential and adaptively chosen queries executed via differentially identifiable mechanisms. That is, we permit the adversary to have full control over which query to ask, and which differentially identifiable mechanism to be used for each query. In addition, the adversary is free to make these choices adaptively based on previous queries outcomes.

Theorem 1. *Given a set of queries functions $\mathcal{Q} = \{q_1, \dots, q_n\}$ ($\forall i \in [1, n], q_i : \mathcal{U} \rightarrow \mathbb{R}$) and a set of n mechanisms M_1, \dots, M_n . Each $M_i, i \in [1, n]$, is (α_i, β_i) -differentially identifiable. Then for all databases D, D^s , where $D \sim D^s$, the combination $\mathcal{M} = (M_1(q_1(D^s)), M_2(q_2(D^s)), \dots, M_n(q_n(D^s)))$ is (α_c, β_c) -differentially identifiable where:*

$$\alpha_c = \sum_{k=1}^n (-1)^{k+1} \sigma_k(\alpha_1, \dots, \alpha_n) \quad \text{and} \quad \beta_c = \sum_{k=0}^n (\sigma_k(\beta_1, \dots, \beta_n)) - 1$$

with σ_k denotes the elementary symmetric polynomials.

Proof. Let us suppose that $\forall i \in [1, n] : M_i(q_i(D^s)) = \tau_i$ and that $\mathcal{R} = \{\tau_i | i \in [1, n]\}$. First, Let us prove by induction that, for all two pairs of adjacent databases $D \sim D'$ and $D \sim D^s$ where $D, D', D^s \in \mathcal{U}$, the belief of the adversary

on D' equals to D^s ($\mathcal{B}_{\mathcal{Q},K}(D' = D^s)$) after the observation of the results of the set of n arbitrary and adaptively chosen queries \mathcal{Q} is bounded as following:

$$\prod_{i=1}^n (1 - \alpha_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \leq \mathcal{B}_{\mathcal{Q},\mathcal{R}}(D' = D^s) \leq \prod_{i=1}^n (1 + \beta_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \quad (3)$$

By definition (Definition 4), Inequality (3) holds for $n = 1$. That is, when using an (α_1, β_1) -differentially identifiable mechanism M_1 to perform q_1 , based on Definition 4 we get:

$$(1 - \alpha_1) \times \mathcal{B}_{\emptyset}(D' = D^s) \leq \mathcal{B}_{q_1, r_1}(D' = D^s) \leq (1 + \beta_1) \times \mathcal{B}_{\emptyset}(D' = D^s) \quad (4)$$

Suppose now that Inequality (3) holds for $n = k$. Then, by denoting $\mathcal{Q}^k = \{q_1, q_2, \dots, q_k\}$ and $\mathcal{R}^k = \{r_1, r_2, \dots, r_k\}$, the following inequality holds:

$$\prod_{i=1}^k (1 - \alpha_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \leq \mathcal{B}_{\mathcal{Q}^k, \mathcal{R}^k}(D' = D^s) \leq \prod_{i=1}^k (1 + \beta_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \quad (5)$$

Let us now prove that the Inequality (3) holds for $n = k + 1$. Since the adversary will observe the result of the query q_{k+1} after observing the results of the previous k queries q_1, \dots, q_k . The adversary belief on D' equals to D^s before observing the output of q_{k+1} is $\mathcal{B}_{\mathcal{Q}^k, \mathcal{R}^k}(D' = D^s)$. By considering the fact that q_{k+1} is performed using the $(\alpha_{k+1}, \beta_{k+1})$ -differentially identifiable mechanism M_{k+1} , based on Definition 4, we get:

$$(1 - \alpha_{k+1}) \times \mathcal{B}_{\mathcal{Q}^k, \mathcal{R}^k}(D' = D^s) \leq \mathcal{B}_{\mathcal{Q}^{k+1}, \mathcal{R}^{k+1}}(D' = D^s) \leq (1 + \beta_{k+1}) \times \mathcal{B}_{\mathcal{Q}^k, \mathcal{R}^k}(D' = D^s) \quad (6)$$

Since, $0 < \alpha < 1$ and that we supposed that Inequality (5) holds, we can use its left side to show that:

$$\prod_{i=1}^{k+1} (1 - \alpha_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \leq (1 - \alpha_{k+1}) \times \mathcal{B}_{\mathcal{Q}^k, \mathcal{R}^k}(D' = D^s) \quad (7)$$

Then using the fact that $\beta > 0$ together with the right side of Inequality (5), we get:

$$(1 + \beta_{k+1}) \times \mathcal{B}_{\mathcal{Q}^k, \mathcal{R}^k}(D' = D^s) \leq \prod_{i=1}^{k+1} (1 + \beta_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \quad (8)$$

Then based on Inequalities (6), (7), and (8) we get:

$$\prod_{i=1}^{k+1} (1 - \alpha_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \leq \mathcal{B}_{\mathcal{Q}^{k+1}, \mathcal{R}^{k+1}}(D' = D^s) \leq \prod_{i=1}^{k+1} (1 + \beta_i) \times \mathcal{B}_{\emptyset}(D' = D^s) \quad (9)$$

which prove that Inequality (3) holds for $n = k + 1$, and by induction it holds for all $n \in \mathbb{N}^*$. Now, based on the fundamental theorem of symmetric polynomials we have:

$$\begin{aligned} \prod_{i=1}^n (1 - \alpha_i) &= 1 + \sum_{k=1}^n (-1)^k \sigma_k(\alpha_1, \dots, \alpha_n) \\ &= 1 - \underbrace{\sum_{k=1}^n (-1)^{k+1} \sigma_k(\alpha_1, \dots, \alpha_n)}_{\alpha_c} \end{aligned}$$

and

$$\begin{aligned} \prod_{i=1}^n (1 + \beta_i) &= \sum_{k=0}^n \sigma_k(\beta_1, \dots, \beta_n) \\ &= 1 + \underbrace{\sum_{k=1}^n (\sigma_k(\beta_1, \dots, \beta_n))}_{\beta_c} - 1 \end{aligned}$$

6 Satisfying Differential Identifiability

Given the above, in this section, we show how to achieve (α, β) -DI. For this, we first define the *identifiability sensitivity* of a query as following.

Definition 5 (Query Identifiability Sensitivity). *For a given query function $q : \mathcal{U} \rightarrow \mathbb{R}$, the query identifiability sensitivity of q is*

$$\Theta_q = \max_{D, D_1, D_2 \in \mathcal{U}} |q(D_1) - q(D_2)|$$

where D_1 and D_2 are adjacent to D .

Note that the Identifiability Sensitivity of a query is different than its Global Sensitivity (Definition 2) used in DP. The Identifiability Sensitivity of a query q represents, for all two pairs of adjacent databases $(D \sim D_1)$ and $(D \sim D_2)$ in \mathcal{U} , the maximum difference between the outputs that q return when executed over D_1 and D_2 .

Motivated by the difficulty for a data publisher to know the prior knowledge of an adversary about each individual in the database, we firstly investigate the achievement of the (α, β) -DI model without taking into consideration the prior knowledge of the adversary. The following theorem defines a prior-free Laplace distribution-based mechanism that achieves (α, β) -DI.

Theorem 2 (Prior-free mechanism). *Let $Lap(\lambda)$ be the Laplace distribution having a density function $h(x) = \frac{1}{2\lambda} \exp(-\frac{|x-\mu|}{\lambda})$ where $\lambda(> 0)$ is a scale factor and μ is a mean. For a given query function q , a randomized mechanism M_L*

that returns $q(X) + Y$ as an answer where Y is drawn i.i.d from $Lap(\lambda)$ satisfies (α, β) -DI for any λ such that:

$$\lambda \geq \max \left(\frac{\Theta_q}{\log(1 + \beta)}, \frac{-\Theta_q}{\log(1 - \alpha)} \right)$$

Proof. Since $M_L = q(X) + Y$ where Y is drawn i.i.d from $Lap(\lambda)$, then, for all two pairs of adjacent databases $(D \sim D_1)$ and $(D \sim D_2)$ in \mathcal{U} , we have:

$$\begin{aligned} \frac{Pr[M_L(q(D'_1)) = \tau]}{Pr[M_L(q(D'_2)) = \tau]} &= \frac{\exp(-\frac{|\tau - q(D'_1)|}{\lambda})}{\exp(-\frac{|\tau - q(D'_2)|}{\lambda})} \\ &= \exp\left(\frac{|r - q(D'_2)| - |r - q(D'_1)|}{\lambda}\right) \end{aligned}$$

we deduce then the following inequality:

$$\exp\left(-\frac{|q(D'_1) - q(D'_2)|}{\lambda}\right) \leq \frac{Pr[M_L(q(D'_1)) = \tau]}{Pr[M_L(q(D'_2)) = \tau]} \leq \exp\left(\frac{|q(D'_1) - q(D'_2)|}{\lambda}\right) \tag{10}$$

Then using Definition 5, we get:

$$\exp\left(-\frac{\Theta_q}{\lambda}\right) \leq \frac{Pr[M_L(q(D'_1)) = \tau]}{Pr[M_L(q(D'_2)) = \tau]} \leq \exp\left(\frac{\Theta_q}{\lambda}\right) \tag{11}$$

In other hand, using Definition 3, and for all two pairs of adjacent databases $(D \sim D'_i)$ and $(D \sim D^s)$ in \mathcal{U} , we have

$$\begin{aligned} \mathcal{B}_{q,\tau}(D'_i = D^s) &= \frac{Pr[M_L(q(D'_i)) = \tau] \times Pr[D'_i = D^s]}{\sum_{D'_j \in \mathcal{D}'} Pr[D'_j = D^s] \times Pr[M_L(q(D'_j)) = \tau]} \\ &= \frac{Pr[D'_i = D^s]}{Pr[D'_i = D^s] + \sum_{D'_j \in \mathcal{D}', D'_j \neq D'_i} Pr[D'_j = D^s] \times \frac{Pr[M_L(q(D'_j)) = \tau]}{Pr[M_L(q(D'_i)) = \tau]}} \end{aligned}$$

Then using Inequality 11 we deduce

$$\frac{Pr[D'_i = D^s]}{Pr[D'_i = D^s] + \exp\left(\frac{\Theta_q}{\lambda}\right) \sum_{D'_j \in \mathcal{D}', D'_j \neq D'_i} Pr[D'_j = D^s]} \leq \mathcal{B}_{q,\tau}(D'_i = D^s) \tag{12}$$

And

$$\mathcal{B}_{q,\tau}(D'_i = D^s) \leq \frac{Pr[D'_i = D^s]}{Pr[D'_i = D^s] + \exp\left(-\frac{\Theta_q}{\lambda}\right) \sum_{D'_j \in \mathcal{D}', D'_j \neq D'_i} Pr[D'_j = D^s]} \tag{13}$$

Now, based on the fact that $\sum_{D'_j \in \mathcal{D}', D'_j \neq D'_i} Pr[D'_j = D^s] = 1 - Pr[D'_i = D^s]$,

Inequality (12) can be transformed as

$$\frac{Pr[D'_i = D^s]}{Pr[D'_i = D^s] \left(1 - \exp\left(\frac{\Theta_q}{\lambda}\right) + \frac{\exp\left(\frac{\Theta_q}{\lambda}\right)}{Pr[D'_i = D^s]} \right)} \leq \mathcal{B}_{q,\tau}(D'_i = D^s)$$

$$\frac{1}{1 - \exp\left(\frac{\Theta_q}{\lambda}\right) + \frac{\exp\left(\frac{\Theta_q}{\lambda}\right)}{Pr[D'_i = D^s]}} \leq$$
(14)

Since $1 - \exp\left(\frac{\Theta_q}{\lambda}\right) \leq 0$ and by considering $Pr[D'_i = D^s] = \mathcal{B}_\emptyset(D'_i = D^s)$ (Definition 3), we obtain

$$\exp\left(-\frac{\Theta_q}{\lambda}\right) \leq \frac{\mathcal{B}_{q,\tau}(D'_i = D^s)}{\mathcal{B}_\emptyset(D'_i = D^s)}$$
(15)

We apply the same transformations to Inequality (13) to get

$$\frac{\mathcal{B}_{q,\tau}(D'_i = D^s)}{\mathcal{B}_\emptyset(D'_i = D^s)} \leq \exp\left(\frac{\Theta_q}{\lambda}\right)$$
(16)

Using Inequalities (15) and (16) together with Definition 4, $M_L = q(X) + Y$ where Y is drawn i.i.d from $Lap(\lambda)$ satisfies (α, β) -DI if:

$$1 - \alpha \leq \exp\left(-\frac{\Theta_q}{\lambda}\right) \quad \text{and} \quad \exp\left(\frac{\Theta_q}{\lambda}\right) \leq 1 + \beta$$

Rearranging yields

$$\lambda \geq \frac{\Theta_q}{\log(1 + \beta)} \quad \text{and} \quad \lambda \geq \frac{-\Theta_q}{\log(1 - \alpha)}$$

Finally, we obtain the following

$$\lambda \geq \max\left(\frac{\Theta_q}{\log(1 + \beta)}, \frac{-\Theta_q}{\log(1 - \alpha)}\right)$$

The previous theorem uses Laplace distribution to satisfy (α, β) -DI without taking into consideration the prior knowledge of the adversary about the presence of each individual in the database D^s . The proposed construction seems to be useful to satisfy (α, β) -DI in case in which the prior knowledge of the adversary could not be known in advance. Unfortunately, in practice, it is not possible to properly instantiate our previous construction, i.e., to find the right values of α and β that make the model useful for an adversary having arbitrary prior belief. That is, in one hand, the values of α and β should be non-negligible so that the model provides an acceptable utility level for the queries that will be performed

by the adversary over the database. In the other hand, the value of α and β should not be bigger enough to allow the adversary to be sure about the presence of any individual in the database. Let us suppose that the adversary is not fully sure that an individual ι is in the database D^s (i.e., $Pr[\iota \in D^s] < 1$). If we consider only the left-hand side of Inequality (2), for any value of $\alpha \in]0, 1[$, our model will still ensure that the adversary cannot be 100% sure that ι is in D^s . Nevertheless, the previous construction may allow the adversary to be pretty much sure that ι is in the database D^s (i.e., for $D' \sim D : \mathcal{B}_{q,\tau}(D' = D^s)$ is very close to zero). Things are much more difficult for choosing the right value of β . According to the definition of our model (Definition 4), to prevent the adversary from knowing with certainty all individuals in the database D^s , the data publisher should choose a β value such that: $\mathcal{B}_\emptyset(D' = D^s) \times (1 + \beta) < 1$. Unfortunately, satisfying the previous condition becomes not possible if the adversary prior on $D' = D^s$ is not taken into consideration.

Seeking to overcome the previous limitation, we define a prior-dependent Laplace distribution based mechanism for achieving (α, β) -DI. The following theorem gives a lower bound for the quantity of Laplace noise to be added to the response of a query q to achieve (α, β) -DI for a given adversary's prior distribution \mathbb{P} .

Theorem 3 (Prior-dependent mechanism). *For all database $D \in \mathcal{U}$ of size $n(> 1)$, let \mathcal{D}' be the set of D 's adjacent databases. For a given prior distribution \mathbb{P} , For a given query function $q : \mathcal{U} \rightarrow \mathbb{R}$, a randomized mechanism M_L that returns $q(X) + Y$ as an answer where Y is drawn i.i.d from $Lap(\lambda)$ satisfies (α, β) -DI for any λ such that:*

$$\lambda \geq \Theta_q \times \max \left(\log \left(\frac{1 + P_{min}(\alpha - 1)}{(1 - \alpha)(1 - P_{min})} \right)^{-1}, \log \left(\frac{(1 + \beta)(1 - P_{min})}{1 - P_{min}(1 + \beta)} \right)^{-1} \right)$$

where $0 < \alpha < 1, 0 < \beta < (1/P_{max}) - 1$, $P_{min} = \min_{D_j, D^s \in \mathcal{D}'} Pr[D_j = D^s]$, and $P_{max} = \max_{D_j, D^s \in \mathcal{D}'} Pr[D_j = D^s]$.

We note that in the previous theorem, condition $\beta < (1/P_{max}) - 1$ is used to be sure that for any possible values of α and β , the usage of M_L will effectively prevent the adversary from knowing with certainty the content of the database $D^s \in \mathcal{D}'$. Obviously, P_{max} 's value should be lesser than 1. Otherwise, there are no possible values for α and β that can prevent the adversary from knowing with certainty the content of the database D^s , since he/she already does.

Proof. To prove the previous theorem, we start by following the same steps as in the proof of Theorem 2 to get Inequalities (12) and (13). By considering the fact that $\sum_{D'_j \in \mathcal{D}', D'_j \neq D'_i} Pr[D'_j = D^s] = 1 - Pr[D'_i = D^s]$, we transform Inequality

(13) to get Inequality (14) which will be transformed as following:

$$\frac{1}{Pr[D'_i = D^s] + \exp\left(\frac{\Theta_\alpha}{\lambda}\right)(1 - Pr[D'_i = D^s])} \leq \frac{\mathcal{B}_{q,\tau}(D'_i = D^s)}{\mathcal{B}_\emptyset(D'_i = D^s)} \quad (17)$$

Since $P_{min} \leq Pr[D'_i = D^s] \leq P_{max}$, we have:

$$\frac{1}{P_{min} \left(1 - \exp\left(\frac{\Theta_q}{\lambda}\right)\right) + \exp\left(\frac{\Theta_q}{\lambda}\right)} \leq \frac{1}{Pr[D'_i = D^s] + \exp\left(\frac{\Theta_q}{\lambda}\right) (1 - Pr[D'_i = D^s])} \tag{18}$$

Using Inequalities (17) and (18) together with Definition 4, M_L satisfies (α, β) -DI if:

$$1 - \alpha \leq \frac{1}{P_{min} \left(1 - \exp\left(\frac{\Theta_q}{\lambda}\right)\right) + \exp\left(\frac{\Theta_q}{\lambda}\right)}$$

Since $P_{min} \leq 1/n$, we have: $1 + P_{min}(\alpha - 1) > 0$. Then, rearranging yields

$$\lambda \geq \Theta_q \log \left(\frac{1 + P_{min}(\alpha - 1)}{(1 - \alpha)(1 - P_{min})} \right)^{-1} \tag{19}$$

On the other hand, by considering the fact that $\sum_{D'_j \in \mathcal{D}', D'_j \neq D'_i} Pr[D'_j = D^s] = 1 - Pr[D'_i = D^s]$, we transform Inequality (13) to get

$$\frac{\mathcal{B}_{q,\tau}(D'_i = D^s)}{\mathcal{B}_0(D'_i = D^s)} \leq \frac{1}{Pr[D'_i = D^s] + \exp\left(\frac{-\Theta_q}{\lambda}\right) (1 - Pr[D'_i = D^s])} \tag{20}$$

Then, considering the fact that $P_{min} \leq Pr[D'_i = D^s] \leq P_{min}$, we have:

$$\frac{1}{Pr[d_i = d^s] + \exp\left(\frac{-\Theta_q}{\lambda}\right) (1 - Pr[d_i = d^s])} \leq \frac{1}{P_{min} \left(1 - \exp\left(\frac{-\Theta_q}{\lambda}\right)\right) + \exp\left(\frac{-\Theta_q}{\lambda}\right)} \tag{21}$$

Using Inequalities (20) and (21) together with Definition 4, M_L satisfies (α, β) -DI if:

$$\frac{1}{P_{min} \left(1 - \exp\left(\frac{-\Theta_q}{\lambda}\right)\right) + \exp\left(\frac{-\Theta_q}{\lambda}\right)} \leq 1 + \beta \tag{22}$$

Since $P_{min} \leq 1/n$, for all $n > 1$, we have: $1 - P_{min}(1 + \beta) > 0$. Then, rearranging yields

$$\lambda \geq \Theta_q \log \left(\frac{(1 + \beta)(1 - P_{min})}{1 - P_{min}(1 + \beta)} \right)^{-1} \tag{23}$$

Finally, based on Inequalities (19) and (23), we have:

$$\lambda \geq \Theta_q \times \max \left(\log \left(\frac{1 + P_{min}(\alpha - 1)}{(1 - \alpha)(1 - P_{min})} \right)^{-1}, \log \left(\frac{(1 + \beta)(1 - P_{min})}{1 - P_{min}(1 + \beta)} \right)^{-1} \right)$$

In contrast to the original Differential Identifiability model propose in [9] which assumes that the prior probability of an individual being in D^s is the same for all individuals, our previous construction defines a Laplace distribution-based mechanism that provides an (α, β) -differentially identifiable outputs for any arbitrary prior distribution.

7 Linking Differential Identifiability and Differential Privacy

In this section, we establish a fundamental connection between DP model and our DI model by showing that the parameter ϵ used in the DP model can be directly translated to the parameters α and β used in our DI model.

Theorem 4. *Let M_L be a mechanisms that satisfies (α, β) -DI for a given query $q : \mathcal{U} \rightarrow \mathbb{R}$ by returning $q(X) + Y$ where Y is drawn i.i.d from $Lap(\lambda)$. M_L satisfies ϵ -DP where*

$$\epsilon = \frac{\Delta_q}{\Theta_q} \times \max \left(\log \left(\frac{1 + P_{min}(\alpha - 1)}{(1 - \alpha)(1 - P_{min})} \right)^{-1}, \log \left(\frac{(1 + \beta)(1 - P_{min})}{1 - P_{min}(1 + \beta)} \right)^{-1} \right) \quad (24)$$

Proof. Since M_L satisfies (α, β) -DI, then using Theorem 3 we have:

$$\lambda = \Theta_q \times \max \left(\log \left(\frac{1 + P_{min}(\alpha - 1)}{(1 - \alpha)(1 - P_{min})} \right)^{-1}, \log \left(\frac{(1 + \beta)(1 - P_{min})}{1 - P_{min}(1 + \beta)} \right)^{-1} \right) \quad (25)$$

In other hand, based on Differential Privacy's Laplace mechanism definition [5], we know that M_L satisfies ϵ -DP when

$$\lambda = \frac{\Delta_q}{\epsilon} \quad (26)$$

Finally using Eqs. (25) and (26), we get (24).

Choosing the appropriate value of ϵ is continuing to be an open problem in DP. The connection we created between ϵ -DP and (α, β) -DI models in Theorem 4 will allow to choose the appropriate ϵ value given the risk of identifying the presence of an individual in the database specified by α and β .

8 Evaluation

We now evaluate the applicability of our model. For this, we use the Adult Database from the UCI Machine Learning Repository [1] as \mathcal{U} (the universe of individuals). The database contains information about 32562 individuals collected from the 1994 U.S. Census. The information about each individual is provided through 9 categorical and 5 numerical attributes. In this evaluation, we consider only numerical attributes. In order to evaluate the applicability of our model, we quantify, for several aggregate queries (e.g., sum, average, max, min, etc.), the error ratio caused by the usage our prior-dependent mechanism (Theorem 3) when the values of α and β are varied. Since, it is not possible to graphically illustrate the variation of the error ratio in function of more than two

variables (i.e., α , β , and \mathbb{P}), for the prior distribution \mathbb{P} , we will consider two main cases. First, we will consider a very weak adversary \mathcal{A}_w . That is, he/she does not have any information about the individuals in the database $D^s \in \mathcal{U}$ (uniform prior distribution: $P_{min} = P_{max} = 1/32562$). Second, we will consider a strong adversary \mathcal{A}_s which have significant prior information about the presence of a subset of individuals in D^s . More precisely, we will suppose that \mathcal{A}_s knows with certainty that some individuals are in D^s . This means that $P_{min} = 0$ since for certain database $D' \in \mathcal{U}$ we have $\mathcal{B}_\emptyset^{\mathcal{A}_s}[D' = D^s] = 0$. Moreover, we suppose that, before interacting with the database, \mathcal{A}_s 's best confidence on the set of identities present in the database cannot be larger than $1/10$. Formally, this means that $P_{max} = 1/10$ since there exists $D' \in \mathcal{U}$ such that $\mathcal{B}_\emptyset^{\mathcal{A}_s}[D' = D^s] = 1/10$.

Table 2. Used aggregate queries and their identifiability sensitivity

Attribute	Query	Identifiability sensitivity (Θ)
Age	Average	27×10^{-4}
Capital-gain	Min	114
Capital-loss	Max	445
Hours-per-week	Sum	99

Table 2 shows the set of aggregate queries that we used in the evaluation of our model. For each query, we give the attribute over which it is executed and its corresponding identifiability sensitivity value. We note that counting queries are not considered in this evaluation since by definition, they have an identifiability sensitivity equal to zero which means that revealing the exact result of a counting query performed over the database D^s will not disclose any information to the adversary about the content of the database D^s .

Figure 1 illustrates the error ratio included in the differentially identifiable result of each query when the parameters α and β are varied and when the weak adversary \mathcal{A}_w is considered. The different plots show, first, that the smaller the values of α and β (i.e., the higher the desired privacy), more noise are included in the response. Second, according to Figs. 1(a) and (d), our model provides a very good compromise between privacy and query response precision. For example, for the *Sum(hours_per_week)* query (Fig. 1(d)), our (α, β) -DI construction reduces the error rate to 9×10^{-3} for $\alpha = \beta = 8 \times 10^{-3}$. Third, although the high identifiability sensitivity of the queries *Min(capital_gain)* (Fig. 1(b)) and *Max(capital_loss)* (Fig. 1(c)), our (α, β) -DI construction provides an acceptable compromise between privacy and query response precision. As an example, our model provides an $(0.5, 1)$ -differentially identifiable answer for the query *Max(capital_loss)* (Fig. 1(c)) with an error rate of 0.1.

When considering the strong adversary \mathcal{A}_s (Fig. 2), our construction still provides a very close privacy/utility trade-off compared to the one provided when the weak adversary \mathcal{A}_w is considered, except for the query

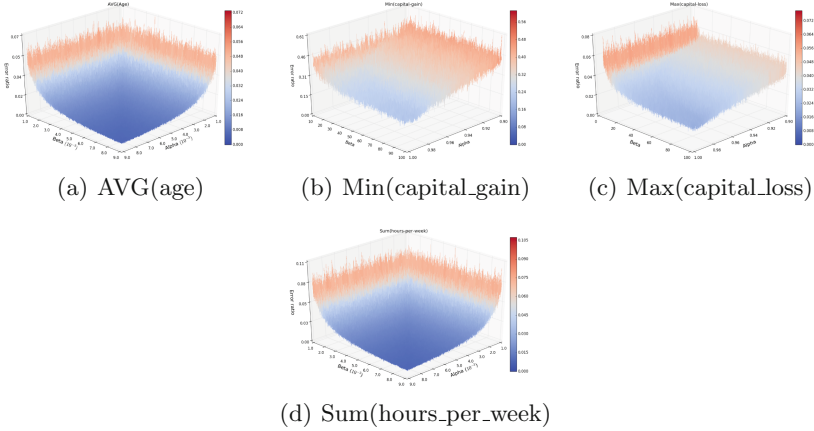


Fig. 1. Noise ratio for the adversary \mathcal{A}_w

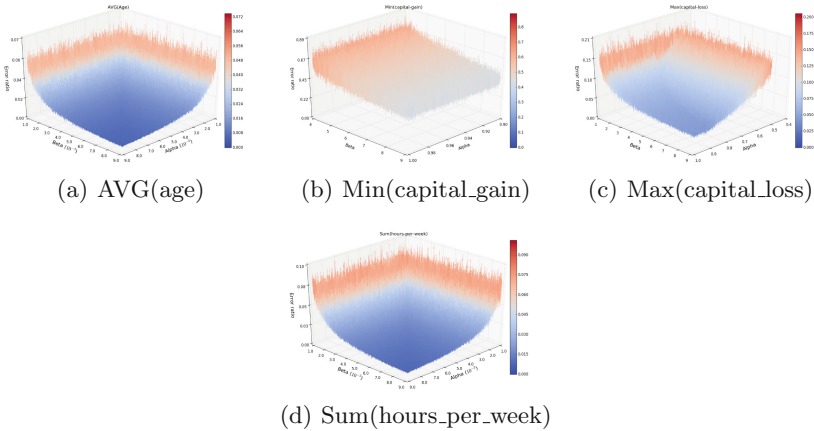


Fig. 2. Noise ratio for the adversary \mathcal{A}_s

$Min(capital\ gain)$ (Fig. 2(b)). For this query, our construction provides answers with little bit more noise compared to answers provided when the weak adversary \mathcal{A}_w is considered (Fig. 1(b)). This mainly caused by the reduced range of β 's values (i.e., α should be less than 9 according to Theorem 3) that can be used without letting \mathcal{A}_s be 100% sure about the content of the database D^s .

9 Related Work

Several privacy definitions have been proposed in last two decades. The most sticking out ones are k -anonymity [14], l -diversity [13], and t -closeness [10]. Dwork pointed out their weaknesses in [4] and argues that privacy problems

should be considered in a more formal way. Following this reflexion, the notion of DP was proposed in [3] and several approaches for satisfying it were developed to support low sensitive queries such as counting, mean, and median queries. Several relaxations of the original DP model have been proposed in order to make DP more efficient for high sensitive queries. (ϵ, δ) -DP was proposed in [5] by introducing new parameter δ that will be used to upper bound the probability that ϵ -DP is not satisfied. Generic DP is a generalization of the DP model proposed in [7]. It allows more flexible definitions for neighboring databases and conditions that the model should satisfy.

In [8], authors showed that for DP, it is not possible to ensure an acceptable privacy/utility compromise without making assumptions about the manner with which the data are generated. In this paper, we provide similar result by showing no possible acceptable privacy/utility compromise can be provided for our DI model without making assumptions about the adversary prior knowledge.

Cormode showed in [2] that DP is not useful for preventing inferential disclosure by demonstrating that one can use differentially private outputs to infer sensitive information with non-trivial accuracy. Lee and Clifton [9] argued that the parameter ϵ used in DP limits only how much one individual can affect the resulting model. It cannot be used to limit how much information is revealed about an individual. They then propose ρ -DI which captures membership disclosure under very specific adversarial background knowledge that we believe seldom satisfied in the real environments. Machanavajjhala et al. [12] proposed a model called ϵ -privacy aiming to limit the impact that one entity can have on the belief of the adversary. Unfortunately ϵ -privacy does not support interactive and adaptive data querying. Membership Privacy [11] proposed a model that uses Bayesian inference to bound the probability of identifying an individual in the database. However, the proposed model fails to bound the probability that an adversary figure out the set of individuals in a database.

10 Conclusion

This paper presents the new differential identifiability model allowing to bound the quantity of disclosed information about the presence of an individual in a database while considering an adversary with arbitrary prior knowledge. We showed that our proposed model can be satisfied using the general Laplace noise addition mechanism used traditionally in differential privacy. We proved that there is a direct connection between our (α, β) -differential identifiability and ϵ -differential privacy, and we showed through a set of experimentations that our model provides a good privacy/utility trade-off for most aggregate queries.

References

1. UCI machine learning repository. <https://archive.ics.uci.edu/ml/index.php>. Accessed 10 Apr 2018
2. Cormode, G.: Personal privacy vs population privacy: Learning to attack anonymization. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2011, pp. 1253–1261. ACM, New York (2011). <https://doi.org/10.1145/2020408.2020598>
3. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_1
4. Dwork, C.: An ad omnia approach to defining and achieving private data analysis. In: Bonchi, F., Ferrari, E., Malin, B., Saygin, Y. (eds.) PInKDD 2007. LNCS, vol. 4890, pp. 1–13. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78478-4_1
5. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
6. Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy. In: 2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 51–60. IEEE (2010)
7. Kifer, D., Lin, B.R.: Towards an axiomatization of statistical privacy and utility. In: Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 147–158. ACM (2010)
8. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, pp. 193–204. ACM, New York (2011). <https://doi.org/10.1145/1989323.1989345>
9. Lee, J., Clifton, C.: Differential identifiability. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012, pp. 1041–1049. ACM, New York (2012). <https://doi.org/10.1145/2339530.2339695>
10. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: privacy beyond k-anonymity and l-diversity. In: IEEE 23rd International Conference on Data Engineering, ICDE 2007, pp. 106–115. IEEE (2007)
11. Li, N., Qardaji, W., Su, D., Wu, Y., Yang, W.: Membership privacy: a unifying framework for privacy definitions. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & #38; Communications Security, CCS 2013, pp. 889–900. ACM, New York (2013). <https://doi.org/10.1145/2508859.2516686>
12. Machanavajjhala, A., Gehrke, J., Götze, M.: Data publishing against realistic adversaries. Proc. VLDB Endow. **2**(1), 790–801 (2009). <https://doi.org/10.14778/1687627.1687717>
13. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkatasubramanian, M.: l-diversity: privacy beyond k-anonymity. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, pp. 24–24. IEEE (2006)
14. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10**(05), 557–570 (2002)



PP-MCSA: Privacy Preserving Multi-channel Double Spectrum Auction

Zhili Chen¹✉, Sheng Chen¹, Hong Zhong¹, Lin Chen², and Miaomiao Tian¹

¹ School of Computer Science and Technology, Anhui University, Hefei 230601, China
{zlchen,mtian}@ahu.edu.cn, shengchen9403@gmail.com

² Lab. Recherche Informatique (LRI-CNRS UMR 8623), Univ. Paris-Sud,
91405 Orsay, France
zhongh@mail.ustc.edu.cn, chen@lri.fr

Abstract. Auction is widely regarded as an effective way in dynamic spectrum redistribution. Recently, considerable research efforts have been devoted to designing privacy-preserving spectrum auctions in a variety of auction settings. However, none of existing work has addressed the privacy issue in the most generic scenario, double spectrum auctions where each seller sells multiple channels and each buyer buys multiple channels. To fill this gap, in this paper we propose PP-MCSA, a Privacy Preserving mechanism for Multi-Channel double Spectrum Auctions. Technically, by leveraging garbled circuits, we manage to protect the privacy of both sellers' requests and buyers' bids in multi-channel double spectrum auctions. As far as we know, PP-MCSA is the first privacy-preserving solution for multi-channel double spectrum auctions. We further theoretically demonstrate the privacy guarantee of PP-MCSA, and extensively evaluate its performance via experiments. Experimental results show that PP-MCSA incurs only moderate communication and computation overhead.

1 Introduction

Today, more and more emerging wireless technologies, such as Wifi, 4G, are penetrating into our daily work and life. At the same time, the traditional static and rigid spectrum allocation scheme renders the utilization of radio spectrum severely inefficient and unbalanced. According to the survey [1], many statically allocated spectrum channels are left idle by their current owners, exaggerating the gap between the ever-increasing spectrum demand of wireless services and the spectrum scarcity. Therefore, to improve and balance spectrum utilization, dynamic spectrum redistribution has been advocated to reallocate spectrum among primary and secondary users.

Spectrum auction is widely regarded as an effective way in dynamic spectrum redistribution. A large body of existing studies are focused on designing truthful spectrum auctions, where the auctioneer is assumed to be trusted, and bidders are stimulated to reveal their true valuations of spectrum channels. However, in many practical scenarios, the auctioneer is by nature self-interested and not

trusted. It may disclose the true valuations of bidders, which may cause serious privacy vulnerabilities [2]. For example, a dishonest auctioneer may take advantage of learning the bidders' bids, and then tamper with the auction results so as to increase its own profit. Or the auctioneer may sell bidders' historical bids for profit. Therefore, privacy preservation is critical in spectrum auctions.

There has been significant research attention on privacy preserving auctions, such as [3–5]. These schemes do not consider spectrum reusability, and thus cannot be applied in spectrum auctions. Recently, a handful of propositions addressed privacy issues in spectrum auctions, such as [2, 6, 7], but most of them focus on protecting privacy for single-sided spectrum auctions. Only a few solutions such as [8] and [9], provide secure designs for double spectrum auctions. However, they assume that in the auction each seller sells only one spectrum channel and each buyer buys only one spectrum channel. Such *one-channel* assumption makes the problem much more tractable, but leaves open the most generic and practical version, the double spectrum auctions, involving multiple spectrum sellers selling multiple channels to multiple buyers [10].

To fill this gap, in this paper, we propose PP-MCSA, a Privacy-Preserving mechanism for Multi-Channel double Spectrum Auctions. Specifically, we manage to protect both sellers' request privacy and buyers' bid privacy for the double spectrum auction mechanism True-MCSA [10] that supports multi-channel auctions. To preserve privacy, we introduce in the auction framework of PP-MCSA a third party, namely an agent, who cooperates with the auctioneer to perform secure auction computations, as shown in Fig. 1. In such a framework, each seller m submits its request value s_m (i.e., the lowest per-channel selling price) and request number c_m (i.e., the number of selling channels) to the auctioneer. Similarly, each buyer n does the same thing with its bid value b_n (i.e., the highest per-channel buying price) and bid number (i.e., the number of buying channels). All submissions are appropriately encrypted such that all sensitive information (i.e. request values, bid values and bid numbers) are protected from either the auctioneer or the agent, but can be securely retrieved and computed with the cooperation between the two parties. Therefore, as long as the auctioneer and the agent do not collude with each other (Note that this assumption is essentially necessary, otherwise the privacy cannot be achieved.), PP-MCSA leaks nothing about the sensitive information to anyone except what can be revealed from the published auction outcome.

We list our main contributions as follows:

- We propose the first privacy-preserving and practical multi-channel double spectrum auction mechanism by combing public-key encryptions and garbled circuits, filling the research gap that there is no privacy consideration in multi-channel double spectrum auctions before.
- We design and optimize data-oblivious algorithms for multi-channel double spectrum auction mechanism True-MCSA, which is rather complex in auction logic, and address both the privacy and efficiency challenges.
- We fully implement PP-MCSA, and conduct extensive experiments to evaluate its computation and communication overhead.

The reminder of this paper is structured as follows. Section 2 briefly reviews related work. In Sect. 3, the underlying mechanism is introduced and the privacy goal is given. We describe the design challenges and rationale in Sect. 4, and present the detailed design of PP-MCSA and prove its privacy in Sect. 5. In Sect. 6, we implement PP-MCSA, and evaluate its performance in terms of computation and communication overheads. Finally, the paper is concluded in Sect. 7.

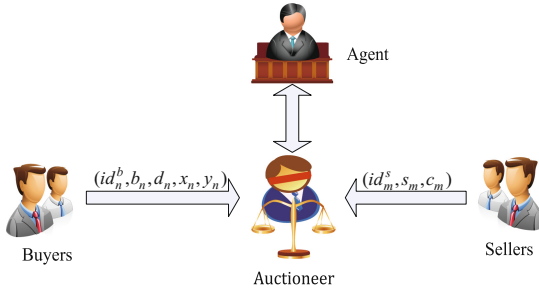


Fig. 1. Privacy-preserving auction framework for PP-MCSA

2 Related Work

In this section, we briefly review the existing works on privacy-preserving auction design, and distinguish our work from the existing ones.

2.1 Spectrum Auction

Spectrum auctions are widely used to redistribute spectrum. In the past few years, many researches have focused on designing truthful spectrum auctions. For example, Zhou et al. put forward TRUST [11], the first truthful double spectrum auction framework exploiting spectrum reusability. Chen et al. proposed the first truthful single-sided auction mechanism TAMES [12] for heterogeneous spectrum auctions, which allows buyers to freely bid their different preferences to heterogeneous spectrum channels. Later, Feng et al. presented the first double auction mechanism for heterogeneous spectrum transaction [13]. Chen et al. proposed the first double multi-channel spectrum auction scheme, True-MCSA [10]. However, all the above studies did not address the privacy preservation issues.

2.2 Privacy-Preserving Spectrum Auction

In the past decade, there have been a great number of schemes for privacy-preserving auctions [3–5]. These schemes were originally designed for traditional goods (e.g., painting, stamps), where each commodity can only be allocated to one bidder. Unfortunately, when directly applied to spectrum

auctions, they suffer severe under-utilization due to the lack of spectrum reusability consideration.

In recent years, quite a few research efforts have been made for the studies on privacy-preserving spectrum auctions [2, 7–9, 14, 15]. Most, if not all, of them have focused on privacy preservation for single-sided spectrum auctions [2, 7, 14, 15]. Different from these works, our work addresses the generic case of double spectrum auctions. There have been a few schemes for privacy-preserving double spectrum auctions [8, 9]. But these schemes only addressed privacy issues for one-channel double spectrum auctions. As far as we know, we are the first to consider privacy preservation for multi-channel double spectrum auctions.

3 Underlying Mechanism and Privacy Goal

In this section, we introduce the underlying mechanism of the double multi-channel spectrum auction, and define the cryptographical protocol privacy.

3.1 TRUE-MCSA Auction Mechanism

Consider a single-round double multi-channel spectrum auction where there is a coordinator as the auctioneer, M primary spectrum users as the sellers, and N secondary spectrum users as the buyers. Consider the general case where each seller sells multiple channels, and each buyer requests multiple channels. The auction is sealed-bid and private, and each bidder (seller or buyer) submits its request or bid to the auctioneer by itself, without knowing any information about other bidders' submissions.

More specifically, in the spectrum auction, a seller m 's request is denoted by (s_m, c_m) ($s_m > 0, c_m \geq 1$), meaning that the seller m requires the minimum per-channel payment s_m to sell c_m channels; a buyer n 's bid is denoted by (b_n, d_n) ($b_n > 0, d_n \geq 1$), representing that the buyer n is willing to pay the maximum price b_n for each channel, and wants to buy at most d_n channels. We call s_m and c_m the seller m 's request value and request number; and call b_n and d_n the buyer n 's bid value and bid number.

An existing solution to the above-mentioned double multi-channel spectrum auction problem is True-MCSA auction mechanism [10]. We will use True-MCSA as our underlying double multi-channel spectrum auction mechanism. A brief review of True-MCSA auction can be found in Appendix A.

3.2 Cryptographical Protocol Privacy

Implicitly, True-MCSA assumes that the auctioneer is trusted. However, if this is not the case, True-MCSA simply leaks all requests and bids to the untrusted auctioneer, and thus no privacy is guaranteed.

To protect the privacy of bidders in the case of an untrusted auctioneer, we introduce an agent to cooperatively perform the auction with the auctioneer. Intuitively, our privacy goal is that as long as the auctioneer and the agent do

not collude with each other (one of them may be semi-honest), nothing about the sensitive inputs (i.e., bid values, bid numbers, and request values) of bidders is leaked to them through the auction, except what is revealed from the auction outcome. We formally present this privacy definition as follows.

Definition 1 (Privacy against semi-honest adversaries). Let $f(x, y)$ be a two-party deterministic auction functionality with inputs x and y from the auctioneer and the agent, respectively, and a common auction outcome $f(x, y)$ for both parties. Suppose that protocol Π computes functionality $f(x, y)$ between the auctioneer and the agent. Let $V_A^\Pi(x, y)$ (resp. $V_B^\Pi(x, y)$) represent the auctioneer’s (resp. the agent’s) view during an execution of Π on (x, y) . In other words, if (x, \mathbf{r}_A^Π) (resp. (y, \mathbf{r}_B^Π)) denotes the auctioneer’s (resp. the agent’s) input and randomness, then

$$\begin{aligned} V_A^\Pi(x, y) &= (x, \mathbf{r}_A^\Pi, m_1, m_2, \dots, m_t), \text{ and} \\ V_B^\Pi(x, y) &= (y, \mathbf{r}_B^\Pi, m_1, m_2, \dots, m_t) \end{aligned}$$

where $\{m_i\}_{i=1}^t$ denote the messages passed between the two parties. Let $O^\Pi(x, y)$ denote the auction outcome after an execution of Π on (x, y) . Then we have $O^\Pi(x, y) = f(x, y)$ for **correctness**, and say that protocol Π **protects privacy** against semi-honest adversaries if there exist probabilistic polynomial time (PPT) simulators S_1 and S_2 such that

$$S_1(x, f(x, y)) \stackrel{c}{\equiv} V_A^\Pi(x, y) \tag{1}$$

$$S_2(y, f(x, y)) \stackrel{c}{\equiv} V_B^\Pi(x, y) \tag{2}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability.

4 PP-MCSA: Design Challenges and Rationale

In this section, we summarize the main challenges in our design, followed by our design rationale to tackle them.

4.1 Design Challenges

Recently, some secure mechanisms for double spectrum auctions, such as PS-TRUST or SDSA [8, 9], have been proposed. However, they all assumed that in the spectrum auction a seller sells one channel, and a buyer buys one channel, and none of them addressed the privacy preservation issue in double multi-channel spectrum auctions. To protect privacy in double multi-channel spectrum auctions like TRUE-MCSA, we face two challenges indicated as follows.

The first one is the *privacy challenge*. As described in Appendix A, TRUE-MCSA involves complex operations in both “VBG splitting and bidding” and “winner determination” steps. How to perform such operations securely by protecting the sensitive inputs is our first challenge.

The second one is the *efficiency challenge*. Straightforwardly securing the auction in our context may result in heavy overhead and thus may degrade the overall performance. Thus, how to achieve practical efficiency in terms of performance with privacy guarantee consists of our second challenge.

4.2 Design Rationale

In order to tackle these two challenges above, we leverage garbled circuits [16, 17] to carefully design the boolean circuits corresponding to the auction mechanism. Specifically, to achieve privacy, we designate binary flags to indicate various conditions, and implement the auction functionality based on these flags in a data-oblivious way; to achieve efficiency, we carefully cache some intermediate values, so that unnecessary repeated circuits are avoided.

5 PP-MCSA: Design Details and Proofs

In this section, we elaborate our privacy preserving spectrum auction protocol, namely PP-MCSA, and prove that it is secure against semi-honest adversaries.

5.1 Protocol Framework

In this subsection, we present the protocol framework of PP-MCSA. Generally speaking, PP-MCSA is a secure protocol for double multi-channel spectrum auctions executed between the auctioneer and the agent. We distinguish two types of inputs, *insensitive* and *sensitive* ones, among which the sensitive input needs to be protected in the spectrum auction. We combine public-key encryption with garbled circuits to protect the sensitive input throughout the auction. As shown in Fig. 2, our protocol consists of three phases, namely, *submission*, *group formation*, and *garbled auction computation*, as specified as follows.

Phase I: Submission

In this phase, sellers and buyers encrypt their respective sensitive inputs, and then send all the necessary inputs to auctioneer. Sensitive inputs include all sellers’ request values, all buyers’ bid values and bid numbers, while the insensitive inputs include all sellers’ IDs and request numbers, and all buyers’ IDs and geographic locations. For sensitive inputs, we split all of them into two parts, and then encrypt them respectively with the auctioneer’s public key pk_A and the agent’s public key pk_B . For insensitive inputs, we directly send them to the auctioneer. The tuples that are submitted by sellers and buyers are presented as follows.

Seller m : $(id_m^s, \langle [s_m^{(1)}]_{pk_A}, [s_m^{(2)}]_{pk_B} \rangle, c_m)$ for $m = 1, 2, \dots, M$
Buyer n : $(id_n^b, (x_n, y_n), \langle [b_n^{(1)}]_{pk_A}, [b_n^{(2)}]_{pk_B} \rangle, \langle [d_n^{(1)}]_{pk_A}, [d_n^{(2)}]_{pk_B} \rangle)$ for $n = 1, 2, \dots, N$

where $[\cdot]_{pk_A}$ and $[\cdot]_{pk_B}$ denote encryptions with pk_A and pk_B , respectively, and $x^{(1)} + x^{(2)} = x \pmod{2^B}$ for any value x , where B is the bit length used.

Additionally, we assume that all communication channels are authenticated and secure, and no one can eavesdrop the data transmitted on the channels.

Phase II: Group Formation

Upon receiving the inputs from sellers and buyers, the auctioneer firstly constructs a conflict graph using all buyers’ geographic locations. Then, according

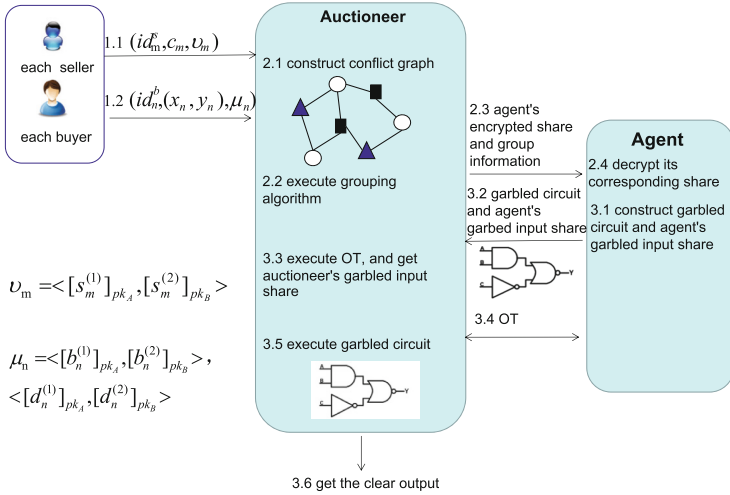


Fig. 2. Protocol framework: First, each buyer or seller submits its input with sensitive parts properly split and encrypted; Next, the auctioneer constructs a conflict graph of buyers, executes buyer grouping algorithm and forwards encrypted input shares to the agent; Then, the agent obtains its corresponding input shares by decrypting the encrypted ones, constructs a garbled circuit based on the auction circuit, garbles its input shares, and sends the garbled circuit and garbled input shares to the auctioneer; Finally, the auctioneer obtains its garbled input shares through running an oblivious transfer with the agent, and executes the garbled circuit and outputs the clear result.

to the conflict graph, the auctioneer executes a bid-independent grouping algorithm to divide buyers into different groups, such that any two members of the same group do not conflict with each other. After group formation, the auctioneer gets group set $G = \{G_1, G_2, \dots, G_T\}$, where the size of group G_t is denoted by N_t . An example of group formation is illustrated in step 2.1 in Fig. 2, where nodes represent buyers, edges represent conflict relations between buyers, nodes with the same shape represent members in the same group, and thus three groups are formed. At the end of this phase, the auctioneer sends the agent's encrypted shares of sensitive inputs, and the grouping information to the agent. Then, both the auctioneer and the agent can obtain their respective shares of sensitive inputs by decrypting the corresponding encrypted shares with their public keys.

Phase III: Garbled Auction Computation

In this phase, the agent constructs a garbled circuit based on the auction circuit which we will design in the next subsection, garbles its shares of sensitive inputs, and generates the output decoder which can decode the garbled output. The garbled circuit, the agent's garbled input shares, and the output decoder are then sent to the auctioneer. Upon receiving these data, the auctioneer executes oblivious transfers (OTs) with the agent to get its garbled shares of sensitive inputs. Finally, with both garbled shares of sensitive inputs and the insensitive

inputs in hand, the auctioneer computes the garbled circuit to get a garbled auction result, and obtains the clear auction result by decoding the garbled one with the output decoder.

The crux of this phase is to design a boolean circuit for our underlying spectrum auction, True-MCSA. A boolean circuit is in essence the binary representation of a data-oblivious algorithm, whose execution path does not depend on its input. In our case, we only need to design auction algorithms which are data-oblivious for sensitive inputs. In the next subsection, we detail our design of such data-oblivious algorithms.

5.2 Data-Oblivious Auction Algorithms

In our context, we only need to protect the sensitive inputs from both sellers and buyers. Thus, we only need to perform sensitive input related operations in the garbled circuits. From here on, we represent a garbled x by $\llbracket x \rrbracket$, meaning that x needs to be protected and should remain in the garbled form throughout the computations. Our data-oblivious spectrum auction is further composed of four steps as follows.

- (1) **Initialization.** In our algorithms, we use arrays of tuples to represent both sellers' and buyers' information. Specifically, we use an array of seller tuples \mathbb{S} to represent all sellers, an array of buyer group tuples \mathbb{G} to represent all buyer groups, an array of buyer tuples \mathbb{G}_t to represent all buyers in the group t ($t = 1, \dots, T$), and an array of virtual buyer group (VBG) tuples \mathbb{G}_t^v to represent all VBGs derived from the group t . The four types of tuples are designed as follows.

Seller tuple: $(id_j^s, s_j, c_j, w_j^s), j \in [1..M]$

Group tuple: $(id_t^g, b_t^g, N_t), t \in [1..T]$

Buyer tuple: $(id_{t,q}^b, b_{t,q}, d_{t,q}, w_{t,q}^b), q \in [1..N_t], t \in [1..T]$

VBG tuple: $(id_t^g, \pi_{t,k}, n_{t,k}, w_{t,k}^v), k \in [1..D], t \in [1..T]$

In a seller tuple, id_j^s , s_j and c_j are the ID, the request value, and the request number of seller j , respectively, while w_j^s is a binary flag indicating whether the seller is a winner (1) or not (0). In a group tuple, id_t^g , b_t^g and N_t are the ID, the minimum buyer bid, and the size of group t . In a buyer tuple, $id_{t,q}^b$, $b_{t,q}$, $d_{t,q}$ are the ID, the bid value, and the bid number of buyer q in the group t ; $w_{t,q}^b$ describes whether a buyer is a winner. In a VBG tuple, $\pi_{t,k}$ and $n_{t,k}$ are the bid, and the size of VBG k derived from group t . $w_{t,k}^v$ is a binary flag indicating whether the VBG k is a winning VBG (1) or not (0). Additionally, D is the maximum bid number of all buyers, which is set as a parameter at the beginning of the auction.

We initialize the arrays \mathbb{S} , \mathbb{G} , \mathbb{G}_t and \mathbb{G}_t^v as follows, where the “null” symbol \perp is a placeholder.

$$\mathbb{S} = \begin{pmatrix} j : 1 & \cdots & M \\ id_j^s : id_1^s & \cdots & id_M^s \\ s_j : \llbracket s_1 \rrbracket & \cdots & \llbracket s_M \rrbracket \\ c_j : c_1 & \cdots & c_M \\ w_j^s : 0 & \cdots & 0 \end{pmatrix}, \quad \mathbb{G} = \begin{pmatrix} t : 1 & \cdots & T \\ id_t^g : id_1^g & \cdots & id_T^g \\ b_t^g : \perp & \cdots & \perp \\ N_t : N_1 & \cdots & N_T \end{pmatrix}$$

$$\mathbb{G}_t = \begin{pmatrix} q : 1 & \cdots & N_t \\ id_{t,q}^b : id_{t,1}^b & \cdots & id_{t,N_t}^b \\ b_{t,q} : \llbracket b_{t,1} \rrbracket & \cdots & \llbracket b_{t,N_t} \rrbracket \\ d_{t,q} : \llbracket d_{t,1} \rrbracket & \cdots & \llbracket d_{t,N_t} \rrbracket \\ w_{t,q}^b : 0 & \cdots & 0 \end{pmatrix}, \quad \mathbb{G}_t^v = \begin{pmatrix} k : 1 & \cdots & D \\ id_t^g : id_1^g & \cdots & id_t^g \\ \pi_{t,k} : \perp & \cdots & \perp \\ n_{t,k} : \perp & \cdots & \perp \\ w_{t,k}^v : 0 & \cdots & 0 \end{pmatrix}$$

- (2) **VBG splitting and bidding.** In this step, a data-oblivious algorithm should be designed for VBG splitting and bidding. The challenge is that this process depends on both buyers’ bid values and their bid numbers, which are sensitive inputs and should be protected.

To design the data-oblivious algorithm, one difficulty is that we do not know the buyers’ bid numbers since they are protected in garbled form, and thus we do not know how many VBGs should be derived from each buyer group. To overcome this difficulty, we assume that the maximum bid number $D = \max_t D_t$ is known, and hence derive exactly D VBGs from each buyer group. To protect both bid values and bid numbers, we keep them and their related computation results in garbled form, while use appropriate logic circuit to implement all required operations. The resulted algorithm is shown in Algorithm 1. Note that we only implement MMIN as the VBG bidding method, while GMAX can be similarly implemented.

Some explanations about Algorithm 1 are as follows.

First, for each group t , the algorithm compares every pair of neighboring buyer tuples (i.e., tuples j and $j + 1$ in \mathbb{G}_t for $j = 1$ to $N_t - 1$) in terms of their bid values, and swaps the two tuples if the former is smaller than the later, such that finally the tuple with the minimum bid value is placed at the last position of \mathbb{G}_t (Line 2 to 5). Note that in Line 4, function $swap(\mathbb{G}_t, \llbracket \lambda \rrbracket, j, j + 1)$ swaps the two tuples j and $j + 1$ of \mathbb{G}_t if $\lambda = 1$. For each field x of the tuples, the swapping function can be implemented using the following circuit [18]:

$$x'_j \leftarrow ((x_j \oplus x_{j+1}) \cdot \lambda) \oplus x_j$$

$$x'_{j+1} \leftarrow x'_j \oplus (x_j \oplus x_{j+1})$$

where x'_j and x'_{j+1} represent the resulted field values. This circuit is very efficient for garbled circuits, since it needs only one non-XOR gate for swapping each pair of bits. Using the free XOR technique, garbled circuits can execute all XOR gates nearly for free, and thus their performances are determined by the number of non-XOR gates executed.

Algorithm 1. Data-oblivious VBG splitting and bidding**Require:** Tuple arrays \mathbb{G} and $\{\mathbb{G}_t\}_{t=1}^T$ **Ensure:** The tuple array \mathbb{G}_t^v

```

1: for  $t = 1 \rightarrow T$  do
2:   for  $j = 1 \rightarrow N_t - 1$  do
3:      $[\lambda] \leftarrow ([b_{t,j}] < [b_{t,j+1}]);$ 
4:      $swap(\mathbb{G}_t, [\lambda], j, j + 1);$ 
5:   end for
6:   for  $k = 1 \rightarrow D$  do
7:      $[n_{t,k}] \leftarrow 0;$ 
8:     for  $j = 1 \rightarrow N_t - 1$  do
9:        $[\gamma] \leftarrow ([d_{t,j}] \geq k);$ 
10:       $[n_{t,k}] \leftarrow [n_{t,k}] + [\gamma];$ 
11:    end for
12:     $[\pi_{t,k}] \leftarrow [b_{t,N_t}] \cdot [n_{t,k}];$ 
13:  end for
14: end for
      return  $\mathbb{G}_t^v$ 

```

Second, Lines 6 to 13 compute the D VBGs for each group t . To compute the k th VBG, the bid number of each group member except the last one (who has the minimum bid value) is compared with k (Line 9), and if it is not smaller than k , the group member is added to the VBG (Line 10). Finally, the bid value of the k th VBG is computed (Line 12).

Note that in the computations, the sensitive inputs, i.e. $b_{t,j}$'s and $d_{t,j}$'s, and their related computation results, i.e., λ 's, γ 's, $n_{t,k}$'s and $\pi_{t,k}$'s, are all kept in garbled form, such that the sensitive inputs can be well protected.

- (3) **Winner determination.** This step applies a variant of McAfee framework to determine winners as shown in Sect. 3. Since this process contains numerous operations, such as comparisons and selections, depending on requests or bids, designing its data-oblivious version is challenging. In order to address this challenge, our main idea is to introduce some appropriate binary flags to indicate different conditions, and construct suitable circuits based on them to data-obliviously achieve the required functions. We describe the data-oblivious winner determination in Algorithm 2.

The details of Algorithm 2 are described as follows.

First, both seller tuples and VBG tuples are appropriately sorted as required in McAfee framework (Lines 1 to 4). In Line 1, the total number of selling channels L is computed in the clear, since initially all request numbers c_j 's are not protected. In Line 3, all VBG tuples from different groups are merged into a uniform VBG tuple array $\mathbb{G}^v = \{id_k^v, \pi_k, n_k, w_k^v\}_{k=1}^T$, where $id_k^v \in \{id_t^v\}_{t=1}^T$, and then in Line 4 \mathbb{G}^v is sorted in term of π_k 's. Note that once sorted (Lines 2 & 4), all fields of \mathbb{S} and \mathbb{G}^v become garbled, otherwise the ranking information of s_i 's and π_k 's would be leaked.

Algorithm 2. Data-oblivious winner determination**Require:** Tuple arrays \mathbb{S} and $\{\mathbb{G}_t^v\}_{t=1}^T$ **Ensure:** The winning seller tuple array \mathbb{W}^s , the winning VBG tuple array \mathbb{W}^v , and the critical request value φ

- 1: Compute $L \leftarrow \sum_{i=1}^M c_i$;
- 2: Sort \mathbb{S} in no-descending order of s_i 's, s.t.

$$\llbracket s_1 \rrbracket \leq \llbracket s_2 \rrbracket \leq \dots \leq \llbracket s_M \rrbracket$$

- 3: Merge $\mathbb{G}^v \leftarrow \bigcup_{t=1}^T \mathbb{G}_t^v$;
- 4: Sort \mathbb{G}^v in no-increasing order of π_k 's, s.t.

$$\llbracket \pi_1 \rrbracket \geq \llbracket \pi_2 \rrbracket \geq \dots \geq \llbracket \pi_K \rrbracket$$

- 5: $Q \leftarrow \min\{L, K\}$; $\llbracket \varphi \rrbracket \leftarrow 0$; $\llbracket W \rrbracket \leftarrow 0$;
- 6: **for** $i = 1 \rightarrow Q$ **do**
- 7: $\llbracket \lambda_M \rrbracket \leftarrow 0$; $\llbracket \delta_M \rrbracket \leftarrow 0$;
- 8: $\llbracket j_i \rrbracket \leftarrow 0$; $\llbracket \varphi_i \rrbracket \leftarrow 0$;
- 9: $\llbracket W_i \rrbracket \leftarrow 0$;
- 10: **for** $j = M - 1 \rightarrow 1$ **do**
- 11: $\llbracket \lambda_j \rrbracket \leftarrow \llbracket \sum_{l=1}^j c_l \rrbracket < i$;
- 12: $\llbracket \delta_j \rrbracket \leftarrow \llbracket \lambda_j \rrbracket \oplus \llbracket \lambda_{j+1} \rrbracket$;
- 13: $\llbracket j_i \rrbracket \leftarrow \llbracket j_i \rrbracket + \llbracket \delta_j \rrbracket \cdot j$;
- 14: $\llbracket \varphi_i \rrbracket \leftarrow \llbracket \varphi_i \rrbracket + \llbracket \delta_j \rrbracket \cdot \llbracket s_{j+1} \rrbracket$;
- 15: $\llbracket W_i \rrbracket \leftarrow \llbracket W_i \rrbracket + \llbracket \delta_j \rrbracket \cdot \sum_{l=1}^j c_l$;
- 16: **end for**
- 17: $\llbracket \omega_i \rrbracket \leftarrow \llbracket (\sum_{l=1}^i \llbracket \pi_l \rrbracket) \geq i \cdot \llbracket \varphi_i \rrbracket \rrbracket$;
- 18: **if** $i > 1$ **then**
- 19: $\llbracket \varphi \rrbracket \leftarrow \llbracket \varphi \rrbracket + \llbracket \varphi_{i-1} \rrbracket \cdot (\llbracket \omega_{i-1} \rrbracket \oplus \llbracket \omega_i \rrbracket)$;
- 20: $\llbracket W \rrbracket \leftarrow \llbracket W \rrbracket + \llbracket W_{i-1} \rrbracket \cdot (\llbracket \omega_{i-1} \rrbracket \oplus \llbracket \omega_i \rrbracket)$;
- 21: **end if**
- 22: **end for**
- 23: $\llbracket \varphi \rrbracket \leftarrow \llbracket \varphi \rrbracket + \llbracket \varphi_Q \rrbracket \cdot \llbracket \omega_Q \rrbracket$; $\llbracket W \rrbracket \leftarrow \llbracket W \rrbracket + \llbracket W_Q \rrbracket \cdot \llbracket \omega_Q \rrbracket$;
- 24: Reveal $\llbracket W \rrbracket$, and $\mathbb{W}^v \leftarrow$ the first W tuples of \mathbb{G}^v ;
- 25: Reveal $\llbracket j_{W+1} \rrbracket$, and $\mathbb{W}^s \leftarrow$ the first j_{W+1} tuples of \mathbb{S} ;
- 26: Reveal $\llbracket \varphi \rrbracket$ as the critical request value;
- 27: Resort \mathbb{W}^v in increasing order of id_t^g 's, and then in no-increasing order of π_k 's;
- 28: Resort \mathbb{W}^s increasing order of id_j^s 's;
- return** $\mathbb{W}^s, \mathbb{W}^v, \varphi$;

Second, winners are determined with two nested for loops (Lines 5 to 22). Specifically, the outer loop iterates over each possible trade i , and computes $\llbracket \omega_i \rrbracket$ indicating whether trade i is profitable (Line 17), the critical request value $\llbracket \varphi \rrbracket$ (Line 19), and the number of winning VBGs $\llbracket W \rrbracket$ (Line 20). While the inner loop computes the index j_i of the last winning seller (Lines 8 & 13), the critical request value φ_i (Lines 8 & 14), the number of winning VBGs W_i (Lines 9 & 15), given trade i is the last profitable trade. Note all these computations are performed in the garbled form.

More specifically, to find the index j_i of the last profitable seller provided the last profitable trade i , we introduce two vectors of flags, i.e., λ_j 's and δ_j 's, where

λ_j : indicates whether $j \leq j_i$, i.e., $\sum_{l=1}^j c_l < i$ ($\lambda_j = 1$) or not ($\lambda_j = 0$) (Lines 7 & 11).

δ_j : indicates whether $j = j_i$ ($\delta_j = 1$) or not ($\delta_j = 0$) (Lines 7 & 12).

According to the auction logic, the two flag vectors should take values as the following pattern:

$$\begin{pmatrix} j : 1 \cdots j_i - 1 & j_i & j_i + 1 \cdots M \\ \lambda_j : 1 \cdots 1 & 1 & 0 \cdots 0 \\ \delta_j : 0 \cdots 0 & 1 & 0 \cdots 0 \end{pmatrix}$$

Thus, $\delta_j = \lambda_j \oplus \lambda_{j+1}$ holds (Line 12).

With similar idea, we compute the profitable flags $\llbracket \omega_i \rrbracket$'s, and the critical request value $\llbracket \varphi \rrbracket$ (which is the request value of the critical seller) and the number of winning VBGs $\llbracket W \rrbracket$ (Lines 17 to 20, and Line 23).

It is worth noting that in Line 14, we use s_{j+1} instead of s_j , since the critical seller is next to the last winning seller. Additionally, in Lines 11, 15 & 17, for simplicity, we repeatedly use the sum equations of c_l 's or π_l 's. However, in real implementation, it is not necessary to repeatedly compute the sums. Optimally, we can compute each sum just once, and cache them for later use.

Finally, some garbled results are appropriately revealed. Specifically, the number of winning VBGs $\llbracket W \rrbracket$ is revealed, and the first W tuples of \mathbb{G}^v form the winning VBG tuple array \mathbb{W}^v (Line 24). Then, $\llbracket j_{W+1} \rrbracket$ is revealed as the number of winning sellers, and the first j_{W+1} seller tuples of \mathbb{S} form the winning seller tuple array \mathbb{W}^s (Line 25). Next, $\llbracket \varphi \rrbracket$ is revealed as the critical request value (Line 26). At the same time, \mathbb{W}^v and \mathbb{W}^s are appropriately resorted, such that the bid order of winning VBGs from different groups and the request order of winning sellers will not be revealed when decoded in the later (Lines 27 & 28). At last, \mathbb{W}^v , \mathbb{W}^s , and φ are returned.

- (4) **Pricing.** In this step, we compute the selling prices for winning sellers and the buying prices for winning buyers, as described in Algorithm 3. Specifically, each winning seller m sells all its c_m channels, and is paid by its selling price $p_m^s = c_m \cdot \varphi$ (Lines 1 to 14). Each winning VBG k is charged by its bid π_k , which is evenly shared by the winning buyers in the VBG. Thus, each winning buyer $n \in G_t$ is charged by its buying price $p_n^b = \min(d_n, D_t) \cdot b_t^g$, where $D_t = \sum_{V \in \mathbb{W}^v} [V.id_k^v = id_t^g]$ is the total number of winning channels for group G_t , and $b_t^g = \pi_{t,k} / (n_{t,k} - 1)$ is the minimum bid value of group G_t . Note that Lines 5 to 9 compute the set of winning buyer groups G^w , and Lines 10 to 18 compute the winning buyers in all winning groups and their prices.

Algorithm 3. Pricing

Require: The winning seller tuple array \mathbb{W}^s , the winning VBG tuple array \mathbb{W}^v , and the critical request value φ ;

Ensure: Winners and their clearing prices;

```

1: for  $E \in \mathbb{W}^s$  do
2:   Reveal  $E.id_m^s$ ;
3:   Seller  $m$  sells  $c_m$  channels, and is paid with  $p_m^s \leftarrow c_m \cdot \varphi$ ;
4: end for
5:  $G^w \leftarrow \emptyset$ ;
6: for  $V \in \mathbb{W}^v$  do
7:   Reveal  $V.id_k^v$  as  $id_t^g$ ;
8:    $G^w \leftarrow G^w \cup \{t\}$ ;
9: end for
10: for  $t \in G^w$  do
11:    $D_t = \sum_{V \in \mathbb{W}^v} [V.id_k^v = id_t^g]$ ;
12:   for  $q = 1 \rightarrow N_t - 1$  do
13:     Reveal  $id_{t,q}^b$  as  $id_n^b$ ;
14:      $\llbracket h_t \rrbracket \leftarrow \min(\llbracket d_n \rrbracket, D_t)$ ;
15:     Reveal  $\llbracket h_t \rrbracket$  and  $\llbracket b_t^g \rrbracket$ ;
16:     Buyer  $n$  buys  $h_t$  channels, and pays  $p_n^b \leftarrow h_t \cdot b_t^g$ ;
17:   end for
18: end for
return All winners and their prices;

```

5.3 Security Analysis

In this section, we prove that our protocol preserves privacy in the sense of cryptography.

Theorem 1. *As long as the auctioneer and the agent do not collude with each other, PP-MCSA preserves privacy against semi-honest adversaries.*

Proof: The proof of privacy for both Phases I and II is trivial. The reasons are as follows. In Phase I no secure computations are involved, sensitive inputs are secretly shared between the auctioneer and the agent, and hence the view of adversary can be easily simulated. While in Phase II, group formation is completely dependent on sensitive inputs, and no privacy issues need to be considered. Therefore, we mainly prove the privacy of Phase III.

To prove the privacy of garbled auction computation phase, we actually need to prove the privacy of Algorithms 1, 2 and 3 separately, and then by applying the sequential composition theory [19] we can conclude that the phase III preserve privacy, and thus the whole protocol also preserve privacy.

We now examine the design of Algorithms 1, 2 and 3. We can see that for every sensitive input related operation, the algorithms compute it in a garbled circuit, and they also store every sensitive input related value by garbled values. At the same time, all garbled values that are revealed in the algorithms carry no more information than what can be inferred from the auction outcome. That is,

these algorithms do not revealed any information about the sensitive information except what can be revealed from the auction outcome. By the privacy definition in Sect. 3.2, when one party (the auctioneer or the agent) is corrupted, the view of the adversary can be easily simulated (e.g., an encrypted or garbled value can be simulated by a random number of the same bit length). As a result, we can conclude that our algorithms achieve the privacy of garbled circuits, whose privacy is formally proved in [16].

Therefore, as long as the auctioneer and the agent do not collude with each other, PP-MCSA preserves privacy. \square

6 Performance Evaluation

6.1 Experimental Setting

We implement our protocol in two cases: *original* implementation and *improved* implementation. In the original implementation, we implement our algorithms literally, while in the improved implementation, we implement them with cache optimization, where we compute the repeatedly used sums (i.e., Lines 11, 15 & 17 in Algorithm 2) just once and cache them for later use. Our experiments are carried out on top of FastGC [20], a java-based framework for the garbled circuit computations. We simulate the auctioneer and the agent with two processes on the same computer. Experimental settings are as follows: buyers are randomly distributed in a $2000\text{m} \times 2000\text{m}$ area, and the interference radius is 400m. The request values of sellers and the bid values of buyers are generated randomly in the intervals $[1..150]$ and $[1..50]$, respectively. The both request numbers and bid numbers are generated randomly in the interval $[1..10]$, and thus D is set to 10 which is the maximum bid number. Throughout our experiments, we use bit length 16, unless otherwise stated, and each point represents the average of 10 times simulation runs.

In the simulation, we run our protocol on a 64-bit Windows 7 Desktop with Intel(R) Core(TM) i5 CPU @3.3GHz and 8GB of memory. We focus on the following two performance metrics:

- *Computation overhead*: total running time for executing our protocol by the auctioneer and the agent.
- *Communication overhead*: total communication cost (data size of all messages that are sent between the auctioneer and the agent).

6.2 Result Analysis

We conduct experiments to compare the performance of the original and improved implementations in two cases: (1) when the number of sellers varies; (2) when the number of buyers varies. We further trace the performance of the improved implementation (3) when the bit length of request values and bid values varies; and (4) when bigger numbers of sellers and buyers vary.

(1) Number of sellers varies. Figure 3 illustrates the comparisons of both computation and communication overheads between the original and improved implementations, when the number of sellers M increases from 50 to 100, and the number of buyers is fixed at $N = 500$, and $N = 600$. We can see that both running time and communication cost of the original implementation increase much faster than those of the improved implementation. The reason is that the cache optimization in the improved implementation (Lines 11, 15 and 17 in Algorithm 2) avoids repeating the addition computations in the nested loops, and thus greatly reduces the computation and communication overheads.

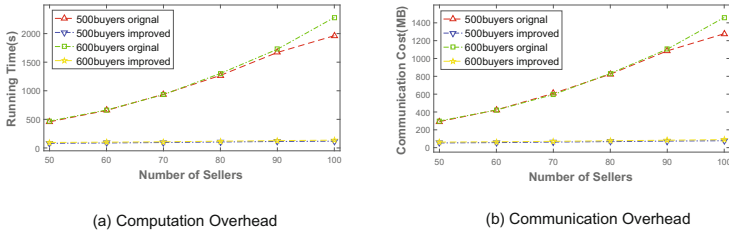


Fig. 3. Comparisons of computation and communication overheads between the original and improved implementations as the number of sellers M varies.

(2) Number of buyers varies. Figure 4 shows the performance comparisons between the original and improved implementations, when the number of sellers is fixed to $M = 100$ and $M = 110$, and the number of buyers N increases from 200 to 600. Similar to Figs. 3 and 4 demonstrates that the improved implementation is much more efficient than the original one in term of computation and communication overheads. In the same way, the cache optimization is the source of this performance improvement.

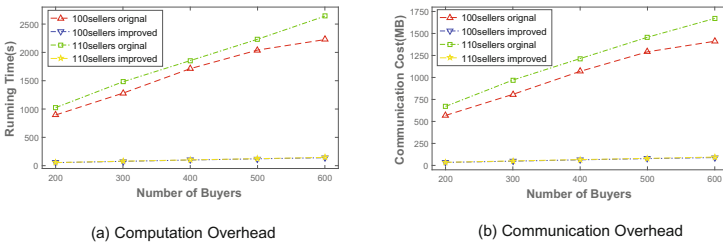


Fig. 4. Comparisons of computation and communication overheads between the original and improved implementations as the number of buyers N varies.

(3) Bit length varies. Figure 5 traces the impact on performance when changing the bit length of bid values and request values in the improved implementation. We vary the bit length from 10 to 20, while fix the number of sellers at

$M = 80$, and the number of buyers at $N = 500$. We can observe that both computation and communication overheads grow linearly as the bit length increases. This is natural, since most of the elemental boolean circuits (e.g., addition, comparison) grow linearly in size when the bit length of its input values increases.

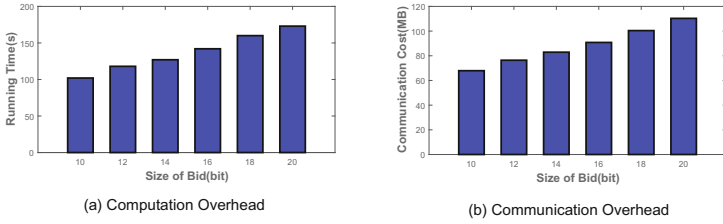


Fig. 5. Comparisons of computation and communication overheads between the original and improved implementations as the bit length varies.

(4) Then bigger numbers of sellers and buyers vary. Figure 6 traces the performance of the improved implementation when the number of buyers varies from 1500 to 3500, for the number of sellers $M = 300, 400$ and 500 , respectively. This figure shows that our improved implementation is rather efficient in both computation and communication performance for bigger numbers. For example, all running times are within 23 min, and all communication costs are within 1600 MB. Meanwhile, both computation and communication overheads scale gracefully as the numbers of sellers and buyers increase.

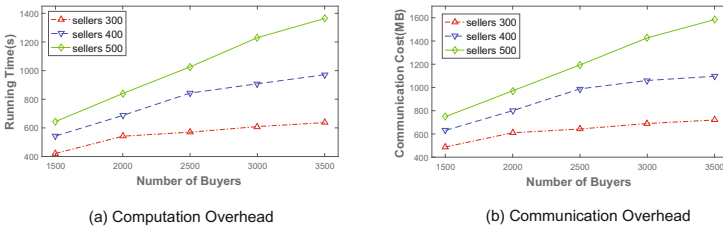


Fig. 6. Computation and communication overheads of the improved implementation as the big numbers of sellers and buyers vary.

7 Conclusion

In this paper, we have proposed PP-MCSA, the first privacy-preserving mechanism for multi-channel double spectrum auctions to our knowledge. To address the challenges imposed by the multi-channel double spectrum auction scenario, we have leveraged garbled circuits in our protocol design. Specifically, we design

data-oblivious algorithms whose execution path does not depend on their sensitive inputs and then turn these algorithms into garbled circuits to address the privacy challenge. Then, we use cache optimization, which caches some intermediate values to avoid repeated circuits, to improve the garbled circuits and hence address the efficiency challenge. Finally, we have theoretically proved the privacy of PP-MCSA, and experimentally shown that it incurs limited computation and communication overheads.

Acknowledgment. The work is supported by the Natural Science Foundation of China under Grant No. 61572031 & 61502443. We thank the anonymous reviewers for their valuable comments that helped improve the final version of this paper.

A True-MCSA Auction

True-MCSA is a truthful double spectrum auction mechanism that allows multi-channel requests from both buyers and sellers, while ensures spectrum reusability. The symbols of the auction can be described in Table 1. Specifically, TRUE-MCSA is composed of the following four steps:

- (1) **Bid-independent Buyer Group Formation:** In this step, the conflict graph of buyers is constructed in term of their geographic locations, and buyers that do not interfere with each other are grouped into the same group. In this way, buyers in the same group can use the same channels without interference. Note that the group formation algorithm should be bid-independent, otherwise bid manipulation is allowed, and hence the auction becomes untruthful.

Table 1. Key symbols for TRUE-MCSA

M, N	Numbers of sellers and buyers
T	Numbers of buyer groups
s_m, c_m	Seller m 's request value and request number
b_n, d_n	Buyer n 's bid value and bid number
(x_n, y_n)	Location of buyer n
D_t	Maximal number of channel of group t
π	Bid vector of virtual buyer group
S	Request vector of sellers
$j(i)$	The seller in the i^{th} trade
k_l	The last profitable trade
L	Sum of sellers channel number
G	$G = \{G_t\}_{t=1}^T$, Global bid set of groups
G_t	The tuple of group t
G_t^v	The tuple of virtual buyer group t

(2) **Virtual Buyer Group (VBG) Splitting and Bidding:** To address the multi-channel requests from buyers, this step splits a buyer group G_t into $D_t = \max_{i \in G_t} d_i$ virtual buyer groups (VBGs), where each VBG only requests for one channel.

After splitting a buyer group into VBGs, we come up with the VBG bidding. Paper [10] proposed two VBG bidding algorithms, member-minimized (MMIN) and group-maximized (GMAX). We only review MMIN as follows. To bid for each VBG derived from a buyer group, the group member with the minimum bid is chosen as the critical buyer, which is removed from all the derived VBGs. Then, each VBG bids with the minimum bid (i.e., the critical buyer's bid) multiplying its size after removing.

(3) **Winner Determination:** At this point, suppose after VBG splitting and bidding we get totally K VBGs with bid values $\{\pi_k\}_{k=1}^K$. Recall that we have M sellers with request values $\{s_m\}_{m=1}^M$. Then, this step applies McAfee's framework to winner determination as follows.

First of all, the sellers' request values s_m 's are sorted in non-decreasing order, and the VBGs' bid values π_k 's are sorted in non-increasing order:

$$O' : s_1 \leq s_2 \leq \dots \leq s_M$$

$$O'' : \pi_1 \geq \pi_2 \geq \dots \geq \pi_K$$

Then, each seller's request value s_m is rewritten as many times as the number c_m of channels he bid, resulting in the bid mapping between sellers and VBGs as follows:

$$\begin{aligned}
 O' : & \overbrace{s_1 \leq \dots \leq s_1}^{c_1} \leq \overbrace{s_2 \leq \dots \leq s_2}^{c_2} \leq \dots \leq \overbrace{s_M \leq \dots \leq s_M}^{c_M} \\
 O'' : & \pi_1 \geq \dots \geq \pi_{c_1} \geq \pi_{c_1+1} \geq \dots \geq \pi_{c_1+c_2} \geq \dots \geq \pi_{1+\sum_{t=1}^{M-1} c_t} \geq \dots \geq \pi_K
 \end{aligned}$$

Finally, find the last profitable trade k_l as:

$$k_l = \arg \max_{i \leq \min\{L, K\}} \{ \sum_{t=1}^i \pi_t \geq i \cdot s_{j(i)} \}$$

Here, L represents the total number of channels provided by sellers, namely, $L = \sum_{j=1}^M c_j$, and $j(i)$ computes the seller index j when the trade index is i , namely,

$$j(i) = 1 + \arg \max_{0 \leq h \leq M-1} \{ \sum_{t=1}^h c_t < i \}.$$

As a result, the last profitable seller is $j(k_l)$. In order to achieve truthfulness, the last profitable seller, as well as all trades involving the seller, should be sacrificed to price the winners. Then the auction winners are the first $j(k_l) - 1$ sellers in O' and the first $k = \sum_{t=1}^{j(k_l)-1} c_t \leq k_l - 1$ VBGs in O'' .

- (4) **Pricing:** Each buyer in the same winning VBG pays an even share of the VBG bid, and each winning channel is paid by the price $s_{j(k_l)}$. As a result, each winning buyer pays the sum of what it pays in all the winning VBGs it belongs to, and each winning seller is paid with the product of multiplying its request number and the price $s_{j(k_l)}$.

References

1. Valenta, V., Maršálek, R., Baudoin, G., Villegas, M., Suarez, M., Robert, F.: Survey on spectrum utilization in Europe: measurements, analyses and observations. In: Proceedings of CROWNCOM (2010)
2. Pan, M., Sun, J., Fang, Y.: Purging the back-room dealing: secure spectrum auction leveraging paillier cryptosystem. *IEEE JSAC* **29**(4), 866–876 (2011)
3. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of EC (1999)
4. Yokoo, M., Suzuki, K.: Secure generalized Vickrey auction without third-party servers. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 132–146. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27809-2_17
5. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: Robust, privacy protecting and publicly verifiable sealed-bid auction. In: Proceedings of ICICS (2002)
6. Huang, H., Li, X., Sun, Y., Xu, H.: PPS: privacy-preserving strategyproof social-efficient spectrum auction mechanisms. *IEEE Trans. Parallel Distrib. Syst.* **26**(5), 1393–1404 (2014)
7. Huang, Q., Tao, Y., Wu, F.: SPRING: a strategy-proof and privacy preserving spectrum auction mechanism. In: Proceedings of INFOCOM (2013)
8. Chen, Z., et al.: PS-TRUST: provably secure solution for truthful double spectrum auctions. In: Proceedings of INFOCOM (2014)
9. Chen, Z., Wei, X., Zhong, H., Cui, J., Xu, Y., Zhang, S.: Secure, efficient and practical double spectrum auction. In: Proceedings of IEEE/ACM IWQoS, pp. 1–6 (2017)
10. Chen, Z., Huang, H., Sun, Y., Huang, L.: True-MCSA: a framework for truthful double multi-channel spectrum auctions. *IEEE Trans. Wirel. Commun.* **12**(8), 3838–3850 (2013)
11. Zhou, X., Zheng, H.: TRUST: a general framework for truthful double spectrum auctions. In: Proceedings of INFOCOM, pp. 999–1007 (2009)
12. Chen, Y., Zhang, J., Wu, K., Zhang, Q.: TAMES: a truthful double auction for multi-demand heterogeneous spectrums. *IEEE Trans. Parallel Distrib. Syst.* **25**(11), 3012–3024 (2014)
13. Feng, X., Chen, Y., Zhang, J., Zhang, Q., Li, B.: TAHES: a truthful double auction mechanism for heterogeneous spectrums. *IEEE Trans. Wirel. Commun.* **11**(11), 4038–4047 (2012)
14. Huang, Q., Gui, Y., Wu, F., Chen, G.: A general privacy-preserving auction mechanism for secondary spectrum markets. *IEEE/ACM Trans. Netw.* **24**(3), 1881–1893 (2016)
15. Sun, Y., et al.: Privacy-preserving strategyproof auction mechanisms for resource allocation in wireless communications. In: Proceedings of BigCom (2016)
16. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: FOCS (1982)

17. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. *J. Cryptol.* **22**, 161–188 (2009)
18. Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., Boneh, D.: Privacy-preserving matrix factorization. In: *Proceedings of ACM CCS* (2013)
19. Goldreich, O.: *Foundations of Cryptography: Volume 2-Basic Applications*. Cambridge University Press, Cambridge (2004)
20. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: *USENIX Security Symposium*, vol. 201, no. 1 (2011)

Full Paper Session VI: Signature Schemes



Hierarchical Group Signatures with Verifier-Local Revocation

Lin Hou^{1,2}(✉), Renzhang Liu³, Tian Qiu^{1,2}, and Dongdai Lin^{1,2}

¹ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{houlin,qiutian,ddlin}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

³ Westone Cryptologic Research Center, Westone Information Industry INC., Beijing, China
liu.renzhang05391@westone.com.cn

Abstract. Group signatures are important when it comes to authentication with privacy. Hierarchical group signatures, as a proper generalization of group signatures, have splendid applications in e-commerce. One key issue for such schemes is to support membership revocation in an efficient as well as secure way. To this end, the notion of group signatures with verifier-local revocation was proposed and well-studied, where the revocation messages are sent only to verifiers. However, such issue has not been formally studied in the context of hierarchical group signatures. In this paper, we raise and formalize the new notion of hierarchical group signatures with verifier-local revocation, and propose a semi-generic construction from group signatures with verifier-local revocation. When instantiating it with a variant of the group signature scheme proposed by Gordon, Katz and Vaikuntanathan, a lattice-based construction is implicitly given.

Keywords: Group signatures · Authentication with privacy
Lattice-based cryptography

1 Introduction

Digital signatures are ubiquitous as a main approach for authentication. However, ordinary digital signatures (via PKI) inherently expose signers' identities, and such privacy is much desired in many real-world scenarios, e.g., e-commerce, e-cash, anonymous online communications and more. To solve this issue, several privacy-oriented signatures were proposed, such as ring signatures [25], traceable signatures [13], domain-specific pseudonymous signatures [6], and especially, *group signatures* (GS) [9]. Loosely speaking, a group signature scheme has both anonymity and traceability. The former means that group members can sign on behalf of the group, without leaking out their identities; on the other hand, given some valid message-signature pair, traceability enables some designated

manager to run a open algorithm using some secret tracing key and figure out the actual signer.

Up to now, many generalized notions of group signatures were proposed, such as sub-group signatures [4], multi-group signatures [4], group blind signatures [22] and *hierarchical group signatures* (HGS) [26]. Hierarchical group signatures was brought up by Wikström *et al.* in the context of anonymous credit card systems. Imagine a balanced tree of depth two. The root stands for some payment network, and nodes at depth one are distinct card-issuing banks, while leaves are their users (card-holders). In a transaction, a user signs on the transaction information to generate a signature, of which the validity with respect to some *single* public verification key can be easily checked by the merchant; the merchant sends the message-signature pair to the payment network, and the latter will figure out then route it to the user's bank; eventually, the bank traces to the user, and debits its account. One admirable feature of such system is that by such hierarchical tracing, nothing except those absolutely necessary will be revealed to each party: the merchant is convinced that the transaction information has been signed by some valid user, but cannot know its issuing bank (let alone its identity); the payment network can route the transaction to the user's bank, but infeasible to figure out its identity; in contrast, the bank must be able to determine the exact identity to debit the correct account.

Related Work. In their foundational work [5], Bellare *et al.* formalized two properties for static group signatures, namely *full-anonymity* and *full-traceability*. Plenty of subsequent work has been done within this framework. They also proposed a generic GS construction from trapdoor permutations, which essentially reflects a *sign-encrypt-proof* designing paradigm.

On the other hand, lattice-based cryptography [1] has seen a flourish of research works in recent years. In our interest, Gordon, Katz and Vaikuntanathan gave the first group signature scheme from lattice assumptions [12] (abbreviated as the GKV scheme), and we refer to the full version or the original paper for a detailed description of their scheme; besides, there are several lattice-based schemes [8, 14–21, 23] with different security models, different levels of efficiency and functionality.

Wikström *et al.* introduced the notion of hierarchical group signatures [26]. Without loss of generality, they considered a *balanced tree* depicting the hierarchy, where inner nodes are managers and leaves are signers. Given a valid message-signature pair, a *path-following tracing*, namely an iterative process where some father node (initialized as the root manager) traces then routes the pair to some child node, will always locate some signer who is (amongst) the actual generator(s); on the other hand, nobody can non-trivially figure out the actual signer without such hierarchical tracing. These are formalized into the *traceability* and *anonymity* properties for HGS in the framework of Bellare *et al.* [5]. Moreover, Wikström *et al.* gave a generic construction assuming the existence of a family of trapdoor permutations.

Motivations. Wikström *et al.* did foundational works for HGS in the static setting, where no dynamic joining or revocation will be allowed once the system is set up. However in practice, there are scenarios where revocation is desired or even necessary, for example, when some signer misbehaves or accidentally exposes its secret signing key and thus has to be removed from the original hierarchy. *Verifier-local revocation* is a highly efficient approach early brought up in the setting of group signatures, with which revocation messages are *only* sent to signature verifiers. Naturally, we wonder how to make an HGS scheme *efficiently revocable*.

Our Contributions and Main Techniques. We formalize the new notion of *hierarchical group signatures with verifier-local revocation* (HGS-VLR) and propose a semi-generic construction from the existing notion of *group signatures with verifier-local revocation* (GS-VLR). Moreover, we implicitly provide an instantiation from lattice assumptions, using a variant of the GKV group signature scheme.

An HGS-VLR scheme consists of three algorithms: a key generation algorithm, a signing algorithm and a verification algorithm. Unlike regular HGS, HGS-VLR has no explicit tracing algorithm. A father node possessing all children's revocation tokens can trace implicitly using the verification algorithm. Correctness says that for any honestly generated message-signature pair, it will pass the verification if and only if no ancestor of the signer (including itself) has been revoked. As for anonymity, we propose the full-version of anonymity for HGS-VLR where the adversary is given all signers' secret keys. In contrast, an insider-version of anonymity is usually considered in the context of GS-VLR, namely the adversary cannot obtain the secret keys of challenge identities.

The main difficulties lie in how to properly define the traceability property for HGS-VLR. If we stick to the original path-following tracing as in defining traceability for HGS, inconsistencies will occur. Instead, we introduce *a whole-depth tracing* in the model, which involves *all managers at the penultimate depth* in a joint and independent tracing of some valid message-signature pair. In our model, the adversary is claimed a success if it comes up with some valid message-signature pair, and it holds either all managers at the penultimate depth cannot open this pair, or there exists some manager at the penultimate depth traces it to some honest signer. The implications and rationalities of the traceability property as such defined will be detailed in Sect. 4.

With our new notions and models, a semi-generic HGS-VLR construction from GS-VLR naturally arises. We regard all parties at the same depth as a group respectively, and generate keys and revocation tokens for them. A signer is given all signing keys of its ancestors (including itself), and for tracing purposes, a manager is given all revocation tokens of its direct children. To generate a signature, the signer produces compositional group signatures for all its ancestors. The correctness, anonymity and traceability can be easily reduced to those of the underlying GS-VLR respectively.

Outline of This Paper. We first recall some lattice knowledge. In Sect. 3, we recall the notion of GS-VLR, and show that a variant of the GKV scheme is a fully-anonymous GS-VLR scheme. In Sect. 4, we introduce and formalize our new notion, and give a semi-generic construction. Its GKV instantiation is straightforward, when combining work done in Sect. 3. We conclude this paper in Sect. 5. Due to lack of spaces, we refer interested readers to the full version for all proofs.

2 Preliminaries

In this section, we briefly introduce some background on lattice.

2.1 Notations

Let $x||y$ denote the concatenation of two binary strings x and y . Vectors are assumed to be in column form and are written using bold lower-case letters, e.g. \mathbf{x} , and let $\|\mathbf{x}\|$ denote the Euclidean norm of a vector \mathbf{x} . Matrices are written as bold capital letters. For a matrix \mathbf{X} , let $\|\mathbf{X}\|$ denote the maximum of the Euclidean norms of the columns of \mathbf{X} and $\tilde{\mathbf{X}}$ is the *Gram-Schmidt orthogonalization* of \mathbf{X} .

We denote the set $\{1, \dots, N\}$ by $[N]$, where $N \in \mathbb{N}$. If S is some finite set, we denote its cardinality by $|S|$, and denote choosing a uniformly random element from S by $s \stackrel{\$}{\leftarrow} S$; the uniform distribution on S is denoted $U(S)$. If A is a randomized algorithm, then $[A(x, y, \dots)]$ denotes the set of all outputs having positive probability on inputs x, y, \dots . We use the standard big- O notation to classify the growth of functions. **Oracles** are written bold to be distinguished from algorithms.

2.2 Lattices

Definition 1 (Lattice). Let $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ be n ($\leq m$) linearly independent vectors in \mathbb{R}^m . The **lattice** generated by \mathbf{B} , denoted by $\mathcal{L}(\mathbf{B})$, is the set of all the integer linear combination of the vectors in \mathbf{B} , and $\mathbf{B} \in \mathbb{R}^{m \times n}$ is called a **basis** of $\mathcal{L}(\mathbf{B})$. Namely, $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^n x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\} = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$.

Definition 2 (Shortest Vector Problem, SVP). Given a basis $\mathbf{B} \in \mathbb{R}^{m \times n}$ of $\mathcal{L}(\mathbf{B})$, find the shortest nonzero vector in $\mathcal{L}(\mathbf{B})$, denoted by $\lambda_1(\mathcal{L}(\mathbf{B}))$.

It has been proved that the SVP problem is NP-hard under randomized reduction [2]. We use its promise variant, namely the GapSVP_γ problem.

Definition 3 (GapSVP_γ). An instance of the problem is given by a pair (\mathbf{B}, r) where $\mathbf{B} \in \mathbb{Z}^{m \times n}$ is a lattice basis and $r \in \mathbb{Q}$. In YES instance, $\lambda_1(\mathcal{L}(\mathbf{B})) \leq r$. In NO instance, $\lambda_1(\mathcal{L}(\mathbf{B})) > \gamma \cdot r$. The goal is to determine which case the input instance is.

Two classes of random lattices are widely used in cryptography:

Definition 4 ($\mathcal{L}^\perp(\mathbf{B})$). Fixing $q, m, n \in \mathbb{N}$ and given a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, the m -dimensional lattice $\mathcal{L}^\perp(\mathbf{B})$ is defined as:

$$\mathcal{L}^\perp(\mathbf{B}) = \{\omega \in \mathbb{Z}^m \mid \mathbf{B}\omega \equiv 0 \pmod{q}\}.$$

Definition 5 ($\mathcal{L}(\mathbf{B}^T)$). Fixing $q, m, n \in \mathbb{N}$ and given a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, the m -dimensional lattice $\mathcal{L}(\mathbf{B}^T)$ is defined as:

$$\mathcal{L}(\mathbf{B}^T) = \{\mathbf{y} \in \mathbb{Z}^m \mid \exists s \in \mathbb{Z}^n, \text{ s.t. } \mathbf{y} \equiv \mathbf{B}^T \mathbf{s} \pmod{q}\}.$$

Alwen *et al.* have given an efficient algorithm to generate a random lattice along with its trapdoor basis:

Theorem 1 [3]. There is a $\mathcal{P.P.T}$ algorithm `TrapSamp` that, on input $1^n, 1^m, q$, with $q \geq 2$ and $m \geq 8n \log q$, outputs $(\mathbf{A}, \mathbf{T}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}^{m \times m}$ such that the distribution on \mathbf{A} is statistically close to $U(\mathbb{Z}_q^{n \times m})$, and with probability all but negligible in n :

1. the columns of \mathbf{T} form a basis of the lattice $\mathcal{L}^\perp(\mathbf{A})$;
2. $\|\mathbf{T}\| = O(n \log q)$ and $\|\tilde{\mathbf{T}}\| = O(\sqrt{n \log q})$.

Gentry, Peikert and Vaikuntanathan [10] focused on the applications of such short bases. Specifically, taking $q = \text{poly}(n)$, $m \geq 8n \log q$, $s = \omega(\sqrt{n \log q \log n})$, a family of one-way preimage-sampleable functions is defined in the following:

1. `GPVGen`(1^n):
 - (1) run $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapSamp}(1^n, 1^m, q)$;
 - (2) define the function: $f_{\mathbf{A}}(\mathbf{e}) = \mathbf{A}\mathbf{e} \pmod{q}$, with domain $\{\mathbf{e} \in \mathbb{Z}^m \mid \|\mathbf{e}\| \leq s\sqrt{m}\}$ and range \mathbb{Z}_q^n .
2. `SampleSIS`($\mathbf{A}, \mathbf{T}, s, \mathbf{u}$):
 - (1) compute some $\mathbf{t} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{t} \equiv \mathbf{u} \pmod{q}$ using standard linear algebra;
 - (2) sample $\mathbf{e} \leftarrow D_{\mathcal{L}^\perp(\mathbf{A})+\mathbf{t},s}$ using the trapdoor basis \mathbf{T} .

The above function is one-way if `GapSVP` $_\gamma$ is hard in the worst case for polynomial approximation factor γ [3].

Theorem 2 [12]. There is a $\mathcal{P.P.T}$ algorithm `SuperSamp` that, on input $1^n, 1^m, q$, with $q \geq 2$ and $m \geq n + 8n \log q$, and $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ whose columns span \mathbb{Z}_q^n , outputs $(\mathbf{A}, \mathbf{T}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}^{m \times m}$ such that $\mathbf{A}\mathbf{B}^T = 0 \pmod{q}$ and the distribution on \mathbf{A} is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$ subject to this condition. Moreover, with probability all but negligible in n :

1. the columns of \mathbf{T} form a basis of the lattice $\mathcal{L}^\perp(\mathbf{A})$;
2. $\|\tilde{\mathbf{T}}\| = O(\log n \cdot \sqrt{mn \log q})$.

We now describe the LWE problem. Fix a positive integer n , integers $m \geq n$ and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on \mathbb{R}^m . Define the following two distributions over $\mathbb{Z}_q^{n \times m} \times [0, q)^m$:

1. $\text{LWE}_{m,q,\chi}$ is the distribution obtained by choosing uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, sampling $\mathbf{e} \leftarrow \chi$, and outputting $(\mathbf{A}, \mathbf{A}^T \mathbf{s} + \mathbf{e} \pmod q)$.
2. $\text{U}_{m,q}$ is the distribution obtained by choosing uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and uniform $\mathbf{y} \in [0, q)^m$, and outputting (\mathbf{A}, \mathbf{y}) .

Formally, for m, q and χ that may depend on n , we say the $\text{LWE}_{m,q,\chi}$ problem is hard, if the following is negligible for any $\mathcal{P.P.T}$ algorithm \mathcal{D} :

$$|Pr[\mathbf{s} \leftarrow \mathbb{Z}_q^n; (\mathbf{A}, \mathbf{y}) \leftarrow \text{LWE}_{m,q,\chi}(\mathbf{s}) : \mathcal{D}(\mathbf{A}, \mathbf{y}) = 1] - Pr[(\mathbf{A}, \mathbf{y}) \leftarrow \text{U}_{m,q} : \mathcal{D}(\mathbf{A}, \mathbf{y}) = 1]|$$

The error distribution χ we will use in this paper is the discrete Gaussian distribution $D_{\mathbb{Z}^m, \alpha q}$. We write $\text{LWE}_{m,q,\alpha}(\mathbf{s})$ as an abbreviation for $\text{LWE}_{m,q,D_{\alpha q}}(\mathbf{s})$, and $\widehat{\text{LWE}}_{m,q,\alpha}(\mathbf{s})$ for $\text{LWE}_{m,q,D_{\mathbb{Z}^m, \alpha q}}(\mathbf{s})$, where $D_{\alpha q}$ is the continuous Gaussian distribution.

Lemma 1 [12]. *For any $m = m(n), q = q(n), \alpha = \alpha(n)$ satisfying $\alpha q = \omega(\sqrt{\log n})$, hardness of the $\text{LWE}_{m,q,\alpha}$ problem implies hardness of the $\widehat{\text{LWE}}_{m,q,\alpha\sqrt{2}}$ problem.*

3 A Fully Anonymous Group Signature with Verifier-Local Revocation

In this section, we present a lattice-based *group signature scheme with verifier-local revocation* (GS-VLR) holding full-anonymity and traceability as defined by a variant of the GKV scheme. Note that a lattice-based GS-VLR scheme has already been proposed by Langlois *et al.* [15] at PKC 2014. However, that scheme only holds some weaker insider-anonymity, and cannot be used in initializing our semi-generic construction. Now, we begin with recalling the notion of GS-VLR [7].

3.1 Group Signatures with Verifier-Local Revocation

Formally, a GS-VLR scheme $\mathcal{GS} = (\text{GKg}, \text{GSig}, \text{GVf})$ is a tuple of three poly-time algorithms:

1. $\text{GKg}(1^n, 1^N)$. The randomized key generation algorithm takes as input the security parameter $n \in \mathbb{N}$, and the group size $N \in \mathbb{N}$. It outputs a group public key gpk , all members' secret keys $gsk := (gsk_1, \dots, gsk_N)$, and all members' revocation tokens $grt := (grt_1, \dots, grt_N)$.
2. $\text{GSig}(gpk, gsk_i, m)$. The randomized signing algorithm takes as input the group public key gpk , the signing key gsk_i of member $i \in [N]$, and a message $m \in \{0, 1\}^*$. It outputs a signature σ .

3. $\text{GVf}(gpk, RL, m, \sigma)$. The deterministic verification algorithm takes as input the group public key gpk , a set of revocation tokens RL , a message m , and a candidate signature σ . It returns either 1 or 0. The latter indicates that either σ is not a valid signature, or the member who generated it has been revoked.

Correctness. For all $n, N \in \mathbb{N}$, all $(gpk, grt, gsk) \leftarrow [\text{GKg}(1^n, 1^N)]$, all $i \in [N]$, and all $m \in \{0, 1\}^*$, the following holds with overwhelming probability:

$$\text{GVf}(gpk, RL, m, \text{GSig}(gpk, gsk_i, m)) = 1 \iff grt_i \notin RL.$$

Implicit Tracing Algorithm. Given a valid message-signature pair (m, σ) with respect to some revocation list RL , the one possessing all revocation tokens grt traces by the following algorithm: for $i \in [N]$, run the verification algorithm $\text{GVf}(gpk, RL_i := \{grt_i\}, m, \sigma)$, and output the first index for which the verification algorithm outputs 0; otherwise output a symbol \perp . Apparently, if a GS-VLR scheme is correct, so is the above implicit tracing algorithm (the honestly generated signature will always trace to its originator).

Oracles. To formalize the security properties, the following oracles are used:

1. $\text{GSig}(\cdot, \cdot)$: for queries $(m, i) \in \{0, 1\}^* \times [N]$, it returns $\sigma \leftarrow \text{GSig}(gpk, gsk_i, m)$.
2. $\text{Corrupt}(\cdot)$: a corrupt set \mathcal{C} is initialized as empty; for queries $i \in [N]$, it responds with gsk_i , and add i into \mathcal{C} .
3. $\text{Revoke}(\cdot)$: for queries $i \in [N]$, it answers with grt_i .

Anonymity. In the *insider-anonymity* experiment as shown in Fig. 1, the adversary's goal is to determine which of two keys generated a signature. He is not given access to *either key or revocation token*. The deprivation of the challenge revocation tokens is necessary, if the GS-VLR scheme is correct. In contrast, the deprivation of the challenge signing keys may not be a must.

Actually, in the constructions proposed by Boneh *et al.* [7], the revocation token grt_i of some member i can be derived from its secret signing key gsk_i . This is an admirable feature on aspect of efficiency, however leaving room for improving security. Specifically, a stronger version of anonymity for GS-VLR, called *full-anonymity* can be considered.

Definition 6 (Full-Anonymity). A GS-VLR scheme \mathcal{GS} holds *full-anonymity* if for all $\mathcal{P.P.T}$ adversaries \mathcal{A} , the following advantage function is negligible in n :

$$\text{Adv}_{\mathcal{A}, \mathcal{GS}}^{\text{GS-VLR-full-anonymity}}(n) = |\Pr[1 \leftarrow \text{Expt}_{\mathcal{A}, \mathcal{GS}}^{\text{GS-VLR-full-anonymity}}(1^n, N)] - \frac{1}{2}|.$$

The full-anonymity differs from the insider-version in that the adversary as shown in Fig. 2 is equipped with all members' secret signing keys.

$\mathbf{Expt}_{\mathcal{A}, \mathcal{GS}}^{GS-VLR-insider-anonymity}(1^n, N)$ <hr/> 1: $(gpk, grt, gsk) \leftarrow \mathbf{GKg}(1^n, 1^N)$ 2: $(i_0, i_1, m, st) \leftarrow \mathcal{A}^{\mathbf{GSig}(\cdot, \cdot), \mathbf{Corrupt}(\cdot), \mathbf{Revoke}(\cdot)}(gpk)$ 3: $b \xleftarrow{\$} \{0, 1\}; \sigma \leftarrow \mathbf{GSig}(gpk, gsk_{i_0}, m)$ 4: $b' \leftarrow \mathcal{A}^{\mathbf{GSig}(\cdot, \cdot)}(st, \sigma)$ 5: return 1 if : 6: $b == b'$ 7: $i_0, i_1 \notin \mathcal{C}$ and were never queried to $\mathbf{Revoke}(\cdot)$ 8: else return 0
--

Fig. 1. Insider-anonymity for GS-VLR.

$\mathbf{Expt}_{\mathcal{A}, \mathcal{GS}}^{GS-VLR-full-anonymity}(1^n, N)$ <hr/> 1: $(gpk, grt, gsk) \leftarrow \mathbf{GKg}(1^n, 1^N)$ 2: $(i_0, i_1, m, st) \leftarrow \mathcal{A}^{\mathbf{Revoke}(\cdot)}(gpk, gsk)$ 3: $b \xleftarrow{\$} \{0, 1\}; \sigma \leftarrow \mathbf{GSig}(gpk, gsk_{i_0}, m)$ 4: $b' \leftarrow \mathcal{A}(st, \sigma)$ 5: return 1 if : 6: $b == b'$ 7: i_0, i_1 were never queried to $\mathbf{Revoke}(\cdot)$ 8: else return 0

Fig. 2. Full-anonymity for GS-VLR.

Traceability. In the traceability experiment as shown in Fig. 3, the adversary’s goal is to forge a signature that cannot be traced to one of unrevoked malicious members in his coalition using the implicit tracing algorithm.

$\mathbf{Expt}_{\mathcal{A}, \mathcal{GS}}^{GS-VLR-traceability}(1^n, N)$ <hr/> 1: $(gpk, grt, gsk) \leftarrow \mathbf{GKg}(1^n, 1^N)$ 2: $(m, \sigma, RL) \leftarrow \mathcal{A}^{\mathbf{Corrupt}(\cdot), \mathbf{GSig}(\cdot, \cdot)}(gpk, grt)$ 3: return 1 if : 4: $\mathbf{GVf}(gpk, RL, m, \sigma) = 1$ 5: σ traces to someone out of $\mathcal{C} \setminus RL$, or the implicit tracing fails 6: $\mathbf{GSig}(m, i)$ was never queried for $i \notin \mathcal{C}$ 7: else return 0

Fig. 3. Traceability for GS-VLR.

Definition 7 (Traceability). A GS-VLR scheme \mathcal{GS} holds traceability if for all P.P.T adversaries \mathcal{A} , the following advantage function is negligible in n :

$$Adv_{\mathcal{A}, \mathcal{GS}}^{GS-VLR-traceability}(n) = \Pr[1 \leftarrow \mathbf{Expt}_{\mathcal{A}, \mathcal{GS}}^{GS-VLR-traceability}(1^n, N)].$$

Note that if the verification algorithm satisfies that $\mathbf{GVf}(gpk, RL, m, \sigma) = 1 \iff \mathbf{GVf}(gpk, \{grt_i\}, m, \sigma) = 1$ for all $grt_i \in RL$, the success condition that σ traces to someone out of $\mathcal{C} \setminus RL$ can be equivalently modified as that σ traces to someone out of \mathcal{C} . This is because σ will never be traced to some $i^* \in (\mathcal{C} \cap RL)$, otherwise by the definition of the implicit tracing algorithm, we have $\mathbf{GVf}(gpk, \{grt_{i^*}\}, m, \sigma) = 0$ which contradicts with $\mathbf{GVf}(gpk, \{grt_{i^*}\}, m, \sigma) = 1$.

3.2 A Fully Anonymous GS-VLR Scheme from Lattice Assumptions

Let n be the security parameter, $q = \text{poly}(n)$, $m \geq 8n \log q$ and $s \geq C\sqrt{n \log q} \cdot \omega(\sqrt{\log m})$ be parameters of the system. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^n$ be a hash function, to be modeled as a random oracle. The GS-VLR scheme is demonstrated as follow:

1. $\text{GKg}(1^n, 1^N)$: for $i \in [N]$, compute $(\mathbf{B}_i, \mathbf{S}_i) \leftarrow \text{TrapSamp}(1^n, 1^m, q), (\mathbf{A}_i, \mathbf{T}_i) \leftarrow \text{SuperSamp}(1^n, 1^m, q, \mathbf{B}_i)$. Output $\text{gpk} := ((\mathbf{A}_i, \mathbf{B}_i))_{i=1}^N$ as the group public key, $\text{gsk} := (\mathbf{T}_i)_{i=1}^N$ as members' signing keys, $\text{grt} := (\mathbf{S}_i)_{i=1}^N$ as members' revocation tokens.
2. $\text{GSig}(\text{gpk}, \text{gsk}_i, m)$: it works exactly the same as the signing algorithm of the GKV scheme.
3. $\text{GVf}(\text{gpk}, RL, m, \sigma)$: parse the signature σ as $(\mathbf{r}, \mathbf{c}_1, \dots, \mathbf{c}_N, \pi)$. If π is not valid, output 0; for $i \in [N]$, calculate $\mathbf{h}_i \leftarrow H(m \parallel \mathbf{r} \parallel i)$, if the equation $\mathbf{A}_i \mathbf{c}_i \equiv \mathbf{h}_i \pmod{q}$ does not hold, output 0; for $\mathbf{S}_{i_\ell} \in RL$ ($\ell = 1, \dots, |RL|$), calculate $\mathbf{e}'_{i_\ell} \leftarrow \mathbf{S}_{i_\ell}^T \mathbf{c}_{i_\ell} \pmod{q}, \mathbf{e}_{i_\ell} \leftarrow \mathbf{S}_{i_\ell}^{T-1} \mathbf{e}'_{i_\ell}$, and if $\|\mathbf{e}_{i_\ell}\| \leq s\sqrt{m}$, output 0. Otherwise output 1.

The only difference between the GKV scheme and ours lies in the verification procedure. Namely, we incorporate the original GKV verification algorithm and open algorithm into our new verification algorithm.

For the correctness and security of our scheme, we have the following theorems.

Theorem 3. *Our GS-VLR scheme is correct.*

Theorem 4. *Let m, q, s be described as above. If $\text{LWE}_{m, q, \alpha}$ is hard for $\alpha = s/(q\sqrt{2})$, and GapSVP_γ is hard for $\gamma = O(n \log^4 n)$, and the proof system used is witness-indistinguishable, our GS-VLR scheme is fully anonymous and traceable.*

4 Hierarchical Group Signatures with Verifier-Local Revocation

In this section, we formalize the new notion of *hierarchical group signatures with verifier-local revocation* (HGS-VLR), and propose a semi-generic construction from GS-VLR.

4.1 Syntax and Correctness

We follow the notations from [26]. There are two types of parties: *signers* denoted as S_α for α in some index set \mathcal{I} and *managers* denoted as M_α for indices α described below. If a manager directly manages a set of signers $\{\alpha \mid \alpha \in \beta \subset \mathcal{I}\}$, we denote it by M_β ; if a manager directly manages a set of managers $\{M_{\beta_1}, \dots, M_{\beta_\ell}\}$, we denote it by M_γ where $\gamma = \{\beta_1, \dots, \beta_\ell\}$.

All parties are organized in a balanced tree \mathcal{T} of depth $\delta \in \mathbb{N}$, where signers are leaves and managers are inner nodes. For $i \in \{0, \dots, \delta\}$, let \mathcal{T}^i denote all nodes at depth i ; we denote all leaves by $\mathcal{L}(\mathcal{T})$ and the root by ρ . When there is no risk of confusion, we write α instead of M_α or S_α .

An HGS-VLR scheme $\mathcal{HGS} = (\text{HKg}, \text{HSig}, \text{HVf})$ consists of three poly-time algorithms:

1. $\text{HKg}(1^n, \mathcal{T})$. The randomized key generation algorithm takes as input the security parameter $n \in \mathbb{N}$, and a balanced tree \mathcal{T} of size polynomially bounded in n . It outputs a tuple of maps $(\text{hpk}, \text{hrt}, \text{hsk})$, where hpk and hrt associates each node $\alpha \in \mathcal{T}$ with a public value $\text{hpk}(\alpha)$ and a revocation token $\text{hrt}(\alpha)$, and hsk associates each leaf $\alpha \in \mathcal{L}(\mathcal{T})$ with a secret signing key $\text{hsk}(\alpha)$.
2. $\text{HSig}(\text{hpk}(\mathcal{T}), \text{hsk}(\alpha), m)$. The randomized signing algorithm takes as input the public map $\text{hpk}(\mathcal{T})$, a message $m \in \{0, 1\}^*$, and the secret signing key $\text{hsk}(\alpha)$ of some signer $\alpha \in \mathcal{L}(\mathcal{T})$, and returns a signature σ .
3. $\text{HVf}(\text{hpk}(\mathcal{T}), \text{RL}, m, \sigma)$. The deterministic verification algorithm takes as input the public map $\text{hpk}(\mathcal{T})$, a revocation list $\text{RL} \subset \text{hrt}(\mathcal{T})$ composed of the tokens associating with already revoked parties, a message m , and a candidate signature σ . It returns either 1 or 0, and the latter means either that σ is not a valid signature, or (at least) one on the path from the signer to the root (both are included) has been revoked.

The key generation algorithm HKg is run by some trusted key generator TKG , akin to the circumstance in HGS. The map $\text{hpk}(\mathcal{T})$ is made public, and each signer $\alpha \in \mathcal{L}(\mathcal{T})$ is given its secret signing key $\text{hsk}(\alpha)$. We initialize the public revocation list RL as empty, and any party $\alpha \in \mathcal{T}$ can be revoked by simply adding its revocation token $\text{hrt}(\alpha)$ into RL .

Correctness. An HGS-VLR scheme is *correct*, if for all $n \in \mathbb{N}$, all balanced trees \mathcal{T} of depth $\delta \in \mathbb{N}$ and size polynomially bounded in n , all $(\text{hpk}, \text{hrt}, \text{hsk}) \in [\text{HKg}(1^n, \mathcal{T})]$, all $m \in \{0, 1\}^*$, and all $\alpha \in \mathcal{L}(\mathcal{T})$, the following holds with overwhelming probability:

$$\text{HVf}(\text{hpk}(\mathcal{T}), \text{RL}, m, \text{HSig}(\text{hpk}(\mathcal{T}), \text{hsk}(\alpha), m)) = 1 \iff \{\text{hrt}(\gamma)\}_{\gamma \in \text{Ancestor}(\alpha)} \cap \text{RL} = \emptyset,$$

where $\text{Ancestor}(\alpha)$ denotes all nodes on the path from α to ρ with both included. We highlight that, if some manager $\beta \in (\mathcal{T} - \mathcal{L}(\mathcal{T}))$ is ever revoked by adding $\text{hrt}(\beta)$ into RL , all signers whom β (maybe indirectly) manages are no longer valid, even without adding their tokens into RL explicitly.

Implicit Tracing Algorithm. Each manager $\beta \in (\mathcal{T} - \mathcal{L}(\mathcal{T}))$ is given all revocation tokens $\{\text{hrt}(\gamma)\}_{\gamma \in \beta}$ of its direct children, to run an implicit tracing algorithm inherent to HGS-VLR. Specifically, given a valid message-signature pair (m, σ) , a manager β does the following:

1. For $\gamma \in \beta$, run $\text{HVf}(\text{hpk}(\mathcal{T}), \text{RL}_\gamma := \{\text{hrt}(\gamma)\}, m, \sigma)$;
2. Output the first index for which the verification algorithm says 0 and terminate; if the pair (m, σ) passes all verifications, output a symbol \perp .

This algorithm is correct, if the HGS-VLR scheme is correct as defined. Namely, given a valid message-signature pair (m, σ) honestly generated by some signer $\alpha \in \mathcal{L}(\mathcal{T})$, a *path-following tracing*, where some father node (initialized as the root) traces then passes the pair to some child, will always locate the signer α eventually.

To explain, let $\beta_0 := \rho \ni \beta_1 \ni \dots \ni \beta_\delta := \alpha$ be the path. If the HGS-VLR scheme is correct, then by definition $\text{HVf}(hpk(\mathcal{T}), RL, m, \sigma) = 1 \iff \{hrt(\beta_i)\}_{i=0}^\delta \cap RL = \phi$ holds with overwhelming probability; it follows that with overwhelming probability, $\text{HVf}(hpk(\mathcal{T}), \{hrt(\beta)\}, m, \sigma) = 0 \iff \beta = \beta_i$ for some $i \in \{0, 1, \dots, \delta\}$. Then for i from 0 to $\delta - 1$, the manager β_i will always trace (m, σ) to β_{i+1} independently with probability $(1 - \text{negl}(n))$, where $\text{negl}(n)$ is a negligible function with respect to n . Then the probability of locating α is $(1 - \text{negl}(n))^\delta$, which is still overwhelming since δ is polynomial in n .¹

4.2 Security Model

We formalize two security requirements for HGS-VLR, namely *full-anonymity* and *traceability* using the experiments as shown in Fig. 4 and Fig. 5 respectively. Overall, we use the framework of Bellare *et al.*, and thus these two properties are strong enough to capture all related security requirements, e.g., unforgeability, exculpability, collision resistance, framing, unlinkability and so on as argued by Bellare *et al.* [5]. To begin with, we specify the following oracles:

1. **HCorrupt**(\cdot): a corrupt set \mathcal{C} is initialized as empty; for queries $\alpha \in \mathcal{L}(\mathcal{T})$, it responds with $hsk(\alpha)$, and adds α into the set \mathcal{C} .
2. **HRevoke**(\cdot): for queries $\alpha \in \mathcal{T}$, it returns $hrt(\alpha)$.
3. **HSig**(\cdot, \cdot): for queries $(m, \alpha) \in \{0, 1\}^* \times \mathcal{L}(\mathcal{T})$, it returns $\sigma \leftarrow \text{HSig}(hpk(\mathcal{T}), hsk(\alpha), m)$.

Full-Anonymity. Assume that a message m has been signed by either $\alpha^{(0)}$ or $\alpha^{(1)} \in \mathcal{L}(\mathcal{T})$. Let B denote all nodes on paths from $\alpha^{(0)}$ and $\alpha^{(1)}$ up to their first common ancestor α_t , including $\alpha^{(0)}$ and $\alpha^{(1)}$ but excluding α_t . One having access to arbitrary element of $\{hrt(\beta)\}_{\beta \in B}$ can trivially determine who the signer is, if the HGS-VLR scheme is correct. Full-anonymity says that nobody can determine whether $\alpha^{(0)}$ or $\alpha^{(1)}$ signed the message without access to $\{hrt(\beta)\}_{\beta \in B}$, even if it is given all secret signing keys $hsk(\mathcal{L}(\mathcal{T}))$, and is allowed to select the challenge identities $\alpha^{(0)}$ and $\alpha^{(1)}$ as well as the challenge message m by itself.

Note that no generality is lost by having access to the oracle **HRevoke** only before σ is computed, since \mathcal{A} has decided on $\alpha^{(0)}$ and $\alpha^{(1)}$ and can obtain any $hrt(\alpha)$ with $\alpha \notin B$ before it receives σ . When \mathcal{T} is a depth one tree, the experiment in Fig. 4 reduces to the experiment in Fig. 2 for GS-VLR.

¹ The negligible functions might be different for different β_i , however, we express them uniformly by $\text{negl}(n)$ for simplicity, since both lead to a overwhelming probability.

<p>Expt$_{\mathcal{A}, \mathcal{HGS}}^{HGS-VLR-full-anonymity}(1^n, \mathcal{T})$</p> <hr style="border: 0.5px solid black;"/> <p>1 : $(hpk, hrt, hsk) \leftarrow \text{HKg}(1^n, \mathcal{T})$</p> <p>2 : $(\alpha^{(0)}, \alpha^{(1)}, m, st) \leftarrow \mathcal{A}^{\text{HRevoke}(\cdot)}(hpk(\mathcal{T}), hsk(\mathcal{L}(\mathcal{T})))$</p> <p>3 : $b \xleftarrow{\\$} \{0, 1\}; \sigma \leftarrow \text{HSig}(hpk(\mathcal{T}), hsk(\alpha^{(b)}), m)$</p> <p>4 : $b' \leftarrow \mathcal{A}(st, \sigma)$</p> <p>5 : return 1 if :</p> <p>6 : $b == b'$</p> <p>7 : HRevoke(β) was never queried for $\beta \in B$</p> <p>8 : else return 0</p>

Fig. 4. Full-anonymity for HGS-VLR.

Definition 8 (Full-Anonymity). An HGS-VLR scheme holds full-anonymity if for all $\mathcal{P.P.T}$ adversaries \mathcal{A} , the following advantage function is negligible in n :

$$\text{Adv}_{\mathcal{A}, \mathcal{HGS}}^{HGS-VLR-full-anonymity}(n) = |\Pr[1 \leftarrow \text{Expt}_{\mathcal{A}, \mathcal{HGS}}^{HGS-VLR-full-anonymity}(1^n, \mathcal{T})] - \frac{1}{2}|.$$

Traceability. In the experiment as shown in Fig. 5, the adversary \mathcal{A} is provided with all revocation tokens $hrt(\mathcal{T})$ and allowed to adaptively corrupt a coalition of signers (denoted by \mathcal{C}). To win, \mathcal{A} must come up with a valid message-signature pair (m, σ) with respect to some revocation list RL , and one of the following must hold: (a) all managers at the penultimate depth cannot figure out an identity by the implicit tracing algorithm; or (b) there exists one manager at the penultimate depth tracing to someone not in the coalition $\mathcal{C} \setminus RL$. For simplicity, let O denote the open results of all managers at the penultimate depth, and we formalize (a) and (b) into two expressions, namely $O = \{\perp\}$ and $O \cap (\mathcal{L}(\mathcal{T}) - \mathcal{C} \setminus RL) \neq \emptyset$ respectively.

Now we demonstrate the implications. Given some valid message-signature pair with respect to some revocation list, it is always feasible to figure out the actual (and unrevoked) generator while nobody will be framed. Specifically, if the path-following tracing does not fail, it will always locate the actual (and unrevoked) generator; if the path-following tracing fails somewhere, all managers at the penultimate depth can do a whole-depth tracing, and figure out some unrevoked malicious signer, while no honest signer will be traced.

Overall, when the path-following tracing fails, attacks against traceability are detected, or contradicting with the correctness; on the other hand, the whole-depth tracing stands for the capability to find out the attackers. Honestly, the latter is less efficient than the former, but it will merely be used if the punishment of misbehaving is heavily enough (this seems rather sound in the context of anonymous credit card systems). Note that such detect-and-punish paradigm is widely used in the e-cash setting, e.g., in solving the double-spending problem [24].

$\mathbf{Expt}_{\mathcal{A}, \mathcal{HGS}}^{HGS-VLR-traceability}(1^n, \mathcal{T})$ <hr style="border: 0.5px solid black;"/> 1 : $(hpk, hrt, hsk) \leftarrow \mathbf{HKg}(1^n, \mathcal{T})$ 2 : $(m, \sigma, RL) \leftarrow \mathcal{A}^{\mathbf{HCorrupt}(\cdot), \mathbf{HSig}(\cdot, \cdot)}(hpk(\mathcal{T}), hrt(\mathcal{T}))$ 3 : return 1 if : 4 : $\mathbf{HVf}(hpk(\mathcal{T}), RL, m, \sigma) = 1$ 5 : $O \cap (\mathcal{L}(\mathcal{T}) - \mathcal{C} \setminus RL) \neq \phi$, or $O = \{\perp\}$ 6 : $\mathbf{HSig}(m, \alpha)$ was never queried for $\alpha \notin \mathcal{C}$ 7 : else return 0
--

Fig. 5. Traceability for HGS-VLR.

The experiment above reduces to the experiment in Fig. 3 for GS-VLR, when \mathcal{T} is a depth one tree. Moreover, when nobody is corrupted, namely $\mathcal{C} = \phi$, the requirement of *unforgeability* is reflected.

Definition 9 (Traceability). *An HGS-VLR scheme holds traceability if for all $\mathcal{P.P.T}$ adversaries \mathcal{A} , the following advantage function is negligible in n :*

$$Adv_{\mathcal{A}, \mathcal{HGS}}^{HGS-VLR-traceability}(n) = \Pr[1 \leftarrow \mathbf{Expt}_{\mathcal{A}, \mathcal{HGS}}^{HGS-VLR-traceability}(1^n, \mathcal{T})].$$

4.3 A Semi-generic HGS-VLR Construction

As we will see, the construction is quite natural under our model, but definitely nontrivial. First, we reduce the full-anonymity of HGS-VLR to that of the underlying GS-VLR; however, similar reduction is not right when it comes to the insider-anonymity counterpart (we can similarly define insider-anonymity for HGS-VLR). Second, it essentially reflects a different designing paradigm from Wikström *et al.*'s, which adds the capability of revocation to the original notion. Now we show the semi-generic construction of HGS-VLR from GS-VLR.

Specifically, let \mathcal{T} be some balanced tree of depth δ , and let $\mathcal{GS} = (\mathbf{GKg}, \mathbf{GSig}, \mathbf{GVf})$ be the underlying GS-VLR. We construct the HGS-VLR scheme as following:

1. $\mathbf{HKg}(1^n, \mathcal{T})$: for $i \in \{0, \dots, \delta\}$, run $(gpk^i, grt^i, gsk^i) \leftarrow \mathbf{GKg}(1^n, 1^{|\mathcal{T}^i|})$, where $grt^i := \{grt_\beta\}_{\beta \in \mathcal{T}^i}$ and $gsk^i := \{gsk_\beta\}_{\beta \in \mathcal{T}^i}$. The public map hpk is defined as: $hpk(\rho) = \{gpk^i\}_{i=0}^\delta$, $hpk(\alpha) = \phi$ for $\alpha \in (\mathcal{T} - \rho)$; the secret map hsk is defined as: $hsk(\alpha) = \{gsk_\gamma\}_{\gamma \in \mathbf{Ancestor}(\alpha)}$ for $\alpha \in \mathcal{L}(\mathcal{T})$, where $\mathbf{Ancestor}(\alpha)$ denotes all nodes on the path from α to ρ with both included; $hrt(\beta) = grt_\beta$ for $\beta \in \mathcal{T}$.
2. $\mathbf{HSig}(hpk(\mathcal{T}), hsk(\alpha), m)$: for $i \in \{0, \dots, \delta\}$, run $\sigma_i \leftarrow \mathbf{GSig}(gpk^i, gsk_\gamma, m)$, where $\{\gamma\} = \mathbf{Ancestor}(\alpha) \cap \mathcal{T}^i$, and output the signature $\sigma := (\sigma_0, \dots, \sigma_\delta)$.

3. $\text{HVf}(hpk(\mathcal{T}), RL, m, \sigma)$: parse the candidate signature σ as $(\sigma_0, \dots, \sigma_\delta)$. For $i \in \{0, \dots, \delta\}$, run $\text{GVf}(gpk^i, RL_i, m, \sigma_i)$, where $RL_i := \{grt_\beta \in RL \mid \beta \in \mathcal{T}^i\}$ denotes all revocation tokens in RL associating with parties at depth i , and if $\text{GVf}(gpk^i, RL_i, m, \sigma_i) = 0$, terminate and output 0; if (m, σ) passes all verifications, output 1.

For the integrity of the revocation functionality, we generate both token and signing key for the root ρ . In other words, $hrt(\rho)$ exists only for revocation, while other tokens are used either in the implicit tracing (thus a manager β is given all children's tokens $\{hrt(\gamma)\}_{\gamma \in \beta}$), or removing someone out of the hierarchy by adding its token into RL . If there is no need for revoking the root, the scheme can be modified by simply dropping the group composed of the root.

For the correctness and security of our construction, we have the following theorems.

Theorem 5. *If the underlying GS-VLR is correct, the HGS-VLR scheme resulted from our semi-generic construction is also correct.*

Theorem 6. *The HGS-VLR scheme described above holds full-anonymity and traceability, if the underlying GS-VLR scheme holds full-anonymity and traceability.*

When we instantiate the semi-generic construction by the variant of the GKV scheme as shown in Sect. 3.2, a concrete construction from lattice assumptions is given. Overall, we regard all parties at the same depth as a group respectively. As for the HGS-VLR scheme, its anonymity is reduced to the anonymity of all groups; however, its traceability is reduced to the traceability of the group composed of all leaves. This leaves rooms for improving efficiency, in the meaning that for the first $(\delta - 1)$ groups, the underlying scheme may only holds anonymity and correctness.

5 Summary

The significance of this paper is embodied on two aspects: first, from HGS to HGS-VLR, we provide the former with efficient revocation approach by introducing and formalizing the latter new notion; second, in contrast with the generic HGS construction, we do not employ extra building blocks (e.g., an anonymous encryption scheme [11]) in our semi-generic HGS-VLR construction, and this somehow unifies the studies of HGS-VLR and GS-VLR. The expansion of signature's size is in proportion to the tree's depth. However, this won't bother much since the depth is usually small in applications. For future works, on one hand, it is desirable to depict a fully dynamic case by further adding the capability of dynamic joining; on the other hand, constructing efficient schemes in the lattice setting seems to be some long-term open problem, as reflected in the studies of GS.

Acknowledgement. The authors would like to thank the anonymous reviewers of ICICS 2018 for helpful comments.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, STOC 1996, pp. 99–108. ACM, New York. <https://doi.org/10.1145/237814.237838>
2. Ajtai, M.: The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 10–19. STOC 1998. ACM, New York (1998). <https://doi.org/10.1145/276698.276705>
3. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. *Theory Comput. Syst.* **48**(3), 535–553 (2011). <https://doi.org/10.1007/s00224-010-9278-3>
4. 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
5. 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
6. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: Domain-specific pseudonymous signatures for the german identity card. In: Gollmann, D., Freiling, F.C. (eds.) Information Security, pp. 104–119. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33383-5_7
7. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, pp. 168–177. ACM, New York (2004). <https://doi.org/10.1145/1030083.1030106>
8. Camenisch, J., Neven, G., Rückert, M.: Fully anonymous attribute tokens from lattices. In: Visconti, I., De Prisco, R. (eds.) Security and Cryptography for Networks, pp. 57–75. Springer, Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-32928-9_4
9. 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
10. Gentry, C., Vaikuntanathan, V., Peikert, C.: How to use a short basis: trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, 17–20 May 2008 (2008)
11. Goldwasser, S., Macali, S.: Probabilistic Encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)
12. Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 395–412. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_23
13. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
14. 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

15. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: Krawczyk, H. (ed.) PKC 2014, pp. 345–361. Springer, Heidelberg (2014). <https://doi.org/10.1007/s12204-017-1837-1>
16. 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
17. 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
18. Libert, B., Mouhartem, F., Nguyen, K.: A lattice-based group signature scheme with message-dependent opening. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 137–155. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_8
19. 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
20. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Lattice-based group signatures: achieving full dynamicity with ease. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) Applied Cryptography and Network Security, pp. 293–312. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_15
21. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Constant-size group signatures from \hat{A} lattices. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography - PKC 2018, pp. 58–88. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_3
22. Lysyanskaya, A., Ramzan, Z.: Group blind digital signatures: a scalable solution to electronic cash. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 184–197. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055483>
23. 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
24. Peter, W.: Digital Cash: Commerce on the Net. Academic Press, Cambridge (1996)
25. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
26. Trolin, M., Wikström, D.: Hierarchical group signatures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) Automata, Languages and Programming, pp. 446–458. Springer, Berlin Heidelberg (2005). https://doi.org/10.1007/11523468_37



Achieving Full Security for Lattice-Based Group Signatures with Verifier-Local Revocation

Maharage Nisansala Sevbandi Perera¹(✉) and Takeshi Koshihara²

¹ Graduate School of Science and Engineering,
Saitama University, Saitama, Japan
perera.m.n.s.119@ms.saitama-u.ac.jp

² Faculty of Education and Integrated Arts and Sciences,
Waseda University, Tokyo, Japan
tkoshihara@waseda.jp

Abstract. Even though Verifier-local revocation mechanism seems to be the most flexible revocation method that suits for any size of groups it could not reach strong security yet. Verifier-local revocation technique needs to update only the verifiers with revocation messages when a member is revoked while most of the revocation mechanisms require to re-initialize the group or track changes of the group. The first lattice-based group signature scheme with verifier-local revocability was suggested by Langlois, Ling, Nguyen, and Wang (PKC 2014). However, their scheme relies on a weaker security notion. On the other hand, Bellare, Micciancio, and Warinschi (EUROCRYPT 2003) proposed formal security definitions called full-anonymity and full-traceability for static groups. Achieving full-anonymity for schemes with verifier-local revocation is technically challenging because those schemes use a token system. This paper provides a scheme with verifier-local revocation that achieves the full-anonymity and full-traceability.

Keywords: Lattice-based group signatures · Verifier-local revocation
Full-anonymity · Full-Traceability

1 Introduction

In the setting of group signatures introduced by Chaum and van Heyst [9], group members can generate signatures for the group anonymously (anonymity). On the other hand, the group manager can extract the identity of the group member who created the signature (traceability). Thus, the original group signature scheme has two core requirements, anonymity and traceability. Later more requirements such as unlinkability, unforgeability, and framing resistance have been proposed. However, the precise meaning of those requirements not always clear and sometimes their meaning overlap each other. Bellare et al. [2] (BMW03 model) proposed strong and formal definitions for the core requirements of the group signatures with two security notions called, *full-anonymity*

and *full-traceability*. The full-anonymity and the full-traceability, which imply all the existing security notions provide a conceptual simplification since it requires to check only two security properties in a group signature scheme. However, the BMW03 model is for static groups, not for dynamic groups. In real-life almost all the group settings are stateless. Thus, member registration and member revocation requirements are essential when applying the group signature schemes in practice.

When a member is misbehaved, he should be punished. For instance, if a member issued a signature for an unnecessary document, he should be removed from the group. Member revocation in group signature schemes requires restricting members signing in future after revoking them. There are several member revocation methods. For instance, one revocation method is generating and distributing new keys for each member and verifiers or requesting each member to update their keys and generating the group public key newly. Since this requires to update all the unrevoked members and the verifiers, it is inconvenient to implement practically. Bresson et al. [5] suggested another revocation technique by extending the signing procedure of the scheme given in [8]. Their revocation method requires signers to prove at the zero-knowledge that his identity is not in the public list of revoked identities. However, this method causes the linear growth of the size of the group signatures with the number of revoked members. Thus it is a burden for the signers. Brickell [6] proposed a revocation method called Verifier-local revocation (VLR), which was subsequently formalized by Boneh et al. [4] in their scheme. VLR requires to pass revocation messages only to the verifiers when a member is revoked. In real-life scenarios, since the number of verifiers is much less than the number of members, passing messages only to the verifiers are efficient than any other revocation technique. Most of the group signature schemes (e.g., [3, 16]) operate in the bilinear map setting which will be insecure once quantum computers become a reality.

Lattice-based cryptography is the most prominent solution for the post-quantum cryptography. It provides provable security under worst-case hardness assumptions. Gorden et al. [11] suggested the first lattice-based group signature scheme. However, the sizes of both the group public key and the signature in their scheme increase with the number of members (N) (linear-barrier problem). Thus, it cannot apply to large groups. Then Camenisch et al. [7] presented a lattice-based group signature scheme with anonymous attribute token system, which still experiences the linear-barrier problem. Later, Languillaumie et al. [13] presented a scheme with a solution to the linear-barrier problem. However, the first three lattice-based group signature schemes follow LWE-based PKE (public-key encryption) scheme, and they are only for static groups.

Langlois et al. [14] proposed the first lattice-based group signature scheme which facilitates member revocation and free of LWE-based PKE. They have used VLR as the member revocation technique, and their scheme is more efficient while based on weaker security assumptions. In terms of security, their scheme satisfies a weaker security notion called *selfless-anonymity*. The VLR group signature schemes cannot employ the full-anonymity described in the BMW03 model directly because VLR group signature schemes use a token

system to manage member revocation. Thus, each member has a token other than their secret signing key. In the full anonymity game between a challenger and an adversary as described in the BMW03 model, all the member secret signing keys are given to the adversary at the beginning. In VLR group signature schemes, revocation tokens cannot be given to the adversary because he can identify the signer of a signature using tokens. Other than that, secret signing keys cannot be given to him because he can derive the revocation token from the secret signing keys.

The present lattice-based VLR group signature schemes raise a problem, that is whether it is possible to design a VLR lattice-based group signature scheme in the BMW03 model that achieves the full-anonymity.

1.1 Our Contribution

The lattice-based VLR group signature scheme in [14] relies on the selfless-anonymity. Stronger security for VLR schemes can be achieved in two ways. One approach is by using a restricted-version of full anonymity. The other process is changing the methods in the scheme. We provide a new group signature scheme that can achieve the full-anonymity using the second method.

The previous lattice-based group signatures failed to obtain the full-anonymity because anyone possessing revocation tokens can execute signature verification algorithm and confirm whether the relevant member created the signature or not. For instance, in the anonymity game between a challenger and an adversary, if the adversary knows the revocation tokens of the challenging indices, then he can execute `Verify` with revocation tokens he has. If `Verify` returns `Invalid`, then he knows that the owner of the revocation token generated the signature. Thus, this leads to an assumption that the verifiers should not see the revocation tokens, especially the challenging indices' revocation tokens. Based on this assumption, new security notions were proposed [18, 19]. However, none of them are as strong as full-anonymity because they do not provide all the revocation tokens to the adversary. Thus those security notions are restricted version of full-anonymity.

This paper suggests a scheme that can provide all the revocation tokens to the adversary even the challenged indices' revocation tokens. In original VLR schemes, when revoking a member, the group manager adds the revoking member's token into a list called revocation list (RL) and passes RL to the verifiers. Thus, `Verify` has an additional input RL, and the verifiers have to check whether the singer's revocation token is not in the list other than verifying the signature. We suggest a new revocation method for VLR schemes that the group manager has to sign each revocation token before adding to RL. On the other hand, at the signature verification, the verifier has to check whether the revocation tokens in the list are signed by the group manager other than checking the signer's revocation token is not in the list and signature is valid. Thus, even the adversary obtains any revocation token he cannot execute `Verify` because the adversary does not know the group manager's secret key. Now, we can apply the

full-anonymity for our VLR group signature scheme and provide all the member secret signing keys and revocation tokens including the challenging indices’ details to the adversary at the full-anonymity game.

As a result, we deliver a new lattice-based group signature scheme using VLR with new revocation and verification methods, that satisfies the full-anonymity.

2 Preliminaries

2.1 Notations

For any integer $k \geq 1$, we denote the set of integers $\{1, \dots, k\}$ by $[k]$. We denote matrices by bold upper-case letters such as \mathbf{A} , and vectors by bold lower-case letters, such as \mathbf{x} . We assume that all vectors are in column form. While the concatenation of matrices $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$, is denoted by $[\mathbf{A}|\mathbf{B}] \in \mathbb{R}^{n \times (m+k)}$ the concatenation of vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^k$ is denoted by $(\mathbf{x}||\mathbf{y}) \in \mathbb{R}^{m+k}$. If S is a finite set, $b \stackrel{\$}{\leftarrow} S$ means that b is chosen uniformly at random from S .

Throughout this paper, we present the security parameter as n and the maximum number of members in a group as $N = 2^\ell \in \text{poly}(n)$. We choose other parameters as in scheme [14] as given in Table 1.

Table 1. Parameters of the scheme

Parameter	Value or asymptotic bound
Modulus q	$\omega(n^2 \log n)$
Dimension m	$\geq 2n \log q$
Gaussian parameter σ	$\omega(\sqrt{n \log q \log n})$
Integer norm bound β	$\lceil \sigma \cdot \log m \rceil$ s.t. $(4\beta + 1)^2 \leq q$
Number of decomposition p	$\lceil \log \beta \rceil + 1$
Sequence of integers: $\beta_1, \beta_2, \beta_3, \dots, \beta_p$	$\beta_1 = \lceil \beta/2 \rceil; \beta_2 = \lceil (\beta - \beta_1)/2 \rceil;$ $\beta_3 = \lceil (\beta - \beta_1 - \beta_2)/2 \rceil; \dots; \beta_p = 1$
Number of protocol repetitions t	$\omega(\log n)$

Let $\mathcal{H}: \{0, 1\}^* \rightarrow \{1, 2, 3\}^t$, and $\mathcal{G}: \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times m}$ be hash functions, modeled as random oracles. We use one-time signature scheme $\mathcal{OTS} = (\text{OGen}, \text{OSign}, \text{Over})$, where OGen is the key generation algorithm of \mathcal{OTS} key pair $(\mathbf{ovk}, \mathbf{osk})$, OSign is signature generation and Over is signature verification functions.

2.2 Lattices

Let q be a prime and $\mathbf{B} = [\mathbf{b}_1 | \dots | \mathbf{b}_m] \in \mathbb{Z}_q^{r \times m}$ be linearly independent vectors in \mathbb{Z}_q^r . The r -dimensional lattice $\Lambda(\mathbf{B})$ for \mathbf{B} is defined as

$$\Lambda(\mathbf{B}) = \{\mathbf{y} \in \mathbb{Z}^r \mid \mathbf{y} \equiv \mathbf{B}\mathbf{x} \pmod q \text{ for some } \mathbf{x} \in \mathbb{Z}_q^m\},$$

which is the set of all linear combinations of columns of \mathbf{B} and m is the rank of \mathbf{B} .

We consider a discrete Gaussian distribution for a lattice. The Gaussian function centered in a vector \mathbf{c} with parameter $s > 0$ is defined as $\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|(\mathbf{x}-\mathbf{c})/s\|^2}$ and the corresponding probability density function proportional to $\rho_{s,\mathbf{c}}$ is defined as $D_{s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/s^n$ for all $\mathbf{x} \in \mathbb{R}^n$. The discrete Gaussian distribution with respect to a lattice Λ is defined as $D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = D_{s,\mathbf{c}}(\mathbf{x})/D_{s,\mathbf{c}}(\Lambda) = \rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(\Lambda)$ for all $\mathbf{x} \in \Lambda$. Since \mathbb{Z}^m is also a lattice, we can define a discrete Gaussian distribution for \mathbb{Z}^m . By $D_{\mathbb{Z}^m,\sigma}$, we denote the discrete Gaussian distribution for \mathbb{Z}^m around the origin with the standard deviation σ .

2.3 Lattice-Related Properties

The security of our scheme depends on the hardness of Learning With Errors (LWE) and two homogeneous and Inhomogeneous Short Integer Solution Problems (SIS and ISIS).

Definition 1 (LWE [17]). *LWE is parametrized by $n, m \geq 1, q \geq 2$, and χ . For $\mathbf{s} \in \mathbb{Z}_q^n$, the distribution $A_{s,\chi}$ is obtained by sampling $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and $e \leftarrow \chi$, and outputting the pair $(\mathbf{a}, \mathbf{a}^T \cdot \mathbf{s} + e)$.*

There are two versions of LWE problem, *Search-LWE* and *Decision-LWE*. While Search-LWE requires to find the secret \mathbf{s} , Decision-LWE requires to distinguish LWE samples and samples chosen according to the uniform distribution. We use the hardness of Decision-LWE problem.

For a prime power q , $b \geq \sqrt{n}\omega(\log n)$, and distribution χ , solving $LWE_{n,q,\chi}$ problem is at least as hard as solving $SIVP_\gamma$ (*Shortest Independent Vector Problem*), where $\gamma = \tilde{O}(nq/b)$ [21].

Definition 2 (SIS [17,21]). *Given m uniformly random vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$, forming the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a nonzero vector $\mathbf{x} \in \Lambda^\perp(\mathbf{A})$ such that $\|\mathbf{x}\| \leq \beta$ and $\mathbf{A}\mathbf{x} = 0 \pmod q$.*

Definition 3 (ISIS [14]). *Given m uniformly random vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$, forming the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a vector $\mathbf{x} \in \Lambda_{\mathbf{u}}^\perp(\mathbf{A})$ such that $\|\mathbf{x}\| \leq \beta$.*

For any $m, \beta = \text{poly}(n)$, and for any $q \geq \beta \cdot \omega(\sqrt{n} \log n)$, solving $SIS_{n,m,q,\beta}$ problem or $ISIS_{n,m,q,\beta}$ problem with non-negligible probability is at least as hard as solving $SIVP_\gamma$ problem, for some $\gamma = \tilde{O}(\beta\sqrt{n})$ [10].

2.4 Lattice-Related Algorithms

We use a randomized nearest-plane algorithm, called, SampleD [10,15] and preimage sampleable trapdoor functions (PSTFs) GenTrap [1,10,15].

- SampleD($\mathbf{R}, \mathbf{A}, \mathbf{u}, \sigma$) outputs $\mathbf{x} \in \mathbb{Z}^m$ sampled from the distribution $D_{\mathbb{Z}^m,\sigma}$ for any vector \mathbf{u} in the image of \mathbf{A} , a trapdoor \mathbf{R} and $\sigma = \omega(\sqrt{n} \log q \log n)$. The output \mathbf{x} should satisfy the condition $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \pmod q$.

- **GenTrap**(n, m, q) is an efficient randomized algorithm that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor matrix \mathbf{R} for given any integers $n \geq 1, q \geq 2$, and sufficiently large $m = 2n \log q$. The distribution of the output \mathbf{A} is $\text{negl}(n)$ -far from the uniform distribution.

2.5 VLR Group Signature

The VLR group signature scheme consists of three PPT algorithms [4] since the *implicit tracing algorithm* is used to trace the misbehaved users.

- **KeyGen**(n, N): This randomized PPT algorithm takes as inputs the security parameter n and the maximum number of group members N , and outputs a group public key \mathbf{gpk} , a vector of user secret keys $\mathbf{gsk} = (\mathbf{gsk}[0], \mathbf{gsk}[1], \dots, \mathbf{gsk}[N-1])$, and a vector of user revocation tokens $\mathbf{grt} = (\mathbf{grt}[0], \mathbf{grt}[1], \dots, \mathbf{grt}[N-1])$.
- **Sign**($\mathbf{gpk}, \mathbf{gsk}[d], M$): This randomized algorithm takes a secret signing key $\mathbf{gsk}[d]$ and a message $M \in \{0, 1\}^*$ as inputs and returns a signature Σ .
- **Verify**($\mathbf{gpk}, \text{RL}, \Sigma, M$): This deterministic algorithm verifies whether the given Σ is a valid signature using the given group public key \mathbf{gpk} and the message M . Then validates that the signer not being revoked using RL .

Implicit Tracing Algorithm: Any VLR group signature scheme has an *implicit tracing algorithm* that uses \mathbf{grt} as the tracing key and traces a signature to at least one group user who generated it. For an input valid signature Σ on a message M run **Verify**($\mathbf{gpk}, \text{RL}, \Sigma, M$) for each $i = 0, \dots, N-1$. It outputs the index of the first user for the verification algorithm returns invalid. The tracing algorithm fails if this algorithm verifies properly for all users on the given signature.

3 Definitions of the Security Notations

In this section first, we describe the core requirements presented in the original group signatures. Then we define the full-anonymity and the full-traceability delivered in the BMW03 model. Later, we describe the selfless-anonymity notion provided in the group signatures with VLR. Finally, we discuss the reasons for the difficulties of achieving the full-anonymity for the existing VLR group signature schemes.

Simply saying,

- *Anonymity* requires that no adversary without group manager’s key recovers the identity of the user from its signature, which is generated by one of the indices from two indistinguishable indices.
- *Traceability* requires that no adversary forges a signature that cannot be traced.

3.1 Full Anonymity and Full Traceability

Bellare et al. [2] delivered a standard security model (BMW03 model) for group signatures with two strong security properties, *full anonymity* and *full traceability*. Definitions of the *full anonymity* and *full traceability* are provided below.

Full Anonymity

The full-anonymity game between a challenger and an adversary is as follows. The adversary is strong as he has given all the member secret keys. At the beginning of the game, all the user secret keys **gsk** and the public key **gpk** are given to the adversary, and he can see the outcome of the tracing algorithm.

- **Initial Phase:** The challenger C runs **KeyGen** to obtain $(\mathbf{gpk}, \mathbf{gmsk}, \mathbf{gsk})$. Then gives $(\mathbf{gpk}, \mathbf{gsk})$ to the adversary A .
- **Query Phase:** The adversary A can access the opening oracle, which results with $\text{Open}(\mathbf{gmsk}, M, \Sigma)$ when queried with a message M and a signature Σ .
- **Challenge Phase:** The adversary A outputs a message M and two distinct identities i_0, i_1 . The challenger C selects a bit $b \xleftarrow{\$} \{0,1\}$, generates a signature Σ^* , and sends to the adversary A . The adversary still can query the opening oracle except the signature challenged.
- **Guessing Phase:** Finally, A outputs a bit b' , the guess of b . If $b' = b$, then the adversary A wins.

Definition 4. Let A be an adversary against the anonymity of a group signature scheme GS . The advantage of A in the above full-anonymity game is

$$\mathbf{Adv}_{GS,A}^{anon}(n, N) = |\Pr[\mathbf{Exp}_{GS,A}^{anon}(n, N) = 1] - 1/2|.$$

A group signature scheme is full-anonymous if $\mathbf{Adv}_{GS,A}^{anon}$ is negligible.

Full Traceability

As explained in [2] the group public key **gpk** and the group manager's secret key **gmsk** are given to the adversary A at the beginning of the game, and the adversary A makes queries as the following game.

- **Initial Phase:** The challenger C runs **KeyGen** to obtain $(\mathbf{gpk}, \mathbf{gmsk}, \mathbf{gsk})$. Then gives **gpk** and **gmsk** to the adversary A and sets $U \leftarrow \emptyset$.
- **Query Phase:** The adversary A can do the following queries.
 1. Signing: The adversary A requests a signature for any message M and user index i , and the challenger C returns $\Sigma = \text{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], M)$.
 2. Corruption: The adversary A queries for the secret key of any user i . The challenger C adds i to U and returns $\mathbf{gsk}[i]$.
- **Challenge Phase:** A outputs a message M^* and a signature Σ^* .
- The forgery adversary A wins if the followings are true.
 1. Σ^* is accepted as a valid signature on the message M^* .

2. Σ^* traces to some user outside the coalition U or tracing algorithm fails.
3. Σ^* not obtained by signing on M^* .

Definition 5. Let A be an adversary against the traceability of a group signature scheme GS . The advantage of A in the above full-traceability game is

$$\mathbf{Adv}_{GS,A}^{trace}(n, N) = \Pr[\mathbf{Exp}_{GS,A}^{trace}(n, N) = 1].$$

A group signature scheme is full-traceable if $\mathbf{Adv}_{GS,A}^{trace}$ is negligible.

3.2 Selfless-Anonymity

Selfless-anonymity is a relaxed anonymity, and it differs from the full-anonymity by the limitations it has. The selfless-anonymity provides none of the member secret keys to the adversary, but only the group public key is given. However, even with these weaknesses, the selfless-anonymity facilitates any member to determine whether his secret signing key is used to generate a particular signature if he forgets whether he signed the message.

The selfless-anonymity game between a challenger and an adversary is as follows.

The adversary in the selfless-anonymity game is weaker than the adversary in the full anonymity game since the adversary has not given any secret key in the selfless-anonymity game. The adversary has to determine which of the two adaptively chosen keys generated the challenging signature.

- **Initial Phase:** The challenger C runs KeyGen to obtain $(\mathbf{gpk}, \mathbf{gsk}, \mathbf{grt})$. Then gives \mathbf{gpk} to the adversary A .
- **Query Phase:** The adversary A can make the following queries.
 1. **Signing:** The adversary A requests a signature for any message $M \in \{0, 1\}^*$ with any user index i , and C returns $\Sigma = \mathbf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], M)$.
 2. **Corruption:** The adversary A queries for the secret key of any user i , and the challenger C returns $\mathbf{gsk}[i]$.
 3. **Revocation:** The adversary A queries for the revocation token of any user i , and the challenger C returns $\mathbf{grt}[i]$.
- **Challenge Phase:** The adversary A outputs a message M^* and two distinct identities i_0, i_1 , such that A did not make the corruption or revocation queries for i_0, i_1 . The challenger C selects a bit $b \xleftarrow{\$} \{0, 1\}$, computes signature $\Sigma^* = \mathbf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i_b], M^*)$ for i_b , and sends the challenging signature Σ^* to the adversary A .
- **Restricted Queries:** Even after the challenge phase the adversary A can make queries but with following restrictions.
 - **Signing:** The adversary A can query as before.
 - **Corruption:** The adversary A cannot query for i_0 or i_1 .
 - **Revocation:** The adversary A cannot query for i_0 or i_1 .
- **Guessing Phase:** Finally, the adversary A outputs a bit b' , the guess of b . If $b' = b$, then A wins.

Definition 6. Let A be an adversary against the anonymity of a VLR group signature scheme DGS . The advantage of A in the above selfless-anonymity game is

$$Adv_{DGS,A}^{anon}(n, N) = |\Pr[Exp_{DGS,A}^{anon}(n, N) = 1] - 1/2|.$$

A VLR group signature scheme is selfless-anonymous if $Adv_{DGS,A}^{anon}$ is negligible.

3.3 Difficulties of Achieving the Full-Anonymity for VLR Schemes

The full-anonymity is suggested for static groups. Thus, members have only secret signing keys. Even the secret signing key is used to generate signatures, by using the secret signing keys nobody can guess the signer. But the members in VLR schemes have another secret attribute called revocation token. Revealing revocation tokens to the outsiders makes the scheme insecure. For instance, if an adversary knows the user i_0 's revocation token $grt[i_0]$, then the adversary can confirm whether the user i_0 generated the given signature or not by executing `Verify` by replacing `RL` with $grt[i_0]$ as depicted in Fig. 1. According to the full-anonymity game in Fig. 1 if Σ_b is generated by user i_0 , then `Verify` return `Res` as Invalid for i_0 . Thus it confirms that user i_0 generated the signature. Moreover, since VLR group signatures derive the revocation tokens from the secret signing keys, the selfless-anonymity also restricts revealing the secret signing keys.

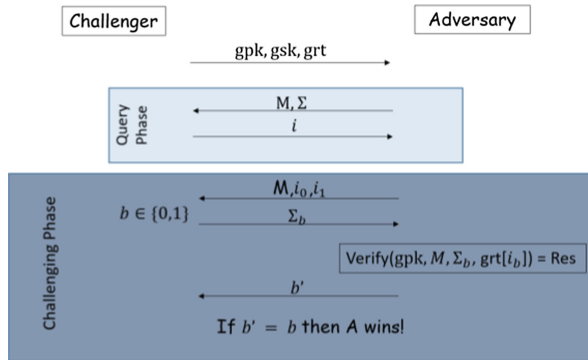


Fig. 1. Full anonymity for VLR schemes

Because of these reasons, to obtain stronger security for VLR group signature schemes, we need a restricted version of full anonymity or new scheme with different methods.

4 New Lattice-Based VLR Scheme

The new scheme requests the group manager to sign revoking member's token before adding to the revocation list `RL`. Thus the group manager signs the revoking member's revocation token grt using the group manager secret key $gmsk$.

Accordingly, at the signature verification, the verifier has to check whether the revocation tokens in RL are signed by the group manager. For this, the verifier executes `Verify` with the group manager’s public key. Because of this reason an adversary who knows the revocation token of any member i cannot replace RL in `Verify(gpk, M, Σ, RL)` with the i ’s revocation token $\mathbf{grt}[i]$ and check whether the user i generated the signature or not. The signature verification algorithm rejects verifying the given signature because the adversary is providing a revocation token which is not signed by the group manager.

In the full-anonymity game depicted in Fig. 1 when the adversary tries to execute `Verify` with the revocation token of i_0 and i_1 he gets Invalid as the response in both cases because he fails to provide tokens with the group manager’s signature. Thus, the adversary cannot understand the signer of the given signature. Therefore, the new scheme can employ the full-anonymity by giving all the members’ secret signing keys and tokens to the adversary.

4.1 Description of the Scheme

We use the scheme in [14] as the base and construct our new scheme as follows.

Key Generation: This randomized algorithm $\text{KeyGen}(n, N)$ works as below.

1. Run PPT algorithm $\text{GenTrap}(n, m, q)$ to get $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ and a trapdoor \mathbf{T}_A .
2. Sample $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\mathbf{A}_i^b \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for each $b \in \{0, 1\}$ and $i \in [\ell]$.
3. Set the matrix $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1^0 | \mathbf{A}_1^1 | \dots | \mathbf{A}_\ell^0 | \mathbf{A}_\ell^1] \in \mathbb{Z}_q^{n \times (2\ell+1)m}$.
4. Run $\text{GenTrap}(n, m, q)$ to obtain $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor \mathbf{T}_B .
5. For each group member select a ℓ -bit string as the index d and generate secret signing keys and revocation tokens as below.
 - (a) Let $d = d[1] \dots d[\ell] \in \{0, 1\}^\ell$ be the binary representation of index d .
 - (b) Sample vectors $\mathbf{x}_1^{d[1]}, \dots, \mathbf{x}_\ell^{d[\ell]} \leftarrow D_{\mathbb{Z}^m, \sigma}$.
 - (c) Compute $\mathbf{z} = \sum_{i=1}^\ell \mathbf{A}_i^{d[i]} \cdot \mathbf{x}_i^{d[i]} \bmod q$.
 - (d) Get $\mathbf{x}_0 \in \mathbb{Z}^m \leftarrow \text{SampleD}(\mathbf{T}_A, \mathbf{A}_0, \mathbf{u} - \mathbf{z}, \sigma)$.
 - (e) Let $\mathbf{x}_1^{1-d[1]}, \dots, \mathbf{x}_\ell^{1-d[\ell]}$ be zero vectors $\mathbf{0}^m$.
 - (f) Define $\mathbf{x} = (\mathbf{x}_0 || \mathbf{x}_1^0 || \mathbf{x}_1^1 || \dots || \mathbf{x}_\ell^0 || \mathbf{x}_\ell^1) \in \mathbb{Z}^{(2\ell+1)m}$.
If $\|\mathbf{x}\|_\infty \leq \beta$ then proceed else repeat from (b).
 - (g) Let the user secret signing key be $\mathbf{gsk}[d] = \mathbf{x}^{(d)}$ and revocation token be $\mathbf{grt}[d] = \mathbf{A}_0 \cdot \mathbf{x}_0 \in \mathbb{Z}_q^n$.

Finally we obtain, the group public key $\mathbf{gpk} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$, the group manager’s secret key $\mathbf{gmsk} = \mathbf{T}_B$, the group manager’s public key $\mathbf{gmpk} = \mathbf{B}$, group members’ secret signing keys $\mathbf{gsk} = (\mathbf{gsk}[0], \mathbf{gsk}[1], \dots, \mathbf{gsk}[N - 1])$, and their revocation tokens $\mathbf{grt} = (\mathbf{grt}[0], \mathbf{grt}[1], \dots, \mathbf{grt}[N - 1])$.

Signing: The randomized algorithm $\text{Sign}(\mathbf{gpk}, \mathbf{gsk}, M)$ generates Σ on a message M as follows.

1. Generate a one-time-signature \mathcal{OTS} key pair $(\mathbf{ovk}, \mathbf{osk})$ using OGen .
2. Sample $\rho \xleftarrow{\$} \{0, 1\}^n$, let $\mathbf{V} = \mathcal{G}(\mathbf{A}, \mathbf{u}, M, \rho) \in \mathbb{Z}_q^{m \times n}$.

3. Sample $\mathbf{e} \leftarrow \chi^m$.
4. Compute $\mathbf{v} = \mathbf{V} \cdot (\mathbf{A}_0 \cdot \mathbf{x}_0) + \mathbf{e} \pmod q$ ($\|\mathbf{e}\|_\infty \leq \beta$ with overwhelming probability and $(\mathbf{A}_0 \cdot \mathbf{x}_0)$ is the revocation token \mathbf{grt} of user i).
5. Repeat the zero knowledge interactive protocol of the commitment described in Sect. 4.2 $t = \omega(\log n)$ times with the public parameter $(\mathbf{A}, \mathbf{u}, \mathbf{V}, \mathbf{v})$ and prover's witness (\mathbf{x}, \mathbf{e}) to make the soundness error negligible and proof that user is certified. Then make it non-interactive using the Fiat-Shamir heuristic as a triple, $\Pi = (\{CMT^{(k)}\}_{k=1}^t, CH, \{RSP^{(k)}\}_{k=1}^t)$, where $CH = (\{Ch^{(k)}\}_{k=1}^t) = \mathcal{H}(M, \mathbf{A}, \mathbf{u}, \mathbf{V}, \mathbf{v}, \{CMT^{(k)}\}_{k=1}^t) \in \{1, 2, 3\}^t$.
6. Compute $\mathcal{OTS}; sig = \text{OSig}(\mathbf{osk}, \Pi)$.
7. Output signature $\Sigma = (\mathbf{ovk}, M, \rho, \mathbf{v}, \Pi, sig)$.

Verification: $\text{Verify}(\mathbf{gpk}, M, \Sigma, RL = \{\{\mathbf{u}_i\}_i\})$ verifies the given signature Σ is valid on the given message M and signer is a valid member as follows.

1. Parse the signature Σ as $(\mathbf{ovk}, M, \rho, \mathbf{v}, \Pi, sig)$.
2. If $\text{Over}(\mathbf{ovk}, \Pi, sig) = 0$ then return 0.
3. Get $\mathbf{V} = \mathcal{G}(\mathbf{A}, \mathbf{u}, M, \rho) \in \mathbb{Z}_q^{m \times n}$.
4. Parse Π as $(\{CMT^{(k)}\}_{k=1}^t, \{Ch^{(k)}\}_{k=1}^t, \{RSP^{(k)}\}_{k=1}^t)$.
5. If $(Ch^{(1)}, \dots, Ch^{(t)}) \neq \mathcal{H}(M, \mathbf{A}, \mathbf{u}, \mathbf{V}, \mathbf{v}, \{CMT^{(k)}\}_{k=1}^t)$ return 0 else proceed.
6. For $k = 1$ to t run the verification steps of the commitment scheme to validate $RSP^{(k)}$ with respect to $CMT^{(k)}$ and $Ch^{(k)}$. If any of the conditions fails then output invalid and hold.
7. For each $\mathbf{u}_i \in RL$,
 - (a) Parse \mathbf{u}_i as $(\mathbf{grt}_i, \Sigma_{rt_i})$.
 - (b) Check whether \mathbf{grt}_i is signed by the group manager by executing $\text{Verify}(\mathbf{gmpk}, \mathbf{grt}_i, \Sigma_{rt_i})$, where \mathbf{gmpk} is the group manager's public key. If $\text{Verify}(\mathbf{gmpk}, \mathbf{grt}_i, \Sigma_{rt_i})$ returns Invalid then return Invalid.
 - (c) Compute $\mathbf{e}'_i = \mathbf{v} - \mathbf{V} \cdot \mathbf{grt}_i \pmod q$ to check whether there exists an index i such that $\|\mathbf{e}'_i\|_\infty \leq \beta$. If so return invalid.
8. Return valid.

Revoke: The algorithm $\text{Revoke}(\mathbf{gpk}, \mathbf{gmsk}, \mathbf{grt}[i], RL)$ takes the group manager's secret key \mathbf{gmsk} , revoking member's revocation token $\mathbf{grt}[i]$, and latest revocation list RL and proceeds as follows.

1. Generate a signature for the revoking token as $\Sigma_{rt_i} = \text{Sign}(\mathbf{gmsk}, \mathbf{grt}[i])$.
2. Add revoking token and generated signature to RL , $RL \leftarrow RL \cup (\mathbf{grt}_i, \Sigma_{rt_i})$.
3. Return RL .

4.2 The Underlying ZKAoK for the Group Signature Scheme

Zero-Knowledge Interactive Protocol is the main building block of the scheme as it allows a signer to argue that he is a certified group member who has a valid secret key and who has not been revoked.

Let COM be the statistically hiding and computationally binding commitment scheme described in [12].

Our scheme can be seen as an adaptation of [14]. Thus we can use the protocol described in [14]. We use matrix $\mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1^0|\mathbf{A}_1^1|\dots|\mathbf{A}_\ell^0|\mathbf{A}_\ell^1] \in \mathbb{Z}_q^{n \times (2\ell+1)m}$, $\mathbf{V} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{u} \in \mathbb{Z}_q^n$, and $\mathbf{v} \in \mathbb{Z}_q^m$ as the public parameters. The witness of the prover is the vector $\mathbf{x}^{(d)} = (\mathbf{x}_0|\mathbf{x}_1^0|\mathbf{x}_1^1|\dots|\mathbf{x}_\ell^0|\mathbf{x}_\ell^1) \in \Sigma^{(2\ell+1)m}$ for some $d \in \{0, 1\}^\ell$ and vector $\mathbf{e} \in \mathbb{Z}^m$. While keeping prover's identity d in secret he has to convince the verifier that,

1. $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \pmod q$ and d is hidden in $\mathbf{x}^{(d)}$.
2. $\|\mathbf{e}\|_\infty \leq \beta$ and $\mathbf{V} \cdot (\mathbf{A}_0 \cdot \mathbf{x}_0) + \mathbf{e} = \mathbf{v} \pmod q$.

5 Analysis of the Scheme

This paper provides a new scheme that satisfies the full-anonymity. However, the restricted versions of full-anonymity called almost-full anonymity [19] and dynamical-almost-full anonymity [18] are efficient than the proposed scheme because those schemes do not require the group manager to sign member revoking tokens. Moreover, in the selfless-anonymity, any user can check whether he created a particular signature or not. But in the proposed scheme this is not possible since the users do not know the group manager's secret key. However, in terms of security, the new scheme is much stronger than any other security applied for VLR schemes.

5.1 Correctness

For all \mathbf{gpk} , \mathbf{gmsk} , \mathbf{gmpk} , \mathbf{gsk} , and \mathbf{grt} ,

$\text{Verify}(\mathbf{gpk}, M, \text{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], M), RL) = \text{Valid} \iff \mathbf{grt}[i] \notin RL$ and

For all $(\mathbf{grt}_i, \Sigma_{rt_i})$ in RL, $\text{Verify}(\mathbf{gmpk}, \mathbf{grt}_i, \Sigma_{rt_i}) = \text{Valid}$.

Verify in the proposed scheme only accepts signatures generated on given messages and which are only generated by active members. If the revocation token of the signer is in RL, then his signature is not accepted by Verify. Similarly Sign also checks whether the signer can satisfy those requirements. The underlying interactive protocol confirms that only active members can generate signatures and signers have to possess valid secret signing key.

5.2 Anonymity

Theorem 1. *In the random oracle model, the proposed scheme is full anonymous based on the hardness of Decision-LWE $_{n,q,\chi}$ problem.*

Proof. We define a sequence of games conducted between a challenger C and an adversary A , where the advantage of the adversary is negligible in the last game. Game 0 is the original full-anonymity game which provides all the members' secret signing keys and revocation tokens to the adversary at the beginning. The adversary can request the index of the signer by giving a signature. We prove that the games are indistinguishable, based on \mathcal{OTS} , the zero-knowledge property of

the underlying argument system, and the hardness of the *Decision-LWE* $_{n,q,\chi}$ problem. Game 4 is the last game which is independent of the bit $b \in \{0, 1\}$.

Game 0: The challenger C runs $\text{KeyGen}(1^n, 1^N)$ to obtain the group public key $\mathbf{gpk} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$, the group manager's secret key $\mathbf{gmsk} = \mathbf{T}_{\mathbf{B}}$, the group manager's public key $\mathbf{gmpk} = \mathbf{B}$, group members' secret signing keys $\mathbf{gsk} = (\mathbf{gsk}[0], \mathbf{gsk}[1], \dots, \mathbf{gsk}[N-1])$, and their revocation tokens $\mathbf{grt} = (\mathbf{grt}[0], \mathbf{grt}[1], \dots, \mathbf{grt}[N-1])$. The challenger C gives the group public key \mathbf{gpk} and all the group members' secret keys \mathbf{gsk} and revocation tokens \mathbf{grt} to the adversary A . In the query phase, A can request to reveal index of the signer for any signature. In the challenge phase, the adversary A sends two indices (i_0, i_1) together with a message M^* and the challenger C generates and sends back the challenging signature $\Sigma^* = (\mathbf{ovk}, M^*, \rho, \mathbf{v}, \Pi, sig)$ for a random bit $b \leftarrow \{0, 1\}$. The adversary's goal is to identify which index is used to generate the challenging signature. A returns b' . If $b' = b$ then the experiment returns 1 or 0 otherwise.

Game 1: In this game, the challenger C makes a slight modification with respect to Game 0. In the real experiment (Game 0) the one-time key pair $(\mathbf{ovk}, \mathbf{osk})$ is generated at the signature generation. In this game, C generates the one-time key pair $(\mathbf{ovk}^*, \mathbf{osk}^*)$ at the beginning of the game. If the adversary A accesses the opening oracle with a valid signature $\Sigma = (\mathbf{ovk}, M, \rho, \mathbf{v}, \Pi, sig)$, where $\mathbf{ovk} = \mathbf{ovk}^*$, C returns a random bit and aborts. However, A comes up with a signature Σ , where $\mathbf{ovk} = \mathbf{ovk}^*$ contradicts the strong unforgeability of \mathcal{OTS} , and since \mathbf{ovk}^* is independent of the adversary's view, the probability of $\mathbf{ovk} = \mathbf{ovk}^*$ is negligible. Even after seeing the challenging signature if A comes up with a valid signature Σ where $\mathbf{ovk} = \mathbf{ovk}^*$, then sig is a forged one-time signature, which defeats the strong unforgeability of \mathcal{OTS} . Thus, we assume that A does not request for opening of a valid signature with \mathbf{ovk}^* and the challenger aborting the game is negligible.

Game 2: In this game, without honestly generating the legitimate non-interactive proof Π , the challenger C simulates the proof Π^* without using the witness. C invokes the simulator for each $k \in [t]$ and then programs the random oracle \mathcal{H} accordingly. The challenging signature $\Sigma^* = (\mathbf{ovk}^*, M^*, \rho, \mathbf{v}, \Pi^*, sig)$ is statistically close to the challenging signature in the previous game because the argument system is statistically zero-knowledge. Thus Game 2 is indistinguishable from Game 1.

Game 3: In this game, the challenger C replaces the original revocation token by a vector sampled uniformly random. The original game has $\mathbf{v} = \mathbf{V} \cdot \mathbf{grt}[i_b] + \mathbf{e} \bmod q$, where \mathbf{V} is uniformly random over $\mathbb{Z}_q^{m \times n}$ and \mathbf{e} is sampled from the error distribution χ . In this game C samples a vector $\mathbf{t} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$ uniformly and computes $\mathbf{v} = \mathbf{V} \cdot \mathbf{t} + \mathbf{e} \bmod q$. The challenger C replaces only the revocation token $\mathbf{grt}[i_b]$ with \mathbf{t} . The rest of the game is same as Game 2. Thus, the two games are statistically indistinguishable.

Game 4: Game 3 has $\mathbf{v} = \mathbf{V} \cdot \mathbf{t} + \mathbf{e}_1 \bmod q$. In this game the challenger C makes \mathbf{v} truly uniform by sampling $\mathbf{y} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^m$ and setting $\mathbf{v} = \mathbf{y}$. Thus,

C makes revocation token totally independent of the bit b . In Game 3, (\mathbf{V}, \mathbf{v}) pair is a proper $LWE_{n,q,\chi}$ instance. Thus, the distribution of the pair (\mathbf{V}, \mathbf{v}) is computationally close to the uniform distribution over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. Game 3 and Game 4 are indistinguishable, under the assumption of the hardness of $LWE_{n,q,\chi}$ problem. If the adversary can distinguish \mathbf{v} from \mathbf{y} , then he can solve Decision-LWE problem.

Hence, these games prove that the new scheme is secure with full anonymity.

5.3 Traceability

Theorem 2. *Based on the hardness of $SIS_{n,(\ell+1)\cdot m,q,2\beta}^\infty$ problem, the proposed scheme is traceable, in the random oracle model.*

We construct a PPT algorithm \mathcal{F} that solves SIS problem with non-negligible probability. The forgery \mathcal{F} is given the verification key (\mathbf{A}, \mathbf{u}) and then he generates the key pair $(\mathbf{B}, \mathbf{T}_B)$. The forgery \mathcal{F} passes $\mathbf{gpk} = (\mathbf{A}, \mathbf{u}, \mathbf{B})$ and $\mathbf{gmsk} = \mathbf{T}_B$ and responds to the A 's queries as follow.

- **Signatures queries:** If A queries signature of user d on a random message M , then \mathcal{F} returns simulated $\Sigma = \text{Sign}(\mathbf{gpk}, \mathbf{gsk}[d], M)$.
- **Corruption queries:** The corruption set CU is initially set to be empty. If A queries the secret key of any user d , then \mathcal{F} adds d to the set CU and returns $\mathbf{gsk}[d]$.
- Queries to the random oracles \mathcal{H}, \mathcal{G} are handled by consistently returning uniformly random values in $\{1, 2, 3\}^t$. For each $k \leq q_{\mathcal{H}}$, we let r_k denote the answer to the k -th query.

Finally, A outputs a message M^* , revocation data RL^* and a non-trivial forged signature Σ^* , which satisfies the requirements of the traceability game, where Σ^* such that $\text{Verify}(\mathbf{gpk}, M^*, \Sigma^*, RL^*) = \text{Valid}$ and implicit tracing algorithm fails, or returns a user index j^* outside of the coalition $CU \setminus RL^*$.

\mathcal{F} exploits the forgery as below.

We require that the adversary A always queries \mathcal{H} on input $(M^*, \mathbf{A}, \mathbf{u}, \mathbf{V}^*, \mathbf{v}^*, \{CMT^{(k)}\}_{k=1}^t)$. As a result, with probability at least $\epsilon - 3^{-t}$, there exists certain $\kappa^* \leq q_{\mathcal{H}}$ such that the κ^* -th oracle queries involve the tuple $(M^*, \mathbf{A}, \mathbf{u}, \mathbf{V}^*, \mathbf{v}^*, \{CMT^{(k)}\}_{k=1}^t)$. For any fixed κ^* run A many times and input as in the original run. For each repeated run, A returns same output $r'_{\kappa^*}, \dots, r'_{\kappa^*-1}$ for the first $\kappa^* - 1$ queries as in initial run and from the κ^* -th query onwards return fresh random values $r'_{\kappa^*}, \dots, r'_{q_{\mathcal{H}}} \stackrel{\$}{\leftarrow} \{1, 2, 3\}^t$. The forking lemma [[20], Lemma 7] implies that, with probability larger than $1/2$, algorithm \mathcal{F} can obtain a 3-fork involving tuple $(M^*, \mathbf{A}, \mathbf{u}, \mathbf{V}^*, \mathbf{v}^*, \{CMT^{(k)}\}_{k=1}^t)$ after less than $32 \cdot q_{\mathcal{H}} / (\epsilon - 3^{-t})$ executions of A . Let the responses of \mathcal{F} with respect to the 3-fork branches be

$$r_{\kappa^*}^{(1)} = (Ch_1^{(1)}, \dots, Ch_t^{(1)}); r_{\kappa^*}^{(2)} = (Ch_1^{(2)}, \dots, Ch_t^{(2)}); r_{\kappa^*}^{(3)} = (Ch_1^{(3)}, \dots, Ch_t^{(3)}).$$

A simple calculation shows that $Pr[\exists j \in \{1, \dots, t\} : \{Ch_i^{(1)}, Ch_i^{(2)}, Ch_i^{(3)}\} = \{1, 2, 3\}] = 1 - (7/9)^t$.

Under the condition of the existence of such index i , one parses the 3 forgeries corresponding to the fork branches to obtain $(RSP_i^{(1)}, RSP_i^{(2)}, RSP_i^{(3)})$.

Then by using the knowledge extractor ζ of the underlying argument system, we can extract vectors (\mathbf{y}, \mathbf{e}) . These vectors satisfy the followings.

1. $\mathbf{y} = (\mathbf{y}_0 \| \mathbf{y}_1^0 \| \mathbf{y}_1^1 \| \dots \| \mathbf{y}_\ell^0 \| \mathbf{y}_\ell^1)$ for some $d \in \{0, 1\}^\ell$, and $\mathbf{A} \cdot \mathbf{y} = \mathbf{u} \pmod q$.
2. $\|\mathbf{e}^*\|_\infty \leq \beta$ and $\mathbf{V}^* \cdot (\mathbf{A}_0 \cdot \mathbf{y}_0) + \mathbf{e}^* = \mathbf{v}^* \pmod q$.

Remaining proof is same as the proof given in [14]. Thus finally, we can obtain a vector, which is a valid solution to the SIS problem. This concludes the proof of traceability.

6 Conclusion

This paper provides a new scheme with new methods for member revocation and signature verifications. As a result, the proposed scheme was able to achieve the full-anonymity becoming the first lattice-based group signature scheme with VLR that achieves the full-anonymity in comparison with known lattice-based group signature schemes. However, the group manager has to sign every revoking members' s token. This leads to an open problem because the security of the scheme depends on the trust of the group manager. If the group manager's information is revealed, then the scheme is not secure.

Acknowledgments. This work is supported in part by JSPS Grant-in-Aids for Scientific Research (A) JP16H01705 and for Scientific Research (B) JP17H01695.

References

1. Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., Wee, H.: Functional encryption for threshold functions (or fuzzy IBE) from lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 280–297. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_17
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
3. 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
4. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM-CCS 2004, pp. 168–177. ACM (2004)
5. Bresson, E., Stern, J.: Efficient revocation in group signatures. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_15

6. Brickell, E.: An efficient protocol for anonymously providing assurance of the container of the private key. Trusted Comp. Group, April 2003 (2003, submitted)
7. 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
8. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052252>
9. 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
10. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: ACM 2008, pp. 197–206. ACM (2008)
11. Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 395–412. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_23
12. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_23
13. 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
14. 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
15. 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
16. 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
17. Peikert, C.: A decade of lattice cryptography. *Found. Trends Theoret. Comput. Sci.* **10**(4), 283–424 (2016). <https://doi.org/10.1561/04000000074>
18. Perera, M.N.S., Koshiha, T.: Achieving almost-full security for lattice-based fully dynamic group signatures with verifier-local revocation. In: ISPEC 2018. LNCS (2018, to Appear)
19. Perera, M.N.S., Koshiha, T.: Fully dynamic group signature scheme with member registration and verifier-local revocation. In: ICMC 2018. Mathematics and Computing (2018, to Appear)
20. Pointcheval, D., Vaudenay, S.: On provable security for digital signature algorithms. Ecole Normale Supérieure (Paris). Laboratoire d’Informatique (1996)
21. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005, pp. 84–93. ACM Press (2005)



Towards Practical Lattice-Based One-Time Linkable Ring Signatures

Carsten Baum¹, Huang Lin^{2(✉)}, and Sabine Oechsner³

¹ Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
`carsten.baum@biu.ac.il`

² ASTRI Security Lab, Hong Kong, China
`linhuang@astri.org`

³ Department of Computer Science, Aarhus University, Aarhus, Denmark
`oechsner@cs.au.dk`

Abstract. Ring signatures, as introduced by Rivest, Shamir, and Tauman (Asiacrypt '01), allow to generate a signature for a message on behalf of an ad-hoc set of parties. To sign a message, only the public keys must be known and these can be generated independently. It is furthermore not possible to identify the actual signer based on the signature. Ring signatures have recently gained attention due to their applicability in the construction of practical anonymous cryptocurrencies, where they are used to secure transactions while hiding the identity of the actual spender. To be applicable in that setting, ring signatures must allow to determine when a party signed multiple transactions, which is done using a property called linkability.

This work presents a linkable ring signature scheme constructed from a lattice-based collision-resistant hash function. We follow the idea of existing schemes which are secure based on the hardness of the discrete logarithm problem, but adapt and optimize ours to the lattice setting. In comparison to other designs for (lattice-based) linkable ring signatures, our approach avoids the standard solution for achieving linkability, which involves proofs about correct evaluation of a pseudorandom function using heavy zero-knowledge machinery.

1 Introduction

Digital signatures are one of the most important concepts in the area of cryptography. They permit a party to generate a key pair (SK, PK) , give PK to the public and add certain information Ω - called the signature - to a message

C. Baum—Supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

S. Oechsner—This work has been supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No. 669255 (MPCPRO). Part of work done while visiting NTT Secure Platform Laboratories.

m . Ω is derived using the private (or signing) key SK and later allows a verifier, equipped with the public verification key PK , to attest that the signer indeed generated Ω for this specific message m . Verification is done in a way such that only a party who possesses certain secret information that only the signer has, namely the secret signing key SK , can generate a valid signature for PK .

Ring signatures, which were first suggested by Rivest, Shamir, and Tauman [40], relax the condition of having exactly one pair (SK, PK) for signing and verification to a certain extent. They allow a party among a set of N participants to sign a message on behalf of all of them. Here it is crucial that the verifier cannot identify the party that signed the message, while nobody outside of the N participants should be able to sign a message as if he was a participant himself. In comparison to group signatures, the set of parties does not need to be known ahead of time, but only when the signature is generated. Therefore, no key-generation algorithm which generates correlated randomness for all N parties needs to be involved and the rings can be set up ad-hoc¹.

For such a ring signature, each signer could issue an arbitrary number of signatures. Fujisaki and Suzuki introduced the notion of traceable ring signatures [17], where the signer signs a message with respect to a list of ring members and a public issue such as an election. There is a public procedure to determine whether two signatures come from one signer, i.e., the signer is linked if a signer signs the same message with respect to the same list of ring member and same issue twice [16]. A related idea is so-called *linkable ring signatures*, in which case the true signer will be linked when he signs two messages (different or identical) with respect to the same ring. In a more restricted version of linkable ring signatures, *one-time linkable ring signatures*, a signer is linked as soon as he reveals two signatures. This property has proven to be vital in the construction of cryptocurrencies, such as to prevent double spending attacks and to preserve the anonymity of a spender since the address or the respective secret key in the design of the anonymous cryptocurrency is supposed to be one-time [37].

1.1 Related Work

Lattice-Based Signature Schemes. The line of work on lattice-based signature schemes was, to the best of our knowledge, initiated by Goldreich et al. [19], while the first practical construction was based on NTRU [22]. A scheme that fits into this line of work is the provably secure construction due to Gentry et al., also called hash-and-sign [18]. This approach, where the signing key is a secret trapdoor which is used to sample a short lattice vector, was further developed in [9, 15]. A different direction, called Fiat Shamir with Aborts, was first explored by Lyubashevsky [28, 29]. Very efficient signature schemes such as Tesla [21] and Dilithium [14] have been designed within this framework.

¹ We relax this a bit and assume that there exists a CRS which is known to all parties and which allows them to derive their respective key pairs (SK, PK) .

(Linkable) Ring Signature Schemes. There exists a wealth of literature on ring signature and linkable ring signature schemes such as [6, 16, 17, 27, 40] and we only list some of the relevant works here. However, the above mentioned signature schemes have a signature size that is linearly dependent on the number of users N in the ring. The Groth-Kohlweiss framework [20] is based on homomorphic commitments and provides a ring signature scheme with a logarithmic signature size. Franklin and Zhang [16] propose a general framework for linkable ring signatures. They extend the “PRF made public” paradigm by Bellare and Goldwasser [5] in order to provide linkability by combining a PRF evaluation of the secret key with a NIZK proof of correct evaluation. The smallest ring signatures to date have constant signature size and are based on accumulators. The construction by Dodis et al. [13] uses accumulators based on the strong RSA assumption, while Nguyen’s [36] relies on pairing-based cryptography. There exists also a linkable version of [13] by Tsang and Wei [42] that retains the constant-sized signatures. There exist candidates for post-quantum ring signature schemes such as hash-based [12, 23] or multi-variate-quadratic-equation based constructions [35]. Neither of them provide linkability in their current form, but they can potentially be extended to do so.

Lattice-Based Ring Signature Schemes. Lattice-based ring signatures were first introduced explicitly through the work of Brakerski and Tauman-Kalai [10] who proposed a general framework for ring signatures in the standard model and showed how to instantiate it based on the SIS assumption. The resulting signatures have size $O(mN)$ for message length m and ring size N . Subsequently, Wang and Sun [43] proposed two ring signatures schemes from the SIS assumption in the random oracle and standard model, respectively, both of linear signature size. The first ring signature scheme based on the LWE assumption was proposed by Melchor et al. [33] and is an extension of [28] to the ring signature setting. Like the previous schemes, it yields signatures of linear size. Recently, Libert et al. [25] proposed the first lattice-based ring signature scheme with only logarithmic signature size using a Merkle-tree based construction.

Concurrent Work. In concurrent work, Torres et al. [41] present a construction that is very similar to ours. When comparing the actual parameters of both, we have a larger size of the public keys, but compare favorably in the signature size.

1.2 Our Contribution

We present a lattice-based linkable ring signature scheme based on the Module-SIS and Module-LWE problem. Our scheme has a signature size which is linear in N . It is therefore asymptotically less efficient than e.g. [12, 23, 25]. However, we show that in terms of signature size our construction outperforms or performs as good as [12, 25] for comparable security levels for ring sizes $N \gtrsim 128$ and beats [23] for rings of small size. A comparison can be found in Table 1 below.

Table 1. Comparison with existing work

	[25]	[12] Sponge/Davies-Meyer	[23]	Our work
Size of PK	0.5 KB	32 B	32 B	8 KB
Size ($N = 8$)	1.44 MB	766/477 KB	148 KB	82.5 KB
Size ($N = 32$)	2.29 MB	1200/719 KB	216 KB	305.7 KB
Size ($N = 128$)	3.14 MB	1.59/0.94 MB	285 KB	1.17 MB

The authors of [12] present two different, highly optimized constructions of ring signatures in their work. We mention numbers for both to allow for fair comparison (outperforming one of the two for $N = 128$). We want to stress that using known techniques [14] and by choosing parameters more aggressively it is possible to reduce the public key and signature size in our setting further, but such optimizations are beyond the scope of this work. Furthermore, [12, 23, 25] are not linkable in their current form, so one can expect a further increase in their proof size to compute a linkability tag. Though our work only outperforms [23] for small ($N \leq 20$) ring sizes, this is exactly the range that cryptocurrencies need: the recommended ring size of the most popular cryptocurrency using linkable ring signatures, Monero, at the time of writing was $N = 5$. As mentioned before, using [14], would make it possible to reduce the ring signature size further to also outperform [23] for $N \lesssim 64$.

1.3 Technical Overview

As mentioned before, the standard approach for transforming a ring signature scheme into a linkable ring signature scheme, following Franklin and Zhang [16], is to add a PRF evaluation of the signer’s secret key to the signature, as well as a zero-knowledge proof of correct evaluation of the PRF under one of the secret keys corresponding to the public keys. This generic approach applies to any ring signature scheme and was explored for lattice-based PRFs in [25, 26, 44]. However, such proofs come with quite a substantial overhead. Our construction instead follows the approach of Liu et al. [27] that avoids this technique. The main observation is that the signer in their scheme has two “public” keys: One that is published before signature generation as part of the ring of signers, and the other one that is appended to each signature. Hence, another “public key” under different public parameters that corresponds to the signer’s secret signing key can be used as linkability tag. Since both kinds of public keys share the same algebraic structure, the two “public keys” of the signer, i.e. the actual public key and the linkability tag, can be tied together without appending another non-interactive zero-knowledge proof to the signature.

Since our construction will be based on the (Module-)SIS and (Module-)LWE problem, the public keys of the parties are of the form $PK = \mathbf{A}\mathbf{r}$ for secret key \mathbf{r} and public matrix \mathbf{A} . Linkability will be ensured by providing linkability tags $I = \mathbf{B}\mathbf{r}$ for another public matrix \mathbf{B} . Interestingly, the reason why our

construction achieves only one-time linkability is inherent in this approach: any evaluation $\mathbf{B}\mathbf{r}$ leaks information about \mathbf{r} . If a fresh matrix \mathbf{B} is generated for each ring, then a malicious party can receive more leakage on \mathbf{r} than intended and hence may be able to recover the signer's secret key.

In order to obtain more efficient lattice-based (linkable) ring signatures, it may be tempting to try to instantiate current sublinear-size ring signatures in the lattice setting. Note, however, that this is far from trivial, as these solutions are specifically tailored to a certain assumption like Dodis et al.'s accumulator-based ring signatures [13], or suffer from the well-known problem that hard lattice assumptions do not provide enough algebraic structure to support existing sublinear approaches based on homomorphic operations like that of Groth and Kohlweiss [20].

Paper Organization

In Sect. 2 we will introduce some definitions and lemmas concerning lattice-based constructions which we will need throughout this work. Moreover, we will give definitions for linkable ring signatures (following previous work). Section 3 contains the construction and security statements. The main parts of the proofs are deferred to Appendix A, whereas we discuss the practicality of our scheme in Sect. 4. In this Section, we also provide a sample parameter set for our construction together with estimates for the size of signatures.

2 Preliminaries

We will use $[N]$ as shorthand for the set $\{1, \dots, N\}$. Let R be the cyclotomic ring $R = \mathbb{Z}[X]/\langle X^\nu + 1 \rangle$, where $\nu = 2^p$ and $p \in \mathbb{N}^+$. Let q be an odd prime and define $R_q = \mathbb{Z}_q[X]/\langle X^\nu + 1 \rangle$. Here \mathbb{Z}_q denotes the integers modulo q , which will be represented as elements from the interval $[-\frac{q-1}{2}, \frac{q-1}{2}]$. For $f = \sum_i f_i X^i \in R$, the norms of f are defined as

$$l_1 : \|f\|_1 = \sum_i |f_i|, \quad l_2 : \|f\|_2 = \left(\sum_i |f_i|^2 \right)^{1/2}, \quad l_\infty : \|f\|_\infty = \max_i |f_i|.$$

If $f \in R_q$, then we will represent each coset from \mathbb{Z}_q with its unique representative from the aforementioned interval and consider the norm of the obtained \mathbb{Z} -vector. Let S_β denote the set of elements $x \in R$ with l_∞ -norm at most β . Let $\mathbf{0}_v \in \mathbb{Z}^{v \times v}$ and $\mathbf{I}_v \in \mathbb{Z}^{v \times v}$ denote the zero and identity matrix over \mathbb{Z} .

Remark 1. We use the following standard relations among different l -norms of a vector in R as defined above:

1. If $f, g \in R$ such that $\|f\|_\infty \leq \beta$, $\|g\|_1 \leq \gamma$, then $\|fg\|_\infty \leq \beta\gamma$.
2. If $f \in R$, $\mathbf{g} \in R^v$ satisfy that $\|f\|_2 \leq \beta$, $\|\mathbf{g}\|_\infty \leq \gamma$, then $\|f\mathbf{g}\|_2 \leq \sqrt{v}\beta\gamma$.

We require a subset D of R_q which consists of short invertible elements such that the difference of any two distinct elements from this set is also invertible. It was shown in [32] that as long as q is a prime that satisfies $q \equiv 17 \pmod{32}$ and $q > 2^{20}$, then the set $D = \{d \in R_q \mid \|d\|_\infty \leq 1, \|d\|_1 \leq \kappa\}$ satisfies this requirement. We use \bar{D} to denote the set of values $D + D$ excluding 0.

2.1 Normal Distribution and Rejection Sampling

The continuous normal distribution over \mathbb{R}^ν centered at $\mathbf{u} \in \mathbb{R}^\nu$ with standard deviation σ has probability density function

$$\rho_{\mathbf{u},\sigma}^\nu(\mathbf{x}) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{-\|\mathbf{x} - \mathbf{u}\|_2^2}{2\sigma^2}\right)$$

The *discrete normal distribution* over R^ν centered at $\mathbf{u} \in R^\nu$ with standard deviation σ is given by the distribution function (for all $\mathbf{x} \in R^\nu$)

$$\mathcal{N}_{\mathbf{u},\sigma}(\mathbf{x}) = \rho_{\mathbf{u},\sigma}^{\nu,\nu}(\mathbf{x}) / \rho_{\sigma}^{\nu,\nu}(R^\nu),$$

where we omit the subscript \mathbf{u} when it is zero. We use the following standard tail-bound due to Banaszczyk:

Lemma 1. *Let $\mathcal{N}_{\mathbf{u},\sigma}$ be defined as above. Then*

$$\Pr [\|\mathbf{z}\|_2 > 2\sigma\sqrt{v\nu} | \mathbf{z} \leftarrow \mathcal{N}_{\sigma}^v] < 2^{-v\nu}$$

For our ring signature scheme, we use rejection sampling to hide the secret signing key. The basic idea of rejection sampling is to abort the protocol with a certain probability such that the distribution of the response is independent of the secret input. We adopt the rejection sampling lemma from [29]:

Lemma 2. *Let V be a subset of R^ν in such that all elements have $\|\cdot\|_2$ -norms less than T , $\sigma \in \mathbb{R}$ such that $\sigma = \omega(T\sqrt{\log(v\nu)})$, and $h : V \rightarrow \mathbb{R}$ be a probability distribution. Then there exists an $M = O(1)$ such that the output distribution of the following two algorithms \mathcal{A} , \mathcal{S} is within statistical distance $2^{-\omega(\log(v\nu))}/M$:*

A:

1. $\mathbf{u} \leftarrow h$
2. $\mathbf{z} \leftarrow \mathcal{N}_{\mathbf{u},\sigma}^v$
3. output (\mathbf{u}, \mathbf{z}) with probability $\min\left(\frac{1}{M} \frac{\mathcal{N}_{\sigma}^v(\mathbf{z})}{\mathcal{N}_{\mathbf{u},\sigma}^v(\mathbf{z})}, 1\right)$

S:

1. $\mathbf{u} \leftarrow h$
2. $\mathbf{z} \leftarrow \mathcal{N}_{\sigma}^v$
3. output (\mathbf{u}, \mathbf{z}) with probability $1/M$

Moreover, the probability that \mathcal{A} outputs a value is at least $\frac{1-2^{-\omega(\log(v\nu))}}{M}$.

In [29], the author remarks that if $\sigma = \alpha T, \alpha > 0$ and $M = \exp(12/\alpha + 1/(2\alpha^2))$ then the output of both algorithms will be within statistical distance $2^{-100}/M$ and \mathcal{A} will output a value with probability at least $\frac{1-2^{-100}}{M}$. As an example, assume that we want the signing algorithm to succeed in each iteration with probability $1/3$, i.e. we want to set $M = 3$. Then following the reasoning in [29], we can set $\sigma = 11 \cdot T$. This means that the output of the signing algorithm is indistinguishable from the simulator except with probability $\approx 2^{-98}$, which we deem sufficient for our application.

2.2 Module-SIS and Module-LWE

The security of our linkable ring signature scheme will be based on the hardness of two problems, Module-SIS and Module-LWE [24]. These problems are variants of the well-known SIS [1] and LWE [39] problems, but over modules that are defined over polynomial rings. This is a generalized version of the Ring-SIS and Ring-LWE problems [30,31,38]. Using Module-lattice assumptions comes with two advantages: (i) while they are a generalization of ideal-lattice assumptions, they still retain some structure which is necessary to construct a large space of short, invertible elements which is necessary for our construction; and (ii) there is evidence that module lattices of larger rank are less prone to certain attacks than ideal-lattices [3,8].

The homogeneous Module-SIS problem consists of finding a vector \mathbf{r} of small norm such that $\mathbf{A}\mathbf{r} = 0$ for a given, structured matrix \mathbf{A} .

Definition 1 (MSIS $_{h,v,t}$). *Given $\mathbf{A} \leftarrow R_q^{h \times v}$, find $\mathbf{r} \in R^v$ such that $\mathbf{A}\mathbf{r} = 0$ and $0 < \|\mathbf{r}\|_2 \leq t$.*

Our scheme also uses the Decisional Module-LWE problem. In D-MLWE, the problem consists of distinguishing noisy linear equations from random.

Definition 2 (D-MLWE $_{h,v,\beta}$). *Let $\mathbf{A} \leftarrow R_q^{h \times v}$. Then distinguish the distributions*

$$(\mathbf{A}, \mathbf{A}\mathbf{r}) \text{ and } (\mathbf{A}, \mathbf{u})$$

where $\mathbf{r} \leftarrow S_\beta^v$ and $\mathbf{u} \leftarrow R_q^h$.

Here, we use a special instance of the Module-LWE problem where the secret has the same distribution as the noise².

If two samples (with different matrices, but same secret vector \mathbf{r}) are issued by the challenger, then this can still be related to a D-MLWE instance but with different parameters, as the following proposition shows.

Proposition 1. *Let $\mathbf{A}, \mathbf{B} \leftarrow R_q^{h \times v}$, $\mathbf{r} \leftarrow S_\beta^v$ and $\mathbf{c}, \mathbf{d} \leftarrow R_q^h$. Then*

$$(\mathbf{A}, \mathbf{A}\mathbf{r}, \mathbf{B}, \mathbf{B}\mathbf{r}) \approx_c (\mathbf{A}, \mathbf{c}, \mathbf{B}, \mathbf{d})$$

given the D-MLWE $_{2h,v,\beta}$ -problem is hard.

Proof. Consider the matrices $\mathbf{E} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$, and $\mathbf{E}\mathbf{r} = \begin{bmatrix} \mathbf{A}\mathbf{r} \\ \mathbf{B}\mathbf{r} \end{bmatrix}$. Then distinguishing the above distributions is equivalent to distinguishing

$$(\mathbf{E}, \mathbf{E}\mathbf{r}) \approx_c \left(\mathbf{E}, \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \right)$$

This is the definition of the D-MLWE $_{2h,v,\beta}$ problem. □

² This equivalent formulation is possible in our setting, as only one LWE sample will be issued per secret. The definition might seem unusual at first, as one regularly defines the LWE distribution as $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. We can use the following transformation, which is well-known: note that the given equation is equivalent to writing $\mathbf{A}\mathbf{s}_1 + \mathbf{I}_h\mathbf{s}_2$ instead. By aligning this into a single matrix product of \mathbf{A}' with $(\mathbf{s}_1|\mathbf{s}_2)$ and multiplying the resulting challenge with a uniformly random $r \in R_q$, we obtain Definition 2.

Our construction will moreover rely on a third problem, namely the Search Module-LWE problem. It can be seen as an inhomogeneous MSIS instance where the target is known to have a short preimage under \mathbf{A} .

Definition 3 (S-MLWE $_{h,v,\beta}$). *Sample a uniformly random $\mathbf{r} \leftarrow S_\beta^v$. Given $(\mathbf{A} \leftarrow R_q^{h \times v}, \mathbf{s} = \mathbf{A}\mathbf{r})$ find $\mathbf{r}' \in R^v$ such that $\mathbf{A}\mathbf{r}' = \mathbf{s}$ and $0 < \|\mathbf{r}'\|_\infty \leq \beta$.*

Fixing h, v, β of an S-MLWE-instance, it is easy to see that any algorithm \mathcal{A} that solves S-MLWE-instances can also solve D-MLWE-instances with the same parameters in comparable time and with similar probability. For the converse direction, Langlois and Stehlé [24] showed that, for certain parameter sets, S-MLWE can be reduced to D-MLWE.

2.3 Linkable Ring Signatures

The formal syntax and security model of linkable ring signatures, sometimes also called linkable spontaneous anonymous group signatures, can be found in [17, 27]. Definitions of linkable ring signatures with adaptation to the cryptocurrency scenario can be found in [37]. Our definitions are in the spirit of [17, 20, 27].

Definition 4 (Linkable Ring Signature). *A linkable ring signature scheme consists of five algorithms:*

- Setup**(1^λ): *Generates and outputs public parameters PP available to all users.*
- KGen**(PP): *Generates a public key PK and a private signing key SK .*
- Sign** $_{PP,SK_\ell}(m, L)$: *Outputs a signature Ω on the message $m \in \{0, 1\}^*$ with respect to the ring $L = (PK_1, \dots, PK_N)$. Here, (PK_ℓ, SK_ℓ) is a valid key pair output by **KGen**(PP), and $PK_\ell \in L$.*
- Vfy**(m, L, Ω): *Verifies a purported ring signature Ω on a message m with respect to the ring of public keys L . It outputs a bit $b \in \{0, 1\}$.*
- Link**($m_1, m_2, \Omega_1, \Omega_2$)³: *Takes as inputs two messages m_1, m_2 as well as two signatures Ω_1 and Ω_2 and outputs $b \in \{0, 1\}$.*

The above algorithms form a linkable ring signature scheme if the following three definitions of correctness, signer anonymity, linkability and exculpability are fulfilled.

Definition 5 (Correctness). *Let $N \geq 1$. Then $\forall t \in [N], \forall \{i_1, \dots, i_t\} \subset [N], k \in \{i_1, \dots, i_t\}$ and $\forall m \in \{0, 1\}^*$ it holds that*

$$\Pr \left[\mathbf{Vfy}(m, L, \Omega) = 0 \mid \begin{array}{l} PP \leftarrow \mathbf{Setup}(), \\ \{PK_i \leftarrow \mathbf{KGen}(PP)\}_{i \in [N]}, \\ L = (PK_{i_1}, \dots, PK_{i_t}), \\ \Omega = \mathbf{Sign}_{PP,SK_k}(m, L) \end{array} \right] \leq \text{negl}(\lambda)$$

³ Different from the definition of **Link** algorithm in the existing linkable ring signature schemes [17, 27], our definition does not take L as inputs since we are talking about one-time linkable ring signature.

Signer anonymity captures the intuition that if the targeted signer is not corrupted, then the probability that the adversary can identify him as the true signer among all uncorrupted parties is negligible.

Definition 6 (Signer Anonymity). Let $L = (PK_1, \dots, PK_N)$ be a list of public keys and D_t be any set of $0 \leq t < N$ signing keys such that $\forall SK_i \in D_t \exists PK_i \in L : (PK_i, SK_i)$ is generated by **KGen**. A ring signature scheme is signer anonymous if for any PPT algorithm \mathcal{E} , on inputs of any message m , sets L, D_t as defined above and any valid signature Ω on L and m generated using $SK_\ell \notin D_t$, then

$$\left| \Pr[\mathcal{E}(m, L, D_t, \Omega) = \ell] - \frac{1}{N-t} \right| \leq \text{negl}(\lambda).$$

Let $PP \leftarrow \mathbf{Setup}(1^\lambda)$. For the following two definitions we assume the existence of two oracles $\mathcal{O}_K, \mathcal{O}_S$:

Key generation oracle \mathcal{O}_K : On input of a bit b generate a random keypair $(PK, SK) \leftarrow \mathbf{KGen}(PP)$. If $b = 0$ then output PK , otherwise (PK, SK) .

Signing oracle \mathcal{O}_S : On input (L, m, i) where $L = (PK_1, \dots, PK_N)$ are public keys generated by \mathcal{O}_K , $i \in [N]$ and \mathcal{O}_K did not output SK_i and $m \in \{0, 1\}^*$, return $\Omega \leftarrow \mathbf{Sign}_{PP, SK_i}(m, L)$. If a key in L was not queried before, then output \perp .

The idea behind the Linkability definition is as follows: if the same signer generates two signatures, then the algorithm **Link** will identify this with overwhelming probability. It is important that this not only holds against honest use of the algorithm **Sign**, but arbitrary adversaries.

Definition 7 (Linkability). Let \mathcal{A} be a PPT algorithm with oracle access to $\mathcal{O}_K, \mathcal{O}_S$. \mathcal{A} is given 1^λ and PP as input and outputs a list $L \subseteq \bar{L}$ (where \bar{L} is the set of all keys queried from \mathcal{O}_K) of length N together with $N + 1$ values $\{(m_i, \Omega_i)\}_{i \in [N+1]}$. Then the scheme is linkable if, for every such \mathcal{A} ,

$$\Pr \left[\begin{array}{l} \forall i \in [N+1] : \mathbf{Vfy}(m_i, L, \Omega_i) = 1, \\ \forall i, j \in [N+1], i \neq j : \mathbf{Link}(m_i, m_j, \Omega_i, \Omega_j) = 0 \end{array} \right] \leq \text{negl}(\lambda).$$

The above only talks about the setting of generating signatures without being traceable. Equally important is the setting where signatures are signed by two different parties, where we require that their tags must be distinct. This then, of course, in particular includes the case of the **Sign** algorithm. This property is important in the setting of cryptocurrencies where one might otherwise be able to issue fake transactions on behalf of another party.

Definition 8 (Exculpability). Let \mathcal{A} be a PPT algorithm with oracle access to $\mathcal{O}_K, \mathcal{O}_S$. \mathcal{A} is given 1^λ and PP as input and outputs a list $L \subseteq \bar{L}$ (where \bar{L} is the set of all keys queried from \mathcal{O}_K) of length N together with two pairs $(m_1, \Omega_1), (m_2, \Omega_2)$ with $\mathbf{Vfy}(m_1, L, \Omega_1) = \mathbf{Vfy}(m_2, L, \Omega_2) = 1$, not both queried to \mathcal{O}_S . Let $M \subset L$ be set of PK_i for which \mathcal{A} did not obtain SK_i from \mathcal{O}_K . Then

$$\Pr \left[\mathbf{Link}(L, m_1, m_2, \Omega_1, \Omega_2) = 1 \mid \begin{array}{l} \exists PK_i \in M, \exists m \in \{0, 1\}^*, \\ \exists j \in \{1, 2\} : \\ [\Omega \leftarrow \mathbf{Sign}_{PP, SK_i}(m, L), \\ \mathbf{Link}(m, m_j, \Omega, \Omega_j) = 1] \end{array} \right] \leq \text{negl}(\lambda).$$

Remark 2. In our scheme, we do not give a definition and proof for existential unforgeability. As was observed in [17] the above definitions imply this property, as any algorithm breaking existential unforgeability can be used in a black-box setting to break exculpability (see [17, Theorem 2.6]).

3 Constructing Linkable Ring Signatures

In this section, we will describe our linkable ring signature scheme and prove its security. Our proposed scheme can be considered as an adaption of the linkable ring signature scheme proposed in [27] to the lattice setting. However, while most linkable signature schemes such as the one proposed in [16] require the use of a pseudorandom function to achieve linkability, our scheme demonstrates that the linkability for one-time ring signature schemes can be obtained without using a pseudorandom function to generate the tag.

If a scheme is not one-time, then this PRF is evaluated on the secret (or public) key of the signing party and a description of the actual ring L . In our case, it is not necessary to include the ring L into the tag computation (as the scheme is one-time) and we attach a tag derived from the secret key only. Concretely, each party will have a private key r_i together with a public key $PK_i = \mathbf{A}r_i$, where \mathbf{A} is a random length-compressing matrix and r_i is a vector of small norm. Thus, PK_i is an evaluation of the public collision-resistant hash function $f_A(\cdot) : x \mapsto \mathbf{A}x$ on the private input r_i .

During the signing process, the signer will generate two rings of signatures (similar to [27, 40] but twice): the first is a ring consisting of signatures for all the N public keys and generated using f_A whereas the second ring uses a different CRHF f_B . This function $f_B(\cdot) : x \mapsto \mathbf{B}x$ uses a different public matrix \mathbf{B} having the same dimensions as \mathbf{A} . The crucial point to interleave these rings is that they are built simultaneously, using the same challenges and blinding value in each step. For this to be verifiable, the signer must now include his I_i in the signature, which serves the same purpose as the public key PK_i in the first ring. We will show that the signer is bound to use his own value I_i if he wants to generate a valid signature and will therefore produce a collision if a second signature is revealed.

Let $H : \{0,1\}^* \rightarrow D$ be a cryptographic hash function where D is the challenge space defined in Sect. 2. The algorithms of our scheme are defined as follows:

Setup(1^λ): Sample two random matrices $\mathbf{A}, \mathbf{B} \leftarrow R_q^{h \times v}$ and set $PP = (\mathbf{A}, \mathbf{B})$.

KGen(PP): Sample $\mathbf{r} \leftarrow S_\beta^v$ and then generate the public key $PK = \mathbf{A}\mathbf{r}$ as well as the signing key $SK = \mathbf{r}$.

Sign $_{PP,SK_\ell}(m, L)$:

1. Compute the tag $I_\ell = \mathbf{B}\mathbf{r}_\ell$.
2. Sample $\mathbf{u} \leftarrow \mathcal{N}_\sigma^v$ and set $d_{\ell+1} \leftarrow H(L, I_\ell, m, \mathbf{A}\mathbf{u}, \mathbf{B}\mathbf{u})$.
3. For each $i = \ell + 1, \dots, N, 1, \dots, \ell - 1$:
 - (a) Sample $\mathbf{r}_{z,i} \leftarrow \mathcal{N}_\sigma^v$.
 - (b) Set $t_{i,1} = \mathbf{A}\mathbf{r}_{z,i} - d_i PK_i$ and $t_{i,2} = \mathbf{B}\mathbf{r}_{z,i} - d_i I_\ell$ as well as $d_{(i \bmod N)+1} \leftarrow H(L, I_\ell, m, t_{i,1}, t_{i,2})$.
4. Compute $\mathbf{r}_{z,\ell} = \mathbf{u} + d_\ell \mathbf{r}_\ell$.
5. Abort with probability $1 - \min\left(1, \frac{\mathcal{N}_\sigma^v(\mathbf{r}_{z,\ell})}{M \cdot \mathcal{N}_{d_\ell \mathbf{r}_\ell, \sigma}(\mathbf{r}_{z,\ell})}\right)$, otherwise output the signature $\Omega = (d_1, (\mathbf{r}_{z,i})_{i \in [N]}, I_\ell)$.

Vfy(m, L, Ω):

1. For $i \in [N]$, check whether $\|\mathbf{r}_{z,i}\|_2 \leq 2\sigma\sqrt{v}$, else output 0.
2. For $i \in [N]$, compute $t'_{i,1} = \mathbf{A}\mathbf{r}_{z,i} - d_i PK_i$, $t'_{i,2} = \mathbf{B}\mathbf{r}_{z,i} - d_i I_\ell$ as well as $d_{i+1} = H(L, I_\ell, m, t'_{i,1}, t'_{i,2})$.
3. If $d_1 = H(L, I_\ell, m, t'_{N,1}, t'_{N,2}) = d_{N+1}$ then output 1, else output 0.

Link(Ω_1, Ω_2): Given

$$\Omega_1 = \left(d_1^{(1)}, (\mathbf{r}_{z,i}^{(1)})_{i \in [N]}, I_\ell^{(1)}\right) \text{ and } \Omega_2 = \left(d_1^{(2)}, (\mathbf{r}_{z,i}^{(2)})_{i \in [N]}, I_\ell^{(2)}\right),$$

return 1 if $I_\ell^{(1)} = I_\ell^{(2)}$ and 0 otherwise.

Correctness can easily be verified using Lemmas 1 and 2.

3.1 Security

We now give the security statements of our construction. Due to length constraints, the proofs for these can be found in Appendix A.

Theorem 1 (Signer Anonymity). *The proposed ring signature scheme provides signer anonymity in the (programmable) random oracle model assuming hardness of the D-MLWE $_{2h,v,\beta}$ -problem.*

Theorem 2 (Linkability). *Assume that there exists an algorithm \mathcal{A} that breaks linkability with probability ϵ , in time at most s , with at most q_H queries to \mathcal{O}_K and q_S queries to \mathcal{O}_S . Then there exists an algorithm \mathcal{M} that breaks a MSIS $_{h,v,t}$ -instance with probability $\left(\epsilon - \frac{1}{|\mathcal{D}| - q_H - Nq_S}\right)^2 / ((N^2 + N)q_H)^2$ in time $O(N^2 \cdot q_H \cdot s)$ where $t = 4\sigma\sqrt{v \cdot \nu} + 2 \cdot \kappa \cdot v \cdot \nu^{1.5} \cdot \beta$.*

Theorem 3 (Exculpability). *Assume that there exists an algorithm \mathcal{A} that breaks exculpability with probability ϵ , in time at most s , with at most q_H queries to \mathcal{O}_K and q_S queries to \mathcal{O}_S . Then there exists an algorithm \mathcal{M} that either breaks an S-MLWE $_{2h,v,\beta}$ instance or an MSIS $_{h,v,t}$ -instance with probability*

$$\left(\frac{(N-1)\epsilon}{N} - \frac{1}{|\overline{D}| - q_H - Nq_S} \right)^2 / ((N^2 + N)(q_H + N \cdot q_S))^2$$

in time $O(N \cdot q_H \cdot s)$ where $t = 4\sigma\sqrt{v \cdot \nu} + 2 \cdot \kappa \cdot v \cdot \nu^{1.5} \cdot \beta$.

4 Discussion

We now discuss questions surrounding the practicality of our scheme and hint at future research directions.

Practical Considerations. The runtime of \mathbf{Vfy} is essentially the N -fold runtime of the verification of a regular lattice-based signature scheme. For signing, the computation and sampling of I_ℓ, \mathbf{u} as well as $\mathbf{r}_{r,j}, \mathbf{Ar}_{z,j}, \mathbf{Br}_{z,j}$ for $j \neq \ell$ can be done offline. The size of the total signature is approximately the size of N individual lattice-based signatures, as can be seen in Table 2.

As the basis of our construction, we chose a simple signature scheme without optimizations. Following the outline of our algorithms, one can instantiate it with e.g. [14] and then use their key-compression technique: this optimization is important when it comes to signature size.

Parameter Selection. In our construction, the D-MLWE-instance from Theorem 1 and the S-MLWE-instance in Theorem 3 have the same dimensions and bounds. Moreover, it was already mentioned in Sect. 2.2 that any algorithm which solves the S-MLWE problem in time h with success probability ϵ can be turned into a distinguisher for D-MLWE for the same dimension with essentially the same runtime and success probability. It thus suffices in the parameter selection to look at the D-MLWE-instance only.

Unfortunately, it seems like the security reduction cannot be used for the choice of parameters, as it is inherently non-tight: from the proofs in Sect. 3, we see that the reductions have a huge loss in terms of success probability (both due to the use of the Forking Lemma and because the runtime is proportional to the number of queries of \mathcal{A} to H). If one attempts to obtain a good success probability of the reduction, the estimated runtime gets rather large. We leave a proof with a tighter reduction that can be used to instantiate our construction as an open problem.

Instead, we chose the parameters of our scheme such that the MSIS, D-MLWE-problems are hard given that the reduction succeeds (see Table 2). As baseline, we assume hardness of at least 128 bits using all currently known lattice reduction attacks. This is reflected by requiring that lattice reduction will have to achieve a Root Hermite factor of less than 1.003 to break our

Table 2. Parameter settings for our scheme

Parameter	Recommended choice
q	$\approx 2^{32}$
ν	1024
h	1
v	4
κ	45/90
β (in S_β)	1
σ	31680/63360
t (ℓ_2 MSIS-bound)	$\approx 2^{24}/2^{25}$
Root Hermite factor	< 1.0030
Public key size (per party)	≈ 8 KB/8 KB
Signing key size (per party)	≈ 8.8 KB/8.8 KB
Signature ($N = 1$)	≈ 17.4 KB/17.9 KB
Signature ($N = 8$)	≈ 82.5 KB/86.5 KB
Signature ($N = 32$)	≈ 305.7 KB/321.7 KB
Signature ($N = 128$)	≈ 1.17 MB/1.23 MB

scheme. For the given parameters, the security relies only on Module-SIS/LWE with $h = 1$ i.e. Ring-SIS/LWE, but increasing h, v, κ and thus decreasing ν would allow to base the hardness on Module-SIS/LWE with a larger rank with only a minor increase in the size of the signature.

To choose actual parameters, we use the LWE simulator with sparse secrets from [2, 4] for D-MLWE. Moreover, we use [34] to assess the hardness of our obtained SIS instance⁴. The size estimates in Table 2 are in Kilobytes/Megabytes (as in related work), we bound the size of each coefficient of $\mathbf{r}_{z,i}$ assuming it is within a 6σ -interval.

Post-Quantum Security. It is widely believed that hardness assumptions used in our scheme may offer security in a post-quantum era. On the other hand, it is unlikely that our security proofs carry over to the Quantum Random Oracle Model (QROM, see e.g. [7]): we use adaptive programming of the RO H in Theorem 1, and adaptive rewinding in Theorems 2 and 3. Both of these proof techniques are somewhat inherent to the construction.

⁴ While there might be newer methods to assess the hardness of SIS more precisely, [34] suffices for an estimation of parameters. Moreover, it turned out that using different methods yields hardness estimates (in terms of the Root Hermite factor) that are very close to [34]. Our parameter choices were considered secure at the time of writing, but the reader should refer to the full version of this work for updated parameters.

We note that other candidate constructions in the QROM such as [11, 14] also use a form of RO programming (even though not adaptively). Moreover, though it seems unlikely that the Forking Lemma can be proven in the QROM, there exist no attacks on protocols using these proof techniques which stem from this use of the RO, to the best of our knowledge.

A Proof of Security

A.1 Simulation

The simulation strategy follows a similar pattern as in [27, 40]. In an honestly generated ring signature (where the secret key SK_ℓ is known) the **Sign** algorithm simulates $N - 1$ individual signatures consecutively for all public keys but the one to which its secret key SK_ℓ belongs. For this last public key, it uses the challenge d_ℓ that is obtained for the last signature to *close the ring* using the secret key SK_ℓ . A simulator has no secret key and will instead generate all N individual signatures consecutively this way. To close the ring, it needs to reprogram the random oracle H on the last query to exactly yield the challenge d_1 that is necessary to make all tests in **Vfy** go through. Even though this reprogramming takes place, the challenge d_1 that the RO returns will be fixed in the simulation ahead of time but be chosen uniformly at random. This means that the reprogramming is not detectable. Furthermore, Lemma 2 ensures that the simulation of the ring is indistinguishable.

Concerning the simulation and consistency of the second ring which involves I we note that here I is not obtained from the same secret input \mathbf{r} that is used to derive PK from \mathbf{A} since the simulator does not know SK . Instead, it will choose this value I uniformly at random from the appropriate set. An adversary cannot distinguish between I and the correctly generated counterpart due to Proposition 1.

In fact, the $\text{D-MLWE}_{2h,v,\beta}$ assumption of Proposition 1 attests to the indistinguishability of a pair of quadruples: $(\mathbf{A}, \mathbf{B}, \mathbf{A} \cdot \mathbf{r}, \mathbf{B} \cdot \mathbf{r}) \sim (\mathbf{A}, \mathbf{B}, u, v)$, where u, v are random. One can further reduce the indistinguishability of another pair of quadruples: $(\mathbf{A}, \mathbf{B}, u, v) \sim (\mathbf{A}, \mathbf{B}, \mathbf{A} \cdot \mathbf{r}, v)$ to $\text{D-MLWE}_{h,v,\beta}$ problem, the hardness of which can be deduced from that of $\text{D-MLWE}_{2h,v,\beta}$. Based on hybrid argument, the indistinguishability of the following two quadruples $(\mathbf{A}, \mathbf{B}, \mathbf{A} \cdot \mathbf{r}, v) \sim (\mathbf{A}, \mathbf{B}, \mathbf{A} \cdot \mathbf{r}, \mathbf{B} \cdot \mathbf{r})$ is reduced to the $\text{D-MLWE}_{2h,v,\beta}$ assumption.

A.2 Linkability

Assume that a PPT algorithm \mathcal{A} is run with some certain input and that it generates an output as in the linkability definition. \mathcal{A} makes queries to both the random oracle H and to the two oracles $\mathcal{O}_K, \mathcal{O}_S$ in order to generate these signatures. We construct an algorithm \mathcal{R} which will run \mathcal{A} with multiple inputs and

will attempt to rewind it on one of these inputs with different outputs from the random oracle. During a run, \mathcal{A} will be allowed to make q_H queries to the random oracle directly, but also \mathcal{O}_S indirectly⁵ makes $N \cdot q_S$ queries to H to generate all the queried signatures. \mathcal{R} will simulate $H, \mathcal{O}_S, \mathcal{O}_K$ honestly and will rewind \mathcal{A} with the goal of finding two signatures $\Omega, \hat{\Omega}$ that for some index $\pi \in [N]$ used in signature verification have the same RO query (L, I, m, t_π, t'_π) , but differing $d, \hat{d}, \mathbf{r}, \hat{\mathbf{r}}$ which go into generating this query for each individual signature. Furthermore, we require that the used I has a public key PK_π that was not generated by the simulated oracle⁶. In the full version, we show how to construct such \mathcal{R} that succeeds with probability $\left(\epsilon - \frac{1}{|\mathcal{D}| - q_H - Nq_S}\right)^2 / ((N^2 + N)q_H)^2$ in time $O(N^2 \cdot q_H \cdot s)$.

Using this algorithm \mathcal{R} , we construct another PPT TM \mathcal{M} . This algorithm will obtain a MSIS-challenge \mathbf{A} , use it as the matrix that generates public keys and uses \mathcal{R} to compute the aforementioned signatures. We obtain $d, \hat{d}, \mathbf{r}, \hat{\mathbf{r}}, \pi$ such that $(d - \hat{d})PK_\pi = \mathbf{A}(\mathbf{r} - \hat{\mathbf{r}})$ and $(d - \hat{d})I = \mathbf{B}(\mathbf{r} - \hat{\mathbf{r}})$.

PK_π was generated honestly by \mathcal{O}_K and we have \mathbf{r}_π such that $PK_\pi = \mathbf{A}\mathbf{r}_\pi$. Rewrite the above as $\mathbf{A}(d - \hat{d})\mathbf{r}_\pi = \mathbf{A}(\mathbf{r} - \hat{\mathbf{r}})$. Assume that $(d - \hat{d})\mathbf{r}_\pi = (\mathbf{r} - \hat{\mathbf{r}})$ then by the invertibility of $(d - \hat{d})$ it holds that $I_\pi = \mathbf{B}\mathbf{r}_\pi = \mathbf{B}\left((\mathbf{r} - \hat{\mathbf{r}}) \cdot (d - \hat{d})^{-1}\right) = I$ which contradicts the assumption that I is different from all honestly generated tags. Hence $(d - \hat{d})\mathbf{r}_\pi \neq (\mathbf{r} - \hat{\mathbf{r}})$ and thus $\mathbf{s} = (d - \hat{d})\mathbf{r}_\pi - (\mathbf{r} - \hat{\mathbf{r}}) \neq 0$, while $0 = \mathbf{A}\mathbf{s}$ which yields a solution \mathbf{s} to the MSIS-instance as in Definition 1.

A.3 Exculpability

The algorithm \mathcal{M} which we will construct in the course of this proof will either use the matrix \mathbf{A} in **Setup** to implant an MSIS-challenge or alternatively choose \mathbf{A}, \mathbf{B} from an S-MLWE instance. Whereas in the former case the proof works as above, in the latter one we use a randomly chosen public key and its corresponding tag to embed an S-MLWE challenge. This then means that we cannot correctly simulate the \mathcal{O}_S -oracle as we would need the secret key for it - which is the secret we want to extract! Instead, the proof uses a version of the simulator from signer anonymity.

With respect to the **Link** algorithm from our construction, the definition translates into the requirement that the tags $I^{(1)}, I^{(2)}$ from Ω_1, Ω_2 are equal. Moreover, each $I^{(i)}$ must be identical to an honestly generated identification tag for one of the public keys in L , and \mathcal{A} did not obtain both signatures from \mathcal{O}_S and does not possess the secret key for this public key. Let $I = I^{(1)} = I^{(2)}$.

⁵ These indirect queries are not important when we discuss a signature that does not correspond to any public key.

⁶ We will describe the explicit construction of \mathcal{R} in the full version of this work, but it follows a standard approach using a version of the Forking Lemma.

The algorithm \mathcal{M} will first fairly flip a bit $b \leftarrow \mathcal{B}_{1/2}$. Then it does the following, based on the value of b :

$b = 0$: \mathcal{M} will take a S-MLWE instance (\mathbf{D}, \mathbf{t}) where $\mathbf{D} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} \in R_q^{2h \times v}$ and $\mathbf{t} = \begin{pmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \end{pmatrix} \in R_q^{2h}$ such that $\mathbf{A}, \mathbf{B} \in R_q^{h \times v}$ and $\mathbf{t}_0, \mathbf{t}_1 \in R_q^h$. Assign $PP = (\mathbf{A}, \mathbf{B})$ and choose an index $k \in [N]$. For $j \in [N]$ set

$$(PK_j, SK_j) = \begin{cases} (\mathbf{A}\mathbf{r}_j, (\mathbf{r}_j, \mathbf{B}\mathbf{r}_j)) & \text{if } k \neq j \text{ and for } \mathbf{r}_j \leftarrow S_\beta^v \\ (\mathbf{t}_0, (\perp, \mathbf{t}_1)) & \text{if } k = j \end{cases}$$

We then set the counter $j = 1$. Whenever \mathcal{A} requests a public key from \mathcal{O}_K , then output PK_j and increase j by 1. If $j = k$ and \mathcal{A} requests the secret key then abort. Whenever \mathcal{O}_S is queried, then sign the signature for the queried key s correctly if $s \neq k$, otherwise use the back-patching simulator from the Signer Anonymity proof⁷, but with $I_j = \mathbf{t}_1$.

$b = 1$: \mathcal{M} will take a MSIS instance $\mathbf{A} \in R_q^{h \times v}$ as input, sample $\mathbf{B} \leftarrow R_q^{h \times v}$ uniformly at random and set $PP = (\mathbf{A}, \mathbf{B})$. It will additionally choose $k \in [N]$ uniformly at random. \mathcal{O}_K will generate all keys honestly, but abort if \mathcal{A} queries SK_k . \mathcal{O}_S will run **Sign** honestly.

Assume that \mathcal{A} does not query for SK_k , then the output of \mathcal{A} will be independent of the choice of b due to Theorem 1. If $b = 0$ then \mathcal{A} will be stopped if SK_k is queried, but observe that this abort probability is the same in case $b = 1$ as the key PK_k is perfectly indistinguishable from honestly generated public key PK_j . Moreover, the abort probability in the presence of \mathcal{O}_S is identical due to the construction of the oracle, so the probability that \mathcal{A} outputs something is independent of b . This output probability is $\epsilon' = \epsilon \cdot (N - 1)/N$ by the random choice of k .

In the next step, \mathcal{M} now runs \mathcal{A} using the algorithm \mathcal{R}' (similar to \mathcal{R} from the previous proof it implements a Forking Lemma-type algorithm) which succeeds with probability $\left(\epsilon - \frac{1}{|\mathcal{D}| - q_H - Nq_S}\right)^2 / ((N^2 + N)(q_H + N \cdot q_S))^2$ in time $O(N \cdot q_H \cdot s)$ to obtain signatures that have identical inputs to the random oracle. From \mathcal{R}' obtain values $d, \hat{d}, \mathbf{r}, \hat{\mathbf{r}}, \pi$ such that $(d - \hat{d})\mathbf{A}\mathbf{r}_\pi = (d - \hat{d})PK_\pi = \mathbf{A}(\mathbf{r} - \hat{\mathbf{r}})$ and $(d - \hat{d})\mathbf{I} = \mathbf{B}(\mathbf{r} - \hat{\mathbf{r}})$ where \mathbf{r}_π is the secret key belonging to PK_π . We might either have that $(d - \hat{d})\mathbf{r}_\pi = \mathbf{r} - \hat{\mathbf{r}}$ or that inequality holds. Now if the values are not equal, then we can use the same argument as in linkability to extract a MSIS solution (this covers the case when $b = 1$). But in case of equality the approach does not work - unless we are in the setting where the algorithm \mathcal{M} chose $b = 0$. Now we know that equality holds and \mathbf{r}_π is known to exist as PK_π is a S-MLWE challenge, which we can therefore extract.

⁷ The anonymity simulation does only provide computational indistinguishability as it uses Proposition 1. Here the correctly generated I_j is known and the simulation is statistically indistinguishable, not just computationally.

More formally, if $b = 0$ and $k = \pi$ then \mathcal{M} will output $\mathbf{r}_\pi = (\mathbf{r} - \hat{\mathbf{r}}) \cdot (d - \hat{d})^{-1}$ as $d - \hat{d} \in D'$. If $b = 1$ then it will instead output $(d - \hat{d})\mathbf{r}_\pi + \hat{\mathbf{r}} - \mathbf{r}$. We now calculate the probability that the algorithm \mathcal{M} will output a correct answer to either of the two challenges. Therefore, denote with $\mathbf{X}_=$ the event that $(d - \hat{d})\mathbf{r}_\pi = \mathbf{r} - \hat{\mathbf{r}}$, and with \mathbf{X}_\neq the opposite event. Let \mathbf{M} denote the event that \mathcal{M} outputs something. As our goal is to lower-bound the probability that the output of \mathcal{M} is correct, we need to determine

$$\Pr[\mathcal{M} \text{ gives correct output}] = \Pr[\mathbf{X}_=, b = 0 | \mathbf{M}] + \Pr[\mathbf{X}_\neq, b = 1 | \mathbf{M}]$$

If $b = 0$, then by the choice of k , the probability that $\pi = k$ is at least $1/|L|$ and therefore $\Pr[\mathbf{M} | \mathbf{X}_=, b = 0] \geq 1/N$. Using Bayes' Theorem, we obtain that

$$\begin{aligned} \Pr[\mathbf{X}_=, b = 0 | \mathbf{M}] &= \frac{\Pr[\mathbf{M} | \mathbf{X}_=, b = 0] \cdot \Pr[\mathbf{X}_=, b = 0]}{\Pr[\mathbf{M}]} \\ &\geq \Pr[\mathbf{M} | \mathbf{X}_=, b = 0] \cdot \Pr[\mathbf{X}_=, b = 0] \\ &\geq 1/N \cdot \Pr[\mathbf{X}_=] \cdot \Pr[b = 0] = 1/2N \cdot \Pr[\mathbf{X}_=] \end{aligned}$$

where we use in the last step that the occurrence of $\mathbf{X}_=$ is independent of b .

In case of $b = 1$ we always give output, so we have that $\Pr[\mathbf{M} | \mathbf{X}_\neq, b = 1] = 1$. Using the same reasoning as above, we obtain that $\Pr[\mathbf{X}_\neq, b = 1 | \mathbf{M}] \geq 1/2 \cdot \Pr[\mathbf{X}_\neq]$ which yields an overall bound of $\Pr[\mathcal{M} \text{ gives correct output}] \geq 1/2N$.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC, pp. 99–108 (1996)
2. Albrecht, M.R., et al.: Estimate all the LWE, NTRU schemes! (2018). <https://eprint.iacr.org/2018/331>
3. Albrecht, M.R., Deo, A.: Large modulus ring-LWE \geq module-LWE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 267–296. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_10
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. *J. Math. Cryptol.* **9**(3), 169–203 (2015)
5. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_19
6. Bender, A., Katz, J., Morselli, R.: Ring signatures: stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_4
7. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_3
8. Bos, J., et al.: CRYSTALS - kyber: a CCA-secure module-lattice-based KEM (2017). <https://eprint.iacr.org/2017/634>

9. Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_29
10. Brakerski, Z., Kalai, Y.T.: A framework for efficient signatures, ring signatures and identity based encryption in the standard model (2010). <http://eprint.iacr.org/2010/086>
11. del Pino, R., Lyubashevsky, V., Neven, G., Seiler, G.: Practical quantum-safe voting from lattices. In: CCS 2017 (2017)
12. Derler, D., Ramacher, S., Slamanig, D.: Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 419–440. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_20
13. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in *Ad Hoc* groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_36
14. Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehle, D.: Crystals - dilithium: Digital signatures from module lattices (2017). <http://eprint.iacr.org/2017/633>
15. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 335–352. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_19
16. Franklin, M., Zhang, H.: A framework for unique ring signatures (2012). <http://eprint.iacr.org/2012/577>
17. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 181–200. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_13
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206. ACM (2008)
19. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052231>
20. Groth, J., Kohlweiss, M.: One-out-of-many proofs: or how to leak a secret and spend a coin. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 253–280. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_9
21. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: a signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_31
22. Hoffstein, J., Pipher, J., Silverman, J.H.: NSS: an NTRU lattice-based signature scheme. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 211–228. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_14
23. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures (2018). <https://eprint.iacr.org/2018/475>
24. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* **75**(3), 565–599 (2015)

25. 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
26. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based PRFs and applications to e-cash. ASIACRYPT 2017 (2017). <http://eprint.iacr.org/2017/856>
27. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable and anonymous signature for ad hoc groups. In: ACISP 2004. LNCS, vol. 3108, pp. 325–335. Citeseer (2004)
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
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
30. 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
31. 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
32. Lyubashevsky, V., Seiler, G.: Partially splitting rings for faster lattice-based zero-knowledge proofs. In: EUROCRYPT 2018 (2018). <https://eprint.iacr.org/2017/523>
33. Aguilar Melchor, C., Bettaieb, S., Boyen, X., Fousse, L., Gaborit, P.: Adapting Lyubashevsky’s signature schemes to the ring signature setting. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 1–25. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38553-7_1
34. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_5
35. Mohamed, M.S.E., Petzoldt, A.: RingRainbow – an efficient multivariate ring signature scheme. In: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 2017. LNCS, vol. 10239, pp. 3–20. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57339-7_1
36. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_19
37. Noether, S., Mackenzie, A.: Ring confidential transactions. Ledger **1**, 1–18 (2016)
38. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_8
39. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)
40. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32

41. Torres, W.A., et al.: Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1.0) (2018). <https://eprint.iacr.org/2018/379>
42. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: Deng, R.H., Bao, F., Pang, H.H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 48–60. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31979-5_5
43. Wang, J., Sun, B.: Ring signature schemes from lattice basis delegation. In: Qing, S., Susilo, W., Wang, G., Liu, D. (eds.) ICICS 2011. LNCS, vol. 7043, pp. 15–28. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25243-3_2
44. Yang, R., Au, M.H., Lai, J., Xu, Q., Yu, Z.: Lattice-based techniques for accountable anonymity: composition of abstract Stern’s protocols and weak PRF with efficient protocols from LWR. Cryptology ePrint Archive, Report 2017/781 (2017)

Full Paper Session VII: Attack Analysis and Detection



Slop: Towards an Efficient and Universal Streaming Log Parser

Zhiyuan Zhao¹, Chenxu Wang^{1,2(✉)}, and Wei Rao¹

¹ School of Software Engineering, Xi'an Jiaotong University,
Xi'an 710049, Shaanxi, China

² MoE Key Lab for Intelligent Network and Network Security,
Xi'an Jiaotong University, Xi'an 710049, Shaanxi, China
cxwang@mail.xjtu.edu.cn

Abstract. System logs record useful information such as execution paths and states of running programs. Log analysis is an important part of anomaly detection which is critical for system security. A primary step for log anomaly detection is to extract structured log templates (message types) from a mass of unstructured raw logs. However, conventional log parsers are designed to work offline, which needs to collect logs for a time period and then load all logs into memory for training. This greatly limits its applications to large-scale log analysis. With the continuous increase of log scales, online streaming methods are greatly desired now. Most of existing online methods are designed for specific log systems and there still lacks a universal log parser. In this paper, we present Slop, which is an efficient and universal streaming log parser. To improve the efficiency of Slop, we first group coming log messages into different partitions according to their lengths. Then, we extract the message types from different partitions. This avoids many unnecessary comparisons between logs and existing message types. To improve the universality and accuracy, we investigate the relationships between lengths of message types and the lengths of their raw logs. Based on the uncovered results, we design a nonlinear threshold criterion for message type extraction which is adaptive to several log systems. Finally, we implement a prototype of Slop and conduct extensive experiments to validate its effectiveness and efficiency based on diverse real-world datasets. It is shown that Slop obtains 55%–82% improvements in accuracy and achieves higher efficiency than state-of-the-art methods.

Keywords: Log parsers · Log partitioning · Nonlinear thresholds

1 Introduction

Log systems record the behavior and running states of systems and programs. Logs contain useful information such as execution paths which help operators to detect execution anomalies [1–3]. Log analytics plays important roles in the building of secure and trustworthy systems. In general, log files are composed of

independent lines of unstructured text data, which is called a log message. A primary step for log analysis is to convert unstructured raw logs into structured log templates which are called message types. Implementations of such techniques are generally called as log parsers.

Existing log parsers can be roughly classified into two categories, namely offline methods and online methods. Offline methods such as LKE [4] and LogSig [5] generate message types based on clustering methods which divide log messages into different clusters. Heuristic methods such as SLCT [6] and IPLoM [7] generate candidates of messages types by counting the occurrences of words at different positions. However, offline methods are often limited by system resources as they need load a mass of log data into memory. In addition, log messages used for training are collected in a specific time period. If new message types are added after training, we have to train the parser again. With the development of computer science, the scale of logs is becoming much larger than before, especially with the emergence of distributed systems. For example, a large service system like HDFS can generate around 50 GB logs (120–200 million lines) per hour [8]. Therefore, online streaming methods are greatly demanded.

Drain [9] and Spell [10] are two recent online log parsers. These methods process log messages in a streaming manner, which works incrementally as log messages are being generated. However, there are two weaknesses of current online log parsers. First, there is an improvement space for existing online methods in both accuracy and efficiency. Second, these methods are designed for specific log systems and the parameter settings are not universal. Drain needs to specify the depth of a prefix tree before parsing log messages [9]. Spell generates message types based on a linear threshold criterion which is not capable for most log systems [10].

In this paper, we present Slop, an efficient and universal streaming *log parser*. Based on the intuition that in most cases log messages have the same length if they belong to the same message type, we perform a partition step to group incoming log messages according to their lengths. Then we extract message types for each partition independently. This greatly improves the efficiency by avoiding many unnecessary searches and comparisons. In order to guarantee the accuracy of Slop, we also combine the message types in different partitions since a small number of message types may generate logs with varying lengths (e.g., logs have different numbers of parameters). In order to improve the universality of our method, we investigate the relationships between the lengths of message types and the lengths of raw log messages for several log systems. We find that these two metrics are not linearly correlated. We then propose a nonlinear threshold criterion to extract message types from the raw log messages. This greatly improves the universality and accuracy of our approach. Compared with other methods, Slop is more adaptive and requires less domain knowledge of log systems. Finally, we implement a prototype of our method and conduct extensive experiments to validate its effectiveness and efficiency based on diverse datasets. The results clearly demonstrate that Slop outperforms state-of-the-art methods in both accuracy and efficiency.

In summary, we make the following contributions:

- We improve the efficiency of Slop by grouping incoming log messages into different partitions. This avoids many unnecessary comparisons in the step of message type extraction.
- We guarantee the accuracy of Slop by aggregating message types in different partitions and merging the message types belonging to the same types.
- We improve the universality and accuracy of Slop by proposing a nonlinear threshold criterion in the step of message type extraction.
- We conduct experiments based on the data collected from five real log systems to evaluate the performance and the result clearly shows the superiority of Slop.

The rest of this paper is organized as follows: In Sect. 2, we present the terminologies. Section 3 describes the methodology of Slop. Section 4 shows the selection of a proper threshold criterion for message type extraction. In Sect. 5, we conduct extensive experiments to evaluate the performance of Slop. After a review of related work in Sect. 6, we conclude the work in Sect. 7.

2 Terminologies

In this section, we describe the terminologies used in this paper. Figure 1 presents an example of the log parsing problem. The upper box shows the raw log messages and the lower contains the extracted message types. Log messages 1 and 2 are from BlueGene/L [11], and Log messages 3 to 5 are from HDFS [12]. Message types 1 to 4 are their templates, respectively.

Log messages are independent text lines in a log file. It is a complete log that describes the behavior of the system or an application. A log message is constituted by constant tokens and variable tokens. The upper box in Fig. 1 contains a number of log messages, where constant and variable tokens are distinguished with black and blue colors, respectively. In the rest of this paper, we use m_i to denote a log message.

Message types are generative templates of log messages. A message type consists of constant tokens, which is the common part of a large number of log messages. The variable tokens are replaced by asterisks, as shown in the bottom box in Fig. 1. Message types have the same constant tokens but different numbers of variable tokens belong to the same type. In the rest of this paper, we use t_k to denote a message type.

Tokens are words delimited by whitespace, commas or colons in a log message. For example, RAS, KERNEL, FATAL, r20 = 0x0044397c presented in Fig. 1 are tokens. The length of a log is defined as the number of tokens.

Constant tokens are the common parts in many different log messages. For example, RAS, KERNEL, FATAL are constant tokens of log messages 1 and 2.

Variable tokens are the parts which vary in different log messages. Variable tokens are replaced by asterisks in Fig. 1. For example, the last four tokens in log messages 1 and 2 are variable tokens.

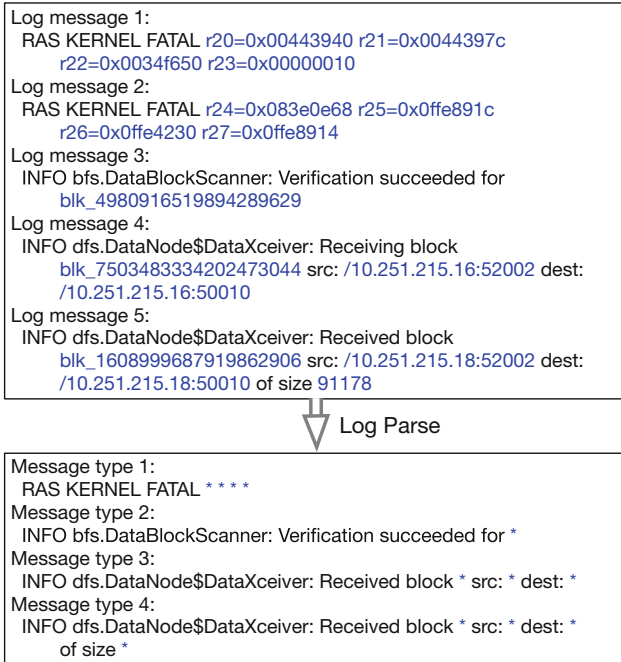


Fig. 1. An example of the log parsing problem (Color figure online)

3 Methodology of Slop

Figure 2 presents the basic workflow of Slop, which consists of five main steps, namely raw log preprocessing, log partitioning, message type prematching, message type extraction, and message type combination.

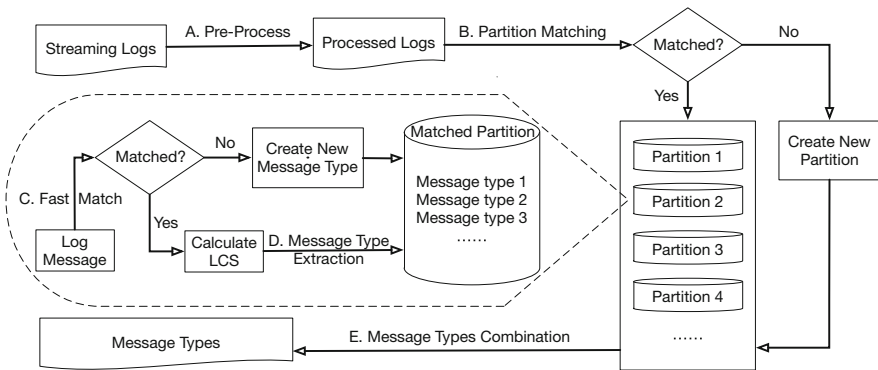


Fig. 2. Structure of slop

3.1 Raw Log Preprocessing

In this step, Slop deletes tokens that are ensured to be variable tokens, e.g., timestamps. Timestamps always increase in a single log file. However, timestamps have fixed positions in log messages and thus the efficiency can be greatly improved if we remove these tokens before log parsing. It is worth noting that this is not a necessary step. If users have no knowledge of where the timestamp is, they can choose to do nothing in this step. The accuracy of the results will not be affected without any preprocessing of the raw logs.

3.2 Log Partitioning

In this step, Slop groups an incoming log message into specific partitions according to the length. As shown in Fig. 2, for an incoming message, if it does not match any partitions according to its length, Slop will create a new partition. Message types are extracted in each partition independently. Each partition contains several message types. In the rest of this paper, we denote a partition as a set P_j , and $P_j = \{t_{1j}, t_{2j}, \dots, t_{kj}\}$, where t_{kj} is the k -th message type in P_j . Clearly, partitioning the messages into small groups reduce many unnecessary comparisons. This is because the number of message types in each partition is much smaller than the total number of message types. In most cases, log messages have the same length if they belong to the same message type. Some log messages belonging to the same message type may have different lengths and thus are grouped into different partitions. We handle this problem by combining all message types and merge message types belonging to the same type. The details are described as in Sect. 3.5.

3.3 Message Type Prematching

When an incoming message m_i is grouped into a partition P_j , Slop first calculates the intersections between m_i and $t_{kj} \in P_j$. If the intersection length satisfies the threshold criterion (see Sect. 4), then m_i is a possible realization of t_{kj} . In the next step Slop extracts the message type and parameters of m_i , and update the message type t_{kj} . If the intersection length does not satisfy the threshold criterion for any message types, then a new message type is created and added to the partition.

3.4 Message Type Extraction

If an incoming message m_i is prematched with a message type t_{kj} , Slop extract the message type and parameters of m_i , and update t_{kj} using the LCS (Longest Common Subsequence) method. The LCS problem is to find the longest subsequence common to all sequences in a set of sequences (often just two sequences). For example, given two sequences $S_1 = ABCDEFG$ and $S_2 = ABKDEFH$, the longest common subsequence of S_1 and S_2 is ABDEF. There are two key points to note here. The first key point is that LCS needs to appear in both sequences

simultaneously. The other is that the order of tokens in LCS needs to be the same order it appears in both sequences.

When we obtain the LCS between m_i and t_{kj} , then the remaining tokens (variable tokens) are parameters. We then replace the variable tokens with asterisks to obtain the message type of m_i . If the obtained message type is different from t_{kj} , we use it substitute t_{kj} .

3.5 Message Types Combination

As mentioned previously, the partitioning step may divide the log messages belonging to the same type into different groups. In this step, Slop combines all message types in different partitions to obtain the final set of message types and merges message types belonging to the same type. Figure 3 presents an example of two log messages from HDFS [12]. At the end of m_1 , there is just one IP address as its parameter while there are two IP addresses in m_2 . Although m_1 and m_2 have different lengths, they belong to the same message type.

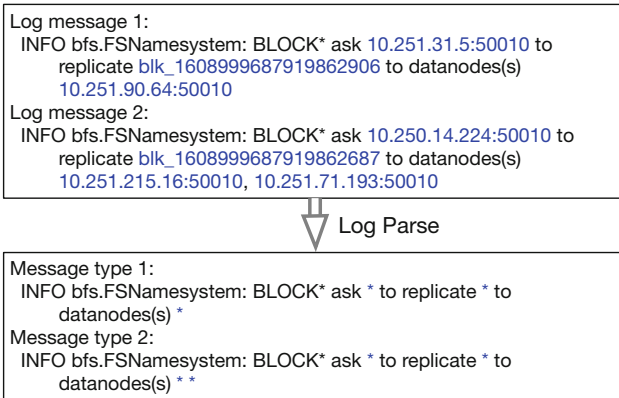


Fig. 3. An example from HDFS logs

To tackle this problem, we propose an algorithm to combine the obtained message types. Assume there are n partitions and each partition contains a number of message types. Denote the set of final message types as P . Slop compares the message types from each partition with that in P . If a message type is a subsequence of the other, then the two message types will be combined as a single message type. The pseudo-code of the algorithm is presented in Algorithm 1. The function $intersection(t_{kj}, t_i)$ calculates the length of common constant tokens between t_{kj} and t_i . $|t_i|$ is the length of the message type t_i , which is the number of constant tokens.

Algorithm 1. Message types combination algorithm

```

Input: Partitions
Output:  $P$ : The final message type set
Initialization:  $P = \emptyset$ 
1 for  $j \leftarrow 1 : n$  do
2   foreach  $t_{kj} \in P_j$  do
3     flag = True;
4     foreach  $t_i \in P$  do
5       if  $\text{intersection}(t_{kj}, t_i) == \min(|t_{kj}|, |t_i|)$  then
6         flag = False;
7         break;
8   if flag then
9     Add  $t_{kj}$  to  $P$ ;

```

It is worth noting that the algorithm is efficient since the total number of message type are usually small. For example, an HDFS [12] log file which contains more than 11 million log messages just has 39 message types. Additionally, the combination can be performed periodically with a much longer time interval. This further improves the efficiency.

4 A Nonlinear Threshold Criterion

In this section, we first present why a proper threshold criterion is necessary for message type extraction and the weaknesses of the linear threshold criterion. We then introduce a nonlinear threshold criterion which is universal for several common log systems.

4.1 The Necessity of a Proper Threshold

Although log messages m_3 and m_4 in Fig. 1 share a common token “INFO”, they do not belong to the same message type. That is, we need a threshold to determine when an LCS can be regarded as a message type. In the rest of this paper, we use l_{ij} to denote the length of LCS between log messages m_i and m_j , and ω to denote the threshold. That is, if and only if the length of the LCS is greater than the threshold ω , these two log messages can be regarded to belong to the same message type.

However, how to decide a proper threshold criterion is a core and difficult problem. Spell introduces a linear threshold criterion which is defined as $\omega = |m_i|/2$ [10], where $|m_i|$ is the length of m_i . For example, the length l_{34} of the LCS of m_3 and m_4 is 1, and $\omega = 3$ (i.e., $6/2$). Since l_{34} is not greater than ω , the LCS “INFO” of m_3 and m_4 cannot be regarded as a message type.

Such a threshold criterion works well for specific log systems. However, it encounters serious problems when applied to other log systems. For example,

Table 1. Statistical results for different log systems

Log systems	Logs	Logs ($l_{ij} \leq \omega$)	MT	MT ($l_{ij} \leq \omega$)
BlueGene/L	2000	1297 (65%)	112	29 (26%)
HDFS	2000	1061 (53%)	14	5 (36%)
HPC	2000	1331 (67%)	44	27 (61%)
Proxifier	2000	1958 (98%)	7	3 (43%)
Zookeeper	2000	324 (16%)	46	12 (26%)

the lengths of log messages m_1 and m_2 in Fig. 1 are both 7, and their LCS l_{12} is of length 3. Thus, the log parser will determine that the two messages belong to different message types. We investigate such a problem in five log systems, namely BlueGene/L, HDFS, HPC, Proxifier, and Zookeeper. He et al. provide the datasets for the five system logs with ground-truth message types in their work [13]. They randomly select 2000 log messages from every dataset and extract their message types. We then examine how many logs and message types that violate the linear threshold criterion. The results are presented in Table 1. It is found that a large fraction of logs (the 3rd column) and message types (the 5th column) disobey the criterion. For instance, nearly 65% log messages have an LCS whose length is not greater than half the length of the message. These messages are generated from about 26% message types. The statistical results clearly show such a linear threshold criterion will result in high message type extraction errors. It is necessary to find a proper threshold criterion which is adaptive to different system logs.

4.2 Nonlinear Threshold

In order to find a proper threshold criterion, we look insight into the relationship between the length of log messages and the length of its corresponding message types based on the above five log systems. The results are shown in Fig. 4. The x-axis represents the length of log messages and the y-axis is the length of message types. We also plot the linear threshold criterion in the figures. It is shown that when the length of log messages is small, the length of most message types is below the line of the linear threshold. This indicates that short message types have a big chance to violate the criterion. Therefore, we should find a proper threshold criterion that satisfies: (i) The threshold curve should approximate to the message type curve and (ii) it should not surpass above the message type curve.

Take the above analysis into consideration, we find a proper nonlinear threshold criterion:

$$\omega(m_i) = \frac{1}{2} |m_i| \tanh\left(\frac{|m_i|}{\max(|m|)} T\right), \quad (1)$$

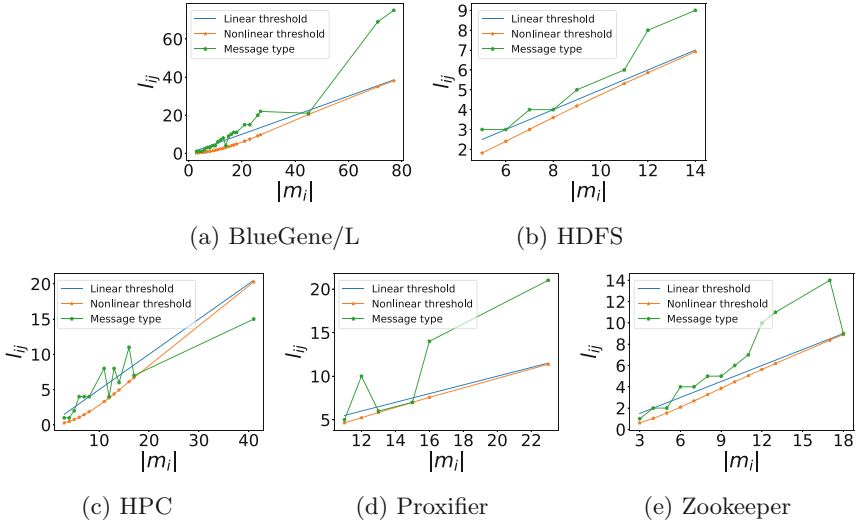


Fig. 4. Effectiveness of \tanh .

where $|m_i|$ is the length of log messages, $\max(|m|)$ is the maximum length of log messages for a specific log system, T is an adjustable constant, and

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (2)$$

The motivation of the above nonlinear threshold criterion is to reconcile the linear one with a $\tanh(x)$ function. However, we have to determine a proper value of the parameter T in order to satisfy the previously mentioned two constraints. The curve of the $\tanh(x)$ function is plotted in Fig. 5. We focus on the first quadrant of the $\tanh(x)$ function since the length of log messages are always positive. It is shown that the value of $\tanh(x)$ increases with the increase of x with a decreasing rate. The values of $\tanh(x)$ approximate to 1.0 when x is large enough. In fact, we want to apply the nonlinearity of $\tanh(x)$ when the length of a log message $|m_i|$ is small. Therefore, we have to scale the value of $\frac{|m_i|}{\max(|m|)}$ in order to satisfy the constraints. We find that when $x \geq 2.64$, it is enough to guarantee the value of $\tanh(x)$ approximate to 1.0. Hence, in this paper we set $T = 2.64$.

We also plot the nonlinear threshold criterion in Fig. 4. We can see that in all datasets, our nonlinear threshold criterion works very well with only one exception in the HPC log system [14]. The exceptional message type has a length of 44, and the logs corresponding to this message type just appear only once in the 2000 log messages.

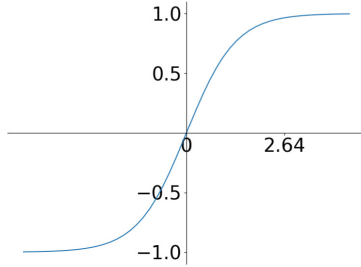


Fig. 5. The Tanh function

5 Experiments

In this section, we first introduce the datasets to conduct the experiments. Then we evaluate the effectiveness and efficiency of our approach by comparing with the state-of-the-art methods.

5.1 Datasets Description

We employ datasets collected from 5 real log systems for the experiments. Table 2 shows the statistical information of the datasets. *BlueGene/L* is collected by LLNL (Lawrence Livermore National Labs). It contains logs collected from a supercomputer called BlueGene/L [11]. *HDFS* is collected from a 203 nodes cluster which is deployed on the Amazon EC2 platform [12]. *HPC* is collected from a cluster, too, which has 49 nodes [14]. *Proxifier* is collected from a software called Proxifier. *Zookeeper* is a dataset used in [9] and it is collected from a 32-node cluster. These datasets can be obtained from [15].

Table 2. Datasets information

System	Log messages	Message types
BlueGene/L	4,747,963	183
HDFS	11,175,629	39
HPC	433,490	50
Proxifier	10,108	9
Zookeeper	74,380	64

5.2 Parameter Settings

We compare Slop with four log parsers namely IPLoM [7], LogSig [5], Spell [10], and Drain [9], which are proposed in 2009, 2011, 2016, and 2017, respectively. We left those methods that are very old like SLCT [6] which is proposed in 2003. We have tuned the parameters of these methods to achieve their best performance. Table 3 shows the parameters for different datasets. For IPLoM, *ct* is used to

avoid further partitioning. Its value ranges in $(0, 1]$. LB is the lower bound to control how to find a bijective [7]. For LogSig, k is the number of message types [5, 9, 13]. Drain [9] has two parameters named *depth* (depth of fixed tree) and *st*. *Depth* is the depth of the fixed tree and *st* is the similarity threshold [9] which is used to select the most suitable log group for a log message. Spell and Slop need no parameters.

Table 3. Parameters of IPLoM, LogSig and Drain

Methods	Parameters	BlueGene/L	HDFS	HPC	Proxifier	Zookeeper
IPLoM	<i>ct</i>	0.4	0.35	0.18	0.55	0.4
	LB	0.01	0.25	0.25	0.25	0.65
LogSig	<i>k</i>	183	39	50	9	64
Drain	Depth	3	4	3	3	4
	<i>st</i>	0.3	0.4	0.5	0.3	0.3

5.3 Effectiveness Evaluation

In this experiment, we evaluate the effectiveness of Slop. The ground-truth of message types for each dataset is obtained as follows: First, we obtain four sets of message types by applying IPLoM [7], Drain [9], Spell [10] and Slop, respectively. We do not use LogSig because it needs the number of message types as its input [5]. If a message type is obtained by more than two methods, we regard it as a ground-truth. For each of the types extracted only by one method, we calculate its similarity with each of the previously obtained ground-truth. The similarity between two message types is defined as follows:

$$s = \frac{|\{t_1\} \cap \{t_2\}|}{|\{t_1\} \cup \{t_2\}|} \quad (3)$$

where $\{t_1\}$ is the set of constant tokens contained in t_1 . A message type is added as a ground-truth if its similarity with any ground-truth message types is smaller than 0.5.

We first examine the performance of the methods for different log systems. We use n_{MT} to denote the number of message types generated by each method and n_G to denote the number of message types in the ground-truth. We then calculate $|n_{MT} - n_G|$ to see which method performs the best for different log systems. The result is shown in Fig. 6. The red line represents the baseline, which is always zero. It is shown that Slop performs better than other methods in almost every dataset. Spell [10] generate much more message types for HDFS [12] because the threshold criterion is not a suitable choice. Log messages whose constant tokens are less than half the length of the log messages will be regarded as a new message type. Thus, Spell extracts many redundant message types which belong to the same message type. Drain [9] also generates too many message types for Proxifier.

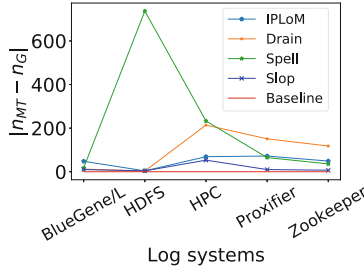


Fig. 6. Number of message types generated by each method (Color figure online)

We then employ the F1-score to evaluate the accuracy of these methods. F1-score is defined as follows:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}. \quad (4)$$

where *Precision* and *Recall* are defined as:

$$Precision = \frac{TP}{TP + FP}. \quad (5)$$

$$Recall = \frac{TP}{TP + FN}. \quad (6)$$

where TP, FP, and FN in the equations are True Positive, False Positive and False Negative, respectively. True Positive means that log messages generated by the same message type are correctly grouped together. False Positive is the case that log messages not belonging to the same message type are misclassified together. False Negative represents that message types belonging to the same message type are parsed as different message types. Table 4 shows the performance of each method on different datasets. It is shown that Slop achieves the highest scores on Precision, Recall and F1-score on all log systems except for the Zookeeper. Although Slop is not as good as IPLoM [7] and Drain [9] on Zookeeper [9], it still has good performance which is close to the highest score. Spell has very low recalls on most datasets because its FNs are too high. Slop reduces the FNs by the employment of the nonlinear threshold criterion.

5.4 Efficiency of Slop

To evaluate the efficiency of Slop, we randomly sample 20%, 40%, 60%, 80%, and 100% log messages from each dataset, respectively. The numbers of log messages in the sampled datasets are shown in Table 5. Figure 7 shows the running time of each log parser. We observe that, compared with other methods, the running time of LogSig [5] increases faster as the log size increases. This is because it uses a matrix to generate message types, whose time complexity is $O(n^2)$. IPLoM [7] shows the best performance on BlueGene/L [11]. This is because log messages from BlueGene/L are more structured than other datasets. So it is faster to count

Table 4. Precision, Recall, F1-score of each methods

Dataset	Metrics	LogSig	IPLoM	Spell	Drain	Slop
BlueGene/L	Precision	0.83	0.97	0.91	0.97	0.99
	Recall	0.25	0.8	0.87	0.72	0.9
	F1-score	0.39	0.87	0.89	0.83	0.94
HDFS	Precision	0.82	0.92	0.92	0.92	0.93
	Recall	0.75	0.86	0.06	0.86	0.93
	F1-score	0.78	0.88	0.11	0.89	0.93
HPC	Precision	0.81	0.33	0.82	0.85	0.97
	Recall	0.56	0.25	0.73	0.69	0.83
	F1-score	0.66	0.29	0.77	0.76	0.9
Proxifier	Precision	0.73	0.89	0.86	0.88	0.89
	Recall	0.73	0.33	0.38	0.1	0.78
	F1-score	0.73	0.48	0.52	0.19	0.82
Zookeeper	Precision	0.82	0.95	0.97	0.95	0.94
	Recall	0.7	0.95	0.68	0.95	0.86
	F1-score	0.76	0.95	0.8	0.95	0.9

the token frequency at each position and search for bijections [7]. However, this method requires more memory than others. Spell [10] is more time-consuming than Drain [9]. Since when there is no matched message type in the prefix tree for a coming log message m_i , it has to calculate the LCS between m_i and each of the extracted message types t_{ij} . The time complexity of calculating LCS between m_i and t_{ij} is $O(|m_i||t_{ij}|)$. On the other hand, the depth of the prefix tree increases as the parsing process goes on. Drain [9] consumes slightly more time than Slop. Drain divides all log messages that contain digits to a same group. Although it restricts the maximum width and depth of the tree, there may be too many logs in a group [9]. So it costs more time for systems whose log messages contain too many digits. As shown in Fig. 7, except for the BlueGene/L dataset, Slop has the best performance among all the methods. This is because Slop partitions log messages by length, which reduces many unnecessary comparisons between log messages and message types.

Table 5. Different sizes of sample datasets

Dataset	20%	40%	60%	80%	100%
BlueGene/L	0.9m	1.8m	2.7m	3.6m	4.5m
HDFS	2.1m	4.3m	6.4m	8.5m	10.7m
HPC	84.7k	169k	254k	338.7k	423k
Proxifier	2k	3.9k	5.9k	7.9k	9.9k
Zookeeper	14.5k	29k	43.6k	58k	72.6k

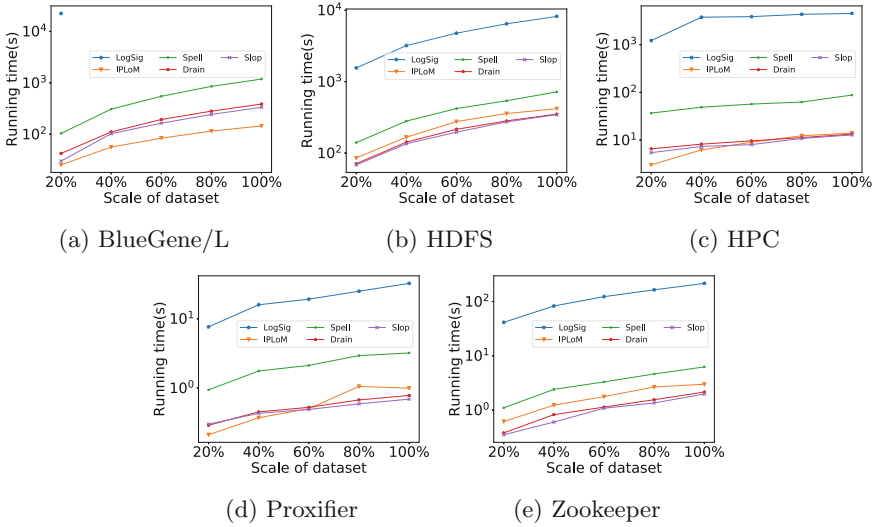


Fig. 7. Efficiency of each method on different log size.

The time complexity of Slop is $O(|P_j||m_i| + |t_{ij}||m_i|)n$, where $|P_j|$ is the number of message types in one partition, $|m_i|$ is the length of the log message, $|t_{ij}|$ is the length of the message type, and n is the number of log messages. For every log message, $|P_j||m_i|$ is the time complexity of prematching, and $|t_{ij}||m_i|$ is the time complexity of calculating LCS. These are only one calculation of LCS for a log message. Obviously, $|m_i|$ and $|t_{ij}|$ can be regarded as constants since they are far less than the number of log messages. The number of message types $|P_j|$ in each partition can also be regarded as a constant. Figure 8 plots the number of message types in each partition. Take the BlueGene/L dataset which has the most message types among the five datasets as an example, Slop produces 173 message types totally. With partitioning, it only needs to compare at most 35 times instead of 173 times in the stage of prematching. In the other four datasets, the number of message types in almost all partitions are less than 20. The number of message types $|P_j|$ in each partition is much smaller than non-partition, which greatly reduces the time complexity.

We also evaluate the average time for each online method to find the message type of a newly incoming log message. Table 6 shows the results. It is shown that Slop has the best performance to find or generate the message type of a log message. Such results clearly demonstrate that Slop satisfies the online processing requirement.

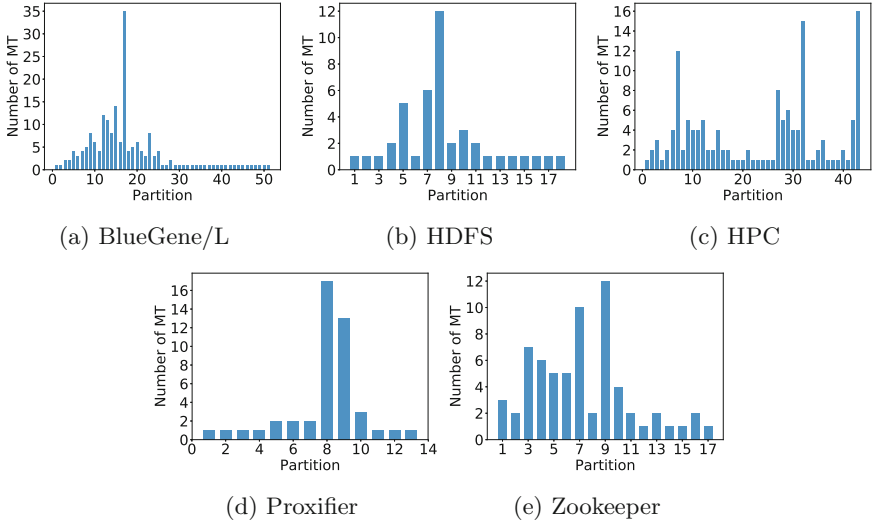


Fig. 8. Number of message types in different partitions.

Table 6. Average lookup time of online methods

Method	BlueGene/L	HDFS	HPC	Proxifier	Zookeeper
Spell (ms)	0.246	0.067	0.201	0.149	0.083
Drain (ms)	0.083	0.031	0.033	0.035	0.03
Slop (ms)	0.074	0.031	0.032	0.032	0.029

6 Related Work

There have been many researches on log parsing, which is critical for log anomaly detections such as DDoS attack detections [3] and performance failure detections [16–20]. To the best of our knowledge, the first log parser is called SLCT [6]. SLCT first calculates the distance between log messages and then employs a clustering technique to divide log messages into different groups, and finally extract the message types. Xu et al. [17, 21] proposed a method to automatically generate message types through source code. However, in practice source codes are often not available in most cases. Through pre-defined regular expressions, message types could also be extracted from raw logs [4]. However, the definition of regular expressions needs domain knowledge, which is a high requirement for most log parser users. IPLoM [7] consists of three steps to group raw logs iteratively and search for bijective relationships between logs. LogSig [5] uses a metric to determine which message type the raw log belongs to. It requires the number of message types as input. However, in practice, it is hard to determine how many message types a specified system has.

To the best of our knowledge, Drain is the latest log parser proposed by He et al. in 2017 [9]. It is an online streaming log parser which parses raw logs using a fixed depth tree. It also needs the user to write regular expressions to pre-process the raw logs, which need domain knowledge like LKE [4]. Moreover, the parameter configurations of Drain [9] varies from system to system. It is usually hard to specify the parameters for different log systems. Spell [10] is another online streaming log parser. It parses log messages through computing longest common subsequence between log messages and message types. Every incoming log message needs to be compared with all message types until the message type of the log message is found. As the number of message types increases, this process becomes time-consuming and there are many unnecessary comparisons between log messages and message types.

In summary, all these methods face a common problem: different log systems need different parameters. However, we often do not know how to specify these parameters for different log systems. Although Spell does not need to tune the parameters by users, it is not adaptive to other systems. In our method, we design a nonlinear threshold criterion which is adaptive to most systems. We also partition the log messages according to their lengths to improve the efficiency.

7 Conclusion

With the continuous increase of log scale, online log parser is greatly desired now. In this paper, we propose Slop, an efficient and universal online streaming log parser. We improve the efficiency of Slop by grouping log messages into partitions. The message types extracted from different partitions are then combined and merged to guarantee its accuracy. We improve the universality of Slop by employing a nonlinear threshold criterion for message type extraction. We implement a prototype of Slop and conduct extensive experiments to evaluate its effectiveness and efficiency. The experimental results show that Slop outperforms the state-of-the-art log parsers in terms of accuracy and efficiency.

Acknowledgment. The research presented in this paper is supported in part by National Natural Science Foundation (No. 61602370, 61672026, 61772411, U1736205), Postdoctoral Foundation (No. 201659M2806, 2018T111066), Fundamental Research Funds for the Central Universities (No. 1191320006), Natural Science Foundation of Shaanxi Province (No. 2016JM6040), Postdoctoral Foundation of Shaanxi Province, CCF-Tencent Open Fund WeBank Special Funding, CCF-NSFOCUS open Fund.

References

1. Lou, J.-G., Fu, Q., Yang, S., Xu, Y., Li, J.: Mining invariants from console logs for system problem detection. In: *USENIX Annual Technical Conference* (2010)
2. Lou, J.-G., Fu, Q., Yang, S., Li, J., Wu, B.: Mining program workflow from interleaved traces. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 613–622. ACM (2010)

3. Yamanishi, K., Maruyama, Y.: Dynamic syslog mining for network failure monitoring. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 499–508. ACM (2005)
4. Fu, Q., Lou, J.-G., Wang, Y., Li, J.: Execution anomaly detection in distributed systems through unstructured log analysis. In: Ninth IEEE International Conference on Data Mining, ICDM 2009, pp. 149–158. IEEE (2009)
5. Tang, L., Li, T., Perng, C.-S.: LogSig: generating system events from raw textual logs. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 785–794. ACM (2011)
6. Vaarandi, R.: A data clustering algorithm for mining patterns from event logs. In: 3rd IEEE Workshop on IP Operations & Management, IPOM 2003, pp. 119–126. IEEE (2003)
7. Makanju, A.A., Zincir-Heywood, A.N., Milios, E.E.: Clustering event logs using iterative partitioning. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1255–1264. ACM (2009)
8. Mi, H., Wang, H., Zhou, Y., Lyu, M.R.-T., Cai, H.: Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems. *IEEE Trans. Parallel Distrib. Syst.* **24**(6), 1245–1255 (2013)
9. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services, ICWS, pp. 33–40. IEEE (2017)
10. Du, M., Li, F.: Spell: streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining, ICDM, pp. 859–864. IEEE (2016)
11. Oliner, A., Stearley, J.: What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, pp. 575–584. IEEE (2007)
12. A. EC2. <https://aws.amazon.com/tw/ec2/>
13. He, S., Zhu, J., He, P., Lyu, M.R.: Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering, ISSRE, pp. 207–218. IEEE (2016)
14. Los Alamos National Security LLC: Operational data to support and enable computer science research, January 2018. <https://institutes.lanl.gov/data/fdata>
15. LogPAI Team: Loghub, January 2018. <https://doi.org/10.5281/zenodo.1147681>
16. Du, M., Li, F., Zheng, G., Srikumar, V.: DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298. ACM (2017)
17. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 117–132. ACM (2009)
18. Ding, R., et al.: Log2: a cost-aware logging mechanism for performance diagnosis. In: USENIX Annual Technical Conference, pp. 139–150 (2015)
19. Zou, D.-Q., Qin, H., Jin, H.: UiLog: improving log-based fault diagnosis by log analysis. *J. Comput. Sci. Technol.* **31**(5), 1038–1052 (2016)
20. Zhang, W., Bastani, F., Yen, I.-L., Hulin, K., Bastani, F., Khan, L.: Real-time anomaly detection in streams of execution traces. In: 2012 IEEE 14th International Symposium on High-Assurance Systems Engineering, HASE, pp. 32–39. IEEE (2012)
21. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.: Online system problem detection by mining patterns of console logs. In: Ninth IEEE International Conference on Data Mining, ICDM 2009, pp. 588–597. IEEE (2009)



Community Discovery of Attribution Trace Based on Deep Learning Approach

Jian Xu^{1,2}, Xiaochun Yun^{1,2,3}, Yongzheng Zhang^{1,2(✉)}, and Zhenyu Cheng^{1,2}

¹ Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093, China
{xujian,zhangyongzheng,chengzhenyu}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences,
Beijing 100093, China

³ National Computer Network Emergency Response Technical Team/Coordination
Center of China, Beijing 100093, China
yunxiaochun@cert.org.cn

Abstract. In order to prevent potential network crime and halt attackers' operation further, collecting information to profile attackers is helpful. Because this exposes the identity of attackers, as well as provides IOC (Indicator of Compromise) to confirm whether devices have been compromised. In this information searching procedure, finding unknown information based on the existing ones is of crucial importance, because it leads to a more comprehensive profile about the attackers. Usually, these information pieces about a particular attacker form a tight connected community. Thus, finding the correct community label for the new incoming information piece based on these existing ones is pivotal for iteratively discovering more unknown information about the attacker. To facilitate this process, we propose to adopt the promising deep learning method to community classification on attribution traces. First, we propose to employ deep learning on extracting attribution trace pattern and then use the fine-tuned DBN (Deep Belief Network) to model the existing communities. At last, we experimentally illustrate the effectiveness of the DBN model in finding the correct community labels by feeding it with test information pieces. The results demonstrate that deep learning is a powerful means for identifying the community label.

Keywords: Deep learning · Attribution trace · Network analysis
Community discovery

1 Introduction

With the highly developed global internet, a variety of network attacks are appearing daily, and this number is increasing [23, 25]. To counter these threats, Network traceback is a sound method, because it directly leads to the exposure of attackers. For the real application, supposing to continuously monitor the APT1 organization [22], we collect information pieces about the attackers,

such as malware MD5, IP address, email addresses used by them from public reports, Google, malware reverse engineering, and compromised device forensic. These information pieces are combined as traceback network to profile the APT1 organization and its attackers. We could exploit this traceback network to verify if machines are compromised by APT1 organization and its attackers through comparing the malware MD5 information pieces that are testified belonging to APT1 traceback network with the file MD5 found in end machine. If there exists a match, it indicates this machine has been compromised by this organization. The malware MD5 here we used are called the IOC (Indicator of Compromise). There are variety forms of IOCs, such as MD5, file existence, cookies, etc.

In general, tight connected information pieces are connected as relevant circles [24] and stay as a compact community. There exist more connections within the community rather than between communities. The more compact the community is, the more related these information pieces are to another. In the attribution trace network, these pieces within the community are strong relevant to the person or the organization we are investigating, while others may not much relevant. So as to continuously add new information pieces into this traceback network for comprehensive organization profiling, We iteratively research these information pieces within the community to discover other relevant but unknown pieces which possibly have been ignored hitherto by analyzers. We also have to determine whether the new incoming information piece belongs to this traceback network or not. Therefore, finding the correct community label for the information piece is of crucial importance as they may help to uncover previously unknown information about the attackers.

Deep learning, as the cutting-edge machine learning method, is very effective in extracting pattern. The pretraining selects better pattern layer by layer automatically. And these patterns extracted by deep learning are usually superior to hand-picked features [12]. Additionally, patterns extracted by pretraining is resistant to breakings launched by attackers. For example, in the past, once the features hand-picked by investigators were analyzed and understood, the attackers usually will modify their implemented features to escape the detection, which renders the previous research works ineffective and finally avoids the traceback mechanism. Based on these grounds, we employ pretraining to extract the community patterns from attribution trace network and model these communities through fine-tuning this model, powering the deep learning model to assign the new piece to the right community.

Our paper proposes a deep learning approach to find the correct community label for information pieces to fulfill the purpose of network attack traceback. To demonstrate the effectiveness of this approach, we use the Enron emails communication dataset to test the model trained by deep learning.

In summary, we make the following contributions to attack traceback in this paper:

- We utilize pretraining to extract connection patterns from both the connection weight and the connection structure of the trace network and demonstrate that the extraction of network trace pattern implemented by deep

learning, which learns optimized deep hierarchical representation. This overcomes the shortcomings accompanied with previous researches, which merely focus on individual features, such as vertex centrality, closeness centrality, eigenvector centrality and clustering coefficients.

- We validate the effectiveness of the deep learning model against the real email communication dataset. And hyper-parameters (i.e. L_2 weight and sparsity), which greatly influence the model, are also discussed and optimized.

The rest of the paper is organized as follows. In Sect. 2, related literatures are reviewed, including the methods based on mesoscopic features, block-modeling and NMF (Nonnegative Matrix Factorization). Our deep learning approach are presented in Sect. 3. We outline our experiments in Sect. 4, and present experimental results. We close with a discussion about the work in Sect. 5.

2 Related Work

Community discovering has been extensively investigated in social network research area [1, 3, 5]. However, adopting community discovery to attribution trace network is a relatively emerging research area.

The traditional methods are usually based on the mesoscopic features, e.g. modularity, network vertex degree and connection weights. Seldom concerns have been given to applying deep learning for trace community discovery. Blondel et al. introduced an efficient modularity-based algorithm which finds communities in large networks [2].

Ferrara analyzed the community structure of the Facebook social network using the mesoscopic feature. It included the statistical features of the meta-network, such as density, average degree, connection weight, and it unveiled the communities representing the aggregation units where users stay together and interact [7]. Emilio Ferrara also presented LogAnalysis, which is a semi-supervised detection of criminal communities in networks reconstructed from phone call records. It analyzed degree centrality, vertex between centrality, closeness centrality, eigenvector centrality and clustering coefficient to profile the criminal communities. The system unveiled a few primary characteristics of criminal communities in real world phone call networks [8]. Most of their works focused on employing local features of vertex to detect community structures. Our work mainly utilizes the connection structure and connection weights of vertices.

De Meo et al. worked on methods to determine whether two vertices belong to the same community according to their similarity, which is based on the knowledge of common connected vertices or vertex group, as well as the analysis of social events in which users are involved [6]. They also proposed CONCLUDE (Complex Network Cluster Detection). It couples the accuracy of global approaches with the scalability of local methods by figuring out the edge importance which keeps the network connected. This edge centrality enables vertices mapping to Euclidean space. And the distance among the points in this space

is employed to discover communities [15]. Their work is somewhat similar to ours, we both exploits the vertex similarity to determining whether two vertices belongs to the same community. However, they used features of the edge centrality, while we employ deep learning to extract representations from connection structure.

Chen et al. tried to uncover crime community by using hierarchical clustering, and they exploited block-modeling to confirm the inter-connection among communities [4]. Their method is mainly a community cluster, while ours devotes to train deep network to recognize vertices belonging to the same community.

He et al. [9] and Jin et al. [10] employed stochastic model to detect communities through nonnegative matrix factorization. It aimed to find a nonnegative membership matrix to reconstruct the adjacency matrix of the network. Their objective functions are usually based on square loss function and Kullback-Leibler divergence. Their work used matrix factorization to discover community structures, while our approach can adapt the community model to new training vertices and continuously improve the community abstraction.

We are aware of Ishai Rosenberg' work, they proposed DeepAPT [20] to attribute malwares to its national developers using Deep Neural Networks (DNN). They recorded the running behaviors of malware in sandbox as raw input to the neural network, from which the DNN extracted features about the malwares. They tested set of 1,000 Chinese and Russian developed malwares, and achieved accuracy rate of 94.6%. Our work is different from theirs because we focus on profiling the attackers and their organization through continuously information mining. While their work concentrated on discovering the authorship for binary malware code.

3 Method

In this section, we propose the deep learning method, including the embedding preprocessing in Sect. 3.2, the deep learning in Sect. 3.3.

3.1 Definition

Attribution trace is defined as the abstract of attributable information piece. It is defined as a tuple, (v_i, v_j, e_{ij}) , which contains two vertices v_i, v_j and their connection e_{ij} . By combining traces with the common vertex, we define the trace network as a network $G = (V, E)$, where V is the set of all vertices, and E is the set of all connections. The network is a collection of all vertices and their connections.

3.2 Vertex Embedding

The input dimension of the deep learning is constant in both training and testing stages. Thus, the vertices in the attribution trace network should map into a space where each vertex's feature dimension is the same. And this mapping

function is requested to reserve the community characters of each vertex. To achieve this purpose, we utilize Deepwalk to embed vertices in the attribution trace network. The dimension of the embeddings is constant, which serves as the input of our deep learning model.

Deepwalk learns embedding of trace vertices by encoding their connection relations into a continuous vector space [18]. For each vertex in the trace network, the walk begins from itself, then adds a new vertex randomly chosen from the neighbors of the last visited vertex until the maximum walk length is met. Deepwalk then uses Word2vec to model and update the embedding of the vertices. Word2vec first represents each $v_i \in V$ as one-hot vector. Given a vertex walk $W^n = (v_0, v_1, \dots, v_n)$ in the random walks, where $v_i \in V$, and V is the set of trace vertices, Word2vec will maximize the $Pr(v_n | v_0, v_1, \dots, v_{n-1})$ over all the walks in the random walk set by exploiting neural network to predict the following vertex in each random walk [16]. After this process, the weights of hidden layer are used to present each vertex as a vector.

Deepwalk is a particularly computationally-efficient predictive model for learning vertex embeddings from the random walks, because if two vertices have the same connection context in the network, they tend to generate the similar random walks. It then embeds vertices in a continuous vector space where connection context similar vertices are mapped to nearby points. This captures the neighborhood similarity and community membership of each vertex in the attribution trace network.

The original implementation of Deepwalk does not exploit the connection weights among vertices. In our paper, we choose a new vertex in the random walk process with the probability proportional to their connection weights, which means if an adjacent vertex has greater weight than other adjacent vertices, it has a bigger chance that the chain walks to this vertex.

Algorithm 1. Deepwalk algorithm with connection weight

Input: $G(V, E)$, *dimension*, *walk.length*,

Output: *embeddings*

```

1: walks  $\leftarrow \emptyset$ 
2: for all  $v_i \in V$  do
3:   walk  $\leftarrow \emptyset$ 
4:   walk.add( $v_i$ )
5:    $v_{cur} \leftarrow v_i$ 
6:   for  $i$  in range(walk.length) do
7:      $v_{next} = rand(v_{cur}.adj, weights)$ 
8:     walk.add( $v_{next}$ )
9:      $v_{cur} \leftarrow v_{next}$ 
10:  end for
11: end for
12: embeddings = word2vec(walks, dimension)
```

The Deepwalk algorithm with connection weight is illustrated in Algorithm 1. It first implements random walks starting from each vertex in the network with the possibility proportional to the connection weight. And all the random walks generated in the first step constitute the walk set which the Word2vec model works on. Then Word2vec model consumes these random walks to update the vertex embeddings. In our research, we set the walk length to 15 vertices, and 200 walks are generated for each vertex. The final embedding dimension is 160.

3.3 Deep Learning

Deep learning is particularly helpful for extracting pattern from complex data [11]. This character especially suits the goal that we classify trace network community. Because the community character of attribution trace is a complex data format that is encoded as the connection structure and connection weight among different traces. The deep learning model extracts the pattern optimized for the community character through layer by layer pretraining. The training procedure of deep learning extracts the patterns which are perfect representations of vertex's community character. The training procedure contains two steps, pretraining and fine-tuning. The core unit of the pretraining stage is the autoencoder.

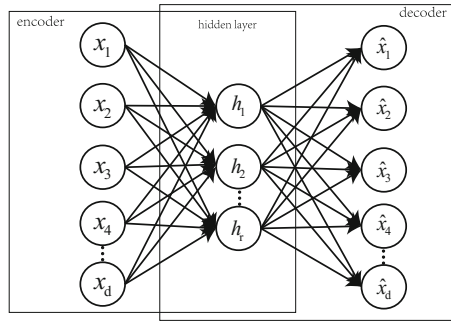


Fig. 1. Autoencoder unit

An autoencoder is comprised of an encoder and a decoder. The structure of autoencoder is depicted in Fig. 1. The encoder transfers the input to hidden pattern represented by the hidden layer output, while the decoder attempts to reverse this process by mapping the hidden pattern to its original input. We consider the vertex embeddings extracted by Deepwalk as $x \in \mathbb{R}^{D_x}$, then the autoencoder tries to replicate the input as \hat{x} according to Eq. 1.

$$\hat{x} = h_d\{w_d(h_e(w_e x + b_e)) + b_d\} \quad (1)$$

where h is the transfer function, w is the weight matrix and the b is the bias vector. The superscript e stands for the encoder layer, while the superscript d stands for the decoder layer.

The autoencoder learns its weights and biases by reconstructing the input through optimizing the cost function. Equation 2 is the cost function. The deep learning model updates the weights and biases through minimizing this cost function. The weights of the hidden layer learnt in this way are close to the global optimal weights, which are later used to transfer the input to the latent representations.

$$E = \frac{1}{N} \sum_{n=1}^N \sum_{d=1}^D (x_{dn} - \hat{x}_{dn})^2 + \lambda \Omega_{weights} + \beta \Omega_{sparsity} \tag{2}$$

where N is the number of vertices in every training batch, D is the vertex dimension that is determined by Deepwalk’s parameter, $\Omega_{weights}$ is the L_2 weight regularization and $\Omega_{sparsity}$ is the sparsity regularization. λ and β stand respectively for the coefficients of the L_2 weight regularization and the sparsity regularization.

The L_2 weight regularization is helpful in keeping the weights small. This is accomplished by summing the square of the weights. The overall term will be punished if this summation is large. This will prevent the model from overfitting the training traces [17].

$$\Omega_{weights} = \frac{1}{2} \sum_l^L \sum_j^N \sum_i^D (w_{ji}^l)^2 \tag{3}$$

where D is the vertex dimension, N is the number of vertices in each training batch and L is the number of neurons in the hidden layer.

The sparsity regularization penalizes the activation of each hidden neuron, $\hat{\rho}_i$, deviating significantly from the hyper-parameter, ρ , by imposing penalty term to the cost function. It will punish the cost function when the average activation value, $\hat{\rho}_i$, of the reconstruct layer neuron is swayed much from the desired value, ρ .

$$\Omega_{sparsity} = \sum_{i=1}^D \rho \log\left(\frac{\rho}{\hat{\rho}_i}\right) + (1 - \rho) \log\left(\frac{1 - \rho}{1 - \hat{\rho}_i}\right) \tag{4}$$

where ρ is a hyper-parameter to which we would like the average activation of each hidden neuron to be close, $\hat{\rho}$ is the activation of each hidden neuron. Here we use the Kullback-Leibler divergence function [13] as a difference measurement of $\hat{\rho}_i$ and ρ . The output of this function equals 0 when $\hat{\rho}_i$ and ρ are the same. D is the number of hidden neurons.

DBN is composed of a stack of encoders with a softmax layer as the final layer. These encoders are the front part of the pretrained autoencoders. The structure of DBN is demonstrated in Fig. 2. In DBN, each layer’s input serves as the visible layer, output serves as the hidden layer. Each hidden layer serves as the visible layer of the next layer [21]. DBN fulfills the training process through three steps. The first step is pretraining, which goes layer by layer. It employs unsupervised training to update the weights of the hidden units in each layer,

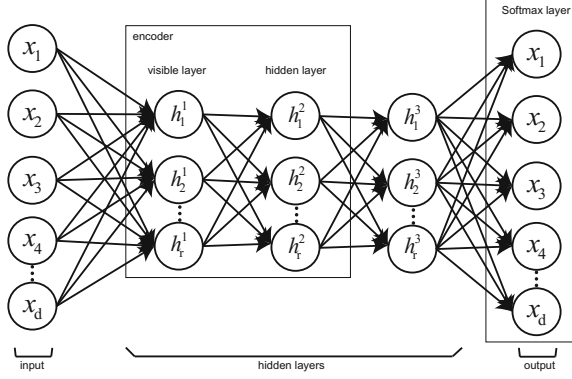


Fig. 2. DBN network

outputs the hidden patterns. During this step, each hidden layer in the DBN is considered as an autoencoder. Then it combines all the encoders trained in the first step and adds a softmax layer. The softmax layer squashes the output of the last hidden layer into vector $\sigma(z)$ of real values, where each entry is in the range $(0, 1)$, and all the entries add up to one. Equation 5 is the softmax function.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K \tag{5}$$

After the pretraining stage, a fine-tuning process is launched using the training traces and their community labels in a supervised training manner to re-train the whole neural network. This process treats all layers of the stacked autoencoders as a single model and precisely adjust the model weights to fit the training traces. This fine-tuning process utilizes the cross-entropy as the cost function, which is displayed in Eq. 6.

$$E(W, b) = -\frac{1}{N} \sum_{i=1}^N [y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)] \tag{6}$$

where N is the number of items in each training batch, the sum is over all training traces, and y is the corresponding target community class label. \hat{y} is the predicted community label of the deep learning model.

In this work, we use the backpropagation algorithm to compute the gradients for all layers displayed through Eqs. 7 to 10.

$$\delta^{n_l} = -(\nabla_{a^{n_l}} E) \bullet f'(z^{(n_l)}) \tag{7}$$

$$\delta^l = [(W^{l+1})^T \delta^{(l+1)}] \bullet f'(z^{(l)}) \tag{8}$$

$$\nabla_{W^{(l)}} E(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T \tag{9}$$

$$\nabla_{b^{(l)}} E(W, b; x, y) = \delta^{(l+1)} \tag{10}$$

Where \bullet denotes the element-wise product operator, $a^{(l)}$ is the output pattern vector in layer l , δ measures how much the node accounts for any errors in the final community output. $f(z)$ denotes the activation function of each layer. The algorithm first performs a feedforward pass. Outputs, $a^{(l)}$, of the hidden layer as well as the final are computed using each layer's forward propagation equation. Then $\nabla_W E(W, b; x, y)$ and $\nabla_b E(W, b; x, y)$ are the partial derivatives and they are calculated from the last layer back to the first layer. Finally, the W and b are updated according to Eqs. 11 and 12.

$$W_{ij} = W_{ij} - \alpha \frac{\partial}{\partial W_{ij}} E(W, b) \quad (11)$$

$$b_i = b_i - \alpha \frac{\partial}{\partial b_i} E(W, b) \quad (12)$$

After this training process, the network predict the labels for the test vertices.

4 Experiment

In this section, we first introduce the dataset we are going to investigate, and then we prune the dataset to extract the trace network. Deepwalk will be utilized to embed these vertices in the trace network. And the ground truth is obtained by applying modularity algorithm. We also discuss the optimization of the hyper-parameters of the deep learning model. And the model is trained from the test traces by applying these parameters. We finally present the deep learning's prediction result.

In this work, we use the Enron email dataset to prove the effectiveness of the proposed method [14]. This dataset contains 517,431 emails from 150 users distributed in 3,500 folders. Each email holds the sender and the receiver's email address, sent date and time, subject, email content and technical details about the email. The 150 users are mostly the Enron senior management members. We extract the email addresses and their connections from the raw email contents, and the extracted dataset contains 79,562 distinctive email addresses and 310,976 communications among them. After this, we further purge this dataset by dropping these emails that are not ended with @enron.com, limiting this investigation to the Enron company. By this time, the dataset is left with 32,190 addresses and 200,534 connections. Each of the connection bears a weight standing for the number of emails communicated by the two email addresses. And we consider the communication undirected, which means the connection weight is the summation number of two direction sent mails. The summary of the dataset is illustrated in Table 1.

We combine these traces in this pruned dataset to compose the trace network we are going to investigate. To evaluate the deep learning model's effectiveness, we employ the modularity algorithm to partition the traces in order to act as the ground truth of our deep learning approach. Modularity separates vertices to communities according to the density of connection within communities. It

Table 1. Dataset summary

Dataset	Vertices	Connections
Original dataset	79,562	310,976
Restrained to @enron.com	32,190	200,534

assumes that a community contains dense connections between vertices within the community, while there are sparse connections among different communities. The community modularity is defined in Eq. 13

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) \tag{13}$$

In the equation, k_v is the degree of v and k_w is the degree of w . m is the total number of connections in the trace network; A_{vw} is the actual number of connections between vertex k_v and k_w . Q is the modularity index. $\delta(c_v, c_w)$ is a function that equals 0, when vertices v and w are in different communities, and equals 1, when they reside in the same community. c_v and c_w are the community label for v and w , respectively.

The modularity algorithm tries to assign traces to different communities by maximizing the modularity index according to Eq. 13. We employ this modularity algorithm to partition the Enron dataset into communities. The communities found by this algorithm is depicted in Fig. 3.

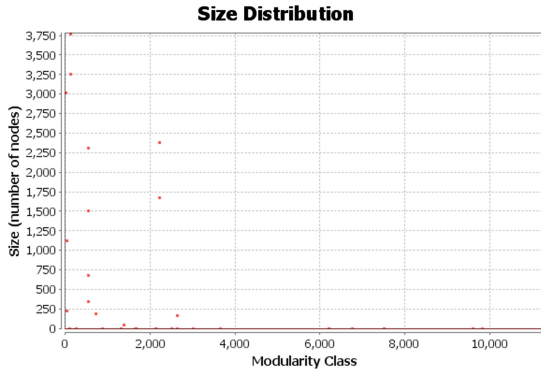


Fig. 3. Modularity based communities

From Fig. 3, we could figure out that most communities rarely contain email addresses. Most traces are included in a few large communities. More precisely, it is shown that the 10 biggest communities each contain over 250 vertices. Additionally, the 149 senior members (There exists a duplicated address in the original 150 email addresses) are scattered in these 10 communities. Based on this

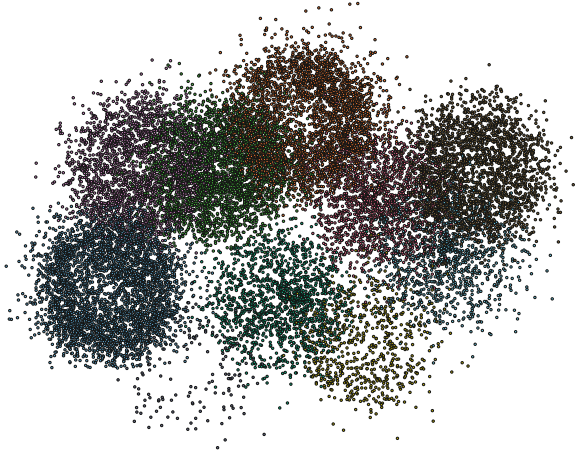


Fig. 4. Email communities

observation, we further limit the dataset to the 10 largest communities. These vertices pertaining to the 10 communities are reserved. It leaves us a dataset with 19,995 vertices and 196,197 connections. We apply this dataset as the ground truth to evaluate the effectiveness of community discovering based on the deep learning approach. Figure 4 is an illustration of the communities separated by modularity algorithm. Table 2 is a breakdown of the statistics about the 10 communities.

The DBN we employed is made up of two encoders as the hidden layers and a softmax layer as the output layer. The number of neurons for the first and second hidden layers are 100 and 80 respectively. The summary of the network parameters is displayed in Table 3.

Table 2. 10 large communities

Community ID	Email addresses	Senior manager emails
1	3,029	55
20	1,133	41
93	3,779	9
102	3,263	2
510	696	1
513	2,322	16
514	1,514	4
2191	1,689	7
2198	2,390	13
2618	180	1

Table 3. DBN parameter

Parameter		Value
Hidden layer	Train function	trainscg
	Cost function	msespase
	L_2 weight	$1.5583e-5$
	Sparsity	0.0485
Softmax layer	Train function	trainscg
	Cost function	crossentropy
Fine tune	Train function	trainscg
	Cost function	crossentropy

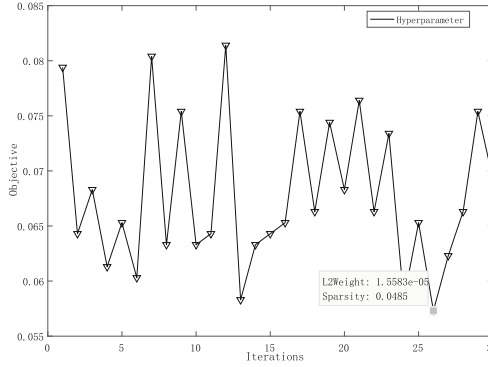


Fig. 5. Hyperparameters

We use scaled conjugate gradient backpropagation (SCG) as the training function for all layers. It updates weights along the conjugate directions which produces a faster convergence. The msesparse is a mean squared error function adjusted by L_2 weight and sparsity regularization according to Eq. 2 we have discussed in Sect. 3.3. The softmax layer and the final fine-tune stage are both supervised learning process, thus we adopt cross-entropy function to measure the difference between the prediction and the target.

The hyper-parameters, L_2 weight Regularization and Sparsity Regularization, influence the prediction greatly, which is significant for the DBN to work properly. Thus, we conduct experiment to find the optimal values by trying different configurations. The search range for L_2 weight Regularization is $[1e-5, 1e-4]$ and Sparsity Regularization is $[1e-5, 1e-1]$. This optimization process iterates 30 times. The results are depicted in Table 4. Figure 5 is the plot of the objective value we are trying to minimize.

Table 4. Hyper-parameter optimization

Iter	L_2 weight	Sparsity	Objective	Iter	L_2 weight	Sparsity	Objective	Iter	L_2 weight	Sparsity	Objective
1	9.43E-05	0.000104	0.0794	11	1.05E-05	0.00893	0.0643	21	1.02E-05	0.00679	0.0764
2	5.73E-05	0.0272	0.0643	12	1.71E-05	0.0317	0.0814	22	1.01E-05	0.00514	0.0663
3	1.26E-05	0.000397	0.0683	13	1.01E-05	0.00677	0.0583	23	1.01E-05	0.00835	0.0734
4	1.59E-05	0.0309	0.0613	14	1.54E-05	0.0273	0.0633	24	1.01E-05	0.00513	0.0583
5	4.61E-05	0.0394	0.0653	15	1.02E-05	0.00458	0.0643	25	4.59E-05	0.0393	0.0653
6	1.00E-05	0.00846	0.0603	16	1.56E-05	0.0486	0.0653	26	1.56E-05	0.0485	0.0573
7	1.01E-05	0.0173	0.0804	17	4.38E-05	0.0134	0.0754	27	4.31E-05	0.00709	0.0623
8	4.27E-05	0.00707	0.0633	18	1.00E-05	0.00622	0.0663	28	4.37E-05	0.00715	0.0663
9	3.36E-05	0.00964	0.0754	19	1.02E-05	0.00772	0.0744	29	1.61E-05	0.0309	0.0754
10	1.04E-05	0.00833	0.0633	20	1.02E-05	0.00657	0.0683	30	5.78E-05	0.0271	0.0693

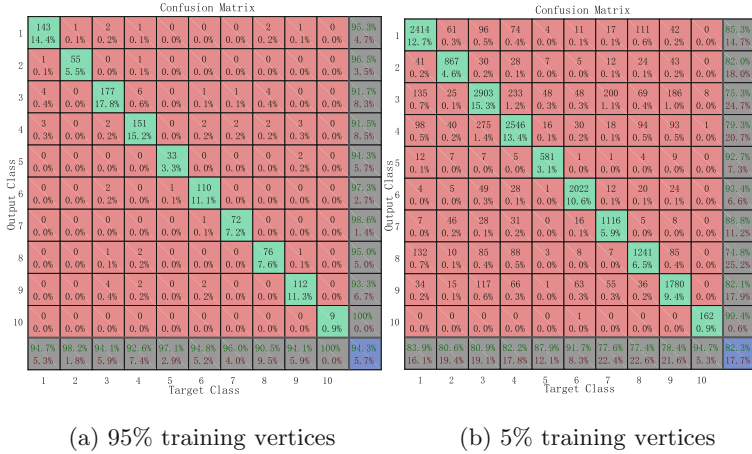


Fig. 6. Recognition result

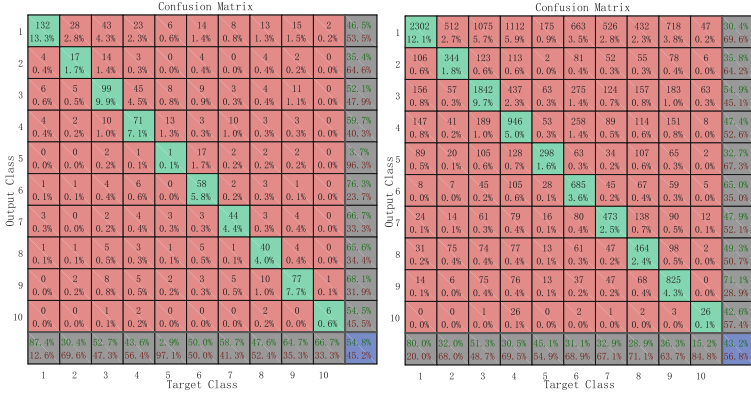
From Fig. 5 and Table 4, we could conclude the proper values of L_2 weight and Sparsity are $1.5583e-05$ and 0.0485 , the corresponding minimized error objective is 0.0573 .

To evaluate the model, we randomly choose 95% of 19,995 vertices from each of the 10 communities to train the DBN network, and then we use the remaining 5% vertices to test the prediction of the learnt deep learning model. The result is quiet promising, and it is demonstrated in Fig. 6(a). The model predicts the community label correctly for 94.3% of test vertices.

And then we further delve into evaluating the effectiveness of this deep learning network when it is fed with small percent of training vertices. We randomly choose only 5% of 19,995 vertices from each of the 10 communities as the training samples in this configuration. And the result is promising, the deep learning network performs very well even under this harsh condition. The result is demonstrated in Fig. 6(b). It assigns the community label correctly for 82.3% of 18,996 test vertices.

To demonstrate the effectiveness of deep learning approach, we compare our approach with the kNN (k nearest neighbor) classification [19]. We set the parameter of kNN to 4 neighbors. The same training and testing dataset setting is used to test kNN. The results are presented in Fig. 7(a) and (b). The overall accuracy is 54.8% when kNN is trained with 95% of vertices, and it drops to 43.2% when trained with 5% of vertices.

The accuracy of kNN depends on the parameter choice of k nearest neighbors. Usually, larger value reduces effect of the noise, but make boundaries between classes less distinct. To mitigate the effects of this parameter, we alternate the nearest neighbors from 1 to 10 in order to test the capability of kNN. The result is in Table 5. It shows the accuracy fluctuates around 55%, and the kNN can at best achieve accuracy of 55.68% when the nearest neighbors are set to 7.

(a) 95% training vertices, $k=4$ (b) 5% training vertices, $k=4$ **Fig. 7.** kNN recognition result**Table 5.** kNN optimization (95% training vertices)

Neighbors	1	2	3	4	5
Accuracy	54.47%	51.46%	55.08%	54.77%	55.58%
Neighbors	6	7	8	9	10
Accuracy	55.58%	55.68%	54.87%	54.57%	54.47%

Seen from results of both deep learning and kNN, deep learning is stronger in extracting community patterns and finding the right community for new traces with great higher accuracy.

5 Conclusion

In conclusion, we proposed a deep learning based approach to discover the community label of attribution traces. The experimental results show that although trained with a small training set, this approach still could produce the promising result. It demonstrates that the deep learning method is applicable in the network-based trace analysis, and it would become a prominent method for trace analysis because of the following reasons. First, deep learning methods use pre-training to automatically extract the patterns, which are hand-picked by experts in the past. The extracted patterns usually perform better than the hand-picked patterns, because the pretraining considers all features that could be utilized to generate patterns. Second, this pretraining and classifying process are not easy to break. Thus, it is hard for attackers to modify features to avoid being detected, so attacker will not come out with countermeasures to avoid this trace-back mechanism. It is worth noting that we used relatively small deep learning structure with 2 hidden layers of 100 and 80 respectively. The result would

become much more promising if we deeper the layers and enlarge the neurons with more training iterations. In our future works, we would further rank vertex importance within the community. Rather than limited to the email communication connections, we also would consider to construct the trace network from variety of connections, such as sharing the same IP address, co-authoring the malware, attending the conference, working for the same organization, etc.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (No. U1736218).

References

1. Bedi, P., Sharma, C.: *Community Detection in Social Networks*. Wiley, Hoboken (2016)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech.* **2008**(10), 155–168 (2008)
3. Catanese, S., Ferrara, E., Fiumara, G.: Forensic analysis of phone call networks. *Soc. Netw. Anal. Min.* **3**(1), 15–33 (2013)
4. Chen, H., Chung, W., Xu, J.J., Wang, G., Qin, Y., Chau, M.: Crime data mining: a general framework and some examples. *Computer* **37**(4), 50–56 (2004)
5. Chopade, P., Zhan, J., Bikdash, M.: Node attributes and edge structure for large-scale big data network analytics and community detection. In: *IEEE International Symposium on Technologies for Homeland Security*, pp. 1–8 (2015)
6. De Meo, P., Ferrara, E., Fiumara, G.: Finding similar users in Facebook, pp. 303–324 (2011)
7. Ferrara, E.: A large-scale community structure analysis in Facebook. *EPJ Data Sci.* **1**(1), 1–30 (2012)
8. Ferrara, E., Meo, P.D., Catanese, S., Fiumara, G.: Detecting criminal organizations in mobile phone networks. *Expert Syst. Appl.* **41**(13), 5733–5750 (2014)
9. He, D., Liu, D., Jin, D., Zhang, W.: A stochastic model for detecting heterogeneous link communities in complex networks (2015)
10. Jin, D., Chen, Z., He, D., Zhang, W.: Modeling with node degree preservation can accurately find communities. *New Media Soc.* **18**(7), 1293–1309 (2016)
11. Krizhevsky, A., Hinton, G.E.: Using very deep autoencoders for content-based image retrieval. In: *Proceedings of the European Symposium on Artificial Neural Networks, ESANN 2011, Bruges, Belgium, 27–29 April 2011* (2012)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: *International Conference on Neural Information Processing Systems*, pp. 1097–1105 (2012)
13. Kullback, S., Leibler, R.A.: On information and sufficiency. *Ann. Math. Stat.* **22**(1), 79–86 (1951)
14. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Math.* **6**(1), 29–123 (2008)
15. Meo, P.D., Ferrara, E., Fiumara, G., Provetti, A.: Mixing local and global information for community detection in large networks. *J. Comput. Syst. Sci.* **80**(1), 72–87 (2014)
16. Mikolov, T., et al.: Efficient estimation of word representations in vector space. In: *International Conference on Learning Representations*, pp. 1–12 (2013)

17. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vis. Res.* **37**(23), 3311–3325 (1997)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations, pp. 701–710 (2014)
19. Peterson, L.: K-nearest neighbor. *Scholarpedia* **4**(2), 1883 (2009)
20. Rosenberg, I., Sicard, G., David, E.O.: DeepAPT: nation-state APT attribution using end-to-end deep neural networks. In: Lintas, A., Rovetta, S., Verschure, P.F.M.J., Villa, A.E.P. (eds.) *ICANN 2017. LNCS*, vol. 10614, pp. 91–99. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68612-7_11
21. Schölkopf, B., Platt, J., Hofmann, T.: Greedy layer-wise training of deep networks. In: *International Conference on Neural Information Processing Systems*, pp. 153–160 (2006)
22. Segal, A.: Mandiant: APT1: exposing one of china’s cyber espionage units
23. Wheeler, D.A., Larsen, G.N.: Techniques for cyber attack attribution (2003)
24. Xu, J., Yun, X., Zhang, Y., Sang, Y., Cheng, Z.: NetworkTrace: probabilistic relevant pattern recognition approach to attribution trace analysis. In: *2017 IEEE Trustcom/BigDataSE/ICISS*, pp. 691–698, August 2017. <https://doi.org/10.1109/Trustcom/BigDataSE/ICISS.2017.301>
25. Yao, S., Chen, J., Du, R., Deng, L., Wang, C.: A survey of security network coding toward various attacks. In: *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 252–259 (2014)



Examine Manipulated Datasets with Topology Data Analysis: A Case Study

Yun Guo¹, Daniel Sun^{2,3}, Guoqiang Li^{2(✉)}, and Shiping Chen⁴

¹ Department of Computer Science and Technology,
Shanghai Jiao Tong University, Shanghai 200240, China
gy2016@sjtu.edu.cn

² School of Software, Shanghai Jiao Tong University,
Shanghai 200240, China
li.g@sjtu.edu.cn

³ Data61, CSIRO, Canberra, ACT 2601, Australia
daniel.sun@data61.csiro.au

⁴ Data61, CSIRO, Sydney, NSW 1710, Australia
shiping.chen@data61.csiro.au

Abstract. Learning and mining technologies have been broadly applied to reveal the value of tremendous data and impact decision-making. Usually, the correctness of decisions roots in the truth of data for these technologies. Data fraud presents everywhere, and even if data were true, could data be maliciously manipulated by cyber-attackers. Methods have been long exploited to examine data authenticity, but are less effective when only values are manipulated without violating scopes and definitions. Then the decisions made from fraud and manipulated data are wrong or hijacked. It has been concluded that data manipulation is the latest technique in “the art of war in cyberspace.” Examining each data instance from its source is exhaustive and impossible, for example recollecting data for national consensus. In this paper, through a case study on the data of banknotes, we exploit *Topological Data Analysis (TDA)* for examining manipulated data. A fraction of data records are examined integrally other than individually. The possibility of using TDA to verify data efficiently is then evaluated. We first test the possibility of using TDA for the above detection, and then discuss the limitations of the state of the art. Although TDA is not so matured, it has been reported to be effective in many applications, and now our work evidences its usage for data anomalies.

Keywords: Data manipulation · Topological features · TDA · Mapper

1 Introduction

Data manipulation refers to selecting, inserting, deleting and updating data in databases, but the term in cyber space refers to that hackers can infiltrate networks, find their way into databases and change information contained in them.

In normal cases, data qualification technologies can be used to examine the values of qualitative or quantitative records, and the violence of logic, numeric scopes, and semantics. In some advanced cases, anomaly detection via learning and mining techniques is effective for outliers, novelties, noise, deviations and exceptions. However, all of these methods are less effective if manipulated data are still with reasonable attributes, such as range, definition, and semantics. For such new records, directly detecting on single record is difficult. Thus we heuristically use mutual relationship between data records to detect on the whole dataset: check whether this dataset contains manipulated records or not.

To utilize the mutual relationships, we resort to topology, which resulted in *Topological Data Analysis (TDA)*. The idea underlying TDA is employing the mutual relationship between data records to find the overall “shape” of a multi-dimensional dataset. The shape found by TDA represents some important topological features of the dataset, such as connectivity, holes, and voids (holes in high dimensions). These shape features are usually insensitive to “small” deformations, and include intrinsic insights of the dataset. TDA thus is able to get conclusions without being disturbing by data manipulations. With the help of TDA, we turn detecting on a single data record into task that finds an essential shape of the whole dataset, and use differences of the shapes to determine if a given dataset contains manipulated data records.

We carried out a case study on a realistic banknote dataset consisted of numerical records of genuine and forged banknotes. We manipulated the forged banknote data records by four types of operations, and used the manipulated datasets to confirm that manipulations of data values cannot change the overall shapes of the datasets. Our experiments were performed using an open source tool, Python Mapper¹, which is an approximation of the classical TDA. Outcomes of our experiments show that using topological shapes to reveal the existence of forged data records is possible. We find that the shapes of different types of datasets present distinct features. For the banknote dataset, the shape of the datasets consisted of totally genuine data records is topologically tree-like, while the shape of the datasets containing forged records has loops. In addition to detecting by topological shapes, we employed statistical learning algorithms to detect on all datasets introduced before. A discovery is that they can detect on the original dataset accurately, too. But for the detection of manipulated datasets, learning algorithms have a rather low accuracy.

2 Related Work

Data Fraud and Detection. Data fraud occurs in a wide range of fields, such as business, politics, science, and health care [13]. To prevent and detect data fraud, various attempts had been made. Most detection methods adopt data mining, knowledge discovery in databases, and machine learning methods. These solutions have been successfully applied in different areas of crimes. Fuzzy rules and neural networks are used in the detection of subscription fraud [9]; the hidden

¹ <http://danifold.net/mapper/index.html>.

Markov model [10] and neural networks [11] are used for card fraud detection; naive Bayesian methods are applied in detection of fraud claim diagnoses [25]. To detect web information fraud, such as fraud search rank and online advertising, multiple data mining methods were used in [18, 21]. Furthermore, [3] examined money laundering, computer intrusion, e-commerce fraud, credit cards fraud and telecommunications fraud, along with detection methods to address these problems.

TDA and Mapper. The initial motivation of TDA is to study the ‘shape’ of dataset. The most widely used tool of TDA is persistent homology (PH), which finds shapes of a dataset at a set of resolutions. The concept of PH was first introduced by Edelsbrunner et al. together with an efficient algorithm and its visualization in 2002 [8]. Carlsson et al. reformulated the initial definition and proposed an equivalent visualization method named ‘barcodes’ [5]. There are some software applications of PH, such as Perseus, Dionysus, DIPHA, javaPlex and Gudhi [15] and several benchmarks for the computation of PH, [8] introduced the standard algorithm for the computation of PH in field of \mathbb{F}_2 and [28] extended it to the general field. Based on PH, TDA provides new insights in the study of data and has been successfully applied in numerous fields, including coverage in sensor networks [23], protein structure [27], stability of fullerene molecules [26], robotics [2, 20], breast-cancer classification [6, 16], and pattern recognition for point cloud data [4] as well as machine learning [1].

Mapper is a special tool of TDA. It can transform a high dimensional dataset into a simplicial complex with far fewer points, while still capturing topological features of the original dataset [24]. It was first introduced by Carlsson et al. for visualization of high-dimensional data in [24]. Mapper has been successfully used for finding subgroups of different data types, such as gene expression of breast tumors, voting data from the US House of Representatives, and player performance data of the NBA [14], breast cancers [17]. Gurjeet Singh, Facundo Mémoli and Gunnar Carlsson applied Mapper for 3D object recognition [24], Aleksandar Savic, Gergely Toth and Ludovic Duponchel applied Mapper for revealing pedogenetic principles of European topsoil systems [22]. Moreover, Tamal K. Dey and Yusu Wang proposed multiscale Mapper [7]. The company, Ayasdi, applied Mapper into its main product, the Ayasdi Iris software. Daniel Mllner realized an open version of Mapper, Python Mapper.

3 Background

TDA uses simplicial complex to study the shape features of a data space. The simplicial complex is a basic concept of combinatorial topology, and an approximation of the original space but has the same topological features as the original space. The simplicial complex of a given dataset is constructed basing on the mutual relationship between data records. Intuitively, the simplicial complex consists of simplexes of basic geometry components, such as nodes, edges, triangles and tetrahedrons in R^3 . The node is the most basic component of a simplex.

A k -dimensional simplex is composed of $k + 1$ affinely independent nodes and is called a k -simplex. Formal definitions are given in Appendix A.

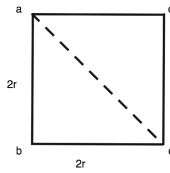


Fig. 1. An example of simplex and simplicial complex. 0-simplexes are nodes: $\{a, b, c, d\}$, 1-simplexes are edges: $\{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}$, then the simplicial complex is the union of them as $\{\{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}$.

Given a dataset S , there are various ways to construct the complex of S , such as the Vietoris-Rips(VR) complex and the Čech complex. We use the VR complex as an example to show how to construct a complex by these classical methods and show a resulting complex in Fig. 1. The VR complex is defined as

$$VR_\varepsilon(S) = \{\sigma \subseteq S \mid d(x, y) \leq 2\varepsilon \text{ for all } x, y \in \sigma\},$$

where the subset $\sigma = \{x_0, \dots, x_k\}$ of S is a k -simplex if and only if the distance of every pair of vertices in $\{x_0, \dots, x_k\}$ is smaller than 2ε . According to the definition of the VR complex with $\varepsilon = r$, the edges $\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}$ in Fig. 1 are 1-simplex since the distance between their endpoints are equal to $2r$, while $\{a, c\}$ is not a 1-simplex since the distance of $\{a, c\}$ is $2\sqrt{2}r$. For the same reason, datasets $\{a, b, c\}$ and $\{a, d, c\}$ in Fig. 1 do not compose a 2-simplex. Thus, there is one hole in this complex and this shape feature exists as long as the range of ε is $[r, \sqrt{2}r)$.

In practice, the above methods have not been widely used for large-scale problems due to the huge complexity of computation. To construct the VR complex one has to check for pairwise distances, and this leads to tremendous computation if there are too many simplexes. [19] has pointed out that the VR complex can have simplex with dimension $|S| - 1$ and thus has up to $2^{|S|} - 1$ simplexes in the worst case. Hence, approximation is usually considered. In this paper, we employ Mapper [24] to compute the complex. For an input X that possibly is a high dimensional dataset, Mapper first maps the original data into a rather lower dimensional space via a k -dimensional filter function f defined on X , and then constructs the complex basing on the codomain of the filter function [24], thus reduces the computation. The detailed procedure of Mapper are introduced in the Appendix B.

4 Applying Mapper Complex in Detection

4.1 Problem Statement

We intend to solve the problem that original forged records are deliberately manipulated again. In this case, the forged records are manipulated basing on

some information about the genuine records, which makes a single manipulated forged record much closer to the genuine records than the original one, thus directly detecting on the single record becomes difficult. To address this problem, we employ mutual relationship between data records to detect on the whole dataset. We introduce Mapper into our detection tasks and use it to construct complex shape of a dataset. The procedure of detection using Mapper can be divided into two stages: (1) finding the valid parameters by means of some historical datasets. (2) constructing the topological shapes of new datasets and comparing the resulting shapes with that of the historical datasets. In the rest of this paper, we denote the historical datasets as training datasets and the new datasets as testing datasets. The valid parameters make the topological shapes of different-type historical datasets distinguishable, and let same features appear in same-type complex shape. Thus, we can use the complex shape of a testing dataset to determine its type: contains forged records (forged), or does not contain (genuine). In our case study, a testing dataset is determined to be genuine if and only if its complex shape only contains features appeared in the historical genuine datasets.

4.2 An Example for Detection

We evaluate our method on a banknote dataset. Our source dataset consists of 762 genuine banknotes records and 610 forged banknotes records. Data record in this dataset has four attributes, which were obtained by applying the wavelet transformation on the original banknote images². We use a part of the original records as training data to find the valid parameters, and test our method on the datasets generated by the remaining records. Detailed experimental settings can be seen in Sect. 5.1. In this part, we illustrate the meaning of the complex shapes, and give an example for detection.

Figure 2 shows the complex shape of three datasets. The complex shape is composed of nodes and edges that connect the nodes. Nodes stands for a cluster of data records that have same range of filter values. The nodes are colored by the average filter value of records in it, red implies high, blue implies low. Number in a node represents the amount of records in this node, and nodes with bigger number have bigger size. Two nodes are connected if there are overlaps in their records. All of them are constructed under parameters $KNN_distance$ filter with $k = 18$, $I = 45$, $O = 30$, $gapsize = 0.2$.

Figure 2(a) is the complex shape of a dataset with 200 genuine records, Fig. 2(b) is the complex shape of a dataset with 200 forged records. The main part of Fig. 2(a) (marked out by the red line) has the following features: (1) The shape of the complex is tree-like, (2) Records of this genuine dataset mainly scatter on the nodes in the tree-like part, (3) The node color in the tree-like part is non-unique. While the totally different notable features of the main parts of Fig. 2(b) (marked out by the blue line) are: (1) Loops, (2) Records of this forged dataset mainly concentrate on the nodes in the loops, (3) Almost all nodes in

² <http://archive.ics.uci.edu/ml/datasets/banknote+authentication>.

the loops are purple. Thus, the complex shape of the genuine dataset and forged dataset can be distinguished by these features. Figure 2(c) is the complex shape of one testing dataset with 200 forged records. The main part (marked out by blue line) of it has the following features: (1) Loops, (2) Most data concentrate on these loops, (3) Almost all nodes in the loops are purple. All of these features appeared in the forged training dataset's complex shape. Thus, we can infer that forged data exist in this dataset.

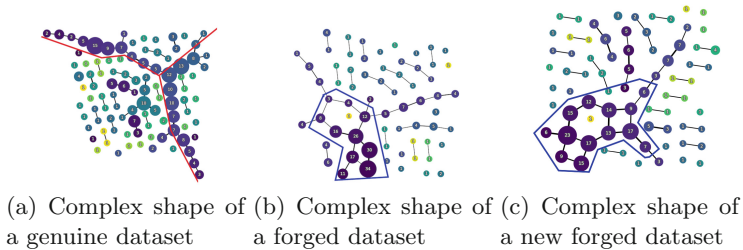


Fig. 2. Example of complex shapes of banknote data.

4.3 The Determination of Parameters

The determination of parameters remains an open issue, we introduce some empirical rules to tune the parameters for the detection tasks. In this paper, all parameters of Mapper are divided into two parts. The filter function and its parameter, denoted as $P1$; the cover parameters and clustering parameters, denoted as $P2$. The determination of parameters is roughly aggregated into two phases.

In the first phase, preliminary results of $P1$ are computed. Requirement for $P1$ at this stage is to make the distributions of the filter values of different-type datasets different. For the banknote data, we make most of the forged data to have relatively smaller filter values, while the genuine data to have randomly distributed filter values. Accordingly, eccentricity with $P = 1.0$ and $kNN_distance$ with $k = 18$ are selected. However, the valid $P1$ we found here may be invalid for other datasets, that is, they cannot make the notable shape features to be contained in a set of same-type datasets.

In the second phase, all parameters of $P2$ are determined as a whole, and the preliminary $P1$ are further checked. $P2$ are required to enable the complex shape of the forged datasets to differ from that of the genuine datasets. Our discovery is that parameters with relatively larger I , larger *gap size* and smaller O could be effective. At such combination, the complex shape of a dataset with centralized filter values contains loops, while that of a dataset with randomly distributed filter values contains no loops. For the verification of a given $P1$, $P2$ are adjusted until same notable shape features presenting in all of the same-type datasets. Figure 3 shows that the tree-like feature is hold in different genuine

datasets and the loops are hold in different forged datasets at $KNN_distance$ with $k = 18$. Thus, $KNN_distance$ with $k = 18$ is valid at last.

5 Experiment and Discussion

5.1 Datasets and Experimental Setup

We denote the source dataset (Sect. 4.2) as $BIData$. For evaluating our method, we generate 49 datasets based on it, the resulting datasets are as follows.

Training Datasets (Table 1). Our training datasets contain two types: $train0$ for training ML models and the others for finding the valid parameters of Mapper. Data records of $train0$ were randomly selected from $BIData$. Data records of $train0i$ and $train1i$ were sampled from $train0$'s genuine part and forged part, respectively.

Original Testing Datasets (Table 2). The rest part of $BIData$ is applied to generate the original testing datasets $test00$ and $test10$, whose records differ from records of $train0$. We use them to show that original testing datasets have the same topological features as that of the training datasets.

Manipulated Testing Datasets (Table 2). They are generated by conducting manipulations on the records of $test10$. Their function is to prove our assumptions that manipulations on the whole dataset cannot change its topological shape. The manipulations are shown as follows:

- (1) $-0.96 * x[1 : 4] + 0.6, x_3 = -x_3, x_4 = -x_4,$
- (2) $-0.96 * x[1 : 4], x_2 = -x_3, x_3 = x_2, x_4 = -x_4,$
- (3) $-0.74 * x[1 : 4], x_2 = -x_3 - 0.15, x_3 = x_2, x_4 = -x_4,$
- (4) $x[1 : 4] + mean(test00 - test10).$

Where x is one data record, x_i is one attribute of x , $i \in \{1, 2, 3, 4, 5\}$, and x_5 records the category (genuine or forged) of one data record.

Mixed Testing Datasets (Table 3). They are generated by mixing the records of $test00$ with that of $test1i$, proportions of each types of the records are detailed in (Table 3). This case is the truest simulation of the fraud, since the datasets that we encounter with in reality usually consisted of both genuine records and forged records. We can thus use their results to show the ability of our method in the realistic situation.

Table 1. Training datasets

Name	train0	train01	train02	train03	train11	train12	train13
Usage	ML	TDA	TDA	TDA	TDA	TDA	TDA
Number	982	200	200	200	200	200	200

Table 2. Original and manipulated testing datasets

Name	test00	test10	test11	test12	test13	test14
Manipulated	0	0	1	1	1	1
Number	200	200	200	200	200	200

Table 3. Mixed testing dataset.

Name	$t1i1$	$t1i2$	$t1i3$	$t1i4$	$t1i5$	$t1i6$	$t1i7$	$t1i8$	$t1i9$
Percentage of $test00$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Percentage of $test1i$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Total number	200	200	200	200	200	200	200	200	200

We use Python Mapper to construct a complex and visualize it. Parameters of Python Mapper are set as follows, $KNN_distance$ filter function with $k = 18$ and cover parameters $I = 45, O = 30$ and clustering parameter $gap\ size = 0.2$. The details about determination of the parameters are introduced in Sect. 4.3. We exploit $train01, train02, train03, train11, train12$ and $train13$ to find these parameters. After obtaining the valid parameters, we construct the complex shapes of all testing datasets and compare them with that of all of the training datasets.

5.2 Results of Mapper

Complex Shapes of Training Datasets. This experiment shows that the genuine datasets and forged datasets can be distinguished via the different shape features in their complex shapes. The genuine features in the main parts (denoted by the red line) of the complex shapes in Fig. 3(a), (b) and (c) are:

- (1) The shape of the complex is tree-like, denoted as δ_1 ;
- (2) Records mainly scatter on the nodes in the tree-like part, denoted as δ_2 ;
- (3) The nodes color in the tree-like part is non-unique, denoted as δ_3 .

The different forged shape features in the main parts (denoted by the blue line) of the complex shapes in Fig. 3(d), (e) and (f) are:

- (1) Loops, denoted as γ_1 ;
- (2) Records mainly concentrate on the nodes in the loops, denoted as γ_2 ;
- (3) Almost all nodes in the loops are purple, denoted as γ_3 .

We empirically adopted these notable features as the standard for judging the type of testing datasets in the subsequent experiments.

Detection on Original Testing Datasets. The insight that uses the complex shape features to determine the type of the testing dataset was proved to be feasible in this experiment. Because the genuine features and forged features

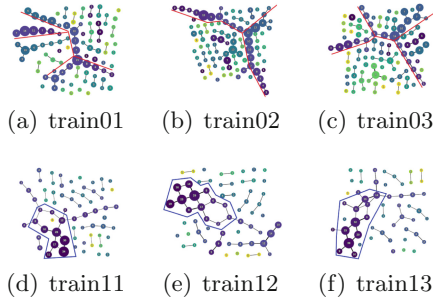


Fig. 3. The complex shapes of training datasets.

introduced before also appeared in the genuine testing dataset and forged testing dataset, respectively. As shown in Fig. 4, the genuine features $\delta_1, \delta_2, \delta_3$ appeared in the main parts (denoted by the red line) of *test00*'s complex shape and the forged features $\gamma_1, \gamma_2, \gamma_3$ appeared in the main parts (denoted by the blue line) of *test10*'s complex shape.

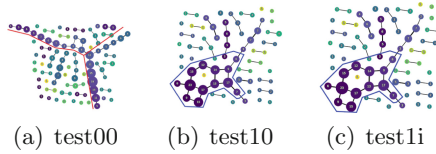


Fig. 4. The complex shape of original testing datasets and modified testing datasets

Detection on Modified Testing Datasets. This experiment shows that Mapper is valid for the detection on *test11, test12, test13* and *test14*. Figure 4(c) shows that the main parts (denoted by the blue line) of the complex shapes of *test11, test12, test13* and *test14* contain the forged features $\gamma_1, \gamma_2, \gamma_3$. Moreover, the complex shapes of *test11...test14* are identical to each other as well as the complex shape of *test10*, indicating that the manipulations on a whole dataset were unable to change its topological structure.

Detection on Mixed Testing Datasets. In this case, Mapper was valid for 28 of 36 datasets. There are 4 invalid cases in *t13i*, 3 invalid cases in *t11i*, 1 invalid case in *t12i* and no invalid cases in *t14i*. The manipulations put on *t11, t12, t13* and *t14* have different degrees of influences on the forged records, indicating that the degrees of manipulations would influence the effectiveness of Mapper. We guess this is because the pairwise-distance was changed considerably for the different degrees of manipulations. For example, the values of the original forged data in *t13i* were multiplied by 0.7, and that of the *t12i* were multiplied by 0.96. The values of the forged data were manipulated with a larger degree in the former case, which resulted in smaller pairwise-distance of the manipulated dataset. In summary, our method is effective for most of the mixed datasets, and thus could

be effective for the real manipulations, since this case is the truest simulation of the realistic fraud. The following details the results. Figure 5 shows that Mapper was valid for the datasets $t111, t112, t113, t116, t118$ and $t119$ but was invalid for $t114, t115$ and $t117$. Figure 6 shows that Mapper was valid for $t121, t122, t123, t124, t125, t127, t128$ and $t129$ but was invalid for $t126$. Figure 7 shows that Mapper was valid for $t131, t132, t133, t136$ and $t138$ but was invalid for $t134, t135, t137$ and $t139$. Figure 8 shows that Mapper was valid for all datasets of $t14i, i \in \{1, \dots, 9\}$.

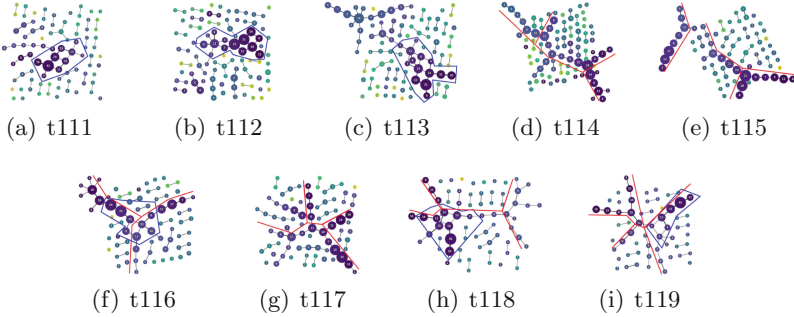


Fig. 5. The complex shapes of $t11i, i \in \{1, \dots, 9\}$.

5.3 Results of ML

Machine learning methods SVM with RBF kernel and the multilayer neural network (NN) were applied for our experiment. The penalty parameter C and Kernel coefficient $gamma$ of SVM were 1.0 and 0.0001. Our NN is a fully connected neural network with two hidden layers with ReLU (rectifier linear units) activation function and a SoftMax layer. Learning rate we used for training NN is 0.0001. We trained the ML models by using training dataset $train0$ and made the training accuracy of SVM to be 0.999 and that of NN to be 0.970. To evaluate the ML models, we computed the accuracy and false positive rate(FPR) of them on all testing datasets. In our case study, FPR can reveal how many forged records that have been manipulated were predicted as genuine by ML models.

Table 4 shows that the ML models trained before can work well on the original testing datasets. The accuracy values given by SVM and NN are larger than 0.965 on $test00$ and $test10$. The value of FPR given by SVM is 0.010 and that given by NN is 0.015, all of which are small enough. This result indicates that our models do not overfit on the training datasets, since the original testing datasets are composed of records differed from records in the training datasets. Table 5 shows that both ML models have a rather poor performance on the manipulated testing datasets for the large FPR values of them. The FPR values of the SVM model are close to 0.5 on $test11, test12, test13$, which means the SVM model misclassified half of the manipulated forged data records in these

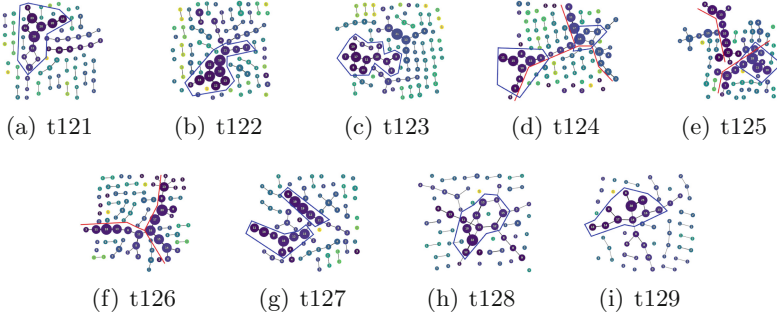


Fig. 6. The complex shapes of $t12i$, $i \in \{1, \dots, 9\}$.

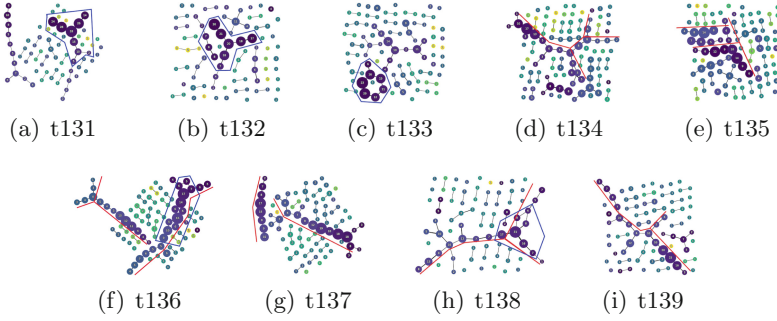


Fig. 7. The complex shapes of $t13i$, $i \in \{1, \dots, 9\}$.

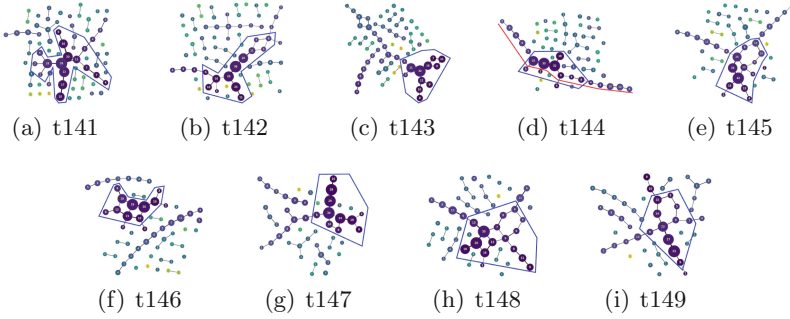


Fig. 8. The complex shapes of $t14i$, $i \in \{1, \dots, 9\}$.

datasets. The FPR values of NN model are close to 0.7 on *test11*, *test12*, *test13*, which makes the NN model to be more unavailable for these datasets than the SVM model. The FPR values of both models on *test14* are larger than 0.95. Thus, both models are totally unavailable for these datasets. Outcomes of the ML models on the manipulated datasets verified our assumption: if records were deliberately manipulated again basing on the information of genuine records, ML models trained on the historical data would be less effective.

Overall, the ML models can work well for 2 datasets that contain no manipulations; However, they are all less effective for datasets that contain manipulated data. Even if ML can detect some outliers in the datasets, it is risky to determine the overall data quality by the individual outliers: Manipulations do not repeat sufficient times in the real world for ML to effectively train models; Manipulations may change the original data in different way to achieve different purposes; Outliers are less effective to indicate data anomalies because values are changed within correct ranges. Unlike the statistical methods, Mapper can be valid in most cases because it does not need to enumerate the alter forms. Since most of the manipulations can not change the internal structure of the datasets.

5.4 Discussion

Our experiments on applying Mapper for detection of manipulated forged banknotes have shown that Mapper is effective at the dataset level. We found six notable shape features for classification between the genuine dataset and the forged dataset. And we carried out a set of experiments to test the effectiveness of Mapper on the testing datasets basing on the difference between the forged features and genuine features. These experiments showed that the detections on the original testing datasets and the manipulated testing datasets are accurate in all cases. We also found that the capacities of Mapper for the detections on the mixed datasets are different in the four cases. Mapper would be more effective when the values of the forged data were manipulated with more slightly manipulations. Overall, Mapper was valid for 33 of 41 testing datasets. We finally got an accuracy of 80.48% for Mapper at the dataset level. Note that Mapper can be applied to any datasets whose different-class subsets contain different topological structures. We just tested the ability of Mapper on the detection of forged banknotes.

Finally, we discuss three of the current limitations of applying Mapper for detection. (1) 100% correct results can not be given for the detection of mixed datasets. (2) Finding the effective parameters of Mapper is difficult. (3) Mapper is unable to give an outcome for each record, making one need to further check to determine which record was forged. These problems will be the focuses of our future work.

Table 4. Results of ML models.

		<i>test00</i>	<i>test10</i>
SVM	acc	0.965	0.990
	FPR	-	0.010
NN	acc	0.965	0.985
	FPR	-	0.015

Table 5. Results of ML models.

		<i>test11</i>	<i>test12</i>	<i>test13</i>	<i>test14</i>
SVM	acc	0.385	0.420	0.440	0.010
	FPR	0.615	0.580	0.560	0.990
NN	acc	0.160	0.205	0.160	0.025
	FPR	0.820	0.795	0.840	0.975

6 Conclusion

This paper investigated the possibility of applying TDA for examining the manipulated data. Our case study showed that the complex shape is a valid measurement for the detection. We found six notable shape features for classification between the genuine datasets and forged datasets. Through applying the complex shape constructed by Mapper for the detection of manipulated data, we found that most of the manipulated datasets can be detected. To be precise, Mapper detected the existence of forged data in 33 of 41 manipulated datasets. The 8 invalid cases will be one of our future focal points. On the whole, our case study revealed the ability of manipulated datasets.

Acknowledgements. This work is supported by the Key Program of National Natural Science Foundation of China with grant No. 61732013, and the Key R&D Project of Zhejiang Province with No. 2017C02036.

A Mathematical foundations of TDA

Definition 1 (Simplex). $p_0 \dots p_k \in R^d$ are affinely independent data points, and $p_1 - p_0, p_2 - p_0, \dots, p_k - p_0$ are linearly independent. Then, the simplex determined by them is the set of points.

$$C = \left\{ \alpha_0 p_0 + \dots + \alpha_k p_k \mid \sum_{i=0}^k \alpha_i = 1 \text{ and } \alpha_i \geq 0, \forall i \right\},$$

where k is the dimension of the simplex. A k -simplex is a simplex that consists of $k + 1$ points.

Definition 2 (Face of the Simplex). The convex hull of any nonempty subset of the $n + 1$ points that define an n -simplex, is called a face of the simplex. Faces are simplexes themselves. In particular, the convex hull of a subset of size $m + 1$ is an m -simplex, called an m -face of the n -simplex.

Definition 3 (Simplicial Complex). A simplicial complex \mathcal{K} is a set of simplexes that satisfies the following conditions: 1. Any face of a simplex from \mathcal{K} is also in \mathcal{K} . 2. The intersection of any two simplexes $\sigma_1, \sigma_2 \in \mathcal{K}$ is either \emptyset or a face of both σ_1 and σ_2 .

Definition 4 (Nerve [24]). Given a covering $\mathbb{U} = \{u_\alpha \mid u_\alpha \subset X\}_{\alpha \in \mathbb{A}}$ of space X , where \mathbb{A} is an indexing set whose item is integer. Intuitively, a covering of X is a collection of sets whose union contains X . The nerve of covering \mathbb{U} is the simplicial complex $N(\mathbb{U})$ whose vertex set is the indexing set \mathbb{A} , and where a family $\{\alpha_0, \alpha_1, \dots, \alpha_k\}$ spans a k -simplex in all $N(u)$ if and only if $u_{\alpha_0} \cap u_{\alpha_1} \cap \dots \cap u_{\alpha_k} \neq \emptyset$.

B Procedure of Mapper

Computing the Distance Matrix This step takes the original dataset X as input and finds the similarity between data. We adopt Euclidean distance to measure the similarity, and compute the distance matrix D .

Computing the Filter Value. This step takes the distance matrix D and one possibly high dimensional filter function f as input. It outputs the filter value of every record. We denote the codomain of f as Z . Python Mapper provides realization for density, eccentricity, graph Laplacians [24], $kNN_distance$ and $dm_eigenvector$ filter functions. We detail eccentricity and $kNN_distance$. The eccentricity filter is defined as:

$$E_p(X_i) = \left(\frac{\sum_{X_j \in X} D_{ij}^p}{N} \right)^{\frac{1}{p}},$$

where p is a parameter, N is the size of X , D_{ij} is the distance between X_i and X_j . The eccentricity filter measures how far a data record deviates from the center of this dataset. The records that are far away from most of the records have relatively larger filter values.³ The filter value of one record given by the $kNN_distance$ is equal to the distance between this record and its k^{th} nearest neighbor, where k is a parameter. The $kNN_distance$ filter measures the aggregation of the dataset at given k .

Computing the Cover. This step takes the filter values as input and then outputs a cover \mathbb{S} of the filter values as well as a cover $\bar{\mathbb{S}} = \{s_\beta\}_{\beta \in A}$ of the original dataset. Mapper employs two methods to find the cover of the filter values, the cube cover for Z in multi-dimensional space [24], and the 1-d uniform cover for $Z \in R$. The 1-d cover uniformly divides Z into I intervals with adjacencies having an overlap ratio of $O\%$, where I and O are two parameters of covering. After covering on Z , a cover $\bar{\mathbb{S}}$ of the original dataset can be constructed through putting data records with the same range of filter values into the same interval.

Computing the Cluster. This step takes the cover $\bar{\mathbb{S}}$ as input, and further clusters each subset s_β to obtain the final cover $\mathbb{U} = \{u_\alpha\}_{\alpha \in \mathbb{A}}$ of the original dataset, where cluster u_α is the subset of s_β . After further dividing $\bar{\mathbb{S}}$ into the ultimate cover $\mathbb{U} = \{u_\alpha\}_{\alpha \in \mathbb{A}}$, the complex can be constructed with the basic nodes in indexing set \mathbb{A} . According to the nerve theory (see Appendix A), the indexing set $\{\alpha_0, \alpha_1 \dots \alpha_k\}$ constructs a k -simplex if and only if $u_{\alpha_0} \cap u_{\alpha_1} \cap \dots \cap u_{\alpha_k} \neq \emptyset$. Then, the Mapper complex can be visualized by Python Mapper. Any domain-specific clustering methods can be used for Mapper⁴. We employ hierarchical clustering [12] with complete-linkage to divide $\bar{\mathbb{S}}$. One parameter *gap size* of Python Mapper is required for the hierarchical clustering. A subset can be divided into more clusters with a smaller *gapsize*.

³ <http://danifold.net/mapper/index.html>.

⁴ Referring to [24] for detail about clustering methods applied for Mapper and its state of arts.

References

1. Adcock, A., Carlsson, E., Carlsson, G.: The ring of algebraic functions on persistence bar codes. *Mathematics* (2013)
2. Bhattacharya, S., Ghrist, R., Kumar, V.: Persistent homology for path planning in uncertain environments. *IEEE Trans. Rob.* **31**(3), 578–590 (2015)
3. Bolton, R.J., Hand, D.J.: Statistical fraud detection: a review. *Stat. Sci.* **17**(3), 235–255 (2002)
4. Carlsson, G.: Topological pattern recognition for point cloud data. *Acta Numerica* **23**, 289–368 (2014)
5. Carlsson, G., Zomorodian, A., Collins, A., Guibas, L.J.: Persistence barcodes for shapes. *Int. J. Shape Model.* **11**(2), 149–187 (2008)
6. Dewoskin, D., Climent, J., Cruz-White, I., Vazquez, M., Park, C., Arsuaga, J.: Applications of computational homology to the analysis of treatment response in breast cancer patients. *Topol. Appl.* **157**(1), 157–164 (2010)
7. Dey, T., Wang, Y.: Multiscale mapper: topological summarization via codomain covers. In: *Twenty-Seventh ACM-SIAM Symposium on Discrete Algorithms*, pp. 997–1013 (2016)
8. Edelsbrunner, Letscher, Zomorodian: *Topological Persistence and Simplification*, vol. 28. *Discrete and Computational Geometry* (2002)
9. Estévez, P., Held, C., Perez, C.: Subscription fraud prevention in telecommunications using fuzzy rules and neural networks. *Expert Syst. Appl.* **31**(2), 337–344 (2006)
10. Gade, S.V.: Credit card fraud detection using hidden Markov model. *Indian Streams Res. J.* **4**(4), 37–48 (2014)
11. Ghosh, S., Reilly, D.: Card fraud detection with a neural-network. *Twenty-Seventh Hawaii Int. Conf. Syst. Sci.* **3**, 621–630 (2011)
12. Johnson, S.: Hierarchical clustering schemes. *Psychometrika* **32**(3), 241–254 (1967)
13. Lenz, H.J.: Data fraud detection: a first general perspective data fraud detection: a first general perspective. *Int. Conf. Enterp. Inf. Syst.* **227**, 14–35 (2014)
14. Lum, P.Y., Singh, G., Lehman, A., Ishkanov, T., Vejdemo-Johansson, M., Alagappan, M., Carlsson, J., Carlsson, G.: Extracting insights from the shape of complex data using topology. *Sci. Rep.* **3**(1), 1236 (2013)
15. Maria, C., Boissonnat, J., Glisse, M., Yvinec, M.: The gudhi library: simplicial complexes and persistent homology, in mathematical software. *Int. Congr. Math. Softw.* **8592**, 167–174 (2014)
16. Nicolau, M., Levine, A.J., Carlsson, G.: Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proc. Natl. Acad. Sci. USA* **108**(17), 7265–7270 (2011)
17. Nicolau, M., Levine, A.J., Carlsson, G.: Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proc. Natl. Acad. Sci. USA* **108**(17), 7265–7270 (2011)
18. Oentaryo, R., Lim, E.P., Finegold, M., Lo, D., Zhu, F.: Detecting click fraud in online advertising: a data mining approach. *J. Mach. Learn. Res.* **15**(1), 99–140 (2014)
19. Otter, N., Porter, M., Tillmann, U., Grindrod, P., Harrington, H.: A roadmap for the computation of persistent homology. *Mathematics* **6**(1), 17 (2017)
20. Pokorný, F., Hawasly, M., Ramamoorthy, S.: Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *Int. J. Rob. Res.* **35**(1–3), 204–223 (2016)

21. Rahman, M., Rahman, M., Carbanar, B., Chau, D.: Search rank fraud and malware detection in Google Play. *IEEE Trans. Knowl. Data Eng. Data Eng.* **PP**(99), 1329–1342 (2017)
22. Savic, A., Toth, G., Duponchel, L.: Topological data analysis (TDA) applied to reveal pedogenetic principles of european topsoil system. *Sci. Total Environ.* **586**, 1091–1100 (2017)
23. de Silva, V., Ghrist, R.: Coverage in sensor networks via persistent homology. *Algebraic Geom. Topol.* **PP**, 339–358 (2007)
24. Singh, G., Mémoli, F., Carlsson, G.: Topological methods for the analysis of high dimensional data sets and 3D object recognition. In: *Eurographics Symposium on Point Based Graphics* (2007)
25. Viaene, S., Derrig, R., Dedene, G.: A case study of applying boosting naive bayes to claim fraud diagnosis. *IEEE Trans. Knowl. Data Eng.* **16**(5), 612–620 (2004)
26. Xia, K., Feng, X., Tong, Y., Wei, G.: Persistent homology for the quantitative prediction of fullerene stability. *J. Comput. Chem.* **36**(6), 408–422 (2014)
27. Xia, K., Wei, G.W.: Persistent homology analysis of protein structure, flexibility, and folding. *Int. J. Numer. Methods Biom. Eng.* **30**(8), 814–844 (2014)
28. Zomorodian, A., Carlsson, G.: Computing persistent homology. *Discret. Comput. Geom.* **33**, 249–274 (2005)

**Full Paper Session VIII: Searchable
Encryption and Identity-Based
Cryptography**



Efficient Multi-keyword Searchable Encryption Based on Multi-input Inner-Product Functional Encryption

Yunong Liang , Zhenfu Cao  , Xiaolei Dong , and Jiachen Shen  

Shanghai Key Laboratory of Trustworthy Computing, East China Normal University,
Shanghai 200062, China

511645000720stu.ecnu.edu.cn, {zfc Cao, dongxiaolei, jcshen}@sei.ecnu.edu.cn

Abstract. With highly development of cloud computing, data owners wish to outsource their data to clouds for computational and storage resource at a lower price. In order to protect the privacy of sensitive information, they should be encrypted before being uploaded to the cloud server. However, in this way, it is hard to find data in encrypted form according to search criticisms. To solve this problem, searchable encryption has merged. In this paper, we propose a secure and efficient searchable encryption scheme supporting multi-keyword search in 1-to-n setting. The scheme is mainly applicable to the scenes that the number of keywords is limited but the number of files is huge such as sharing a comprehensive knowledge base of a certain field. By tactfully leveraging multi-input inner-product functional encryption, the cloud server is able to complete search processes with search tokens which consist of only two items. It reduces communication and transportation overhead significantly. By using an inverted index structure and super-incremental sequence, our scheme achieves efficient multi-keyword search. In addition, our scheme avoids per-query interaction between the data owner and data users. That is to say, the data owner does not need to stay online waiting for data users to search in his archives. On the other hand, the scheme also achieves partial token privacy, index privacy and token privacy at the same time.

Keywords: Searchable encryption · Multi-keyword · Multi-user
Index privacy · Token privacy

1 Introduction

1.1 Background

As we all know, big data has three outstanding features: large volume, high velocity and high variety. Cloud storage is well designed for big data because of its excellent capability to store large volumes of data, to prepare for high velocity of data generation and to process high variety of data. Cloud computing provides great convenience to users and is one of the most popular technologies at present.

© Springer Nature Switzerland AG 2018

D. Naccache et al. (Eds.): ICICS 2018, LNCS 11149, pp. 377–392, 2018.

https://doi.org/10.1007/978-3-030-01950-1_22

Meanwhile, cloud computing (data outsourcing) raises confidentiality and privacy concerns. Simple encryption can protect data confidentiality easily. However, when data users want to search using some specific keyword to get documents of their interest among massive volumes of data, this becomes a new challenge. In order to search by a particular keyword, the data owner has to decrypt the data first before starting the searching process. It is obviously not practical especially when the volume of data is large. Searchable encryption (SE) [5–7, 10, 11, 15, 20] is a cryptographic primitive to address search over ciphertexts. SE allows data users retrieve documents from the cloud server according to some keywords. Searchable encryption has been studied intensively and a mass of keyword search schemes over encrypted cloud data have been proposed.

In the cloud environment, a data owner usually shares his documents with data users. In this paper, we focus on the single-owner/multi-user setting. In this setting, when a data user wants to search over the data owner's documents, he usually needs to ask data owner to produce necessary trapdoor information to help him complete the search. This is to say, the data owner must be online all the time to perform the per-query interaction with data users. However, the primary goal of data owner is to outsource his search services to the cloud server, so we remove the per-query interaction between the data owner and data users in our scheme.

In some practical setting, search with only one keyword may obtain a great quantity of documents and obviously it lacks search precision. Especially when the queried keyword does not accurately describe the documents that data user wants to get. Thus, multi-keyword searchable encryption merged. That is to say, data users can do the research with multi keywords, namely, conjunctive keyword search. Thus, data users can get the documents including all the queried keywords.

Most existing SE schemes assume that the cloud server is honest-but-curious. That is to say, the cloud server may try to find out which keyword the ciphertext is about. Besides, it may also be curious about which keywords data user wants to search. So it is necessary to ensure index privacy and token privacy.

In most literatures, when data users want to search with specific keywords to get their target documents, they have to compute a search token including quite a few items and send it to the cloud server to complete the search process. It may consume a lot of bandwidth and the computing overhead is huge for data users. However, the search token in our scheme consists of only two items by using multi-input inner-product functional encryption. Furthermore, our scheme guarantees index privacy and token privacy simultaneously.

1.2 Design Goal

In this paper, we propose a secure and efficient single-owner/multi-user searchable encryption scheme supporting multi-keyword search. Our design goal is summarized as follows:

1. Our scheme avoids the per-query interaction between the data owner and data users. Namely, the data owner does not need to stay online waiting for data users to search in his archives.
2. By tactfully leveraging multi-input inner-product functional encryption, our scheme allows the cloud server to complete search processes with search tokens which consist of only two items.
3. By using an inverted index structure and super-incremental sequence, our scheme achieves efficient multi-keyword search.
4. Our scheme ensures the correctness of search phase.
5. Our scheme achieves partial token privacy, index privacy and token privacy at the same time.

1.3 Organization

The structure of this paper is as follows:

In Sect. 2, we introduce some related work. Section 3 gives some preliminaries. Then we propose our system model and describe our scheme in detail in Sect. 4 and Sect. 5 respectively. In Sect. 6, we show the correctness and security of our scheme and analyze function and efficiency of our scheme by comparing with other schemes in Sect. 7. In the last section, we summarize this paper.

2 Related Work

2.1 Index

The index structures have an effect on assisting to perfect the scheme. Different index structures have different advantages and disadvantages. Curtmola et al. proposed a searchable encryption scheme in literature [6] based on the inverted index because of its efficiency. Although inverted index structure is efficient on searching, it is not convenient on updating the files. Goh et al. [11] proposed an index structure based on bloom filter. While Chang et al. proposed a vector index in [5].

2.2 Searchable Encryption

The first searchable encryption is proposed by Song et al. [20] in the symmetric key setting. The security notion of searchable encryption was first introduced by Goh [11]. And then, Curtmola et al. [6] presented a stronger security notion, indistinguishability against adaptive chosen-keyword attacks (IND-CKA2). Boneh et al. [7] designed the first searchable encryption with keyword search in the public key setting, but its search efficiency is not fast comparing to the symmetric searchable encryption. All the scheme mentioned above only support single-keyword search.

However, in some settings, a single-keyword may not describe the search precision correctly. Therefore, multi-keyword searchable encryption [2, 4, 8, 12–14, 16–19, 21, 22] has received increasing attention.

Golle et al. [12] first proposed the construction of conjunctive keyword searchable encryption in the single-owner/single-user setting and presented two schemes. In the first scheme, the size of search token is linear with the number of encrypted documents. In the second scheme, the size of search token is constant by using bilinear pairings while the computational cost is still not low. In the literatures [14, 16, 19], conjunctive and disjunctive keyword search are proposed, which make the multi-keyword search semantics get a further extension. Cash et al. [4] proposed the first sublinear symmetric searchable encryption support boolean queries in single-owner/single-user setting and implemented it in a large database [3]. Jarecki et al. [13] extends it to single-owner/multi-user setting, which data owner needs to be online all the time. Besides, the search time mainly depends on the number of files including the least frequency keywords among keywords data user wants to search. That is to say, search efficiency is not high when the keywords data user queries are all high frequency ones.

3 Preliminary

3.1 Notation

We use $s \xleftarrow{R} S$ to denote the operation of uniformly sampling a random element s from a set S . We use PPT to denote a probabilistic polynomial-time algorithm. λ represents the security parameter in this paper. We use lower case boldface italics to denote (column) vectors and upper case boldface italics to denote matrices. For a matrix \mathbf{M} over \mathbb{Z}_q , we have $[\mathbf{M}]_1 := g_1^{\mathbf{M}}$ and $[\mathbf{M}]_2 := g_2^{\mathbf{M}}$, where exponentiation is carried out component-wise. Besides, we use $[n]$ to denote integers no more than n and we use $\langle \mathbf{x}, \mathbf{y} \rangle$ to denote inner product of \mathbf{x} and \mathbf{y} where \mathbf{x} and \mathbf{y} are column vectors with same dimension.

3.2 Asymmetric Bilinear Groups

Let \mathcal{PG} denote a group generator – an algorithm which takes a security parameter λ as input and outputs a description of prime order groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We define \mathcal{PG} 's output as $(q, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where q is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order q . g_1, g_2, g_T are the generator of $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a map with the following properties:

- (1) Bilinearity: $\forall a, b \in \mathbb{Z}_q, e(g_1^a, g_1^b) = e(g_1, g_1)^{ab}$
- (2) Non-degeneracy: $e(g_1, g_2) \neq 1$
- (3) Computability: $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2, e(u, v)$ can be efficiently computed.

3.3 Multi-input Inner-Product Encryption

In inner-product encryption scheme, upon receiving the ciphertext of a vector \mathbf{x} , only the recipients who have the secret key $k_{\mathbf{y}}$ can obtain the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ of \mathbf{x} and \mathbf{y} . While in multi-input inner-product encryption,

only the recipients who have the secret key $k_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n}$ and ciphertexts of vector $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ can obtain the sum of inner product $\langle \mathbf{x}_i, \mathbf{y}_i \rangle$, namely, $\sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i \rangle$.

We will use the definition of Matrix Decision Diffie-Hellman (MDDH) Assumption in [9].

3.4 Matrix Distribution

Let $k, l \in N$, with $l > k$, we call $\mathcal{D}_{l,k}$ a matrix distribution if it outputs matrices in $Z_q^{l \times k}$ of full rank k in polynomial time. We write $\mathcal{D}_k := \mathcal{D}_{k+1,k}$. Without loss of generality, we assume the first k rows of $\mathbf{A} \xleftarrow{R} \mathcal{D}_{l,k}$ form an invertible matrix. Particularly, we use $\mathcal{U}_{l,k}$ to denote the uniform distribution. \mathcal{U}_k stands for $\mathcal{U}_{k+1,k}$. In this work, we are mostly interested in the uniform matrix distribution $\mathcal{U}_{l,k}$.

3.5 $\mathcal{D}_{l,k}$ -Matrix Diffie-Hellman Assumption $\mathcal{D}_{l,k}$ -MDDH

Let $\mathcal{D}_{l,k}$ be a matrix distribution. We say that the $\mathcal{D}_{l,k}$ -Matrix Diffie-Hellman ($\mathcal{D}_{l,k}$ -MDDH) Assumption relative to \mathcal{PG} in \mathbb{G}_s holds if for all PPT adversaries \mathcal{A} , there is no non-negligible function Adv . Namely $Adv_{\mathbb{G}_s, \mathcal{A}}^{\mathcal{D}_{l,k}\text{-MDDH}} = |Pr[\mathcal{A}(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{A}\mathbf{w}]_s) = 1] - Pr[\mathcal{A}(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{u}]_s) = 1]| = \text{negl}(\lambda)$, where the probability is taken over $\mathbf{A} \xleftarrow{R} \mathcal{D}_{l,k}$, $\mathbf{w} \xleftarrow{R} Z_q^k$, $\mathbf{u} \xleftarrow{R} Z_q^{k+1}$ and $s \in \{1, 2\}$.

Lemma 1. *Among all possible matrix distribution $\mathcal{D}_{l,k}$, the uniform matrix distribution $\mathcal{U}_{l,k}$ is the hardest possible instance. We have $\mathcal{D}_{l,k}\text{-MDDH} \Rightarrow \mathcal{U}_{l,k}\text{-MDDH}$. For all PPT adversaries \mathcal{A} , there exists an adversary \mathcal{B} such that $Adv_{\mathbb{G}_s, \mathcal{A}}^{\mathcal{U}_{l,k}\text{-MDDH}} \leq Adv_{\mathbb{G}_s, \mathcal{B}}^{\mathcal{U}_k\text{-MDDH}}$.*

Lemma 2. *For $\mathbf{A} \xleftarrow{R} \mathcal{U}_{l,k}$, $\mathbf{W} \xleftarrow{R} Z_q^{k \times Q}$, $\mathbf{U} \xleftarrow{R} Z_q^{(k+1) \times Q}$, $s \in \{1, 2\}$. $Adv_{\mathbb{G}_s, \mathcal{A}}^{\mathcal{Q}\text{-}\mathcal{U}_{l,k}\text{-MDDH}} = |Pr[\mathcal{A}(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{A}\mathbf{W}]_s) = 1] - Pr[\mathcal{A}(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{U}]_s) = 1]|$. Then, we have for all PPT adversaries \mathcal{A} , there exists an adversary \mathcal{B} such that $Adv_{\mathbb{G}_s, \mathcal{A}}^{\mathcal{Q}\text{-}\mathcal{U}_{l,k}\text{-MDDH}} \leq Adv_{\mathbb{G}_s, \mathcal{B}}^{\mathcal{U}_{l,k}\text{-MDDH}} + \frac{1}{q-1}$.*

4 System Model

In our single-owner/multi-user setting, there are three different kinds of entities: data owner, data user and cloud server. As shown in Fig. 1, the data owner has a collection of files and wants to outsource his search service to the cloud server. The data owner first extracts keywords from the files and constructs inverted indices. It is important to note that our scheme is mainly applicable to the scenes that the number of keywords is limited but the number of files is huge, so the data owner only extracts the most relevant keywords.

And then, the data owner outsources encrypted indices and encrypted files to the cloud server. Besides, the data owner sends partial token and search-authorized secret key to each legitimate data user, with which data user is able

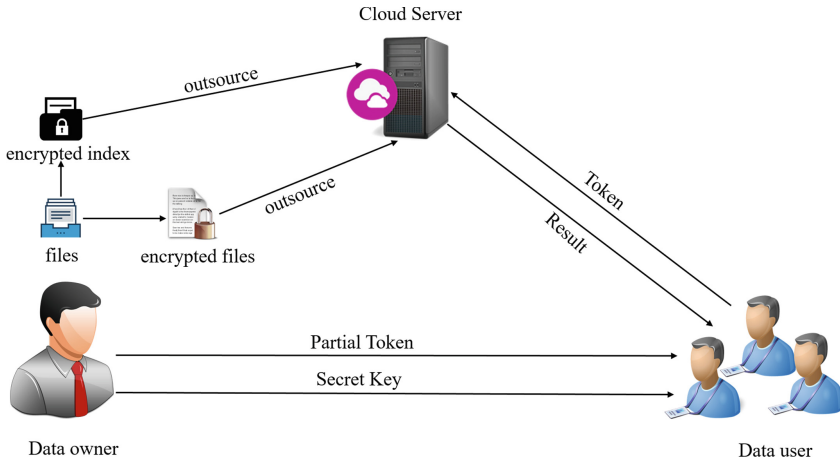


Fig. 1. System model

to generate search token about the keywords he wants to search. When a data user performs a search query, he sends the search token to the cloud server. With the search token and encrypted indices, the cloud server finally returns target documents to the data user.

Formally, our multi-keyword searchable encryption is a tuple of six polynomial-time algorithms $\pi = (Setup, Enc, PartialTokenGen, ClientKGen, TokenGen, Search)$

- $Setup(1^\lambda) \rightarrow (pp, msk)$: is a probabilistic algorithm that the data owner takes security parameter 1^λ as input and generates system master key msk and public parameter pp .
- $Enc(pp, F, W, DU) \rightarrow (C_W, C_F, C_{Indices}, C_{List})$: is a probabilistic algorithm that the data owner takes public parameter pp , a document collection $F = \{f_1, f_2, \dots, f_n\}$, keyword dictionary $W = \{w_1, w_2, \dots, w_m\}$ which is public and a set of legitimate data users DU as input and generate encrypted keywords C_W , encrypted files C_F and encrypted indices $C_{Indices}$. The file encryption is executed by using some simple symmetric encryption due to efficiency concerns. Besides, the data owner generates an encrypted list C_{List} about data users and their corresponding information.
- $PartialTokenGen(pp, msk, \xi) \rightarrow pt$: is a probabilistic algorithm that the data owner takes pp, msk to generate partial token pt for each legitimate data user $\xi \in DU$, with which and his search-authorized private key, a data user can generate search tokens for the keywords he wants to search.
- $ClientKGen(pp, msk, \xi) \rightarrow sk$: is a probabilistic algorithm that the data owner takes pp and msk as input and generates different search-authorized private key sk for each legitimate data user $\xi \in DU$.

- $TokenGen(sk, pt, Q) \rightarrow token$: is a deterministic algorithm that the data users use their private key sk and partial-token pt to produce search tokens $token$ for the keyword set Q they want to query.
- $Search(token, C_W, C_F, C_{Indices}, C_{List}) \rightarrow RST$: is a deterministic algorithm that the cloud server uses search token $token$ to search over encrypted indices $C_{Indices}$. Then it downloads the matched encrypted files RST and returns them to the data user.

5 Construction

In this section, we will introduce our multi-keyword searchable encryption scheme in detail.

- $Setup(1^\lambda)$: Given a bilinear group $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where q is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order q . g_1, g_2, g_T are generators of $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T respectively. Randomly select a matrix \mathbf{A} from $\mathbb{Z}_q^{3 \times 2}$ of full rank, namely randomly select a matrix \mathbf{A} from \mathcal{U}_2 , randomly choose a matrix \mathbf{M} from $\mathbb{Z}_q^{3 \times 3}$, \mathbf{V} from $\mathbb{Z}_q^{2 \times 3}$, and randomly select m vectors $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m$ from \mathbb{Z}_q^2 . Let $\varepsilon = (Setup, Enc, KGen, Dec)$ be a public-key encryption scheme, where $Setup$ is a public key generation algorithm, Enc is an encryption algorithm, $KGen$ is a secret key generation algorithm and Dec is a decryption algorithm. $pk_{server} \leftarrow \varepsilon.Setup(1^\lambda)$ is public key of the cloud server. Then, output public parameter $pp = (g_1, g_2, g_T, q, [\mathbf{A}]_1, pk_{server})$ and master secret key $msk = (\mathbf{M}, \mathbf{V}, \{\mathbf{z}_i\}_{i \in [m]})$
- $Enc(pp, F, W, DU)$: Choose a super-incremental sequence $\alpha_1, \alpha_2, \dots, \alpha_m \in (0, \frac{\log_{g_T} q}{2})$, that is, for $i \in [m]$,

$$\alpha_i > \alpha_1 + \alpha_2 + \dots + \alpha_{i-1}$$

For each keyword $w_i \in W$, use a pseudo random substitution to map i to j , let $\mathbf{x}_i = (w_i, 1, r) \in \mathbb{Z}_q^3$, where r are randomly chosen from \mathbb{Z}_q , choose $\mathbf{y}_i = (y_{i1}, y_{i2}, 1) \in \mathbb{Z}_q^3$ such that $\alpha_j = \langle \mathbf{x}_i, \mathbf{y}_i \rangle$. Besides, choose different r_ξ for each data user $\xi \in DU$, where r_ξ is randomly chosen from \mathbb{Z}_q . Record $\{\mathbf{x}_i\}_{i \in [m]}$, $\{\mathbf{y}_i\}_{i \in [m]}$ and $List = \{\xi, r_\xi\}_{\xi \in DU}$. Then, we compute $C_W = \{g_T^{\alpha_1}, g_T^{\alpha_2}, \dots, g_T^{\alpha_m}\}$ as the ciphertext of keywords W , compute C_F as the ciphertext of files F with some symmetric encryption algorithm and generate encrypted indices $C_{Indices} = \{g_T^{\alpha_j}, Id_{w_i}\}_{j \in [m]}$. Id_{w_i} means a set of file identifiers of files which include keywords w_i . Besides, we use pk_{server} to compute $C_{List} = \varepsilon.Enc(List)$. Finally, the data owner sends $(C_W, C_F, C_{Indices}, C_{List})$ to cloud server.

- $PartialTokenGen(pp, msk, \{\mathbf{x}_i\}_{i \in [m]}, \xi)$: For each legitimate data user $\xi \in DU$, the data owner randomly chooses different $\mathbf{s}_{\xi, i} \in \mathbb{Z}_q^2$ and $r_{pt} \in \mathbb{Z}_q$. Let $\mathbf{r}_{\xi, pt} = (0, 0, r_{pt})$, and compute partial-token as follows.

$$[\mathbf{c}_i]_1 = [\mathbf{A}\mathbf{s}_{\xi, i}]_1 \tag{1}$$

$$[\mathbf{c}'_i]_1 = [\mathbf{M}\mathbf{A}\mathbf{s}_{\xi,i} + \mathbf{x}_i + \mathbf{r}_{\xi pt}]_1 \tag{2}$$

$$[\mathbf{c}''_i]_1 = [\mathbf{V}\mathbf{A}\mathbf{s}_{\xi,i} + \mathbf{z}_i]_1 \tag{3}$$

Send the partial-token $pt = \left([\mathbf{c}_i]_1, [\mathbf{c}'_i]_1, [\mathbf{c}''_i]_1\right)_{i \in [m]}$ to the data user ξ by a secure channel.

- *ClientKGen*($pp, msk, \{\mathbf{y}_i\}_{i \in [m]}, \xi, r_\xi$): For each legitimate user $\xi \in DU$, the data owner randomly chooses different $\mathbf{r}_{\xi_1} \in \mathbb{Z}_q^2$, let $\mathbf{r}_{\xi_{sk}} = (0, r_\xi - r_{pt}, 0) \in \mathbb{Z}_q^3$ and compute secret key as follows.

$$\mathbf{d}_i = \mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T \mathbf{r}_{\xi_1} \tag{4}$$

$$Z_i = \langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle \tag{5}$$

Send the secret key $sk = \left(\{[\mathbf{d}_i]_2, [Z_i]_T, [\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}]_2\}_{i \in [m]}, [r_{\xi_1}]_2\right)$ to the data user ξ by a secure channel.

- *TokenGen*(sk, pt, Q): With partial-token pt , secret key sk and the keywords set $Q = \{w_{q_1}, w_{q_2}, \dots, w_{q_t}\} \subseteq W$ to be searched, data users compute search tokens as follows. We use $e([\mathbf{X}]_1, [\mathbf{Y}]_2)$ to denote $[\mathbf{X}^T \mathbf{Y}]_T$.

$$st = \prod_{i=1}^t \frac{e([\mathbf{c}'_{q_i}]_1, [\mathbf{y}_{q_i} + \mathbf{r}_{\xi_{sk}}]_2) \cdot e([\mathbf{c}''_{q_i}]_1, [r_{\xi_1}]_2) / e([\mathbf{c}_{q_i}]_1, [\mathbf{d}_{q_i}]_2)}{[Z_i]_T} \tag{6}$$

Data users send the search tokens $token = (st, \varepsilon.Enc(t))$ corresponding to the keywords they want to search to the cloud server.

- *Search*($token, C_W, C_F, C_{Indices}, C_{List}$): When the cloud server receives a search token, it first decrypts $\varepsilon.Enc(t)$ to get t and retrieves real search token $rst = g_T^{\sum_{i=1}^t \langle \mathbf{x}_{q_i}, \mathbf{y}_{q_i} \rangle}$ by t and $[r_\xi]_T$ corresponding to user identity ξ and then determines whether g^{α_m} less than the real search token rst , if so, return \perp , which means that the search token is illegal and there is no corresponding keywords. Otherwise, by using binary search, the cloud server determines whether there is a k satisfying $g^{\alpha_k} \leq rst \leq g^{\alpha_{k+1}}$, if so, it means that the keyword corresponding to g^{α_k} is one of the keyword the data user wants to search. Then, it calculates $rst = rst / g^{\alpha_k}$ and repeats the above steps until rst equals to one. Pseudo code is showed in Algorithm 1. Finally, cloud server takes all the file identifiers that contain the keywords to be searched and then returns the ciphertexts of the corresponding file to the data user.

6 Correctness and Security

6.1 Correctness

We now show the correctness of the search phase.

Algorithm 1. Search Process

```

1: if  $g^{\alpha_m} \leq rst$  then
2:   return  $\perp$ 
3: else
4:    $int\ low = 1, high = m;$ 
5:   while  $low \leq high \& \& rst == 1$  do
6:      $mid = (low + high)/2$ 
7:     if  $g^{\alpha_{mid+1}} \leq rst$  then
8:        $low = mid + 1$ 
9:     else if  $g^{\alpha_{mid}} > rst$  then
10:       $high = mid - 1$ 
11:     else  $g^{\alpha_{mid}} \leq rst \leq g^{\alpha_{mid+1}}$ 
12:      the keyword corresponding to  $g^{\alpha_{mid}}$  is one of the target keyword
13:       $rst = rst/g^{\alpha_{mid}}$ 
14:       $high = mid - 1$ 
15:       $low = 1$ 
16:     end if
17:   end while
18: end if

```

The data user first calculates search token as follows:

$$\begin{aligned}
 st &= \prod_{i=1}^t \frac{e([\mathbf{c}'_{q_i}]_1, [\mathbf{y}_{q_i} + \mathbf{r}_{\xi_{sk}}]_2) \cdot e([\mathbf{c}''_{q_i}]_1, [\mathbf{r}_{\xi_1}]_2) / e([\mathbf{c}_{q_i}]_1, [\mathbf{d}_{q_i}]_2)}{[Z_i]_T} \\
 &= \prod_{i=1}^t \frac{g_T^{\langle \mathbf{c}'_{q_i}, \mathbf{y}_{q_i} + \mathbf{r}_{\xi_{sk}} \rangle} \cdot g_T^{\langle \mathbf{c}''_{q_i}, \mathbf{r}_{\xi_1} \rangle} / g_T^{\langle \mathbf{c}_{q_i}, \mathbf{d}_{q_i} \rangle}}{[Z_i]_T} \\
 &= \prod_{i=1}^t \frac{g_T^{\langle M \mathbf{A} s_{\xi, q_i} + \mathbf{x}_{q_i} + \mathbf{r}_{\xi_{pt}}, \mathbf{y}_{q_i} + \mathbf{r}_{\xi_{sk}} \rangle + \langle V \mathbf{A} s_{\xi, q_i} + \mathbf{z}_{q_i}, \mathbf{r}_{\xi_1} \rangle}}{g_T^{\langle \mathbf{z}_{q_i}, \mathbf{r}_{\xi_1} \rangle + \langle \mathbf{A} s_{\xi, q_i}, M^T(\mathbf{y}_{q_i} + \mathbf{r}_{\xi_{sk}}) + V^T \mathbf{r}_{\xi_1} \rangle}} \\
 &= \prod_{i=1}^t g_T^{\langle \mathbf{x}_{q_i} + \mathbf{r}_{\xi_{pt}}, \mathbf{y}_{q_i} + \mathbf{r}_{\xi_{sk}} \rangle} \\
 &= g_T^{\sum_{i=1}^t \langle \mathbf{x}_{q_i}, \mathbf{y}_{q_i} \rangle + r_{\xi}} \tag{7}
 \end{aligned}$$

When the cloud server receives a search token, it first decrypts $\varepsilon.Enc(t)$ to get t and retrieves real search token $rst = g_T^{\sum_{i=1}^t \langle \mathbf{x}_{q_i}, \mathbf{y}_{q_i} \rangle}$ by t and $[r_{\xi}]_T$ according to user's identity ξ . Because $\alpha_1, \alpha_2, \dots, \alpha_m \in (0, \frac{\log_{g_T} q}{2})$ is a super-incremental sequence, we have that $\alpha_i > \alpha_1 + \alpha_2 + \dots + \alpha_{i-1}$. Thus, $g_T^{\alpha_i} > g_T^{\alpha_1 + \alpha_2 + \dots + \alpha_{i-1}}$. Because of the ciphertext of keywords $g_T^{\alpha_j} = g_T^{\langle \mathbf{x}_{q_i}, \mathbf{y}_{q_i} \rangle}$, it means that the product of the ciphertext of keywords that data user wants to search equals to the real search token. When the cloud server retrieves $g_T^{\sum_{i=1}^t \langle \mathbf{x}_{q_i}, \mathbf{y}_{q_i} \rangle}$, it determines whether there is a k satisfying $g_T^{\alpha_k} \leq rst < g_T^{\alpha_{k+1}}$ or not. Obviously, the keywords corresponding to $g_T^{\alpha_{k+1}}, \dots, g_T^{\alpha_m}$ can not be the target keyword. If keyword corresponding to $g_T^{\alpha_k}$ is not the target keyword, the keywords corresponding to $g_T^{\alpha_1}, \dots, g_T^{\alpha_{k-1}}$ must be the target keywords, namely, $g_T^{\alpha_1 + \dots + \alpha_{k-1}} = rst$. However, according to the super-incremental sequence, we

know that $g_T^{\alpha_1+\dots+\alpha_{k-1}} < rst$, that is to say, the keywords corresponding to $g_T^{\alpha_1}, \dots, g_T^{\alpha_{k-1}}$ cannot be the target keywords. Therefore, we know that the keyword corresponding to $g_T^{\alpha_k}$ is one of the target keywords.

6.2 Security

For the files, ciphertexts C_F are semantic security by adopting symmetric encryption, such as AES. Then a probabilistic polynomial-time adversary cannot get any useful information from C_F with non-negligible probability. For the keyword, we have:

Partial-Token Privacy. Partial-Token Privacy means that a probabilistic polynomial-time adversary cannot get any useful information from the partial-token. That is to say, assuming that an adversary gets one item of the partial-token from a legitimate data user ξ , he could not know which keywords the item is about.

- Setup: The challenger plays a role as the system and runs $Setup()$, then it keeps master key msk .
- Init: The challenger runs $Enc()$ to get different reasonable $\{\mathbf{x}_i\}_{i \in [m]}$, $\{\mathbf{y}_i\}_{i \in [m]}$ and $\{\xi, r_\xi\}_{\xi \in DU}$, with which it can run $ClientKGen()$ and $PartialTokenGen()$.
- Query Phase1: The adversary adaptively queries sk and pt about different ξ for polynomial times. The challenger runs $PartialTokenGen()$ and $ClientKGen()$ algorithm and returns $pt \leftarrow PartialTokenGen(pp, msk, \{\mathbf{x}_i\}_{i \in [m]}, \xi)$ and $sk \leftarrow ClientKGen(pp, msk, \{\mathbf{y}_i\}_{i \in [m]}, \xi, r_\xi)$.
- Challenge phase: The adversary randomly selects two keywords w_{i_0} and w_{i_1} and submits the identity ξ he wants to challenge with the restriction that ξ has not queried before. The challenger flips a coin to select $\beta \leftarrow \{0, 1\}$ and then runs $PartialTokenGen()$ algorithm. The challenger returns $PartialTokenGen(pp, msk, \mathbf{x}_{i_\beta}, \xi, r_\xi)$ to the adversary.
- Query Phase 2: The adversary executes queries as Phase1 did.
- Guess: Finally, the adversary gives a guess β' of β and wins the game if $\beta' = \beta$. We can define the advantage of adversary winning the game is $|Pr[\beta' = \beta] - \frac{1}{2}|$.

Theorem 1. *If an adversary wins the game mentioned above with a non-negligible advantage, there is an adversary \mathcal{B} can break MDDH assumption.*

Proof. Specific proofs are detailed in the Appendix A. □

Index Privacy. Index privacy means that a probabilistic polynomial-time adversary cannot get any useful information from encrypted keyword C_w . In other words, the cloud server cannot determine which keyword the ciphertext is for.

- Setup: The challenger plays a role as the system and runs $Setup()$, then it keeps master key msk .
- Query Phase1: The adversary adaptively queries the ciphertext of keyword w for polynomial times, and get $C_w \leftarrow Enc(pp, w)$.
- Challenge phase: The adversary randomly selects two keywords w_{i_0} and w_{i_1} which have not queried before, and sends them to the challenger. The challenger flips a coin to select $\beta \leftarrow \{0, 1\}$ and then returns $C_w \leftarrow Enc(pp, w_{i_\beta})$ to the adversary.
- Query Phase 2: The adversary continues to query the ciphertext of keyword w as Phase1 did with the restriction that w is neither w_{i_0} nor w_{i_1} .
- Guess: Finally, the adversary gives a guess β' of β and wins the game if $\beta' = \beta$. We can define the advantage of adversary winning the game is $|Pr[\beta' = \beta] - \frac{1}{2}|$.

Theorem 2. *If an adversary wins the game mentioned above with a non-negligible advantage, our scheme is secure with index privacy.*

Proof. If the adversary wants to know whether C_w is about w_{i_0} or w_{i_1} , he will analyze the $C_w = g_T^{\langle \mathbf{x}_{i_\beta}, \mathbf{y}_{i_\beta} \rangle} = g_T^{\alpha_j}$. Because α_j is less than $\frac{\log_{g_T} q}{2}$, he could get $\langle \mathbf{x}_{i_\beta}, \mathbf{y}_{i_\beta} \rangle$ by logarithmic operation. However, \mathbf{y}_{i_β} is kept secret by the challenger, so that it is impossible for the adversary to get \mathbf{x}_{i_β} and has no chance to get keyword w_{i_β} . Therefore, the adversary can not know whether β equals to 0 or 1. □

Token Privacy. Token privacy means that given a search token, a probabilistic polynomial-time adversary cannot learn which keyword the search token is for. Namely, the cloud server cannot know which keyword the data user queries.

- Setup: The challenger plays a role as the system and runs $Setup()$, then it keeps master key msk .
- Init: The challenger runs $Enc()$ to get reasonable $\{\mathbf{x}_i\}_{i \in [m]}$, $\{\mathbf{y}_i\}_{i \in [m]}$ and $\{\xi, r_\xi\}_{\xi \in DU}$, with which it can run $ClientKGen()$ to obtain secret key sk . Besides, it runs $PartialTokenGen()$ to get partial token pt .
- Query Phase1: The adversary with ξ adaptively queries search token of keyword w for polynomial times, and get $st \leftarrow Token(sk, pt, w)$.
- Challenge phase: The adversary randomly selects two keywords w_{i_0} and w_{i_1} which have not queried before, and sends them to the challenger. The challenger flips a coin to select $\beta \leftarrow \{0, 1\}$ and then runs $TokenGen()$ algorithm. The challenger returns $st \leftarrow Token(sk, pt, w)$ to the adversary.
- Query Phase 2: The adversary continues to query search token of keyword w as Phase1 did with the restriction that w is neither w_{i_0} nor w_{i_1} .
- Guess: Finally, the adversary gives a guess β' of β and wins the game if $\beta' = \beta$. We can define the advantage of adversary winning the game is $|Pr[\beta' = \beta] - \frac{1}{2}|$.

Theorem 3. *If an adversary wins the game mentioned above with a non-negligible advantage, our scheme is secure with token privacy.*

Proof. If the adversary wants to know whether st is about w_{i_0} or w_{i_1} , he will analyze the $st = g_T^{\langle \mathbf{x}_{i\beta}, \mathbf{y}_{i\beta} \rangle + r\epsilon}$. For the cloud, although he can calculate $g_T^{\langle \mathbf{x}_{i\beta}, \mathbf{y}_{i\beta} \rangle}$ and get $\langle \mathbf{x}_{i\beta}, \mathbf{y}_{i\beta} \rangle$ by logarithmic operation, the cloud has no way to get $\mathbf{y}_{i\beta}$. Therefore, it is incapable of getting $\mathbf{x}_{i\beta}$ and unable to get keyword $w_{i\beta}$. \square

7 Functionality and Efficiency

We compare our scheme with the work in [13,22] and the first scheme of work in [12] in Table 1. From the table, we can see that the size of ciphertext is linear with the number of keywords m in both literature [13] and our scheme. While in literature [22], the ciphertext size is linear with the product of the number of keywords m and the number of files n . And in literature [12], the ciphertext size is linear with the number of keywords data owner extracts. We can easily find that the size of the search token is constant only in our scheme. Obviously, our scheme could significantly reduce the communication and transportation overhead, especially when the number of data users is large and the query frequency is high. Besides, by using an inverted index structure and super-incremental sequence, our scheme achieves efficient multi-keyword search, which is illustrated in Table 1. In addition, our scheme avoids the per-query interaction between data owner and data users. That is to say, the data owner does not need to stay online waiting for data users to search in his archives. Furthermore, our scheme supports multi-keyword search in single-owner/multi-user setting.

Table 1. Calculation overhead

	[12]	[13]	[22]	Ours
Ciphertext size	$o(n\alpha)$	$o(m)$	$o(mn)$	$o(m)$
Token size	$o(n + t)$	$o(ct)$	$o(m)$	2
Search time	$o(m)$	$o(ct)$	$o(wm\log_2 n)$	$o(t\log_2 m)$
Inverted index	×	√	×	√
Multi-keyword	√	√	√	√
Multi-user	×	√	×	√
Non interaction*	N/A	×	N/A	√

*: the interaction between data owner and data users whenever data users perform search queries.

n : the number of files.

m : the number of keywords.

t : the number of keywords data user wants to search.

c : the number of files including the least frequency keywords among keywords data user wants to search.

w : the number of files including the keywords data user wants to search.

α : the number of keyword in each file, which its fixed in [13].

8 Conclusion

In our scheme, search tokens have only two items by tactfully leveraging multi-input inner-product functional encryption, which reduces communication and transportation overhead significantly. The use of inverted index structure and super-incremental sequence makes the multi-keyword search process efficient. In addition, our scheme avoids the per-query interaction between data owner and data users. That is to say, data owner does not need to stay online waiting for data users to search in his archives. What is more, our scheme ensures the correctness of search process and protects the privacy of keywords and plaintext files.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China (Grant No.61632012, 61672239, and U1509219) and in part by “the Fundamental Research Funds for the Central Universities”. Zhenfu Cao and Jiachen Shen are the corresponding authors.

A Proof of Theorem1

Proof. The proof of theorem1 consists of six games. The transitions between contiguous games are summarized in Table 2. $([c_i]_1, [c_i']_1, [c_i'']_1)$ is computed by *PartialTokenGen*(\cdot). $[d_i]_2$ and $[Z_i]_T$ are part of sk computed by *ClientKGen*(\cdot). We use $\mathbf{u} \xleftarrow{R} \mathbb{Z}_q^3 \setminus Span(\mathbf{A})$ and $\mathbf{a}^\perp \xleftarrow{R} \mathbb{Z}_q^3$ so that $\mathbf{A}^T \mathbf{a}^\perp = \mathbf{0}$ and $\langle \mathbf{u}, \mathbf{a}^\perp \rangle = 1$. To analyze Game3, we consider the selective variant of the game: Game3*. Then we prove Game3* through using an information-theoretic argument via Game4*. Both Game3* and Game4* are selective security.

The concrete proof is similar to [1]. Here we just simply outline each game.

Let \mathcal{A} be a PPT adversary, and let $\lambda \in N$ be the security parameter. We define

$$Adv_t(\mathcal{A}) := Pr[Game_t(1^\lambda, \mathcal{A}) = 1] \quad t \in \{0, 1, 2, 3, 3^*, 4^*\}$$

- Game 0: is the partial-token privacy game.
- Game 1: using \mathcal{U}_k -MDDH, we change the distribution of the vectors $[c_i]_1$ computed by *PartialTokenGen*(\cdot). This change depends on the fact that:
 - The distributions $\{s_{\xi,i}\}_{i \in [m]}$ and $\{s_{\xi,i} + s\}_{i \in [m]}$, where $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^2$, $s_{\xi,i} \xleftarrow{R} \mathbb{Z}_q^2$ are identically distributed.
 - By \mathcal{U}_k -MDDH assumption, we can switch $([\mathbf{A}]_1, [\mathbf{A}\mathbf{s}]_1)$ to $([\mathbf{A}]_1, [\mathbf{u}]_1)$, where $\mathbf{A} \xleftarrow{R} \mathcal{U}_2$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^2$, $\mathbf{u} \xleftarrow{R} \mathbb{Z}_q^3$.
 - The uniform distributions over \mathbb{Z}_q^3 and $\mathbb{Z}_q^3 \setminus Span(\mathbf{A})$ are $\frac{1}{q}$ -close.
- Game 2: using an information theoretic argument, we change the way how the vectors $[c_i'']_1$ and $[d_i]_2$ are computed by *PartialTokenGen*(\cdot) and

$ClientKGen()$ respectively. This relies on the fact that the distributions \mathbf{V} and $\mathbf{V} - \mathbf{z}_i(\mathbf{a}^\perp)^T$ are identical. Thus, we have

$$\begin{aligned} [\mathbf{c}_i'']_1 &= [(\mathbf{V} - \mathbf{z}_i(\mathbf{a}^\perp)^T)(\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}) + \mathbf{z}_i]_1 \\ &= [\mathbf{V}(\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}) - \mathbf{z}_i(\mathbf{a}^\perp)^T\mathbf{u} + \mathbf{z}_i]_1 \\ &= [\mathbf{V}(\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u})]_1 \end{aligned} \tag{8}$$

$$\begin{aligned} \mathbf{d}_i &= \mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + (\mathbf{V}^T - \mathbf{a}^\perp \mathbf{z}_i^T)\mathbf{r}_{\xi_1} \\ &= \mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1} - \mathbf{a}^\perp \langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle \end{aligned} \tag{9}$$

- Game 3: we switch $\{[\mathbf{r}_{\xi_1}]_2, \langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle\}_2\}_{i \in [m]}$ to $\{[\mathbf{r}_{\xi_1}]_2, [\tilde{\mathbf{z}}_i]_2\}_2\}_{i \in [m]}$ for all calls to $ClientKGen()$, where $\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2, \dots, \tilde{\mathbf{z}}_m \xleftarrow{R} \mathbb{Z}_q$. This is justified by the distributions $[\mathbf{r}_{\xi_1}^T \parallel \langle \mathbf{z}_1, \mathbf{r}_{\xi_1} \rangle \parallel \dots \parallel \langle \mathbf{z}_m, \mathbf{r}_{\xi_1} \rangle]_2 \in \mathbb{G}_2^{1 \times (2+m)}$ and $[\mathbf{r}_{\xi_1}^T \mathbf{U}^T]_2$, where $\mathbf{U} \xleftarrow{R} \mathcal{U}_{2+m,2}$ are identical. According to $\mathcal{U}_{2+m,2}$ -MDDH, we know that $[\mathbf{r}_{\xi_1}^T \mathbf{U}^T]_2$ is indistinguishable from a random vector over $\mathbb{G}_2^{1 \times (2+m)}$ of the form $[\mathbf{r}_{\xi_1}^T \parallel \tilde{\mathbf{z}}_1 \parallel \dots \parallel \tilde{\mathbf{z}}_m]_2$.
- Game 3*: is the selective variant of Game3, in other words, any adversary playing this game has to commit its challenge queries w_{i_0} and w_{i_1} beforehand.
- Game 4*: is similar to Game3*, except t $[\mathbf{c}_i']_1$ and $[\mathbf{Z}_i]_T$ computed by $PartialTokenGen()$ and $ClientKGen()$ respectively. The transform is true because of the fact $\{\tilde{\mathbf{z}}_i - \langle \mathbf{x}_i + \mathbf{r}_{\xi_{pt}}, \mathbf{y}_i + \mathbf{r}_{\xi_{sk}} \rangle\}_{i \in [m]}$ and $\{\tilde{\mathbf{z}}_i\}_{i \in [m]}$ are identically distributed. Besides, \mathbf{M} and $\mathbf{M} - \mathbf{x}_i(\mathbf{a}^\perp)^T$ are identically distributed.

We build a PPT adversary \mathcal{B}_1 so that

$$Adv_0(\mathcal{A}) - Adv_1(\mathcal{A}) \leq Adv_{\mathbb{G}_1, \mathcal{B}_1}^{\mathcal{U}_k - MDDH}(\lambda) + \frac{1}{q}$$

Because of information theoretic argument, we know that

$$Adv_1(\mathcal{A}) = Adv_2(\mathcal{A})$$

Table 2. Sequence of games for the proof of partial-token privacy

Game	\mathbf{c}_i	\mathbf{c}_i'	\mathbf{c}_i''	\mathbf{d}_i	Z_i
0	$\mathbf{A}\mathbf{s}_{\xi_i}$	$\mathbf{M}\mathbf{c}_i + \mathbf{x}_i + \mathbf{r}_{\xi_{pt}}$	$\mathbf{V}\mathbf{c}_i + \mathbf{z}_i$	$\mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1}$	$\langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle$
1	$\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}$	$\mathbf{M}\mathbf{c}_i + \mathbf{x}_i + v\mathbf{r}_{\xi_{pt}}$	$\mathbf{V}\mathbf{c}_i + \mathbf{z}_i$	$\mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1}$	$\langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle$
2	$\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}$	$\mathbf{M}\mathbf{c}_i + \mathbf{x}_i + \mathbf{r}_{\xi_{pt}}$	$\mathbf{V}\mathbf{c}_i$	$\mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1} - \mathbf{a}^\perp \langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle$	$\langle \mathbf{z}_i, \mathbf{r}_{\xi_1} \rangle$
3	$\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}$	$\mathbf{M}\mathbf{c}_i + \mathbf{x}_i + \mathbf{r}_{\xi_{pt}}$	$\mathbf{V}\mathbf{c}_i$	$\mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1} - \mathbf{a}^\perp \tilde{\mathbf{z}}_i$	$\tilde{\mathbf{z}}_i$
3*	$\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}$	$\mathbf{M}\mathbf{c}_i + \mathbf{x}_i + \mathbf{r}_{\xi_{pt}}$	$\mathbf{V}\mathbf{c}_i$	$\mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1} - \mathbf{a}^\perp \tilde{\mathbf{z}}_i$	$\tilde{\mathbf{z}}_i$
4*	$\mathbf{A}\mathbf{s}_{\xi_i} + \mathbf{u}$	$\mathbf{M}\mathbf{c}_i + \mathbf{r}_{\xi_{pt}}$	$\mathbf{V}\mathbf{c}_i$	$\mathbf{M}^T(\mathbf{y}_i + \mathbf{r}_{\xi_{sk}}) + \mathbf{V}^T\mathbf{r}_{\xi_1} - \mathbf{a}^\perp \tilde{\mathbf{z}}_i$	$\tilde{\mathbf{z}}_i - \langle \mathbf{x}_i + \mathbf{r}_{\xi_{pt}}, \mathbf{y}_i + \mathbf{r}_{\xi_{sk}} \rangle$

There exists a PPT adversary \mathcal{B}_3 so that the

$$Adv_2(\mathcal{A}) - Adv_3(\mathcal{A}) \leq Adv_{\mathbb{G}_2, \mathcal{B}_3}^{\mathcal{U}_k - MDDH}(\lambda) + \frac{1}{q-1}$$

Using complexity leveraging, we build a PPT adversary \mathcal{B}_{3^*} so that

$$Adv_3(\mathcal{A}) \leq m(m-1) \cdot Adv_{3^*}(\mathcal{B}_{3^*})$$

For all adversaries \mathcal{A} , $Adv_{3^*}(\mathcal{A}) = Adv_{4^*}(\mathcal{A})$.

In Game 4*, from Table II, we can easily find that the partial token of $w_{i\beta}$ of ξ is only associate with vector $\mathbf{s}_{\xi, i\beta}$. However, $\mathbf{s}_{\xi, i\beta}$ is randomly chosen from \mathbb{Z}_q^k . That is to say, $\mathbf{A}\mathbf{s}_{\xi, i0}$ and $\mathbf{A}\mathbf{s}_{\xi, i1}$ are statistically indistinguishable. Thus, $Adv_{4^*}(\mathcal{A}) = 0$. Therefore, we obtain that

$$Adv_0(\mathcal{A}) \leq Adv_{\mathbb{G}_1, \mathcal{B}_1}^{\mathcal{U}_k - MDDH}(\lambda) + Adv_{\mathbb{G}_2, \mathcal{B}_3}^{\mathcal{U}_k - MDDH}(\lambda) + \frac{2}{q-1}$$

Using $\mathcal{U}_{i,k} - MDDH$ assumption in $\mathbb{G}_1, \mathbb{G}_2$, we know that $Adv_0(\mathcal{A})$ is negligible in λ .

References

1. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings, pp. 601–626 (2017)
2. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005). https://doi.org/10.1007/11602897_35
3. Cash, D., et al.: Dynamic searchable encryption in very-large databases: data structures and implementation. In: Network and Distributed System Security Symposium (2014)
4. 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
5. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
6. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
8. Dhumal, A.A., Jadhav, S.: Confidentiality-conserving multi-keyword ranked search above encrypted cloud data. *Procedia Comput. Sci.* **79**, 845–851 (2016)

9. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8
10. Fu, Z., Wu, X., Wang, Q., Ren, K.: Enabling central keyword-based semantic extension search over encrypted outsourced data. *IEEE Trans. Inf. Forensics Secur.* **12**(12), 2986–2997 (2017)
11. Goh, E.J.: Secure indexes. IACR Cryptology ePrint Archive 2003, 216 (2003)
12. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_3
13. Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Outsourced symmetric private information retrieval. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 875–888 (2013)
14. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
15. 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
16. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
17. Li, H., Liu, D., Dai, Y., Luan, T.H., Shen, X.S.: Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage. *IEEE Trans. Emerg. Top. Comput.* **3**(1), 127–138 (2015)
18. Li, H., Yang, Y., Luan, T., Liang, X., Zhou, L., Shen, X.: Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. *IEEE Trans. Dependable Secure Comput.* **13**(3), 312–325 (2016)
19. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_27
20. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
21. Hwang, Y.H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73489-5_2
22. Zhu, X., Dai, H., Yi, X., Yang, G., Li, X.: MUSE: an efficient and accurate verifiable privacy-preserving multikeyword text search over encrypted cloud data. *Secur. Commun. Netw.* **2017**(2), 1–17 (2017)



Fast Two-Server Multi-User Searchable Encryption with Strict Access Pattern Leakage

Cédric Van Rompay^(✉), Refik Molva, and Melek Önen

EURECOM, Biot, France
{vanrompa,molva,onen}@eurecom.fr

Abstract. A recent paper showed that most Multi-User Searchable Encryption protocols do not provide any privacy without the assumption that all users can be trusted, an assumption too strong to be realistic for a MUSE system. As to the few MUSE protocols that are not affected, they all suffer from some scalability issues. We present the first MUSE protocol that does protect against user-server collusions, and yet scales very well. The protocol is also very simple. We prove that the leakage of the protocol is limited to the access pattern of queries and we report on performance measurements from a proof-of-concept implementation.

Keywords: Multi-user searchable encryption · Diffie-Hellman Access pattern

1 Introduction

The advent of cloud computing allowed an increasing number of users to delegate tasks to Cloud Service Providers (CSP). However users are reluctant to trust CSPs regarding the handling of their data. Simple client-side encryption would solve the privacy problem, but would prevent any useful operation on the data server-side. This motivated research on Searchable Encryption (SE) (see [3,10] and their references) which goal is to allow a CSP to search some outsourced data on behalf of a user without compromising the privacy of this data and the privacy of the queries.

While current state-of-the-art SE schemes [4,7,9] can efficiently process very large databases, these protocols only consider a single user being both the only one uploading data and the only one searching it. At the same time, research in SE also studied the situation where the dataset is being written and/or searched by several users. Multi-User SE (MUSE) denotes the setting with many readers *and* writers.

MUSE is a recent but active research topic [1,2,8,11,12,14,15,18–21,25,26]; however it seems very difficult to reconcile security and efficiency in MUSE. Prior to the paper of Popa and Zeldovich [18], papers on MUSE were only considering the server as a threat, implicitly assuming that all users were fully trusted.

Popa and Zeldovich were the first to address user-server collusions in MUSE and to present a protocol, MKSE, supposed to provide privacy in such a model. However this protocol was shown in [22] to fail as well to protect privacy against user-server collusions. New MUSE protocols were presented in [12, 20, 21] that seem to reach an acceptable level of privacy against user-server collusions, but they all suffer from scalability issues.

In this paper, we identify different mechanisms present in recent MUSE protocols [12, 20, 21] that trade some privacy for an efficiency increase, and we show that combining them leads to a simple and efficient MUSE protocol which privacy level stays acceptable. The protocol we present, resembling an existing PSI protocol [13], is the first MUSE protocol to have both a very light user workload and a moderate server workload while being secure against user-server collusions.

We prove the security of the protocol using the “simulation technique” [17] in the random oracle model, and we report on performance measurements of a proof-of-concept implementation.

2 Multi-User Searchable Encryption

We give definitions for MUSE that are general enough to apply to all existing constructions. In Sect. 6 we apply these definitions to the protocol we present using a more formal syntax.

A MUSE protocol involves a **server** and a number of **users**. Users can be of type either **reader** or **writer**. A writer owns some **records** and uploads them to the server (in an encrypted form). For each record, the writer owning it can authorize some readers to search it. The **authorization graph** denotes the information of “which reader has access to which record”. A reader can search the records for which she got authorization by sending a **query** to the server in an encrypted form called **trapdoor**. We will only consider **single-keyword search**, meaning that records are defined as sets of keywords, a query consists of a single keyword and we say that a record **matches** a query if the query is present in the record. Keywords are defined as bit strings. At the end of the search procedure, the server sends a **response** back to the querying reader (possibly encrypted) who outputs the ids of records that match the query among the records this reader was authorized to search.

We note $W_d \in \{0, 1\}^*$ the record with id d and we represent the authorization graph by a function $Auth$ such that for any reader $r \in R$ we have $d \in Auth(r)$ if and only if r is authorized to search W_d . If q is the keyword queried by reader r and a is the query result that r outputs at the end of the search protocol, the protocol is correct if the following holds with overwhelming probability:

$$a = \{d \in Auth(r) : q \in W_d\} \quad (1)$$

Regarding security, the adversary we consider is a collusion of the server and some users. We consider the adversary as honest-but-curious (see [17]), as it is common in the literature on MUSE. Following the seminal paper of Curtmola et al. [6], we define the **history** of a MUSE protocol as the records, the queries,

and the authorizations. We define the **leakage** as a function of the history, and we say that a MUSE protocol has some leakage with respect to an adversary if the view of this adversary can be simulated in an indistinguishable way using only the information from this leakage.

We define several notions that will be helpful when describing leakage functions: the **access pattern** denotes the information of which record matched which query. “Access pattern” is thus a synonym of “query result” (see Eq. (1)). The term **benign leakage** will regroup all the information we consider as non-sensitive. It consists of the size of each record, the number of queries from each reader, and the authorization graph. All MUSE protocols reveal this benign leakage. As a result we will sometimes omit the benign leakage, saying that some protocol “only leaks the access pattern” while it also leaks the benign leakage. Finally the **revealed content** denotes the queries and records which the adversary has a legitimate access to through the users it controls. It includes the queries of corrupted readers and the records of corrupted writers, but also the records corrupted readers have access to. For the same reasons, we often omit it as well when describing the leakage of protocols.

3 Preliminaries

Diffie-Hellman Problems. Given some cyclic group \mathbb{G} of order ζ having generator g , the Computational Diffie-Hellman (CDH) problem consists, given (g^a, g^b) in \mathbb{G}^2 , to compute g^{ab} . The Decisional Diffie-Hellman (DDH) problem consists, given any triplet (g^a, g^b, g^c) in \mathbb{G}^3 to outputs “true” if $c = ab$ and “false” otherwise. Groups where the CDH and DDH problems are assumed to be hard are very widely used in practical cryptography. We will note h a cryptographic hash function, modeled as a random oracle, that hashes any bit string into \mathbb{G} .

4 Related Work

The first MUSE protocol was proposed by Hwang and Lee in [14]. It slightly differs from our definition of MUSE because their protocol considers records as tuples of keywords.

A MUSE protocol that is important in our study is the one of Bao, Deng, Ding and Yang in [2], that uses a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ where \mathbb{G}_1 , \mathbb{G}_2 and \mathbb{G}_T be multiplicative cyclic groups of order ζ . In this protocol, a Trusted Third Party (TTP) creates a master key $msk \in \mathbb{Z}_\zeta^*$. Then for each user u (users are both readers and writers in this protocol), the TTP creates a secret user key $k_u \in \mathbb{Z}_\zeta^*$ sent to the user and a value called “complementary key” (later called “delta value”) g_2^{msk/k_u} that is sent to the server. For the creation of both trapdoors and encrypted records, user u encrypts a keyword w as $h_e(w)^{k_u}$ (where h_e is a secure hash function to \mathbb{G}_1) and the server pairs the encrypted keyword with the complementary key to obtain the following:

$$e(h_e(w)^{k_u}, g_2^{msk/k_u}) = e(h_e(w), g_2)^{msk} \quad (2)$$

As a result while each user has her own secret key, the use of the complementary key makes the protocol equivalent to a single user encrypting her records and queries using master key msk , while this master key is in fact only known by the TTP. This kind of MUSE protocols where all records and all trapdoors are re-encrypted under a common secret key are called **single-key**.

A paper by Yang et al. [26] adds a few extensions to the protocol of Bao et al. [2] without changing its basic behaviour. Finally, a paper by Dong et al. [8] presents a protocol that works in a similar fashion, but is based on RSA encryption instead of bilinear pairings.

4.1 The MKSE Protocol

In [18], Popa and Zeldovich present a MUSE protocol named “Multi-Key Searchable Encryption” (MKSE). This protocol introduces radical changes from the previous MUSE protocols in order to address a much more challenging threat model where some users may be colluding with the server. MKSE does not follow the “single-key” structure because a single corrupted user in a single-key structure gives the adversary immediate access to the entire database.

There is no TTP in MKSE; instead, user u creates his own secret key $\gamma_u \in \mathbb{Z}_\zeta^*$ and can authorize user v to search his record by computing the delta value $g_2^{\gamma_u/\gamma_v}$. User u encrypts keyword w as $e(h_e(w), g_2)^{\gamma_u}$, user v encrypts query q as $h_e(q)^{\gamma_v}$ which is transformed by the server using the delta value. Similarly as in Bao et al. [2], we have:

$$e(h_e(q)^{\gamma_v}, g_2^{\gamma_u/\gamma_v}) = e(h_e(q), g_2)^{\gamma_u} \quad (3)$$

The main difference between MKSE and [2] is that in MKSE the encrypted keywords are never transformed, but trapdoors are transformed to match the encrypted keywords. While this requires the server to compute a pairing for each record the querying user is allowed to search, it also ensures that the trapdoor of a user can only be applied on the records this user was allowed to search, mitigating the consequences of user corruptions.

The MKSE protocol had quite some impact. [18] has been cited by a number of papers [23–25], most of them using it as a base and suggesting improvements and extensions to it. Also, the MKSE protocol is at the core of the Mylar platform, presented in [19], that aims at facilitating the development of secure web applications.

4.2 Insecurity of the Iterative Testing Structure and Recent Protocols

In [22], Van Rompay et al. show that none of the previously mentioned MUSE protocols can offer privacy against even a very small number of users colluding with the server, because they all follow a common structure named “iterative testing” in [22]. Interestingly this affects the MKSE protocol as well, despite the fact that it was designed to protect against such collusions.

Intuitively, iterative testing denotes the fact that the server sees encrypted records as lists of encrypted keywords and that search consists in testing each encrypted keyword one by one. When a query matches a record, the server can see which encrypted keyword matched the query. This can reveal when two queries from different readers are similar because they will match the same encrypted keyword. As a result, the corruption of one user can lead to the recovery of queries of other, non colluding users, which in turn can lead to the recovery of keywords in records the colluding reader did not have access to. Results from some simulations in [22] show that even a very small number of colluding users can lead to a major loss of privacy across the whole dataset.

Some recent papers on MUSE [12, 20, 21] present protocols that do not follow the iterative testing structure and achieve privacy against user-server collusions. Nevertheless all these protocols suffer from some form of scalability issues. In the protocol by Hamlin et al. [12] a reader must download and process every single record he is allowed to search before re-uploading the processed version to the server. Similarly in [21], the response received by a reader has a size that is linear with the number of records being searched. This goes against the main goal of cloud computing which is to allow end users with small capacities to process large amounts of data. Finally in the 2018 protocol of Van Rompay et al. [20], while the user workload is small and independent of the number of records searched, the server workload is significant and the absence of an implementation makes it difficult to assert the practicality of the protocol.

4.3 An Unexplored Middle Ground

All MUSE protocols suffer from either insecurity ([2, 8, 18] and derivatives) or scalability issues [12, 20, 21] (See Fig. 1). Among the protocols that are secure against user-server collusions, we note various techniques which trade some security for a gain in efficiency. We suggest to combine these techniques, hoping that their performance advantages add up together, resulting in a level of scalability that was not reached before among this kind of MUSE protocols.

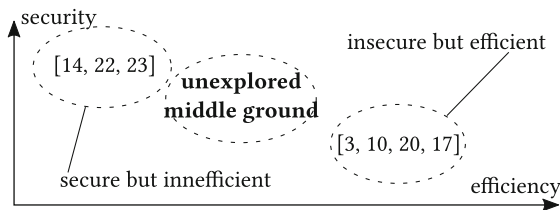


Fig. 1. A representation of our notion of “unexplored middle ground” regarding the security/efficiency dilemma in MUSE.

These techniques consist of:

- In Hamlin et al. [12], trading off access pattern leakage for lower complexity in underlying mechanisms. Leaking the access pattern is very common among single-user SE schemes, and leaking *no more than* the access pattern does not lead to the kind of security issues iterative-testing-based protocols suffer from. Accepting to leak the access pattern avoids using the kind of costly mechanisms present in the protocols from Van Rompay et al. [20,21].
- In the two protocols of Van Rompay et al., the use of two servers that are assumed not to collude together. This kind of assumption is present and well-accepted in various other protocols such as Private Information Retrieval [5]. While relying on such an assumption slightly weakens the privacy guarantees, it is obviously much better than having to assume the absence of any user-server collusion. Having two non-colluding servers makes it easier to protect the privacy of the records and queries, as required by protocols based on iterative testing. Note that *none of the two servers are trusted*, both are modeled as independent honest-but-curious adversaries.

We view the combination of these techniques as a “middle ground” that has not been studied yet, represented in Fig. 1, where security is only slightly weaker than in the latest protocols while scalability could be significantly improved.

5 Idea of the Protocol

We show that accepting to leak the access pattern while assuming the presence of two non-colluding servers leads indeed to a simple, efficient and scalable solution to the MUSE problem. We claim that the MUSE protocol we present may be the best practical tradeoff as of today for the MUSE problem with a very large number of records.

The protocol is similar to a Private Set Intersection (PSI) protocol presented in [13] (see also [16]), which we will call “DH-PSI”, that is solely based on the Diffie-Hellman protocol. A (one-sided) PSI protocol involves a sender with set Y and a receiver with set X , and the receiver must learn $X \cap Y$ and the size of Y while the sender must learn nothing beyond the size of X . Remark that set membership test, which is what our MUSE protocol does, is a special case of set intersection: $q \in W_q$ is equivalent to $\{q\} \cap W_q \neq \emptyset$. In DH-PSI [13], the receiver picks a random value $\alpha \in \mathbb{Z}_\zeta^*$ and sends $\{h(x)^\alpha \forall x \in X\}$ to the sender. The sender picks a random value $\beta \in \mathbb{Z}_\zeta^*$ and sends both $\{(h(x)^\alpha)^\beta \forall x \in X\}$ and $\{h(y)^\beta \forall y \in Y\}$. Finally the receiver computes $\{(h(y)^\beta)^\alpha \forall y \in Y\}$ and is able to see which elements of X are in Y without learning anything about the elements in $Y - X$. Interestingly, this protocol was shown in [16] to be the fastest existing PSI protocol when one set is much larger than the other one, which corresponds to our case. Our protocol can actually be considered as an “outsourced” version of the protocol of [13]. The reader of MUSE would be the receiver in PSI and the writer would be the sender, but instead of interacting together in a direct manner, the receiver sends her masked set to the (non-trusted) **proxy** and her

secret to the server while the sender sends her masked set to the server and her secret key to the proxy. Both the proxy and the server apply the key they received on the masked set they received in order to compute a “double-masked” set. Finally the proxy determines the intersection between the double-masked set it computed and the one transmitted by the server. The result is returned to the reader as the response to its query. The proxy or the server could “cheat” and try to apply other keys or blinding factors, but the result will be of no use unless the server sends extra prepared records to the proxy, which would violate the honest-but-curious model and/or the assumption that the two servers do not collude together.

As to the handling of queries from a same reader, we note that there is no need to renew the blinding factor at each query. This saves a great amount of computation because both the preparation step and the sending of prepared records are skipped. Instead, it suffices that the reader avoids sending two identical queries for a fixed period of time, say a month. The exact time period should be chosen depending on how many query results the reader can remember, and on how fast new keywords are added to records.

6 The Diffie-Hellman AP-MUSE Protocol

- The writer owning record $W_d \in \{0, 1\}^*$ picks record key γ_d at random from \mathbb{Z}_ζ^* . She encrypts this record into \overline{W}_d by computing:

$$\overline{W}_d \leftarrow \{h(w)^{\gamma_d} \ \forall w \in W_d\}$$

She sends \overline{W}_d to the server and γ_d to the proxy.

- The writer owning record W_d can authorize a reader r to search W_d simply by notifying the proxy and the server.
- For each time period l , reader r picks a random blinding factor $\xi_{r,l} \in \mathbb{Z}_\zeta^*$ and sends it to the server.
- When the server receives blinding factor $\xi_{r,l}$, it computes the prepared encrypted record $\overline{\overline{W}}_{d,r,l}$ for each $d \in Auth(r)$:

$$\overline{\overline{W}}_{d,r,l} \leftarrow \{\overline{w}^{\xi_{r,l}} \ \forall \overline{w} \in \overline{W}_d\}$$

It sends $\overline{\overline{W}}_{d,r,l}$ to the proxy.

- $q_{r,l,s}$ denotes the s -th query of reader r during the l -th time period. For each such query, reader r creates the corresponding trapdoor $t_{r,l,s}$:

$$t_{r,l,s} \leftarrow h(q_{r,l,s})^{\xi_{r,l}}$$

$t_{r,l,s}$ is sent to the proxy.

- When receiving trapdoor $t_{r,l,s}$, the proxy does the following steps for each record the querying reader is authorized to search, that is for each $d \in Auth(r)$:

- the proxy computes the transformed trapdoor $t'_{r,l,s,d}$:

$$t'_{r,l,s,d} = t_{r,l,s}^{\gamma_d}$$

- the proxy looks for value $t'_{r,l,s,d}$ in prepared record $\overline{\overline{W}}_{d,r,l}$. If the value is found, we say that W_d matches.

The proxy sends the ids of the matching records to the querying reader.

We assume that a reader does not send similar queries during the same time period, that is, $q_{r,l,s} \neq q_{r,l,s'}$. However queries from different readers during the same time period can be similar, that is, we can have $q_{r,l,s} = q_{r',l,s'}$.

7 Security Analysis

The security of the protocol derives from the hardness of the DDH problem in an almost obvious way. Intuitively, both the proxy and the server receive keywords that are “masked” by some key they do not know, the key being a blinding factor in the case of the proxy and a record key in the case of the server. However we still give a rigorous proof based on the simulation technique as it is usual in the field of Searchable Encryption (see [4]). We prove security against the server and proxy separately, but because the two proofs are very similar we start by giving an overview of them.

We first give a formal definition of security in MUSE, adapted from the definition of “Non-adaptive semantic security” by Curtmola et al. [6].

Definition 1 (Non-adaptive semantic security of a MUSE protocol). *Let MUSE be a MUSE protocol. Let $\mathcal{V}_{\text{MUSE},\mathcal{A}}$ be an algorithm which takes a MUSE history, runs protocol MUSE on this history, and outputs the view of adversary \mathcal{A} during this execution. Let \mathcal{S} be a simulator and κ be the security parameter.*

We say that MUSE is semantically secure with leakage \mathcal{L} with respect to \mathcal{A} (or simply that it has leakage \mathcal{L} w.r.t \mathcal{A}) if for all polynomial-size \mathcal{D}_1 , there exists a polynomial-size simulator \mathcal{S} such that for all polynomial-size \mathcal{D}_2 , the following quantity is negligible in κ :

$$\begin{aligned} & | \Pr[\mathcal{D}_2(st_{\mathcal{D}}, \mathcal{V}(\kappa, \mathcal{H})) = 1; (st_{\mathcal{D}}, \mathcal{H}) \leftarrow \mathcal{D}_1(1^\kappa)] \\ & - \Pr[\mathcal{D}_2(st_{\mathcal{D}}, \mathcal{S}(\kappa, \mathcal{L}(\mathcal{H}))) = 1; (st_{\mathcal{D}}, \mathcal{H}) \leftarrow \mathcal{D}_1(1^\kappa)] | \end{aligned}$$

In each proof we build a simulator that takes the leakage as input and that outputs a simulated view. The only parts of the view that are not trivial to build for the simulator are the encrypted keywords (for privacy against the server) and the trapdoors and prepared keywords (for privacy against the proxy) that are not revealed. Most of them are generated by replacing the call to hash function h , modeled as a programmable Random Oracle (RO), by a uniform random sampling from \mathbb{G} . We say “most of them” because some prepared keywords in the simulated view of the proxy are instead generated by taking a trapdoor and

transforming it (using the record key), in order to have the trapdoor matching the resulting prepared record so that access pattern is preserved.

We show that the output of simulators are indistinguishable from the real view of the adversary with a sequence of hybrid simulators where each hybrid simulates one more element of the view than the previous one. The output of any two successive hybrid simulators are shown to be indistinguishable with a reduction to the DDH problem in \mathbb{G} , using the following embedding of a DDH instance g^a, g^b, g^c :

$$h(w^*) \leftrightarrow g^a, \quad \gamma^* \leftrightarrow b, \quad \bar{w}^* \leftrightarrow g^c$$

Where w^* is the keyword corresponding to the trapdoor (or prepared keyword or encrypted keyword, depending on the case) that is simulated in one hybrid simulator but not in the other, called the **pivot** trapdoor/prepared keyword/encrypted keyword, and γ^* (or ξ^* for privacy against the proxy) is the record key (resp. blinding factor) corresponding to the pivot element. The embedding is done by programming the RO as follows: On input w^* it outputs g^a , and on any other input it outputs $g^{\mathcal{O}[w]}$ where $\mathcal{O}[w]$ is previously picked uniformly at random if it was not already set, as is usually done with ROs. There is a subtlety in the programming of the RO because it must be programmed before the value w^* is available to the reduction. We give more details on this point further below.

The embedding of b is made possible by the way we define the RO: encrypting a keyword using b as the record key or blinding factor is done with the following function:

$$w \mapsto (g^b)^{\mathcal{O}[w]}$$

Finally the embedding of c is done by using g^c as the pivot trapdoor/prepared keyword/encrypted keyword.

The only thing that remains to be done in each of the proof is the argumentation over the correctness of the embedding, essentially making sure that the value we replace by g^c does not appear anywhere else. This is where our requirements that records do not contain duplicates and that a reader does not send two identical queries during the same time period are needed.

Due to space limitations, the full proofs are given in appendix.

About Programming the RO. Programming the random oracle requires to know w^* . The reduction will only know this value when \mathcal{D}_1 returns (see Definition 1), while we need to program the oracle before \mathcal{D}_1 starts. Popa and Zeldovich suggest in [18] a way to overcome this difficulty: The reduction makes a “bet” on which query to the oracle will be for the keyword that will end up being w^* . It can also bet that \mathcal{D}_1 will never query the oracle for this keyword. When \mathcal{D}_1 returns, the reduction can check whether its bet was correct or not. If it was, the reduction can continue, otherwise it halts and gives a random answer to the DDH problem. If the bet was that the keyword is not queried and the bet was correct, this means that the reduction can program the RO using the value of w^* present in

the history returned by \mathcal{D}_1 . Because \mathcal{D}_1 runs in polynomial time, there are only a polynomial number of possible bets, thus a non-negligible advantage of the distinguisher still results in a non-negligible advantage of the reduction.

8 Performance Analysis

Intuitively, what makes the protocol scalable is that the workload of a writer is linear with the number of keywords she uploads, the workload of a reader is linear with the number of queries it sends, the workload of the server is a long-term task which does not affect search time, and the workload of the proxy is no greater than the one of the server in MKSE [18].

To give a more precise performance evaluation, we consider a system with A writers owning B records each, each record containing N keywords, and C readers each having access to D records. We assume that all readers have the same time period.

In our protocol, Each writer must perform $B \times N$ exponentiations and hashing in \mathbb{G} . The server must perform $C \times D \times N$ such exponentiations for each time period, and the proxy must perform D exponentiations for each trapdoor it receives. The workload of readers is only a single exponentiation and hashing per query, and response reception requires essentially no resources (the final response is received in plain text).

Note that preparation and transformation are tasks that are “embarrassingly parallel”, meaning that they can be parallelized with no effort. They also have strong data locality, meaning that each elementary task is applied on a small portion of the whole dataset, allowing the use of distributed infrastructures like MapReduce. Also, record preparation, performed by the server, is a predictable amount of work without any burst and with long-term deadlines. Also note that the proxy can discard prepared records at the end of each time period, making its space consumption about the same as the server.

8.1 Comparison with Other MUSE Protocols

RMO15 [21] and the protocol of Hamlin et al. [12] both have a very heavy reader workload: in RMO15, the reader has to receive and decrypt D responses for each query, and for HSWW18 it has to download, process and upload $D \times N$ keywords at the beginning of the protocol. HSWW18 has a sublinear search time, though, while RMO15 has a very heavy server workload.

When comparing DH-AP-MUSE with RMO18 [20], the most major difference is the server workload. In RMO18 after each trapdoor transformation, the proxy must perform a complex and costly privacy-preserving sub-protocol with the server, which purpose is to prevent the proxy from learning the access pattern. The exact cost of this lookup sub-protocol is difficult to assess, but there are no doubts it is much more expensive than the simple local lookup done by the proxy in DH-AP-MUSE.

8.2 Implementation and Performance Measurements

Another advantage of DH-AP-MUSE is its great simplicity, which makes its implementation an easy task. We implemented the algorithms of DH-AP-MUSE in less than 100 lines of C, using the Sodium crypto library¹. We encoded prepared records as bloom filters.

Performance measurements on a Amazon EC2 t2.micro instance² gave the following running times:

- trapdoor generation and keyword encryption: 0.1 ms per keyword
- record preparation (including insertion of prepared keywords in a bloom filter): 60 μ s per keyword
- trapdoor transformation: 60 μ s per transformation

This means that a server hosted on a single t2.micro instance could handle (in terms of computation) the preparation of records for 100 readers each having access to 40,000 records assuming records of 10,000 keywords each and a time period of one month. The same machine should be able as a proxy to transform the trapdoor of a reader searching 40,000 records in under 3 s. Note that we do not measure communication time, only computation. Hence there is no need to run the algorithms on two different machines for these measurements. Using a machine with a faster CPU (t2.micro has a frequency of 2.40 GHz) or with more cores (using multi-threading) should scale capacity accordingly. “scaling out” using several machines should also increase the capacity in a linear fashion due to the embarrassingly parallel nature and high data locality of the task.

9 Improving Previous MUSE Protocols with Techniques from This Protocol

The previous MUSE protocols of Van Rompay et al. [20,21] can benefit from several techniques used in the presented protocol, namely, the replacement of bilinear pairings by “normal” DDH-hard groups and the periodic renewal of the blinding factor. While these techniques would improve the efficiency of these protocols, the presented protocol would still be much more scalable and the comparisons we made in Sect. 8.1 would still be valid.

10 Conclusion

We presented the first MUSE protocol that protects query and record privacy against user-server collusions while scaling well to very large databases. Moreover, techniques used in this protocols can be used to improve the efficiency of existing protocols. Interesting topics for future work include a more thorough study of the security implications of a access pattern leakage in a MUSE context, which would give a great amount of insight on the practical security of this protocol as well as the protocol of [12].

¹ <https://libsodium.org>.

² <https://aws.amazon.com/ec2/instance-types/>.

Acknowledgements. This work was supported by the EU FP7 ERANET program under grant CHIST-ERA-2016 UPRISE-IOT.

A Privacy Against the Server

Algorithm 1. Simulator for the server view

Input: The benign leakage and revealed content

Create all record keys and all blinding factors ;

for each record id d **do**

if W_d is revealed **then**

 Encrypt it using the record key γ_d previously generated;

else

 Set W_d to a set of random bit strings
 using the length of W_d from the benign leakage;

 Encrypt W_d

Output: All encrypted records, all blinding factors and the record keys of revealed records

We show that the output of Algorithm 1 is indistinguishable from the real view of the server using a sequence of hybrid simulators, where each hybrid simulates one more non-revealed encrypted keyword than the previous hybrid. All hybrids have the entire history as input except the last one that only has the benign leakage and revealed content.

We then show that the output of two successive hybrids are indistinguishable using a reduction to the DDH problem in \mathbb{G} . The reduction performs the following embedding of a DDH problem instance g^a, g^b, g^c as described in the beginning of Sect. 7:

$$h(w^*) \leftrightarrow g^a, \quad \gamma^* \leftrightarrow b, \quad \bar{w}^* \leftrightarrow g^c$$

Where w^* is the “pivot keyword” that is simulated in one hybrid but not in the other (for instance this could be “*the third keyword of the second non-revealed record*”). The embedding is correct if the view corresponds to the output of one hybrid in the case where $c = ab$ and the other hybrid in the case where c is random. The only difference between these two outputs is that the pivot encrypted keyword \bar{w}^* is simulated in one hybrid and properly generated in the other. All other values of the hybrid output must be the same whatever c is. As a result we must check that the value \bar{w}^* does not appear anywhere else in the output. This is satisfied thanks to the fact that records are represented as sets in our protocol, that is, they do not have duplicate elements. As a result, any keyword w of the pivot record must be different from w^* and its encrypted keyword will be either generated as $(g^b)^{\mathcal{O}[w]}$ or with random sampling.

As a result distinguishing the output of two successive hybrid is at least as hard as solving the DDH problem in \mathbb{G} , thus the output of Algorithm 1 is indistinguishable from a real view, and this ends the proof.

B Privacy Against the Proxy

Algorithm 2. Simulator for the proxy view

Input: The access pattern, the benign leakage and revealed content

Create all record keys and all blinding factors ;

for each r, l, s **do**

if $q_{r,l,s}$ is revealed **then**

 | Create $t_{r,l,s}$ as normal;

else

 | Create $t_{r,l,s}$ as a random element of \mathbb{G} ;

for each record $id\ d$, each r s.t. $d \in Auth(r)$ and each l **do**

if W_d is revealed **then**

 | Encrypt and transform as normal;

else

 | Initialize $\overline{\overline{W}}_{d,r,l}$ as an empty set;

for each s such that $d \in a_{r,l,s}$ (known from the access pattern) **do**

 | Add $(t_{r,l,s})^{\gamma_d}$ to $\overline{\overline{W}}_{d,r,l}$;

 Add random elements to $\overline{\overline{W}}_{d,r,l}$ until it has the proper size (known from the benign leakage);

Output: All encrypted records, all blinding factors and the record keys of revealed records

This time we show that the output of Algorithm 2 is indistinguishable from a real view of the proxy using two sequences of hybrids: The first sequence will correspond to the simulation of prepared records and the second sequence to the simulation of trapdoors. The first hybrid of the first sequence corresponds to the real world experiment. Then, each hybrid in the first sequence will simulate one more non-revealed prepared keyword than the previous hybrid. The first hybrid of the second sequence is the last hybrid of the first sequence, that is, it simulates all non-revealed prepared records but none of the trapdoors. Finally each hybrid simulator in the second sequence simulates one more trapdoor than the previous simulator. As a result the last hybrid simulator of the second sequence is Algorithm 2.

We start by showing that two successive simulators from the first sequence have indistinguishable outputs. The pivot keyword is characterized by d^*, i^*, r^*, l^* such that the second hybrid simulates prepared keyword $\overline{\overline{W}}_{d^*, r^*, l^*}[i^*]$ but the first one does not. If $\overline{\overline{W}}_{d^*, r^*, l^*}[i^*]$ is matched by a trapdoor, the output distributions of the two simulators are more than indistinguishable, they are identical. Indeed in this case the second simulator will not generate this prepared keyword at random but by transforming the corresponding trapdoor, and the resulting value will be the same as what the first simulator would have obtained, as a consequence of the correctness of the protocol. If $\overline{\overline{W}}_{d^*, r^*, l^*}[i^*]$ is not matched by any trapdoor though, the second simulator will simulate it

through random sampling, and again we show the two outputs are indistinguishable with a reduction to the DDH problem. This time the embedding of the DDH instance g^a, g^b, g^c is as follows:

$$h(W_{d^*}[i^*]) \leftrightarrow g^a, \quad \xi_{r^*, l^*} \leftrightarrow b, \quad \overline{W}_{d^*, r^*, l^*}[i^*] \leftrightarrow g^c$$

Again, the correctness of the embedding requires that the reduction does not have to use the value g^c for anything else than the pivot prepared keyword. The argument for this is the same as for privacy against the server: a keyword does not appear twice in a record, and other (prepared) records will use different record keys (or different blinding factors).

For the second sequence, there are r^*, l^*, s^* such that the second hybrid simulates t_{r^*, l^*, s^*} but the first one does not. The embedding used is then:

$$h(q_{r^*, l^*, s^*}) \leftrightarrow g^a, \quad \xi_{r^*, l^*} \leftrightarrow b, \quad t_{r^*, l^*, s^*} \leftrightarrow g^c$$

And correctness of the embedding comes from that a reader will not send two identical queries in a same time period.

References


1. Asghar, M.R., Russello, G., Crispo, B., Ion, M.: Supporting complex queries and access policies for multi-user encrypted databases. In: CCSW 2013, Proceedings of the 2013 ACM Cloud Computing Security Workshop, Co-located with CCS 2013, Berlin, Germany, 4 November 2013, pp. 77–88 (2013). <https://doi.org/10.1145/2517488.2517492>
2. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 71–85. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79104-1_6
3. Bschi, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. *ACM Comput. Surv.* **47**(2), 1–51 (2014). <https://doi.org/10.1145/2636328>
4. Cash, D., et al.: Dynamic searchable encryption in very large databases: data structures and implementation. In: Proceedings of NDSS, vol. 14 (2014)
5. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* **45**(6), 965–981 (1998). <https://doi.org/10.1145/293347.293350>
6. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, 30 October–3 November 2006, pp. 79–88 (2006). <https://doi.org/10.1145/1180405.1180417>
7. 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
8. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. In: Atluri, V. (ed.) DBSec 2008. LNCS, vol. 5094, pp. 127–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70567-3_10

9. Faber, S., Jarecki, S., Krawczyk, H., Nguyen, Q., Rosu, M., Steiner, M.: Rich queries on encrypted data: beyond exact matches. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 123–145. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24177-7_7
10. Fuller, B., et al.: SoK: cryptographically protected database search. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017, pp. 172–191 (2017). <https://doi.org/10.1109/SP.2017.10>
11. Grubbs, P., McPherson, R., Naveed, M., Ristenpart, T., Shmatikov, V.: Breaking web applications built on top of encrypted data. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016, pp. 1353–1364 (2016). <https://doi.org/10.1145/2976749.2978351>
12. Hamlin, A., Shelat, A., Weiss, M., Wicks, D.: Multi-Key Searchable Encryption, Revisited (2018). <https://eprint.iacr.org/2018/018>. Cryptology ePrint Archive, Report 2018/018
13. Huberman, B.A., Franklin, M.K., Hogg, T.: Enhancing privacy and trust in electronic communities. In: EC, pp. 78–86 (1999). <https://doi.org/10.1145/336992.337012>
14. Hwang, Y.H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, E., Okamoto, T., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73489-5_2
15. Kiayias, A., Oksuz, O., Russell, A., Tang, Q., Wang, B.: Efficient encrypted keyword search for multi-user data sharing. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016, Part I. LNCS, vol. 9878, pp. 173–195. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_9
16. Kiss, Á., Liu, J., Schneider, T., Asokan, N., Pinkas, B.: Private set intersection for unequal set sizes with mobile applications. PoPETs **2017**(4), 177–197 (2017). <https://doi.org/10.1515/popets-2017-0044>
17. Lindell, Y.: How to simulate it - a tutorial on the simulation proof technique. In: Tutorials on the Foundations of Cryptography, pp. 277–346 (2017)
18. Popa, R.A., Zeldovich, N.: Multi-Key Searchable Encryption. IACR Cryptology ePrint Archive 2013, 508 (2013). <http://eprint.iacr.org/2013/508>
19. Popa, R.A., Stark, E., Valdez, S., Helfer, J., Zeldovich, N., Balakrishnan, H.: Building web applications on top of encrypted data using Mylar. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, 2–4 April 2014, pp. 157–172 (2014). <https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/popa>
20. Van Rompay, C., Molva, R., Önen, M.: Secure and scalable multi-user searchable encryption. IACR Cryptology ePrint Archive 2018, 90 (2018). <http://eprint.iacr.org/2018/090>
21. Van Rompay, C., Molva, R., Önen, M.: Multi-user searchable encryption in the cloud. In: Lopez, J., Mitchell, C.J. (eds.) ISC 2015. LNCS, vol. 9290, pp. 299–316. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23318-5_17
22. Van Rompay, C.V., Molva, R., Önen, M.: A leakage-abuse attack against multi-user searchable encryption. PoPETs **2017**(3), 168 (2017). <https://doi.org/10.1515/popets-2017-0034>
23. Tang, Q.: Nothing is for free: security in searching shared and encrypted data. IEEE Trans. Inf. Forensics Secur. **9**(11), 1943–1952 (2014). <https://doi.org/10.1109/TIFS.2014.2359389>

24. Yang, J., Fu, C., Shen, N., Liu, Z., Jia, C., Li, J.: General multi-key searchable encryption. In: 29th IEEE International Conference on Advanced Information Networking and Applications Workshops, AINA 2015 Workshops, Gwangju, South Korea, 24–27 March 2015, pp. 89–95 (2015). <https://doi.org/10.1109/WAINA.2015.18>
25. Yang, J., Liu, Z., Li, J., Jia, C., Cui, B.: Multi-key searchable encryption without random oracle. In: 2014 International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italy, 10–12 September 2014, pp. 79–84 (2014). <https://doi.org/10.1109/INCoS.2014.143>
26. Yang, Y., Lu, H., Weng, J.: Multi-User Private Keyword Search for Cloud Computing, pp. 264–271. IEEE, November 2011. <https://doi.org/10.1109/CloudCom.2011.43>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6133152>



Identity-Based Functional Encryption for Quadratic Functions from Lattices

Kelly Yun^{1,2}(✉) , Xin Wang^{1,2}, and Rui Xue^{1,2}

¹ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
{yuankaili,wangxin9076,xuerui}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
yunkaili15@mailsucas.ac.cn

Abstract. We present a functional encryption scheme for quadratic functions from lattices under identity-based access control. This represents a practical relevant class of functions beyond multivariate quadratic polynomials and may adapt to many scenarios. Recently, Baltico et al. [10] in Crypto 2017 presented two constructions from pairings which enable efficient decryption only when $\mathbf{x}^\top \mathbf{F}\mathbf{y}$ is contained in a sufficiently small interval to finally compute a discrete logarithm, and one construction is proved selectively secure under standard assumptions and the other adaptively secure in the generic group model (GGM). Our construction is no pairings and no small interval restriction. We formalize the definition of identity-based functional encryption and its indistinguishability security and achieve adaptive security against unbounded collusions under standard assumptions in the random oracle model.

Keywords: Functional encryption · Quadratic functions
Identity-based access control · Learning with errors

1 Introduction

Functional Encryption (FE) is an ambitious generalization of public-key encryption which overcomes the all-or-nothing, user-based access to encrypted data and enables fine grained, role-based access to the data. Namely, functional encryption comes equipped with a key generation algorithm that utilizes a master secret key to generate decryption keys sk_F corresponding to functions F , the key holders only learn $F(x)$ from a ciphertext $\text{Enc}(x)$ and no more information about x is revealed. This is well suited for cloud computing platforms and remote untrustworthy servers to store sensitive private data and allow users to request the result of the function F computing on the underlying data.

Supported by National Natural Science Foundation of China grants No. 61772514, No. 61472414, No. 61602061, and National Key R&D Program of China (2017YFB1400700).

The definition of functional encryption was first formalized by [17, 39] which gave indistinguishability (IND-based) and simulation (SIM-based) security model, and identity-based encryption (IBE) [2, 14, 15, 20, 21, 28, 44], attribute-based encryption (ABE) [11, 16, 31, 33, 42], predicate encryption (PE) [3, 32, 35, 36, 38] and other concrete functionalities [18, 45] in a general framework could all be regarded as specific function classes of functional encryption.

Though Garg et al. [9, 24, 26, 46] constructed functional encryption for general function, their work used brilliant but ill-understood indistinguishability obfuscation(iO) or multi-linear maps machinery that existing constructions [23, 27] were found to be insecure [22, 34], so there is no provably secure instantiation by now. Some work [4, 5, 29, 30] considered general function under bounded collisions from simple primitives or well-understood assumptions. Conversely, there is also some fascinating work that constructs iO from FE schemes [8, 12, 13, 25].

Recently Abdalla et al. [1] built FE for linear functions surprisingly efficiently from standard assumptions like the Decision Diffie-Hellman (DDH) and Learning-with-Errors (LWE) assumptions. Later, Agrawal et al. [4] promoted their schemes from selective security to adaptive security and gave an additional construction from Decision Composite Residuosity (DCR) assumption. Beyond linear functions, Baltico et al. [10] constructed two FE schemes for quadratic functions from pairings which enable efficient decryption only when $\mathbf{x}^\top \mathbf{F}\mathbf{y}$ is contained in a sufficiently small interval to finally compute a discrete logarithm, and one construction is proved selectively secure under standard assumptions and the other adaptively secure in the generic group model (GGM). This motivates the following question:

Can we build adaptively secure FE scheme for quadratic functions without pairings and the small interval restriction?

1.1 Our Results

We answer the above question affirmatively. We propose the first adaptively secure FE scheme for quadratic functions from lattices against unbounded collisions, but under identity-based access control. On the one hand, identity-based functional encryption can be regarded as functional encryption under identity-based control. On the other hand, we can think it as an extension of identity-based encryption what only allow certain identity owner to decrypt partial information or function values. We notice that Sans and Pointcheval [43] consider the identity-based access control as an additional property to expand the possible applications of their unbounded length inner product FE schemes. Here we formalize the identity-based functional encryption definition and indistinguishability security (IND-IBFE-CPA) based on [17, 39]. Namely, we additionally add identity id to the input to KeyGen and Encrypt algorithms, and we need the identity-based access control property to prove adaptive security of our scheme under random oracle model. So constructing adaptively secure FE scheme for quadratic functions under standard model is still an open problem.

In recent years, lattice-based cryptography has been shown to be extremely versatile, leading to a large number of attractive theoretical applications. Lattice problems provide some significant advantages not found in other types of cryptography, based on worst-case assumption, resistant to cryptanalysis by quantum algorithms and lattice cryptography operations are very simple (almost matrix operations), especially to our scheme, without the small interval restriction to finally compute a discrete logarithm. We employ preimage sampling techniques with trapdoor [2, 20, 28] to generate secret keys unlike linear functions schemes from LWE assumption [4] which do not use preimage sampling algorithms with trapdoor.

Overview of Techniques. We utilize $\mathbf{x}^\top \mathbf{F} \mathbf{y}$ form to represent general quadratic functions the same as [10]. Without loss of generality, messages are expressed as pairs of vectors $(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^l \times \mathbb{Z}^l$ of the same length l , and it is easy to see that the case in which one is longer than the other can be captured by padding the shorter one with zero entries, and secret keys are associated with $(l \times l)$ matrices \mathbf{F} , and decryption allows to compute $\mathbf{x}^\top \mathbf{F} \mathbf{y} = \sum_{i,j} f_{i,j} x_i y_j$.

We use dual Regev’s cryptosystem for multi-bit messages [4, 28], which enjoys ciphertexts have size $O(l)$. Namely, we set $Ct_{(\mathbf{x}, \mathbf{y})} = (\mathbf{c}_{01}, \mathbf{c}_{02}, \mathbf{c}_{11}, \mathbf{c}_{12})$:

$$\begin{aligned} \mathbf{c}_{01} &= \mathbf{A}^\top \mathbf{s}_1 + \mathbf{r}'_1, & \mathbf{c}_{11} &= \mathbf{B}^\top \mathbf{s}_2 + \mathbf{r}'_2 \\ \mathbf{c}_{02} &= \mathbf{U}_1^\top \mathbf{s}_1 + \mathbf{r}_1 + \mathbf{x}, & \mathbf{c}_{12} &= \mathbf{U}_2^\top \mathbf{s}_2 + \mathbf{r}_2 + \mathbf{y} \end{aligned}$$

where $\mathbf{s}_1, \mathbf{s}_2$ are chosen at random, $\mathbf{U}_1, \mathbf{U}_2$ are $\mathbb{Z}_q^{n \times l}$ matrices, and $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ are contained in the public key and $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}'_1, \mathbf{r}'_2$ are noises. We have a relation that $\mathbf{A} \mathbf{E}_1 = \mathbf{U}_1, \mathbf{B} \mathbf{E}_2 = \mathbf{U}_2$ where $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{Z}^{m \times l}$ are sampled uniformly from discrete Gaussian probability distributions. We observe that

$$\mathbf{x}^\top \mathbf{F} \mathbf{y} \approx \mathbf{c}_{02}^\top \mathbf{F} \mathbf{c}_{12} - \mathbf{c}_{01}^\top \mathbf{E}_1 \mathbf{F} \mathbf{c}_{12} - \mathbf{c}_{02}^\top \mathbf{F} \mathbf{E}_2^\top \mathbf{c}_{11} + \mathbf{c}_{01}^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top \mathbf{c}_{11}.$$

Thus we set $sk_{\mathbf{F}} = (\mathbf{F}, \mathbf{E}_1 \mathbf{F}, \mathbf{F} \mathbf{E}_2^\top, \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top)$. Then, there is a problem that if one user asks for an \mathbf{F} which is invertible (especially unitary matrix), he will get a pair of $\mathbf{E}_1, \mathbf{E}_2$ from $sk_{\mathbf{F}}$ and he can compute arbitrary $sk_{\mathbf{F}' } = (\mathbf{F}', \mathbf{E}_1 \mathbf{F}', \mathbf{F}' \mathbf{E}_2^\top, \mathbf{E}_1 \mathbf{F}' \mathbf{E}_2^\top)$ corresponding to \mathbf{F}' and decrypt arbitrary $\mathbf{x}^\top \mathbf{F}' \mathbf{y}$ owing to the relation that $\mathbf{A} \mathbf{E}_1 = \mathbf{U}_1, \mathbf{B} \mathbf{E}_2 = \mathbf{U}_2$ always holds.

To circumvent this problem, we employ extension preimage sampling techniques with trapdoor [2, 20]. We additionally use a public matrix $\mathbf{R} \in \mathbb{Z}_q^{n \times l}$ to randomize \mathbf{F} and make the multiplication into the extension preimage sampling algorithms. So in the KeyGen algorithm, the relation becomes $(\mathbf{A} | \mathbf{R} \mathbf{F}) \mathbf{E}_1 = \mathbf{U}_1$ and $(\mathbf{B} | \mathbf{R} \mathbf{F}) \mathbf{E}_2 = \mathbf{U}_2$ where $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{Z}^{(m+l) \times l}$ can be sampled uniformly by extension sampling algorithms with trapdoors $T_{\mathbf{A}}, T_{\mathbf{B}}$.

In order to prove the security, we need to regard $\mathbf{U}_1, \mathbf{U}_2$ as random oracle $\mathbf{U}_1(id), \mathbf{U}_2(id): \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times l}$ to answer secret keys queries for arbitrary identity id except the challenge id^* and arbitrary \mathbf{F} . For different \mathbf{F} , there are distinct $\mathbf{E}_1, \mathbf{E}_2$ which have enough entropy to resist collusion attacks.

1.2 Related Work

Agrawal and Rosen [5] considered bounded collusions schemes from LWE assumption, and they also achieved bounded collusions functional encryption for quadratic functions.

Sans and Pointcheval [43] consider the identity-based access control as an additional property to expand the possible applications of their unbounded length inner product FE schemes. They do not formalize the definition of identity-based functional encryption and its security model, and they only achieve selective security from pairings under random oracle model for their unbounded length inner product FE schemes.

1.3 Organization

In Sect. 2, we introduce some necessary notations and some lemmas, algorithms and assumptions from lattice-based cryptography. We formalize the definitions of identity-based functional encryption (IBFE) and its security model in Sect. 3. Section 4 presents our IBFE scheme for quadratic functions. In Sect. 5, we analyze the security of our scheme. We conclude and propose some open problems in Sect. 6.

2 Preliminary

Notations. We denote vectors by lower-case bold letters (e.g. \mathbf{x}) and are always in column form (respectively, \mathbf{x}^\top is a row vector). Matrices are denoted by upper-case bold letters (e.g. \mathbf{A}) and treat them with their ordered column vector sets $[\mathbf{a}_1, \mathbf{a}_2, \dots]$. We let $\mathbf{M}_1|\mathbf{M}_2$ denote the (ordered) concatenation of the column vector sets of \mathbf{M}_1 and \mathbf{M}_2 , $\mathbf{M}_1\|\mathbf{M}_2$ denote the (ordered) concatenation of the row vector sets of \mathbf{M}_1 and \mathbf{M}_2 , and vectors are similar. For a vector \mathbf{x} , we let $\|\mathbf{x}\|$ denote its l_2 norm and $\|\mathbf{x}\|_\infty$ denote its infinity norm. Similarly, for matrices $\|\cdot\|$ and $\|\cdot\|_\infty$ denote their l_2 and infinity norms respectively.

2.1 Functional Encryption

We recall the syntax of functional encryption, as defined by [17], and their indistinguishability based security definition.

Definition 1 (Functionality). *A functionality F defined over $(\mathcal{K}, \mathcal{M})$ is a function $F : \mathcal{K} \times \mathcal{M} \rightarrow \Sigma \cup \{\perp\}$ where \mathcal{K} is a key space, \mathcal{M} is a message space and Σ is an output space which does not contain the special symbol \perp .*

Definition 2 (Functional Encryption). *A functional encryption scheme FE for a functionality F is a tuple of four algorithms $FE = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ that work as follows:*

Setup (1^λ) takes as input a security parameter 1^λ and outputs a master key pair (mpk, msk) .

KeyGen(msk, K) takes as input the master secret key and a key (i.e. a function) $K \in \mathcal{K}$, and outputs a secret key sk_K .

Encrypt(mpk, M) takes as input the master public key mpk and a message $M \in \mathcal{M}$, and outputs a ciphertext C .

Decrypt(mpk, sk_K, C) takes as input a secret key sk_K and a ciphertext C , and returns an output $v \in \Sigma \cup \{\perp\}$.

For correctness, it is required that for all $(mpk, msk) \leftarrow Setup(1^\lambda)$, all keys $K \in \mathcal{K}$ and all messages $M \in \mathcal{M}$, if $sk_K \leftarrow KeyGen(msk, K)$ and $C \leftarrow Encrypt(mpk, M)$, then it holds with overwhelming probability that $Decrypt(sk_K, C) = F(K, M)$ whenever $F(K, M) \neq \perp$.

Indistinguishability-Based Security. For a functional encryption scheme FE for a functionality F over $(\mathcal{K}, \mathcal{M})$, security against chosen-plaintext attacks (IND-FE-CPA, for short) if no PPT adversary has non-negligible advantage in the following game:

1. The challenger runs $(mpk, msk) \leftarrow Setup(1^\lambda)$ and gives mpk to \mathcal{A} .
2. The adversary \mathcal{A} adaptively makes secret key queries. At each query, \mathcal{A} chooses a key $K \in \mathcal{K}$ and obtains $sk_K \leftarrow KeyGen(msk, K)$.
3. Adversary \mathcal{A} chooses a pair of distinct messages $M_0, M_1 \in \mathcal{M}$ such that $F(K, M_0) = F(K, M_1)$ holds for all Keys K queried in the previous phase. The challenger computes $C^* \leftarrow Encrypt(mpk, M_\beta)$ and return C^* to \mathcal{A} .
4. Adversary \mathcal{A} makes further secret key queries for arbitrary keys $K \in \mathcal{K}$, but under the requirement that $F(K, M_0) = F(K, M_1)$.
5. Adversary \mathcal{A} eventually outputs a bit $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$.

The adversary's advantage is defined to be $Adv_{\mathcal{A}}(\lambda) := |Pr[\beta' = \beta] - 1/2|$.

2.2 Lattices

An m -dimensional lattice \mathcal{L} is a discrete additive subgroup of \mathbb{R}^m . Given positive integers n, m, q and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \pmod q\}$ and $\Lambda_q(\mathbf{A})$ denote the lattice $\{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{A}^\top \cdot \mathbf{s} \pmod q \text{ for some } \mathbf{s} \in \mathbb{Z}^n\}$. For $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \pmod q\}$. Note that if $\mathbf{t} \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$ then $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{t}$ and hence $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ is a shift of $\Lambda_q^\perp(\mathbf{A})$.

Discrete Gaussians. Let σ be any positive real number, $\mathbf{c} \in \mathbb{R}^m$. The Gaussian distribution $\mathcal{D}_{\sigma, \mathbf{c}}$ centered at \mathbf{c} with parameter σ is defined by the probability distribution function $\rho_{\sigma, \mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / \sigma^2)$. For any set $\mathcal{L} \subset \mathbb{R}^m$, define $\rho_{\sigma, \mathbf{c}}(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_{\sigma, \mathbf{c}}(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L}, \sigma, \mathbf{c}}$ over \mathcal{L} centered at \mathbf{c} with parameter σ is defined by the probability distribution function $\rho_{\mathcal{L}, \sigma, \mathbf{c}}(\mathbf{x}) = \rho_{\sigma, \mathbf{c}}(\mathbf{x}) / \rho_{\sigma, \mathbf{c}}(\mathcal{L})$ for all $\mathbf{x} \in \mathcal{L}$.

The following lemma states that the total Gaussian measure on any translate of the lattice is essentially the same.

Lemma 1 [28,37]. *For any m -dimensional lattice Λ , $\sigma \geq \omega(\sqrt{\log m})$, $\mathbf{c} \in \mathbb{R}^m$, $\epsilon \in (0, 1)$, we have*

$$\rho_{\sigma, \mathbf{c}}(\Lambda) \in \left[\frac{1 - \epsilon}{1 + \epsilon}, 1 \right] \cdot \rho_{\sigma}(\Lambda)$$

A sample from a discrete Gaussian with parameter σ is at most $\sqrt{m}\sigma$ away from its center \mathbf{c} with overwhelming probability.

Lemma 2 [28,37]. *For any m -dimensional lattice Λ , $m > n$, center \mathbf{c} , $\sigma \geq \omega(\sqrt{\log m})$, we have*

$$\Pr[\|\mathbf{x} - \mathbf{c}\| > \sqrt{m}\sigma | \mathbf{x} \leftarrow \mathcal{D}_{\Lambda, \sigma, \mathbf{c}}] \leq \text{negl}(n).$$

There is an upper bound on the probability of a discrete Gaussian, equivalently, it is a lower bound on the min-entropy of the distribution.

Lemma 3 [28]. *For any m -dimensional lattice Λ , $\sigma \geq \omega(\sqrt{\log m})$, center \mathbf{c} , positive $\epsilon > 0$, and $\mathbf{x} \in \Lambda$, we have*

$$\mathcal{D}_{\Lambda, \sigma, \mathbf{c}} \leq \frac{1 + \epsilon}{1 - \epsilon} \cdot 2^{-m}.$$

In particular, for $\epsilon < \frac{1}{3}$, the min-entropy of $\mathcal{D}_{\Lambda, \sigma, \mathbf{c}}$ is at least $m-1$.

Ajtai et al. [6,7] showed how to sample an essentially uniform \mathbf{A} , along with a relatively short basis $T_{\mathbf{A}}$.

Lemma 4. *Let n, q, m be positive integers with $q > 2$ and $m \geq 5n \log q$. There is a probabilistic polynomial-time(PPT) algorithm **TrapGen** that outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, T_{\mathbf{A}} \in \mathbb{Z}^{m \times m})$ where the distribution of \mathbf{A} is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$ and $\|T_{\mathbf{A}}\| \leq m \cdot \omega(\sqrt{\log m})$.*

Gentry et al. [28] showed that if $\text{ISIS}_{q,m,2\sigma\sqrt{m}}$ is hard, $f_{\mathbf{A}} : \mathbb{Z}_q^m \rightarrow \mathbb{Z}_q^n$ with $f_{\mathbf{A}}(\mathbf{e}) = \mathbf{A}\mathbf{e} \bmod q$ is one-way function, even collision resistant function where $\|\mathbf{e}\| \leq \sqrt{m}\sigma$. Note that for $m > 2n \log q, \sigma > \omega(\sqrt{\log m})$, $f_{\mathbf{A}}$ is surjective for almost all \mathbf{A} , and the distribution of $\mathbf{u} = \mathbf{A}\mathbf{e} \bmod q$ is statistically close to uniform over \mathbb{Z}_q^n . Furthermore, fix $\mathbf{u} \in \mathbb{Z}_q^n$, a short basis for $\Lambda^{\perp}(\mathbf{A})$ can be used to efficiently sample short vectors from $f_{\mathbf{A}}^{-1}(\mathbf{u})$ without revealing any information about the short basis $T_{\mathbf{A}}$.

Lemma 5. *Let n, q, m be positive integers with $q \geq 2$ and $m \geq 2n \log q$. There is a PPT algorithm **SamplePre** that on input of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a basis $T_{\mathbf{A}}$ for $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and an integer $\sigma \geq \|\widetilde{T_{\mathbf{A}}}\| \cdot \omega(\sqrt{\log m})$, the distribution of the output of $\mathbf{e} \leftarrow \text{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \sigma)$ is with negligible statistical distance of $\mathcal{D}_{\Lambda_q^n(\mathbf{A}), \sigma}$.*

2.3 Algorithm SampleR

The preimage sampling algorithm can be easily generalized to generate preimages of matrices instead of vectors by independently running **SamplePre** algorithm on each column of the matrix, so we overload the notation by directly giving matrices $\mathbf{U} \in \mathbb{Z}_q^{n \times l}$ as inputs to the **SamplePre** algorithm. The following algorithm is reminiscent of the extension preimage sampling algorithm of [2,20].

Algorithm SampleR($\mathbf{A}, \mathbf{M}, T_{\mathbf{A}}, \mathbf{U}, \sigma$)

Inputs:

- a rank n matrix \mathbf{A} in $\mathbb{Z}_q^{n \times m}$ and a matrix \mathbf{M} in $\mathbb{Z}_q^{n \times l}$,
- a short basis $T_{\mathbf{A}}$ of $\Lambda_q^\perp(\mathbf{A})$ and a matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times l}$,
- a gaussian parameter $\sigma > \|\widetilde{T}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log(m+l)})$.

Running:

1. sample a random matrix $\mathbf{E}_{10} \in \mathbb{Z}^{l \times l}$ distributed statistically close to $\mathcal{D}_{\mathbb{Z}^{l \times l}, \sigma}$,
2. compute $\mathbf{Y} = \mathbf{U} - \mathbf{M} \cdot \mathbf{E}_{10} \in \mathbb{Z}_q^{n \times l}$, and run $\mathbf{E}_{11} \leftarrow \text{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{Y}, \sigma)$,
3. output $\mathbf{E}_1 = (\mathbf{E}_{11} \| \mathbf{E}_{10}) \in \mathbb{Z}^{(m+l) \times l}$

Outputs:

Let $\overline{\mathbf{A}} = (\mathbf{A} | \mathbf{M})$. The algorithm outputs a matrix $\mathbf{E}_1 \in \mathbb{Z}^{(m+l) \times l}$ sampled from a distribution statistically close to $D_{\Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}}), \sigma}$. In particular, $\mathbf{E}_1 \subset \Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}})$.

Theorem 1. *Let n, q, m, l be positive integers with $q \geq 2$ and $m \geq 2n \log q$. There is a PPT algorithm **SampleR** that on input of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a basis $T_{\mathbf{A}}$ for $\Lambda_q^\perp(\mathbf{A})$, matrices $\mathbf{M}, \mathbf{U} \in \mathbb{Z}_q^{n \times l}$, and an integer $\sigma \geq \|\widetilde{T}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log(m+l)})$ outputs $\mathbf{E}_1 \leftarrow \text{SampleR}(\mathbf{A}, \mathbf{M}, T_{\mathbf{A}}, \mathbf{U}, \sigma)$ which is with negligible statistical distance of the distribution $\mathcal{D}_{\Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}}), \sigma}$ where $\overline{\mathbf{A}} = (\mathbf{A} | \mathbf{M})$.*

Proof. As the process of the algorithm, we have

$$\begin{aligned} Pr[\mathbf{E}_1] &= Pr[\mathbf{E}_{10}] \cdot Pr[\mathbf{E}_{11} : \mathbf{E}_{10}] \\ &= \rho_\sigma(\mathbf{E}_{10}) \cdot \frac{\rho_\sigma(\mathbf{E}_{11})}{\rho_{\mathcal{D}^{l \times l}, \sigma} \cdot \rho_\sigma(\{\mathbf{E}_{11} : \mathbf{A}\mathbf{E}_{11} = \mathbf{U} - \mathbf{M}\mathbf{E}_{10}\})}. \end{aligned}$$

For a \mathbf{t} satisfying $\mathbf{A}\mathbf{t} = \mathbf{U} - \mathbf{M}\mathbf{E}_{10}$, we have

$$\{\mathbf{E}_{11} : \mathbf{A}\mathbf{E}_{11} = \mathbf{U} - \mathbf{M}\mathbf{E}_{10}\} = \mathbf{t} + \Lambda_q^\perp(\mathbf{A})$$

Then we have

$$\rho_\sigma(\mathbf{t} + \Lambda_q^\perp(\mathbf{A})) \in \left[\frac{1 - \epsilon}{1 + \epsilon}, 1 \right] \cdot \rho_\sigma(\Lambda_q^\perp(\mathbf{A}))$$

for some negligible function ϵ . Besides, we have

$$\begin{aligned} \rho_\sigma(\Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}})) &= \sum \rho_\sigma(\mathbf{E}_1) = \sum_{\mathbf{A}\mathbf{E}_{11}=\mathbf{U}-\mathbf{M}\mathbf{E}_{10}} \rho_\sigma(\mathbf{E}_{11})\rho_\sigma(\mathbf{E}_{10}) \\ &= \sum_{\mathbf{E}_{10} \leftarrow \mathcal{D}^{l \times l}} \rho_\sigma(\mathbf{E}_{10}) \sum_{\mathbf{E}_{11} \leftarrow \mathcal{D}^{m \times l}, \mathbf{A}\mathbf{E}_{11}=\mathbf{U}-\mathbf{M}\mathbf{E}_{10}} \rho_\sigma(\mathbf{E}_{11}) \\ &= \left(\sum_{\mathbf{E}_{10} \leftarrow \mathcal{D}^{l \times l}} \rho_\sigma(\mathbf{E}_{10}) \right) \rho_\sigma(\mathbf{t} + \Lambda_q^\perp(\mathbf{A})) \\ &\in \left(\sum_{\mathbf{E}_{10} \leftarrow \mathcal{D}^{l \times l}} \rho_\sigma(\mathbf{E}_{10}) \right) \cdot \left[\frac{1 - \epsilon'}{1 + \epsilon'}, 1 \right] \cdot \rho_\sigma(\Lambda_q^\perp(\mathbf{A})) \\ &\in \left[\frac{1 - \epsilon'}{1 + \epsilon'}, 1 \right] \cdot \rho_{\mathcal{D}^{l \times l}, \sigma} \cdot \rho_\sigma(\Lambda_q^\perp(\mathbf{A})) \end{aligned}$$

for some negligible function ϵ' . Thus,

$$\begin{aligned} \rho_\sigma(\Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}})) &\in \left[\frac{1 - \epsilon'}{1 + \epsilon'}, 1 \right] \cdot \rho_{\mathcal{D}^{l \times l}, \sigma} \cdot \rho_\sigma(\Lambda_q^\perp(\mathbf{A})) \\ Pr[\mathbf{E}_1] &\in \rho_\sigma(\mathbf{E}_{10}) \cdot \frac{\rho_\sigma(\mathbf{E}_{11})}{\rho_{\mathcal{D}^{l \times l}, \sigma} \cdot \left[\frac{1 - \epsilon}{1 + \epsilon}, 1 \right] \cdot \rho_\sigma(\Lambda_q^\perp(\mathbf{A}))} \\ &\in \left[\frac{1 - \epsilon'}{1 + \epsilon'}, \frac{1 + \epsilon}{1 - \epsilon} \right] \cdot \frac{\rho_\sigma(\mathbf{E}_{10}) \cdot \rho_\sigma(\mathbf{E}_{11})}{\rho_\sigma(\Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}}))} \end{aligned}$$

The distribution of \mathbf{E}_1 is with negligible statistical distance of the distribution $\mathcal{D}_{\Lambda_q^{\mathbf{U}}(\overline{\mathbf{A}}), \sigma}$. This ends the proof. \square

2.4 Learning with Errors

We review the learning with errors (LWE) problem for the most part from [41].

We first introduce the error distribution χ_α , that is, the normal (Gaussian) distribution on \mathbb{T} with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ having density function $\frac{1}{\alpha} \exp(-\pi x^2/\alpha^2)$. Its discretized normal distribution on \mathbb{Z}_q denoted to be the distribution of $\lfloor q \cdot X \rfloor \bmod q$, where X is a random variable with distribution χ_α and $\lfloor x \rfloor$ is the closest integer to $x \in \mathbb{R}$.

The following lemma about the distribution χ_α will be needed to show that decryption works correctly.

Lemma 6 [2]. *Let $\mathbf{x} \in \mathbb{Z}^m$ and $\mathbf{r} \leftarrow \chi_\alpha^m$, then the quantity $\|\mathbf{x}^\top \mathbf{r}\|$ treated as an integer in $[0, q - 1]$ satisfies*

$$\|\mathbf{x}^\top \mathbf{r}\| \leq \|\mathbf{x}\| q \alpha \omega(\sqrt{\log m}) + \|\mathbf{x}\| \sqrt{m}/2$$

with all but negligible probability in m .

For an integer $q \geq 2$ and some probability distribution χ over q , $\mathbf{s} \in \mathbb{Z}_q^n$, define $A_{\mathbf{s},\chi}$ to be the distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ of the variable $(\mathbf{a}, \mathbf{a}^\top \mathbf{s} + x)$ induced by choosing \mathbf{a} uniformly at random from \mathbb{Z}_q^n , $x \leftarrow \chi$.

Learning with Errors (Decision Version). For an integer $q = q(n)$ and a distribution χ on \mathbb{Z}_q , $\text{LWE}_{q,\chi}$ is to distinguish between the distribution $A_{\mathbf{s},\chi}$ for some uniform secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ (via oracle access to the distribution).

Regev [41] demonstrated that for certain moduli q and Gaussian error distribution χ_α , $\text{LWE}_{q,\chi_\alpha}$ is as hard as solving several standard worst-case lattice problems using a quantum algorithm.

Theorem 2. *Let $\alpha(n) \in (0, 1)$ and $q(n)$ be a prime such that $\alpha \cdot q \geq 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves $\text{LWE}_{q,\chi_\alpha}$, then there exists an efficient quantum algorithm for approximating SIVP and GapSVP to within $O(n/\alpha)$ factors in the worst case.*

Peikert et al. [19,40] showed that there is a classical reduction from GapSVP to the LWE problem.

3 Definitions of Identity-Based Functional Encryption

Definition 3 (Identity-Based Functional Encryption). *An identity-based functional encryption (IBFE) scheme for a functionality F is a tuple of four algorithms $\text{IBFE} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ that work as follows:*

Setup (1^λ) takes as input a security parameter 1^λ and outputs a master key pair (mpk, msk) .

KeyGen $(\text{msk}, \text{id}, K)$ takes as input the master secret key, an $\text{id} \in \mathcal{ID}$ and a key (a.k.a. a function) $K \in \mathcal{K}$, and outputs a secret key sk_K .

Encrypt $(\text{mpk}, \text{id}, M)$ takes as input the master public key mpk , an $\text{id} \in \mathcal{ID}$ and a message $M \in \mathcal{M}$, and outputs a ciphertext C .

Decrypt (mpk, sk_K, C) takes as input a secret key sk_K and a ciphertext C , and returns an output $v \in \Sigma \cup \{\perp\}$.

For correctness, it is required that for all $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$, all $\text{id} \in \mathcal{ID}$, all keys $K \in \mathcal{K}$ and all messages $M \in \mathcal{M}$, if $sk_K \leftarrow \text{KeyGen}(\text{msk}, \text{id}, K)$ and $C \leftarrow \text{Encrypt}(\text{mpk}, \text{id}, M)$, then it holds with overwhelming probability that $\text{Decrypt}(sk_K, C) = F(K, M)$ whenever $F(K, M) \neq \perp$.

Definition 4 (IND-IBFE-CPA Security). *For an identity-based functional encryption scheme for a functionality F over $(\mathcal{K}, \mathcal{M})$, security against chosen-plaintext attacks (IND-IBFE-CPA, for short) if no PPT adversary has non-negligible advantage in the following game:*

1. The challenger runs $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$ and gives mpk to \mathcal{A} .
2. The adversary \mathcal{A} adaptively makes secret key queries. At each query, \mathcal{A} chooses an identity $\text{id} \in \mathcal{ID}$ and a key $K \in \mathcal{K}$ and obtains $sk_K \leftarrow \text{KeyGen}(\text{msk}, \text{id}, K)$.

3. Adversary \mathcal{A} chooses an identity $id^* \in \mathcal{ID}$ and a pair of distinct messages $M_0, M_1 \in \mathcal{M}$ such that $F(K, M_0) = F(K, M_1)$ holds for all Keys K queried in the previous phase. The challenger computes $C^* \leftarrow \text{Encrypt}(mpk, id^*, M_\beta)$ and return C^* to \mathcal{A} .
4. Adversary \mathcal{A} makes further secret key queries for arbitrary identities $id \in \mathcal{ID}$ and keys $K \in \mathcal{K}$, but under the restriction that $id \neq id^*$ and $F(K, M_0) = F(K, M_1)$.
5. Adversary \mathcal{A} eventually outputs a bit $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$.

The adversary's advantage is defined to be $\text{Adv}_{\mathcal{A}}(\lambda) := |\text{Pr}[\beta' = \beta] - 1/2|$.

4 Construction of Identity-Based Functional Encryption for Quadratic Functions

Let $\mathbf{U}_1, \mathbf{U}_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times l}$ be hash functions, which can be simply seen as l maps to map id to uniform syndromes in \mathbb{Z}_q^n at random and independently. For ease of exposition, we overload them as matrices.

Setup($1^n, 1^l, P, V$): Utilize **TrapGen** to generate $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $T_{\mathbf{A}} \subset A_q^\perp(\mathbf{A})$, $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $T_{\mathbf{B}} \subset A_q^\perp(\mathbf{B})$, where \mathbf{A}, \mathbf{B} are statistically close to uniform, and $T_{\mathbf{A}}, T_{\mathbf{B}} \in \mathbb{Z}^{m \times m}$. Choose $\mathbf{R} \in \mathbb{Z}_q^{n \times l}$ uniformly at random. Set $\max(\|\mathbf{x}\|_\infty, \|\mathbf{y}\|_\infty) = P$ and $\|\mathbf{F}\|_\infty = V, K = l^2 P^2 V$. Define $\text{mpk} := \{\mathbf{A}, \mathbf{B}, \mathbf{R}, K, P, V\}$ and $\text{msk} := \{T_{\mathbf{A}}, T_{\mathbf{B}}\}$.

Keygen($\text{msk}, id, \mathbf{F}$): Given \mathbf{F} , running **SampleR**($\mathbf{A}, \mathbf{R}\mathbf{F}, T_{\mathbf{A}}, \mathbf{U}_1(id), \sigma$), **SampleR**($\mathbf{B}, \mathbf{R}\mathbf{F}, T_{\mathbf{B}}, \mathbf{U}_2(id), \sigma$) to sample \mathbf{E}_1 and $\mathbf{E}_2 \in \mathbb{Z}^{(m+l) \times l}$ such that $(\mathbf{A}|\mathbf{R}\mathbf{F})\mathbf{E}_1 = \mathbf{U}_1(id)$ and $(\mathbf{B}|\mathbf{R}\mathbf{F})\mathbf{E}_2 = \mathbf{U}_2(id)$. Compute and return the secret key $sk_{\mathbf{F}} = (\mathbf{F}, \mathbf{E}_1\mathbf{F}, \mathbf{F}\mathbf{E}_2^\top, \mathbf{E}_1\mathbf{F}\mathbf{E}_2^\top)$.

Encrypt($\text{mpk}, id, (\mathbf{x}, \mathbf{y})$): Sample $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathbb{Z}_q^n$ uniformly at random, $\mathbf{r}'_1, \mathbf{r}'_2 \leftarrow \chi_{q,\alpha}^m$ and $\mathbf{r}''_1, \mathbf{r}''_2, \mathbf{r}_1, \mathbf{r}_2 \leftarrow \chi_{q,\alpha}^l$ and compute

$$\begin{aligned}
 \mathbf{c}_{01} &= \mathbf{A}^\top \mathbf{s}_1 + \mathbf{r}'_1, & \mathbf{c}_{11} &= \mathbf{B}^\top \mathbf{s}_2 + \mathbf{r}'_2 \\
 \mathbf{c}_{02} &= \mathbf{U}_1(id)^\top \mathbf{s}_1 + \mathbf{r}_1 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x}, & \mathbf{c}_{12} &= \mathbf{U}_2(id)^\top \mathbf{s}_2 + \mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} \\
 \mathbf{c}_{03} &= \mathbf{R}^\top \mathbf{s}_1 + \mathbf{r}''_1, & \mathbf{c}_{13} &= \mathbf{R}^\top \mathbf{s}_2 + \mathbf{r}''_2.
 \end{aligned}$$

Then, return $\mathbf{C} := (\mathbf{c}_{01}, \mathbf{c}_{02}, \mathbf{c}_{03}, \mathbf{c}_{11}, \mathbf{c}_{12}, \mathbf{c}_{13})$.

Decrypt($\text{mpk}, sk_{\mathbf{F}}, \mathbf{C}$): Compute $\mu' = \mathbf{c}_{02}^\top \mathbf{F} \mathbf{c}_{12} - (\mathbf{c}_{01} \|\mathbf{F}^\top \mathbf{c}_{03})^\top \mathbf{E}_1 \mathbf{F} \mathbf{c}_{12} - \mathbf{c}_{02}^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{c}_{11} \|\mathbf{F}^\top \mathbf{c}_{13}) + (\mathbf{c}_{01} \|\mathbf{F}^\top \mathbf{c}_{03})^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{c}_{11} \|\mathbf{F}^\top \mathbf{c}_{13}) \pmod{q^2}$ and output the value $\mu \in \{-K + 1, \dots, K - 1\}$ that minimizes $|(\lfloor \frac{q}{K} \rfloor)^2 \cdot \mu - \mu'|$.

4.1 Parameters and Correctness

For ease of exposition, we omit *id* here. Observe that

$$\begin{aligned}
& \mathbf{c}_{02}^\top \mathbf{F} \mathbf{c}_{12} \\
&= (\mathbf{U}_1^\top \mathbf{s}_1 + \mathbf{r}_1 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} (\mathbf{U}_2^\top \mathbf{s}_2 + \mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) \\
&= (\mathbf{U}_1^\top \mathbf{s}_1)^\top \mathbf{F} \mathbf{U}_2^\top \mathbf{s}_2 + (\mathbf{U}_1^\top \mathbf{s}_1)^\top \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} + (\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \mathbf{U}_2^\top \mathbf{s}_2 + (\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \\
&\quad \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} + \mathbf{r}_1^\top \mathbf{F} (\mathbf{U}_2^\top \mathbf{s}_2 + \mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) + (\mathbf{U}_1^\top \mathbf{s}_1 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \mathbf{r}_2
\end{aligned}$$

$$\begin{aligned}
& (\mathbf{c}_{01} \|\mathbf{F}^\top \mathbf{c}_{03}\)^\top \mathbf{E}_1 \mathbf{F} \mathbf{c}_{12} \\
&= ((\mathbf{A}^\top \mathbf{s}_1 + \mathbf{r}'_1) \|\mathbf{F}^\top (\mathbf{R}^\top \mathbf{s}_1 + \mathbf{r}''_1))^\top \mathbf{E}_1 \mathbf{F} (\mathbf{U}_2^\top \mathbf{s}_2 + \mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) \\
&= ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1 + (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}''_1))^\top \mathbf{E}_1 \mathbf{F} (\mathbf{U}_2^\top \mathbf{s}_2 + \mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) \\
&= ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1)^\top \mathbf{E}_1 \mathbf{F} \mathbf{U}_2^\top \mathbf{s}_2 + ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1)^\top \mathbf{E}_1 \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} \\
&\quad + (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}''_1)^\top \mathbf{E}_1 \mathbf{F} (\mathbf{U}_2^\top \mathbf{s}_2 + \mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) + ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1)^\top \mathbf{E}_1 \mathbf{F} \mathbf{r}_2
\end{aligned}$$

$$\begin{aligned}
& \mathbf{c}_{02}^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{c}_{11} \|\mathbf{F}^\top \mathbf{c}_{13}\) \\
&= (\mathbf{U}_1^\top \mathbf{s}_1 + \mathbf{r}_1 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \mathbf{E}_2^\top ((\mathbf{B}^\top \mathbf{s}_2 + \mathbf{r}'_2) \|\mathbf{F}^\top (\mathbf{R}^\top \mathbf{s}_2 + \mathbf{r}''_2)) \\
&= (\mathbf{U}_1^\top \mathbf{s}_1 + \mathbf{r}_1 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \mathbf{E}_2^\top ((\mathbf{B}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_2 + (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2)) \\
&= (\mathbf{U}_1^\top \mathbf{s}_1)^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{B}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_2 + (\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{B}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_2 \\
&\quad + (\mathbf{U}_1^\top \mathbf{s}_1 + \mathbf{r}_1 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2) + \mathbf{r}_1^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{B}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_2
\end{aligned}$$

$$\begin{aligned}
& (\mathbf{c}_{01} \|\mathbf{F}^\top \mathbf{c}_{03}\)^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{c}_{11} \|\mathbf{F}^\top \mathbf{c}_{13}\) \\
&= ((\mathbf{A}^\top \mathbf{s}_1 + \mathbf{r}'_1) \|\mathbf{F}^\top (\mathbf{R}^\top \mathbf{s}_1 + \mathbf{r}''_1))^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top ((\mathbf{B}^\top \mathbf{s}_2 + \mathbf{r}'_2) \|\mathbf{F}^\top (\mathbf{R}^\top \mathbf{s}_2 + \mathbf{r}''_2)) \\
&= ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1 + (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}''_1))^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top ((\mathbf{B}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_2 + (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2)) \\
&= ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1)^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{B}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_2 + (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}''_1)^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top ((\mathbf{B}^\top \|\mathbf{F}^\top \\
&\quad \mathbf{R}^\top) \mathbf{s}_2 + (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2)) + ((\mathbf{A}^\top \|\mathbf{F}^\top \mathbf{R}^\top) \mathbf{s}_1)^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2)
\end{aligned}$$

$$\begin{aligned}
\mu' &= \mathbf{c}_{02}^\top \mathbf{F} \mathbf{c}_{12} - (\mathbf{c}_{01} \|\mathbf{F}^\top \mathbf{c}_{03}\)^\top \mathbf{E}_1 \mathbf{F} \mathbf{c}_{12} - \mathbf{c}_{02}^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{c}_{11} \|\mathbf{F}^\top \mathbf{c}_{13}\) + (\mathbf{c}_{01} \|\mathbf{F}^\top \mathbf{c}_{03}\)^\top \\
&\quad \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{c}_{11} \|\mathbf{F}^\top \mathbf{c}_{13}\) \\
&= (\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x})^\top \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} + \mathbf{r}_1^\top \mathbf{F} \mathbf{r}_2 + \mathbf{r}_1^\top \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x}^\top \mathbf{F} \mathbf{r}_2 - (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}''_1)^\top \\
&\quad \mathbf{E}_1 \mathbf{F} (\mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) - (\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x} + \mathbf{r}_1)^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2) + (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}''_1)^\top \mathbf{E}_1 \mathbf{F} \\
&\quad \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}''_2)
\end{aligned}$$

$$\begin{aligned} \mathbf{error} &= \mathbf{r}_1^\top \mathbf{F} \mathbf{r}_2 + \mathbf{r}_1^\top \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y} + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{x}^\top \mathbf{F} \mathbf{r}_2 - (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}_1''\|)^\top \mathbf{E}_1 \mathbf{F} (\mathbf{r}_2 + \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}) \\ &\quad - (\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x} + \mathbf{r}_1)^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}_2''\|) + (\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}_1''\|)^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}_2''\|) \end{aligned}$$

We set $\|\mathbf{E}_1\| = \|\mathbf{E}_2\| = \beta \leq \sqrt{(m+l)}\sigma$, and note that

$$\begin{aligned} |\mathbf{r}_1^\top \mathbf{F} \mathbf{r}_2| &\leq l^2 V \alpha^2 q^2 \omega(\log n), \\ |\mathbf{r}_1^\top \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}| &= |\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x}^\top \mathbf{F} \mathbf{r}_2| \leq \lfloor \frac{q}{K} \rfloor \cdot l^2 P V \alpha q \omega(\sqrt{\log n}), \\ |(\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}_1''\|)^\top \mathbf{E}_1 \mathbf{F} \mathbf{r}_2| &= |\mathbf{r}_1^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}_2''\|)| < (m+l)^2 V^2 \beta \alpha^2 q^2 \omega(\log n), \\ |(\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}_1''\|)^\top \mathbf{E}_1 \mathbf{F} \lfloor \frac{q}{K} \rfloor \cdot \mathbf{y}| &= |\lfloor \frac{q}{K} \rfloor \cdot \mathbf{x}^\top \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}_2''\|)| \\ &< \lfloor \frac{q}{K} \rfloor \cdot (m+l) l P V^2 \beta \alpha q \omega(\sqrt{\log n}), \\ |(\mathbf{r}'_1 \|\mathbf{F}^\top \mathbf{r}_1''\|)^\top \mathbf{E}_1 \mathbf{F} \mathbf{E}_2^\top (\mathbf{r}'_2 \|\mathbf{F}^\top \mathbf{r}_2''\|)| &< (m+l)^2 V^3 \beta^2 \alpha^2 q^2 \omega(\log n), \end{aligned}$$

Then, $\mathbf{error} \leq (m+l)^2 P V^3 \beta^2 \alpha^2 q^2 \omega(\log n)$

In order to ensure the correctness, we let $\mathbf{error} \leq \lfloor \frac{q}{K} \rfloor^2 / 4$. We set

$$\alpha^{-1} > K^2 \beta \omega(\sqrt{\log n}), \quad q > \alpha^{-1} \omega(\sqrt{n})$$

Additionally, ensure that **TrapGen** and **SampleR** can work. We set

$$m = 5n \log q, \quad \sigma > m \omega(\log m)$$

5 Security Analysis

Theorem 3. *If LWE_{q, χ_α} is hard with the parameters set as above, then the IBFE scheme for quadratic functions is IND-IBFE-CPA secure in the random oracle model.*

Proof. Let \mathcal{A} be an adversary attacking the CPA security of IBFE, we can construct an adversary \mathcal{B} that breaks the LWE assumption.

\mathcal{B} receives $2(m+2l)$ samples from LWE oracle which be parsed as $(\mathbf{p}_{1i}^*, c_{1i}^*) \in \mathbb{Z}_q^n \times \mathbb{Z}_q, i=1, \dots, m+2l, (\mathbf{p}_{2i}^*, c_{2i}^*) \in \mathbb{Z}_q^n \times \mathbb{Z}_q, i=1, \dots, m+2l$. \mathcal{B} 's goal is to guess whether $c_{ji}^* = \mathbf{p}_{ji}^{*\top} \mathbf{s}_j + r$ or c_{ji}^* are uniformly random from $\mathbb{Z}_q, j = 1, 2$.

Then, \mathcal{B} can simulate \mathcal{A} 's view:

- mpk: \mathcal{B} sets $\mathbf{A} = [\mathbf{p}_{11}^*, \dots, \mathbf{p}_{1m}^*], \mathbf{B} = [\mathbf{p}_{21}^*, \dots, \mathbf{p}_{2m}^*], \mathbf{R} = [\mathbf{p}_{j(m+1)}^*, \dots, \mathbf{p}_{j(m+l)}^*]$ where without loss we assume $\mathbf{p}_{1i}^* = \mathbf{p}_{2i}^*, i=m+1, \dots, m+l$, and sends $(\mathbf{A}, \mathbf{B}, \mathbf{R})$ to \mathcal{A} .

- Queries to hash $\mathbf{U}_1(), \mathbf{U}_2()$: on \mathcal{A} 's distinct query id , if $id = id^*$, return $(\mathbf{U}_1(id^*) = [\mathbf{P}_{1(m+l+1)}^*, \dots, \mathbf{P}_{1(m+2l)}^*], \mathbf{U}_2(id^*) = [\mathbf{P}_{2(m+l+1)}^*, \dots, \mathbf{P}_{2(m+2l)}^*])$, or if id is contained in the list, return $(\mathbf{U}_1(id), \mathbf{U}_2(id))$, otherwise, for an \mathbf{F} , choose $\mathbf{E}_1, \mathbf{E}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{(m+l) \times l}, \sigma}$ so that $(\mathbf{A}|\mathbf{R}\mathbf{F})\mathbf{E}_1 = \mathbf{U}_1(id)$ and $(\mathbf{B}|\mathbf{R}\mathbf{F})\mathbf{E}_2 = \mathbf{U}_2(id)$, and store $(id, \mathbf{F}, \mathbf{U}_1(id), \mathbf{U}_2(id), \mathbf{E}_1, \mathbf{E}_2)$ into the list and return $(\mathbf{U}_1(id), \mathbf{U}_2(id))$, where $\mathbf{E}_1, \mathbf{E}_2$ are uniform and have enough entropy. Note that it does not matter that we have no input \mathbf{F} here, because the number of \mathbf{F} is at most V^{l^2} (a polynomial) and maybe we can store all \mathbf{F} corresponding to an id and the same $\mathbf{U}_1(id), \mathbf{U}_2(id)$. Besides, we note that for a sample \mathbf{E}_1 corresponding to an \mathbf{F} , it is hard to find a distinct \mathbf{F}' satisfying $\mathbf{R}\mathbf{F} = \mathbf{R}\mathbf{F}'$ without loss of generality assuming full rank \mathbf{R} .
- Queries to secret keys: when \mathcal{A} asks for a secret key for (id, \mathbf{F}) , we assume without loss of generality that \mathcal{A} has made all relevant queries to $\mathbf{U}_1, \mathbf{U}_2$. If (id, \mathbf{F}) is contained in the list, \mathcal{B} computes and returns $(\mathbf{F}, \mathbf{E}_1\mathbf{F}, \mathbf{F}\mathbf{E}_2^\top, \mathbf{E}_1\mathbf{F}\mathbf{E}_2^\top)$, otherwise returns a random bit and aborts.
- Challenge ciphertext: when \mathcal{A} submits a challenge id^* (distinct from all its queried id) and a pair of distinct message $(\mathbf{x}_0, \mathbf{y}_0)$ and $(\mathbf{x}_1, \mathbf{y}_1)$ which satisfies $\mathbf{x}_0^\top \mathbf{F}\mathbf{y}_0 = \mathbf{x}_1^\top \mathbf{F}\mathbf{y}_1$ for all queried \mathbf{F} , \mathcal{B} picks $\beta \in \{0, 1\}$ and generates ciphertexts as follows:

$$\begin{aligned}
 \mathbf{c}_{01} &= [c_{11}^*, \dots, c_{1m}^*]^\top, & \mathbf{c}_{02} &= [c_{21}^*, \dots, c_{2m}^*]^\top \\
 \mathbf{c}_{03} &= [c_{1(m+1)}^*, \dots, c_{1(m+l)}^*]^\top, & \mathbf{c}_{13} &= [c_{1(m+1)}^*, \dots, c_{1(m+l)}^*]^\top \\
 \mathbf{c}_{02} &= [c_{1(m+l+1)}^*, \dots, c_{1(m+2l)}^*]^\top + \mathbf{x}_\beta, & \mathbf{c}_{12} &= [c_{2(m+l+1)}^*, \dots, c_{2(m+2l)}^*]^\top + \mathbf{y}_\beta
 \end{aligned}$$

When \mathcal{A} terminates with some output, \mathcal{B} terminates with the same output.

It remains to analyze the reduction. It is easy to see that the master public key $\mathbf{A}, \mathbf{B}, \mathbf{R}$ and the random oracle responses $\mathbf{U}_1, \mathbf{U}_2$ are clearly uniformly random. Thanks to the discrete Gaussian distributions, for different \mathbf{F} , there are distinct $\mathbf{E}_1, \mathbf{E}_2$ which have enough entropy so that the adversary can not forge new $\mathbf{E}'_1, \mathbf{E}'_2$ corresponding to arbitrary \mathbf{F}' and acquire more information than $\mathbf{x}_\beta^\top \mathbf{F}\mathbf{y}_\beta$ through collusion attacks. We claim that the probability that \mathcal{B} does not abort during the simulation is $\frac{1}{Q_{\mathbf{U}_1, \mathbf{U}_2}}$ (this is proved by considering a game in which \mathcal{B} can answer all secret key queries). We showed that if \mathcal{B} does not abort during secret key queries, then the challenge ciphertexts is distributed as encryption of $\beta = 0$ or $\beta = 1$ depending on whether the LWE sample is real or random. Therefore, conditioned on \mathcal{B} not aborting, \mathcal{A} 's view is statistically close to the one provided by the real IBFE CPA security experiment. Then, we have

$$Adv_{\mathcal{B}}^{LWE_{q, \chi_\alpha}} \geq \frac{Adv_{\mathcal{A}}^{IND-IBFE-CPA}}{Q_{\mathbf{U}_1, \mathbf{U}_2}} - negl(n).$$

This concludes the proof. □

6 Conclusions and Open Problems

We propose an adaptively secure IBFE scheme for quadratic functions from lattices in the random oracle model. Constructing adaptively secure FE scheme for quadratic functions under standard model is still an open problem.

We formalize the definitions of identity-based functional encryption (IBFE) and its indistinguishability security (IND-IBFE-CPA) which may apply to many scenarios and applications, and it seems easier to construct IBFE schemes than FE schemes, so we appeal for more constructions for more practical function classes for IBFE.

Lattice-based cryptography have many fascinating properties not found in other types of cryptography, but related techniques are still limited to construct and prove some primitives(e.g. FE), so whether we can construct an FE scheme for polynomial functions from standard assumptions is an appealing open problem.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. 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
3. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_2
4. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
5. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 173–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_7
6. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 1–9. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48523-6_1
7. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: International Symposium on Theoretical Aspects of Computer Science, STACS 2009, pp. 75–86 (2009)
8. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15
9. Ananth, P., Sahai, A.: Functional encryption for turing machines. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 125–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_6

10. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 67–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_3
11. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy, pp. 321–334, May 2007
12. Bitansky, N., Nishimaki, R., Passelègue, A., Wichs, D.: From cryptomania to obfustopia through secret-key functional encryption. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 391–418. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_15
13. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), FOCS 2015, Washington, DC, USA, pp. 171–190 (2015)
14. 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
15. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
16. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
17. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
18. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
19. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC 2013, New York, NY, USA, pp. 575–584 (2013)
20. 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
21. 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
22. Coron, J.-S., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of GGH15 multilinear maps. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 607–628. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_21
23. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_1
24. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 480–511. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_18

25. Garg, S., Mahmoody, M., Mohammed, A.: When does functional encryption imply obfuscation? In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 82–115. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_4
26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science(FOCS), pp. 40–49, October 2014
27. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_20
28. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC 2008, New York, USA, pp. 197–206 (2008)
29. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC 2013, New York, NY, USA, pp. 555–564 (2013)
30. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
31. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC 2013, New York, USA, pp. 545–554 (2013)
32. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_25
33. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, New York, USA, pp. 89–98 (2006)
34. Hu, Y., Jia, H.: Cryptanalysis of GGH map. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 537–565. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_21
35. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
36. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
37. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: IEEE Symposium on Foundations of Computer Science, pp. 372–381 (2004)
38. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_13
39. O’Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, report 2010/556 (2010). <https://eprint.iacr.org/2010/556>

40. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC 2009, New York, NY, USA, pp. 333–342 (2009)
41. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2005, New York, NY, USA, pp. 84–93 (2005)
42. 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/10.1007/11426639_27
43. Sans, E.D., Pointcheval, D.: Unbounded inner product functional encryption, with succinct keys. Cryptology ePrint Archive, report 2018/487 (2018). <https://eprint.iacr.org/2018/487>
44. 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
45. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_14
46. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 678–697. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_33



Key Dependent Message Security for Revocable Identity-Based Encryption and Identity-Based Encryption

Rui Zhang^{1,2} and Yang Tao^{1,2}(✉)

¹ State Key Laboratory of Information Security (SKLOIS),
Institute of Information Engineering (IIE),
Chinese Academy of Sciences (CAS), Beijing, China
{r-zhang, taoyang}@iie.ac.cn

² School of Cyber Security,
University of Chinese Academy of Sciences (UCAS), Huairou, China

Abstract. In a revocable identity-based encryption (RIBE) system, the private key and update key are generated separately and combined together to obtain the decryption key. Since the update key is distributed in a public channel, for each user, the private key and the decryption key are essential to his information security. Careless key management, e.g. full disk encryption may leak the encryption of the private key or decryption key, which actually needs to consider the key dependent message (KDM) security. However, previous research mainly focus on the KDM security of IBE and revocability separately and the KDM security for RIBE scheme is still unclear. In this paper, we consider the KDM security for RIBE schemes for the first time and investigate two KDM security models with respect to the private key and decryption key respectively. First, we present a generic construction of KDM-secure RIBE with the private key from any KDM-secure IBE and RIBE in the selective/adaptive chosen-identity model. Second, we construct a concrete KDM-secure RIBE scheme with the decryption key in the selective chosen-identity model from lattices under the *polynomial* modulus. As an independent interest, we also present an efficient lattice-based KDM-secure IBE scheme in the random oracle model. However, it is only secure in the single key setting in the quantum random oracle model.

Keywords: Lattice-based schemes
Key dependent message security
Revocable identity-based encryption

1 Introduction

Identity-based encryption (IBE) is a special public key encryption (PKE), where a user's public key can be an arbitrary string. It was first advocated by Shamir [19] in 1984, mainly to leverage the difficulty of managing certificates for traditional public key infrastructure (PKI). Similar to its PKE counterpart, IBE was

also born without the revocation mechanism. To solve this problem, Boneh and Franklin [5] mentioned to append the current time t to an id , namely, to encrypt a message, the sender uses $id||t$ instead of id , and the private key will be refreshed according to the time. But it is very inefficient, actually linear in the number of remaining users for both computation and bandwidth, since the private keys of all remaining users should be reissued and distributed at the beginning of each time period. Later, in ACM CCS'08, using broadcast encryption for tree-based structures, Boldyreva, Goyal, and Kumar [4] introduced revocable IBE (RIBE), which reduced the complexity of key update information from linear to logarithmic in the number of users. Subsequent work [8, 11, 18, 20] based on bilinear pairings and lattices made further improvements and/or trade-offs.

On the other hand, key management is essential to the security of any practical system. E.g., in full disk encryption, an adversary may see the encryption of the secret key, which was known as key dependent message (KDM) security [3, 6]¹. There are great efforts on KDM-secure PKE [2, 6, 7, 21], but less attention on KDM-secure IBE. The first KDM-secure IBE was introduced by Alperin-Sheriff and Peikert [1]. They considered the scenario of revocation and proved the KDM security against selective chosen-identity attack (KDM-sID-CPA) with respect to the users' private keys in the multi-key setting. However, since the selective security is a weaker security model and the scheme with large keys is less efficient. Hereafter, there are some further improvements [9, 14] on [1] in the aspects of efficiency and security. In the selective security model, Chen et al. [9] optimized the efficiency of the scheme in [1] with indistinguishability obfuscation ($i\mathcal{O}$) and puncturable PRF, where $i\mathcal{O}$ seems impractical by far. As for security optimization, in [9] and [14], they were both interested in the KDM security against adaptive chosen-identity attack (KDM-ID-CPA). Chen et al. [9] proposed a generic construction of KDM-ID-CPA secure IBE from identity-based hash proof system (IB-HPS) with homomorphic property. In the recent work [14], Kitagawa and Tanaka constructed a generic construction of KDM-ID-CPA secure IBE from KDM-secure symmetric key encryption using IND-ID-CPA IBE and garbled circuits.

Similar to its IBE counterpart, RIBE may also share the same practical problem. In an RIBE system, the private key and update key are generated separately, and combined together to generate the decryption key. In the real world, the private key serves as a long-term key, while the update key (an ephemeral key) is distributed through a public channel. Since for each user, the private key and decryption key are the main secrecy of the system, it may damage the user's data confidentiality when such keys are lost or some information is leaked.

¹ If we consider the KDM security in d pairs of public/secret keys, i.e., the multi-key setting, we denote it as d -KDM security and if $d = 1$, it refers to the KDM security in a single key setting, a weaker security notion than d -KDM with $d \geq 2$.

Therefore, it is necessary to consider the KDM security for RIBE. However, the previous research on revocability and KDM security is discussed separately².

As a result, the problems whether one can construct a KDM-secure RIBE system are still open.

1.1 Our Results

In this paper, we pose an affirmative answer to the above problems. Since there are two kinds of secret keys for each user in the RIBE scheme—private key and decryption key, we investigate the KDM security with respect to the private key and decryption key respectively. We view our work as one step ahead towards bringing IBEs to the real-world usage. Our techniques are summarized as follows. Due to the space limit, our proofs are given in the full version of this paper.

First, we propose a generic construction of KDM-secure RIBE with the private key from a KDM-secure IBE and a RIBE scheme, where the KDM-security with the private key is gained from the underlying KDM-secure IBE and revocation mechanism stems from the underlying RIBE. When encrypting a message, we randomly split the message into two parts and encrypt each part using the corresponding IBE and RIBE respectively. As long as the two building blocks is secure in the selective/adaptive chosen-identity model, our generic construction preserves the security in the selective/adaptive chosen-identity model.

Second, as for the KDM security with the decryption key, instead of combining two unrelated building blocks to achieve KDM security and key revocation, we propose a KDM-sID-CPA secure RIBE scheme from lattices by modifying the RIBE proposed by Chen et al. [8]. Recall such RIBE in the selective chosen-identity model utilized two IBE schemes, one of which is used to deal with the identity and the other is corresponding to the time. To achieve the revocation mechanism, Chen et al. adopted the binary data structure and randomly split the public parameter into two parts to link the identity and time. The private key of each user is a set of vectors corresponding to the nodes in the binary tree. Each non-revoked user can get the update key of one node and generate the decryption key relating to such node. Inspired by the work of [8] and [1], we exploit the framework of [8] and replace the IBE building block corresponding to the identity with a KDM-secure IBE of [1]. If guessing the decryption node correctly (with non-negligible probability), we can answer the KDM queries of the decryption key following the strategy of [1]. Therefore, we can obtain a KDM-sID-CPA secure RIBE but suffering from a super-polynomial modulus as [1]. Furthermore, we optimize the modulus from super-polynomial to polynomial by the noise re-randomization technique [12], which leads to a reasonably weak assumption and much efficiency. By the way, our modulus optimization is also applicable to [1].

As an independent interest, we also present an efficient KDM-ID-CPA secure IBE for arbitrary constant identity clique d under the LWE assumption in the

² In [1], scenarios of revocation has been considered for the IBE scheme, but in the concrete construction, they proposed the KDM-secure IBE instead of KDM-secure RIBE.

random oracle. Unlike [9, 14] using the complicated tools, such as $i\mathcal{O}$ and garbled circuits, our IBE scheme is a GPV-style construction [10] and for each identity id , it is actually an image of the KDM-PKE instance in [1]. In the classical random oracle model, we can link the KDM-PKE public key \mathbf{A}_i and the public parameter \mathbf{A}_{id} of IBE with the help of a trapdoor, thus successfully transforming a KDM-challenge ciphertext of PKE scheme to a KDM-challenge response for our IBE scheme. Therefore, our security of KDM-IBE merely relies on the security of the underlying KDM-PKE scheme. In the quantum random oracle, our security is a direct adaption of [22] and only 1-KDM secure. However, when extending the security to d -KDM security, the existing strategy fails and the detailed discussion is put in Sect. 6.2.

Related Work. As for RIBE, there is another stronger security notion—decryption key exposure resistance (DKER), which is introduced by Seo and Emura [18]. Recently, Katsumata et al. [11] proposed a generic construction of RIBE with DKER and gave the first construction based on lattices. However, it is worth mentioning that our concrete RIBE scheme does not support DKER security, since our decryption key is a simple concatenation of the private key and update key.

2 Preliminary

Notations. Denote real numbers by \mathbb{R} and integers by \mathbb{Z} . Denote column vectors over \mathbb{R} and \mathbb{Z} with lower-case bold letters (e.g. \mathbf{x}), and matrices by upper-case bold letters (e.g. \mathbf{A}). Denote the matrix $[\mathbf{A}_1|\mathbf{A}_2]$ the concatenating the matrix \mathbf{A}_1 and \mathbf{A}_2 . For a positive integer d , let $[d]$ denote the integer set $\{1, \dots, d\}$. If S is a set, $s \xleftarrow{r} S$ denotes sampling randomly s from uniform distribution over S . A function $\text{negl}(n) : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is negligible if sufficiently large $n > n_0$ (n_0 is a constant), $\text{negl}(n) < 1/\text{poly}(n)$. The statistical distance between two random variables X and Y over a countable set D is $\Delta(X, Y) = \frac{1}{2} \sum_{w \in D} |\Pr[X = w] - \Pr[Y = w]|$. Let $\{X_n\}$ and $\{Y_n\}$ be ensembles of random variables indexed by a security parameter n , we say that $\{X_n\}$ and $\{Y_n\}$ are statistically close if $\Delta(X_n, Y_n)$ is negligible function of n . For a matrix $\mathbf{R} \in \mathbb{R}^{l \times t}$, the largest singular value of \mathbf{R} is defined as $s_1(\mathbf{R}) = \max_{\|\mathbf{u}\|=1} \|\mathbf{R}\mathbf{u}\|$. We fix a universal gadget matrix $\mathbf{G} = \mathbf{I}_n \otimes (1, 2, 4, \dots, 2^{k-1}) \in \mathbb{Z}_q^{n \times w}$ for $k = \lceil \log q \rceil$ and $w = nk = n \lceil \log q \rceil$. In this paper, we use $\text{negl}(n)$ to denote a class of negligible functions instead of some fixed function.

2.1 Lattices and Gaussian Measures

An n -dimension (full-rank) lattice $\Lambda \subseteq \mathbb{R}^n$ is the set of all integer linear combinations of some set of independent basis vectors $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\} \subseteq \mathbb{R}^{n \times n}$, $\Lambda = \mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^n z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}$. The dual lattice of $\Lambda \subseteq \mathbb{R}^n$ is defined as $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \langle \Lambda, \mathbf{x} \rangle \subseteq \mathbb{Z}\}$. For integers $n \geq 1$, modulus $q \geq 2$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, an m -dimensional lattice is defined as $\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \in \mathbb{Z}_q^n\} \subseteq \mathbb{Z}^m$.

For any \mathbf{y} in the subgroup of \mathbb{Z}_q^n , we also define the coset $\Lambda_{\mathbf{y}}^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{y} \pmod q\} = \Lambda^\perp(\mathbf{A}) + \bar{\mathbf{x}}$, where $\bar{\mathbf{x}} \in \mathbb{Z}^m$ is an arbitrary solution to $\mathbf{A}\bar{\mathbf{x}} = \mathbf{y}$.

Gaussian Measures. Let Λ be a lattice in \mathbb{Z}^n . For any vector $\mathbf{c} \in \mathbb{R}^n$ and parameter $r > 0$, the n -dimensional Gaussian function $\rho_{r,\mathbf{c}} : \mathbb{R}^n \rightarrow (0, 1]$ is defined as $\rho_{r,\mathbf{c}}(\mathbf{x}) := \exp(-\pi\|\mathbf{x} - \mathbf{c}\|^2/r^2)$. The discrete Gaussian distribution over Λ with parameter r and center \mathbf{c} (abbreviated as $D_{\Lambda,r,\mathbf{c}}$) is defined as $\forall \mathbf{y} \in \Lambda, D_{\Lambda,r,\mathbf{c}}(\mathbf{y}) := \frac{\rho_{r,\mathbf{c}}(\mathbf{y})}{\sum_{\mathbf{y}' \in \Lambda} \rho_{r,\mathbf{c}}(\mathbf{y}')}$, where $\rho_{r,\mathbf{c}}(\Lambda) = \sum_{\mathbf{y} \in \Lambda} \rho_{r,\mathbf{c}}(\mathbf{y})$. When $\mathbf{c} = \mathbf{0}$, we write $D_{\Lambda,r}$ for short.

Lemma 1. ([10], Theorem 4.1). There is a probabilistic polynomial-time algorithm `SampleGaussian` that, given a basis \mathbf{B}_Λ of an n -dimensional lattice Λ , a parameter $r \geq \|\mathbf{B}_\Lambda\| \cdot \omega(\sqrt{\log n})$, and a center $\mathbf{c} \in \mathbb{R}^n$, outputs a sample from a distribution that is statistically close to $D_{\Lambda,r,\mathbf{c}}$.

Lemma 2. ([10, 15, 16]). Let $m \geq Cn \log q$ for some constant $C > 1$.

1. For any n -dimensional lattice Λ , any $\mathbf{c} \in \mathbb{Z}^n$, and any $r \geq \eta_\varepsilon(\Lambda)$,³ where $\varepsilon(n) = \text{negl}(n)$, we have $\|D_{\Lambda+\mathbf{c},r}\| \leq r\sqrt{n}$ with all but $\text{negl}(n)$ probability. In addition, for $\Lambda = \mathbb{Z}$ we have $|D_{\mathbb{Z},r}| \leq r \cdot \omega(\sqrt{\log n})$ except with $\text{negl}(n)$ probability.
2. For any $r > 0$, and for $\mathbf{R} \leftarrow D_{\mathbb{Z},r}^{n \times k}$, we have $s_1(\mathbf{R}) \leq r \cdot O(\sqrt{n} + \sqrt{k})$ except with $\text{negl}(n)$ probability.
3. With all but $\text{negl}(n)$ probability over the uniformly random choice of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the following holds: For $\mathbf{e} \leftarrow D_{\mathbb{Z},r}^m$, where $r = \omega(\sqrt{\log n})$, the distribution of $\mathbf{y} = \mathbf{A}\mathbf{e} \pmod q$ is within $\text{negl}(n)$ statistical distance of uniform, and the conditional distribution of \mathbf{e} given \mathbf{y} is $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),r}$.

Lemma 3. ([12], Lemma 1). Let q, l, m be positive integers and r a positive real satisfying $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log l})\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and \mathbf{x} chosen from $D_{\mathbb{Z},r}^m$. Then for any $\mathbf{V} \in \mathbb{Z}^{m \times l}$ and positive real $\sigma > s_1(\mathbf{V})$, there exists a PPT algorithms `ReRand`($\mathbf{V}, \mathbf{b} + \mathbf{x}, r, \sigma$) that outputs $\mathbf{b}' = \mathbf{b}\mathbf{V} + \mathbf{x}' \in \mathbb{Z}_q^l$ where \mathbf{x}' is distributed statistically close to $D_{\mathbb{Z},2r\sigma}^l$.

A new trapdoor notion was introduced in [15]. The strong trapdoor \mathbf{R} for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ refers that for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, we have $\mathbf{R} \in \mathbb{Z}^{(m-w) \times w}$ such that $\mathbf{A} \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H}\mathbf{G}$.

Lemma 4. ([15], Theorem 5.1). Let \mathbf{R} be a strong trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. There is an efficient randomized algorithm that given \mathbf{R} , any $\mathbf{u} \in \mathbb{Z}_q^n$, and any $r \geq s_1(\mathbf{R}) \cdot \omega(\sqrt{\log n}) \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{A}))$ (for some $\varepsilon(n) = \text{negl}(n)$), samples from a distribution within $\text{negl}(n)$ distance of $D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{A}),r}$.

³ For a lattice Λ and a positive real $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is defined as the smallest real $r > 0$ such that $\rho_{1/r}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$. Especially, for any $\omega(\sqrt{\log n})$ function, $\eta_\varepsilon(\mathbb{Z}^n) \leq \omega(\sqrt{\log n})$.

Learning with Errors Assumption. The learning with errors (LWE) problem was introduced by Regev [17], which is at least as hard as several lattice problems in the worst case. For security parameter λ , let $n = n(\lambda)$ be an integer dimension, let $q = q(\lambda) \geq 2$ be an integer, and let $\chi = \chi(\lambda)$ be a distribution over \mathbb{Z} . The $\text{LWE}_{n,q,\chi}$ problem is to distinguish the two distributions: $\{\mathbf{A}, \mathbf{A}^t \mathbf{s} + \mathbf{x}\}$ and $\{\mathbf{A}, \mathbf{u}\}$ where $\mathbf{A} \xleftarrow{r} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{r} \mathbb{Z}_q^n$, $\mathbf{u} \xleftarrow{r} \mathbb{Z}_q^m$, and $\mathbf{x} \leftarrow \chi^m$. When the error distribution $\chi = D_{\mathbb{Z},\alpha,q}$, the problem is abbreviated as $\text{LWE}_{n,q,\alpha}$.

2.2 KDM-PKE Scheme in [1]

KDM Security. We mainly consider the KDM security from [1,3]. In their definitions, the adversary \mathcal{A} plays a game with the challenger \mathcal{C} , and is able to make queries for encryptions of functions of secret keys. The functions which the adversary queries are restricted in a certain family $\mathcal{F} \subset \{f : \mathcal{K} \rightarrow \mathcal{M}\}$ (\mathcal{F} contains constant functions on \mathcal{M}), where \mathcal{K} is the keyspace of secret keys and \mathcal{M} is the message space of the encryption scheme. In the definition of [1], the adversary assigns two functions $(f_0, f_1) \in \mathcal{F}$ with each query, and must distinguish between the encryptions of f_0 and encryptions of f_1 . Concretely, for the PKE scheme (Setup, Enc, Dec), the d -KDM-CPA security game between an adversary and the challenger parameterized by $\beta \in \{0, 1\}$ proceeds as follows.

- **Setup:** The challenger runs $(\text{PK}_i, \text{SK}_i) \leftarrow \text{Setup}(1^n)$ for $i \in [d]$ and the adversary \mathcal{A} is given the challenge public keys $I = \{\text{PK}_1, \dots, \text{PK}_l\}$ for some $l \leq d$.
- **Query:** \mathcal{A} may adaptively make a polynomial number of queries: \mathcal{A} can make encryption query of the form (i, f_0, f_1) , where $f_0, f_1 \in \mathcal{F}$ and $1 \leq i \leq l$. The challenger computes $\mu \leftarrow f_\beta(\text{SK}_1, \dots, \text{SK}_l)$ and $c \leftarrow \text{Enc}(\text{PK}_i, \mu)$, and responses a ciphertext c .

We say the scheme is d -KDM-CPA secure with respect to \mathcal{F} if the game for $\beta = 0, 1$ are computationally indistinguishable.

The d -KDM security against selective chosen-identity and chosen-message attack for IBE scheme (d -KDM-sID-CPA) was defined in [1]. A stronger security model for IBE, i.e., adaptive chosen-identity and chosen-message attack (d -KDM-ID-CPA), is similar to the above definition for d target identities with the KDM ciphertext of $f(\text{SK}_{\text{id}_1^*}, \dots, \text{SK}_{\text{id}_d^*})$ except that the challenge identities can be chosen adaptively even after seeing the public key.

KDM-PKE Scheme. Let a modulus be $q = p^2$ for a polynomial prime $p \geq r^2 \sqrt{n+m} \cdot \omega(\sqrt{\log n})$, where n, m are integers, r is a Gaussian parameter satisfying $r \geq 2\sqrt{n}$ and λ be a security parameter. The message space is \mathbb{Z}_p . The KDM-secure PKE scheme Π_{PKE} in [1] consists of three algorithms:

- $(\text{PK}, \text{SK}) \leftarrow \overline{\text{Gen}}(1^\lambda)$: Choose $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{z}_0 \leftarrow D_{\mathbb{Z},r}^n$, $\mathbf{z}_1 \leftarrow D_{\mathbb{Z},r}^m$, and let $\mathbf{y} = \mathbf{z}_0 - \mathbf{A}\mathbf{z}_1 = [\mathbf{I}_n \mid -\mathbf{A}]\mathbf{z} \in \mathbb{Z}_q^n$ where $\mathbf{z} = [\mathbf{z}_0^t \mid \mathbf{z}_1^t]^t \in \mathbb{Z}^{n+m}$. The public key PK is (\mathbf{A}, \mathbf{y}) and the secret key SK is \mathbf{z}_1 .

- $\mathbf{c}^t \leftarrow \overline{\text{Enc}}(\mathbf{A}, \mathbf{y}, \mu)$: To encrypt a message $\mu \in \mathbb{Z}_p$, choose $\mathbf{x}_0 \leftarrow D_{\mathbb{Z},r}^n$, $\mathbf{x}_1 \leftarrow D_{\mathbb{Z},r}^m$ and $x' \leftarrow D_{\mathbb{Z},r}$. Output the ciphertext $\mathbf{c}^t = \mathbf{x}_0^t[\mathbf{A}|\mathbf{y}] + [\mathbf{x}_1^t|x'] + [\mathbf{0}|p \cdot \mu] \in \mathbb{Z}_q^{1 \times (m+1)}$.
- $\mu \leftarrow \overline{\text{Dec}}(\mathbf{z}_1, \mathbf{c})$: Compute $\mu' = \mathbf{c}^t \begin{bmatrix} \mathbf{z}_1 \\ 1 \end{bmatrix} \in \mathbb{Z}_q$. Output the $\mu \in \{0, \dots, p-1\} = \mathbb{Z}_p$ such that μ' is closest to $p \cdot \mu \pmod q$.

Theorem 1. ([1]). The above cryptosystem is d -KDM-CPA secure with respect to the set of affine functions over \mathbb{Z}_p under the LWE assumption.

3 Security Model

In this section, we review the definition of revocable identity-based encryption (RIBE), and adapt the KDM security to the RIBE scheme. Since there are two kinds of secret keys for each user—the private key and the decryption key in the RIBE schemes, we consider our KDM security with respect to the private key and decryption key respectively.

An RIBE scheme has seven probabilistic polynomial-time (PPT) algorithms **Setup**, **PriKeyGen**, **KeyUpd**, **DecKeyGen**, **Enc**, **Dec** and **KeyRev** with associated message space \mathcal{M} , identity space \mathcal{I} , and time space \mathcal{T} . We assume that the size of \mathcal{T} is a polynomial in the security parameter. There are three parties: key authority, sender and receiver. Key authority maintains a revocation list RL and state ST. Hereafter, an algorithm is called stateful if RL or ST needs updating for revocation.

- $(\text{PP}, \text{MK}, \text{RL}, \text{ST}) \leftarrow \mathbf{Setup}(1^n, N)$: Taking as input a security parameter n and a maximal number of users N , it outputs public parameters PP, a master secret key MK, a revocation list RL (initially empty) and a state ST.
- $(\text{SK}_{\text{id}}, \text{ST}) \leftarrow \mathbf{PriKeyGen}(\text{PP}, \text{MK}, \text{id}, \text{ST})$: Taking as input public parameters PP, a master secret key MK, an identity $\text{id} \in \mathcal{I}$ and a state ST, it outputs a private key SK_{id} and an updated state ST.
- $\text{KU}_t \leftarrow \mathbf{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST})$: Taking as input public parameters PP, a master secret key MK, a key update time $t \in \mathcal{T}$, a revocation list RL and a state ST, it outputs an update key KU_t .
- $\text{DK}_{\text{id},t}/\perp \leftarrow \mathbf{DecKeyGen}(\text{PP}, \text{SK}_{\text{id}}, \text{KU}_t)$: Taking as input public parameters PP, a private key SK_{id} and an update key KU_t , it outputs a decryption key $\text{DK}_{\text{id},t}$ or a special symbol \perp indicating that id has been revoked.
- $\text{CT}_{\text{id},t} \leftarrow \mathbf{Enc}(\text{PP}, \text{id}, t, \mu)$: Taking as input public parameters PP, an identity $\text{id} \in \mathcal{I}$, an encryption time $t \in \mathcal{T}$ and a message $\mu \in \mathcal{M}$, it outputs a ciphertext $\text{CT}_{\text{id},t}$.
- $\mu \leftarrow \mathbf{Dec}(\text{PP}, \text{DK}_{\text{id},t}, \text{CT}_{\text{id},t})$: Taking as input public parameters PP, a decryption key $\text{DK}_{\text{id},t}$ and a ciphertext $\text{CT}_{\text{id},t}$, it outputs a message $\mu \in \mathcal{M}$.
- $\text{RL} \leftarrow \mathbf{KeyRev}(\text{id}, t, \text{RL}, \text{ST})$: Taking as input an identity to be revoked $\text{id} \in \mathcal{I}$, a revocation time $t \in \mathcal{T}$, a revocation list RL and a state ST, it outputs an updated revocation list RL.

We require the correctness condition holds that \forall polynomially-bounded N , \forall $(PP, MK) \leftarrow \text{Setup}(1^n, N)$, $\forall \mu \in \mathcal{M}$, $\forall \text{id} \in \mathcal{I}$, $\forall \mathbf{t} \in \mathcal{T}$, all possible valid states ST and revocation lists RL , if identity id was not revoked before or at time \mathbf{t} , for $(SK_{\text{id}}, ST) \leftarrow \text{PriKeyGen}(PP, MK, \text{id}, ST)$, $KU_{\mathbf{t}} \leftarrow \text{KeyUpd}(PP, MK, \mathbf{t}, RL, ST)$ and $DK_{\text{id}, \mathbf{t}} \leftarrow \text{DecKeyGen}(PP, SK_{\text{id}}, KU_{\mathbf{t}})$, we have $\Pr[\text{Dec}(PP, DK_{\text{id}, \mathbf{t}}, \text{Enc}(PP, \text{id}, \mathbf{t}, \mu)) \neq \mu] \leq \text{negl}(n)$.

Now we define the KDM security games in the RIBE. For simplicity, we only consider the selective chosen-identity attack model⁴ in the single identity setting and take into account the key revocation and KDM security with respect to function family \mathcal{F} . We define the KDM security game with decryption key and private key respectively.

The KDM security game with the *decryption key* between an adversary \mathcal{A} and a challenger \mathcal{C} is parameterized by some $\beta \in \{0, 1\}$ and proceeds as follows:

- **Initial:** The adversary first outputs the challenge identity id^* and time \mathbf{t}^* , and also some information state it wants to preserve.
- **Setup:** The challenger runs $(PP, MK, RL, ST) \leftarrow \text{Setup}(1^n, N)$, and the adversary \mathcal{A} is given public parameters PP .
- **Query:** \mathcal{A} may adaptively make a polynomial number of queries of the following oracles (the oracles share state):
 - **Extraction Queries:** \mathcal{A} can query $\text{PriKeyGen}(\cdot)$ for identity id , and gets a private key SK_{id} .
 - **Update Queries:** \mathcal{A} can query $\text{KeyUpd}(\cdot)$ for time \mathbf{t} , and gets an update key $KU_{\mathbf{t}}$.
 - **Revocation Queries:** \mathcal{A} can query $\text{KeyRev}(\cdot, \cdot)$ for identity id and time \mathbf{t} , and gets an update RL .
 - **KDM-Encryption Queries:** \mathcal{A} can make encryption queries of the form (f_0, f_1) , where $f_0, f_1 \in \mathcal{F}$. If $DK_{\text{id}^*, \mathbf{t}^*} \neq \perp$, the challenger \mathcal{C} computes $\mu \leftarrow f_\beta(DK_{\text{id}^*, \mathbf{t}^*})$ and $c \leftarrow \text{Enc}(PP, \text{id}^*, \mathbf{t}^*, \mu)$, and responds a ciphertext $CT_{\text{id}^*, \mathbf{t}^*}$. If $DK_{\text{id}^*, \mathbf{t}^*} = \perp$ and $f_\beta = m_\beta$, a constant function in \mathcal{M} , \mathcal{C} returns encryption of m_β , i.e., $CT_{\text{id}^*, \mathbf{t}^*} \leftarrow \text{Enc}(PP, \text{id}^*, \mathbf{t}^*, m_\beta)$. Otherwise, if $DK_{\text{id}^*, \mathbf{t}^*} = \perp$ and f_β is not a constant function in \mathcal{M} , then \mathcal{C} returns the encryption of zero string, i.e., $CT_{\text{id}^*, \mathbf{t}^*} \leftarrow \text{Enc}(PP, \text{id}^*, \mathbf{t}^*, \mathbf{0})$.
- **Guess:** The adversary outputs a bit β' . If $\beta' = \beta$, \mathcal{A} succeeds.

The KDM security game with the *private key* between an adversary \mathcal{A} and a challenger \mathcal{C} is the same as above, except the KDM-Encryption queries. Such KDM-Encryption queries with the private key are as follows.

- **KDM-Encryption Queries:** \mathcal{A} can make encryption queries of the form (f_0, f_1) , where $f_0, f_1 \in \mathcal{F}$. The challenger \mathcal{C} computes $\mu \leftarrow f_\beta(SK_{\text{id}^*})$ and $c \leftarrow \text{Enc}(PP, \text{id}^*, \mathbf{t}^*, \mu)$, and responds a ciphertext $CT_{\text{id}^*, \mathbf{t}^*}$.

⁴ The adaptive chosen-identity model is identical to the selective chosen-identity model, except the target identity and time are adaptively chosen by the adversary after seeing the public key.

We insist that $\text{KeyUpd}(\cdot)$ and $\text{KeyRev}(\cdot, \cdot)$ can be queried on time in a non-decreasing order of time. Besides, the following restrictions should always hold:

1. $\text{KeyRev}(\cdot, \cdot)$ cannot be queried at time t if $\text{KeyUpd}(\cdot)$ was queried on t .
2. If $\text{PriKeyGen}(\cdot)$ was queried on identity id^* , then $\text{KeyRev}(\text{id}^*, \cdot)$ must be queried at time $t \leq t^*$. In other words, id^* must be revoked at time $t \leq t^*$.
3. If $t \geq t^*$ and ID^* is not revoked at t^* , then $\text{PriKeyGen}(\text{ID}^*)$ cannot be queried at time t .

We say that a scheme has KDM security with the decryption key/private key against selective chosen-identity and chosen-plaintext attacks (KDM-sID-CPA) with respect to \mathcal{F} , if the advantage of any PPT adversary \mathcal{A} is bounded by a negligible function $\text{negl}(n)$, i.e. $\text{Adv}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2}| \leq \text{negl}(n)$.

4 KDM Security for RIBE with the Private Key

In this section, we give a generic construction of KDM-secure RIBE with the private key from a KDM-secure IBE and a RIBE scheme. Let $\text{k}.\Pi = (\text{k.Setup}, \text{k.PriKeyGen}, \text{k.Enc}, \text{k.Dec})$ be a KDM-secure IBE scheme with identity space $\text{k}.\mathcal{I}$ and message space $\text{k}.\mathcal{M}$. Let $\text{r}.\Pi = (\text{r.Setup}, \text{r.PriKeyGen}, \text{r.KeyUpd}, \text{r.DecKeyGen}, \text{r.Enc}, \text{r.Dec}, \text{KeyRev})$ be an RIBE scheme with identity space $\text{r}.\mathcal{I}$, time space $\text{r}.\mathcal{T}$ and message space $\text{r}.\mathcal{M}$. We assume $\text{k}.\mathcal{I} = \text{r}.\mathcal{I}, \text{k}.\mathcal{M} = \text{r}.\mathcal{M}$.

Our KDM-secure RIBE $\Pi = (\text{Setup}, \text{PriKeyGen}, \text{KeyUpd}, \text{DecKeyGen}, \text{Enc}, \text{Dec}, \text{KeyRev})$ with identity space \mathcal{I} , message space \mathcal{M} and time space \mathcal{T} . Assuming $\mathcal{I} = \text{k}.\mathcal{I} = \text{r}.\mathcal{I}, \mathcal{M} = \text{k}.\mathcal{M} = \text{r}.\mathcal{M}, \mathcal{T} = \text{r}.\mathcal{T}$, our construction is as follows.

- $(\text{PP}, \text{MK}, \text{RL}, \text{ST}) \leftarrow \text{Setup}(1^n, N)$: Taking as input a security parameter n and a maximal number of users N , run $(\text{k.PP}, \text{k.MK}) \leftarrow \text{k.Setup}(1^n)$ and $(\text{r.PP}, \text{r.MK}) \leftarrow \text{r.Setup}(1^n, N)$. It outputs public parameters $\text{PP}=(\text{k.PP}, \text{r.PP})$, a master secret key $\text{MK}=(\text{k.MK}, \text{r.MK})$, a revocation list RL (initially empty) and a state ST .
- $(\text{SK}_{\text{id}}, \text{ST}) \leftarrow \text{PriKeyGen}(\text{PP}, \text{MK}, \text{id}, \text{ST})$: Taking as input public parameters PP , a master secret key MK , an identity $\text{id} \in \mathcal{I}$ and a state ST , run $\text{k.SK}_{\text{id}} \leftarrow \text{k.PriKeyGen}(\text{k.PP}, \text{k.MK}, \text{id})$ and $\text{r.SK}_{\text{id}} \leftarrow \text{r.PriKeyGen}(\text{r.PP}, \text{r.MK}, \text{id}, \text{ST})$. It outputs a private key $\text{SK}_{\text{id}} = (\text{k.SK}_{\text{id}}, \text{r.SK}_{\text{id}})$ and an updated state ST .
- $\text{KU}_t \leftarrow \text{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST})$: Taking as input public parameters PP , a master secret key MK , a key update time $t \in \mathcal{T}$, a revocation list RL and a state ST , run $\text{r.KU}_t \leftarrow \text{r.KeyUpd}(\text{r.PP}, \text{r.MK}, t, \text{RL}, \text{ST})$. it outputs an update key $\text{KU}_t = \text{r.KU}_t$.
- $\text{DK}_{\text{id}, t} / \perp \leftarrow \text{DecKeyGen}(\text{PP}, \text{SK}_{\text{id}}, \text{KU}_t)$: Taking as input public parameters PP , a private key SK_{id} and an update key KU_t , run $\text{r.DK}_{\text{id}, t} \leftarrow \text{r.DeyKeyGen}(\text{r.PP}, \text{r.SK}_{\text{id}}, \text{r.KU}_t)$. It outputs a decryption key $\text{DK}_{\text{id}, t} = (\text{k.SK}_{\text{id}}, \text{r.DK}_{\text{id}, t})$ or a special symbol \perp if $\text{r.DK}_{\text{id}, t} = \perp$ indicating that id has been revoked.

- $\text{CT}_{\text{id},\text{t}} \leftarrow \text{Enc}(\text{PP}, \text{id}, \text{t}, \mu)$: Taking as input public parameters PP , an identity $\text{id} \in \mathcal{I}$, an encryption time $\text{t} \in \mathcal{T}$ and a message $\mu \in \mathcal{M}$, sample a uniform pair $(\text{k}.\mu, \text{r}.\mu) \in \mathcal{M}^2$ such that $\mu = \text{k}.\mu + \text{r}.\mu$. Run $\text{k}.\text{CT} \leftarrow \text{k}.\text{Enc}(\text{k}.\text{PP}, \text{id}, \text{k}.\mu)$ and $\text{r}.\text{CT} \leftarrow \text{r}.\text{Enc}(\text{r}.\text{PP}, \text{id}, \text{t}, \text{r}.\mu)$. It outputs a ciphertext $\text{CT}_{\text{id},\text{t}} = (\text{k}.\text{CT}, \text{r}.\text{CT})$.
- $\mu \leftarrow \text{Dec}(\text{PP}, \text{DK}_{\text{id},\text{t}}, \text{CT}_{\text{id},\text{t}})$: Taking as input public parameters PP , a decryption key $\text{DK}_{\text{id},\text{t}}$ and a ciphertext $\text{CT}_{\text{id},\text{t}}$, run $\text{k}.\mu \leftarrow \text{k}.\text{Dec}(\text{k}.\text{PP}, \text{k}.\text{SK}_{\text{id}}, \text{k}.\text{CT})$ and $\text{r}.\mu \leftarrow \text{r}.\text{Dec}(\text{r}.\text{PP}, \text{r}.\text{DK}_{\text{id},\text{t}}, \text{r}.\text{CT})$. If $\text{k}.\mu = \perp$ or $\text{r}.\mu = \perp$, it output \perp . Otherwise, it outputs a message $\mu = \text{k}.\mu + \text{r}.\mu \in \mathcal{M}$.
- $\text{RL} \leftarrow \text{KeyRev}(\text{id}, \text{t}, \text{RL}, \text{ST})$: Taking as input an identity to be revoked $\text{id} \in \mathcal{I}$, a revocation time $\text{t} \in \mathcal{T}$, a revocation list RL and a state ST , run $\text{r}.\text{RL} \leftarrow \text{r}.\text{KeyRev}(\text{id}, \text{t}, \text{RL}, \text{ST})$. It outputs an updated revocation list $\text{RL} = \text{r}.\text{RL}$.

The correctness of our scheme Π relies on the correctness of each building block, i.e., $\text{k}.\Pi$ and $\text{r}.\Pi$. The security of Π is given as follows.

Theorem 2. Assuming the IBE scheme $\text{k}.\Pi$ is KDM-sID-CPA secure (resp. KDM-ID-CPA) with the private key with respect to the affine functions \mathcal{F} and the RIBE scheme $\text{r}.\Pi$ is IND-sID-CPA secure (resp. IND-ID-CPA), then the scheme Π is KDM-sID-CPA secure (resp. KDM-ID-CPA) with the private key with respect to \mathcal{F} .

Proof. (Sketch) Since the proofs for the selective chosen-identity model and the adaptive chosen-identity model are the same, we only consider the proof in the selective chosen-identity model. Let id^* and t^* be the target identity and time. Denote Q the number of extraction queries issued by the adversary \mathcal{A} . For $1 \leq i \leq Q$, let SK_{id_i} denote the queried private key on identity id_i . For simplicity, we divide \mathcal{A} into two types:

- Type I : for $\forall i \in [Q]$, $\text{SK}_{\text{id}^*} \neq \text{SK}_{\text{id}_i}$. That means, \mathcal{A} does not issue the extraction query on id^* .
- Type II : for some $i \in [Q]$, $\text{SK}_{\text{id}^*} = \text{SK}_{\text{id}_i}$. That means, \mathcal{A} learns the private key SK_{id^*} by the extraction query, but id^* is revoked at t^* .

Combining the Lemmas 5 and 6, the advantage of \mathcal{A} is negligible. \square

Lemma 5. If \mathcal{A} is a successful adversary of Type I, there exists a PPT simulator \mathcal{B} attacking the underlying KDM-IBE scheme with non-negligible probability.

Lemma 6. If \mathcal{A} is a successful adversary of Type II, there exists a PPT simulator \mathcal{B} attacking the underlying RIBE scheme with non-negligible probability.

Remark 1. Especially, our RIBE construction Π is even d -KDM-secure if the underlying IBE scheme $\text{k}.\Pi$ is d -KDM secure, where d is the private key clique size.

5 KDM Security for RIBE with the Decryption Key

In this section, we present a concrete construction for KDM-secure RIBE with the decryption key in the selective chosen-identity model.

5.1 Our KDM-RIBE Scheme

In our construction, we use the instance of [1] to deal with user’s identity and another IBE instance to handle the time. As in [1], we use an additive group $\mathbb{G} = \mathbb{Z}_q^n$ and a ring $R = \mathbb{Z}_q[x]/(F(x))$, where $F(x)$ is a monic and irreducible polynomial with degree n and \mathbb{G} is an R -module⁵. Let p be the smallest prime divisor of q , and we define $U = \{u_0 = 0, u_1, u_2, \dots\} \subseteq R$ as the set consisting of all the polynomials having coefficients in \mathbb{Z}_p , which makes $|U| = p^n \geq 2^n$ and $u_i - u_j$ a unit for any $i \neq j$. (See detailed discussion in [1].)

Suppose identity id and time t are encoded into $U \setminus \{0\}$ respectively and our RIBE scheme from lattices is described below.

- $(\text{PP}, \text{MK}, \text{RL}, \text{ST}) \leftarrow \mathbf{Setup}(1^n, N)$: On input a security parameter n and a maximal number N of users, perform the following steps:
 1. Sample $\mathbf{R}_1, \mathbf{R}_2 \leftarrow D_{\mathbb{Z}, \omega(\sqrt{\log n})}^{m \times w}$, choose uniformly random $\mathbf{A} \xleftarrow{r} \mathbb{Z}_q^{n \times m}$, $\mathbf{y} \xleftarrow{r} \mathbb{Z}_q^n$ and let $\tilde{\mathbf{A}}_1 = \mathbf{A}\mathbf{R}_1, \tilde{\mathbf{A}}_2 = \mathbf{A}\mathbf{R}_2 \in \mathbb{Z}_q^{n \times w}$.
 2. Let RL be an empty set and BT be a binary-tree with at least N leaf nodes, and set $\text{ST} := \text{BT}$.
 3. Output RL, ST, public parameters $\text{PP} = (\mathbf{A}, \tilde{\mathbf{A}}_1, \tilde{\mathbf{A}}_2, \mathbf{y})$ and a master secret key $\text{MK} = \{\mathbf{R}_1, \mathbf{R}_2\}$.
- $(\text{SK}_{\text{id}}, \text{ST}) \leftarrow \mathbf{PriKeyGen}(\text{PP}, \text{MK}, \text{id}, \text{ST})$: On input public parameters PP, a master secret key MK, an identity $\text{id} \in U \setminus \{0\}$ and a state ST, it picks an unassigned leaf node v from BT and stores id in that node. It then performs the following steps:
 1. For any $\theta \in \text{Path}(v)$, if $\mathbf{y}_{\theta,1}, \mathbf{y}_{\theta,2}$ are undefined, pick $\mathbf{y}_{\theta,1} \xleftarrow{r} \mathbb{Z}_q^n$, set $\mathbf{y}_{\theta,2} := \mathbf{y} - \mathbf{y}_{\theta,1}$ and store them in the node θ . Set $\mathbf{A}_{\text{id}} := [\mathbf{A} | (\text{id})\mathbf{G} - \tilde{\mathbf{A}}_1]$. Sample $\mathbf{z}_0 \leftarrow D_{\mathbb{Z}, r}^n$, $\mathbf{e}_{\theta,1} \leftarrow D_{\Lambda_{\mathbf{z}_0 - \mathbf{y}_{\theta,1}}^\perp(\mathbf{A}_{\text{id}}), r}$ so that $\mathbf{z}_0 - \mathbf{A}_{\text{id}}\mathbf{e}_{\theta,1} = \mathbf{y}_{\theta,1}$.
 2. Output $\text{SK}_{\text{id}} := \{(\theta, \mathbf{e}_{\theta,1})\}_{\theta \in \text{Path}(v)}$.
- $\text{KU}_t \leftarrow \mathbf{KeyUpd}(\text{PP}, \text{MK}, t, \text{RL}, \text{ST})$: On input public parameters PP, a master secret key MK, a time $t \in U \setminus \{0\}$, a revocation list RL and a state ST, it performs the following steps:
 1. For any $\theta \in \text{KUNodes}(\text{BT}, \text{RL}, t)$ ⁶, if $\mathbf{y}_{\theta,1}, \mathbf{y}_{\theta,2}$ are undefined, pick $\mathbf{y}_{\theta,1} \xleftarrow{r} \mathbb{Z}_q^n$, set $\mathbf{y}_{\theta,2} := \mathbf{y} - \mathbf{y}_{\theta,1}$ and store them in the node θ . Set $\mathbf{A}_t := [\mathbf{A} | t\mathbf{G} - \tilde{\mathbf{A}}_2]$. Sample $\mathbf{e}_{\theta,2} \leftarrow D_{\Lambda_{\mathbf{y}_{\theta,2}}^\perp(\mathbf{A}_t), r}$, so that $-\mathbf{A}_t\mathbf{e}_{\theta,2} = \mathbf{y}_{\theta,2}$.
 2. Output $\text{KU}_t := \{(\theta, \mathbf{e}_{\theta,2})\}_{\theta \in \text{KUNodes}(\text{BT}, \text{RL}, t)}$.
- $\text{DK}_{\text{id}, t/\perp} \leftarrow \mathbf{DecKeyGen}(\text{PP}, \text{SK}_{\text{id}}, \text{KU}_t)$: On input public parameters PP, a private key $\text{SK}_{\text{id}} := \{(i, \mathbf{e}_{i,1})\}_{i \in I}$ and update key $\text{KU}_t := \{(j, \mathbf{e}_{j,2})\}_{j \in J}$ for some sets of nodes I and J, it runs the following steps:

⁵ Scalar Multiplication $R \times \mathbb{G} \rightarrow \mathbb{G}$ is defined by identifying each $\mathbf{a} = (a_0, \dots, a_{n-1})^t \in \mathbb{G}$ with the polynomial $a(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} \in R$, multiplying in R , and then mapping back to \mathbb{G} .

⁶ Due to limit page, we omit the description of KUNodes. Refer to the concrete algorithm in [4]. It takes as input a binary-tree BT, a revocation list RL and time t, and outputs a set of nodes.

1. $\forall(i, \mathbf{e}_{i,1}) \in \text{SK}_{\text{id}}, \forall(j, \mathbf{e}_{j,2}) \in \text{KU}_t$, if $\exists(i, j)$ s.t. $i = j$, $\text{DK}_{\text{id},t} \leftarrow (\mathbf{e}_{i,1}, \mathbf{e}_{j,2})$, else if SK_{id} and KU_t do not have any node in common, $\text{DK}_{\text{id},t} \leftarrow \perp$.
 2. Output $\text{DK}_{\text{id},t} := \{(\mathbf{e}_{i,1}, \mathbf{e}_{i,2})\}_{i \in I \cap J}$.
- $\text{CT}_{\text{id},t} \leftarrow \text{Enc}(\text{PP}, \text{id}, t, \mu)$: On input public parameters PP, an identity $\text{id} \in U \setminus \{0\}$, a time $t \in U \setminus \{0\}$, and a message $\mu \in \mathbb{Z}_p$:
1. Set $\mathbf{F}_{\text{id},t} := [\mathbf{A} | (\text{id})\mathbf{G} - \tilde{\mathbf{A}}_1 | t\mathbf{G} - \tilde{\mathbf{A}}_2] \in \mathbb{Z}_q^{n \times (m+2w)}$.
 2. Choose $\mathbf{x}_0 \leftarrow D_{\mathbb{Z},r}^n$, $\mathbf{x}_1^{(1)} \leftarrow D_{\mathbb{Z},r}^m$, $\mathbf{x}_1^{(2)} \leftarrow D_{\mathbb{Z},\gamma}^w$, $\mathbf{x}_2^{(2)} \leftarrow D_{\mathbb{Z},\gamma}^w$, $x_2 \leftarrow D_{\mathbb{Z},\gamma}$, and set $\mathbf{x}_1^t := [(\mathbf{x}_1^{(1)})^t | (\mathbf{x}_1^{(2)})^t | (\mathbf{x}_2^{(2)})^t] \in \mathbb{Z}_q^{1 \times (m+2w)}$.
 3. Compute $c_0 \leftarrow \mathbf{x}_0^t \mathbf{y} + x_2 + p \cdot \mu \in \mathbb{Z}_q$, $\mathbf{c}_1 \leftarrow \mathbf{x}_0^t \mathbf{F}_{\text{id},t} + \mathbf{x}_1^t \in \mathbb{Z}_q^{1 \times (m+2w)}$.
 4. Output the ciphertext $\text{CT}_{\text{id},t} := (c_0, \mathbf{c}_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{1 \times (m+2w)}$.
- $m \leftarrow \text{Dec}(\text{PP}, \text{DK}_{\text{id},t}, \text{CT}_{\text{id},t})$: On input public parameters PP, a decryption key $\text{DK}_{\text{id},t} := (\mathbf{e}_1, \mathbf{e}_2)$, and a ciphertext $\text{CT}_{\text{id},t} := (c_0, \mathbf{c}_1)$, it runs the following steps:
1. Parse \mathbf{c}_1 as $[\mathbf{c}_{1,0} | \mathbf{c}_{1,1} | \mathbf{c}_{1,2}] \in \mathbb{Z}_q^{1 \times m} \times \mathbb{Z}_q^{1 \times w} \times \mathbb{Z}_q^{1 \times w}$.
 2. Compute $\mu' \leftarrow c_0 + [\mathbf{c}_{1,0} | \mathbf{c}_{1,1}] \mathbf{e}_1 + [\mathbf{c}_{1,0} | \mathbf{c}_{1,2}] \mathbf{e}_2$.
 3. Output $\mu \in \mathbb{Z}_p$ s.t. μ' is closest to $p \cdot \mu \bmod q$.
- $\text{RL} \leftarrow \text{KeyRev}(\text{id}, t, \text{RL}, \text{ST})$: On input an identity id , a time t , a revocation list RL and a state ST, the algorithm adds (id, t) to RL for all nodes associated with identity id and returns RL.

5.2 Correctness and Security

We can set the parameters as follows: $m = \Theta(n \log q)$, Gaussian parameter r needs to be large enough for Gaussian sampling i.e. $r \geq \max\{s_1(\mathbf{R}_1), s_1(\mathbf{R}_2)\} \cdot \omega(\sqrt{\log n}) = O(\sqrt{m} + \sqrt{w}) \cdot \omega(\sqrt{\log n})^2 = O(\sqrt{m}) \cdot \omega(\sqrt{\log n})^2$. On the other hand, the hardness of LWE requires $r \geq 2\sqrt{n}$. For the security proof, we need $\gamma' \geq O(m) \cdot r^2 \cdot \omega(\sqrt{\log n})$ and let $\gamma = \sqrt{r^2 + 2\gamma'^2}$, $p = \gamma \cdot \text{poly}(n)$ for a sufficiently large $\text{poly}(n)$ term to ensure correctness and modulus $q = p^2$. Now we prove the correctness.

$$\begin{aligned} \mu' &= c_0 + [\mathbf{c}_{1,0} | \mathbf{c}_{1,1}] \mathbf{e}_1 + [\mathbf{c}_{1,0} | \mathbf{c}_{1,2}] \mathbf{e}_2 \\ &= p \cdot \mu + \mathbf{x}_0^t \mathbf{z}_0 + x_2 + [(\mathbf{x}_1^{(1)})^t | (\mathbf{x}_1^{(2)})^t] \mathbf{e}_1 + [(\mathbf{x}_1^{(1)})^t | (\mathbf{x}_2^{(2)})^t] \mathbf{e}_2. \end{aligned}$$

Thus, the decryption is correct if the error term $|\mathbf{x}_0^t \mathbf{z}_0 + x_2 + [(\mathbf{x}_1^{(1)})^t | (\mathbf{x}_1^{(2)})^t] \mathbf{e}_1 + [(\mathbf{x}_1^{(1)})^t | (\mathbf{x}_2^{(2)})^t] \mathbf{e}_2| < \frac{p}{2}$. By the Cauchy-Schwartz equality and Lemma 2, it holds except with negligible probability.

Theorem 3. The above RIBE scheme is KDM-sID-CPA secure with the decryption key with respect to the affine functions over \mathbb{Z}_p under the above parameters under the LWE assumption.

Remark 2. Our scheme is only 1-KDM-sID-CPA secure and it seems hard to prove the d -KDM using our proof strategy. In our simulation of Type I, we generate the public parameter \mathbf{y} by setting $\mathbf{y} = \mathbf{y}_{\text{id}^*} + \mathbf{y}_{\mathbf{t}^*}$ with \mathbf{y}_{id^*} obtained from the challenger and $\mathbf{y}_{\mathbf{t}^*}$ from our computation. However, when considering the d -KDM security, it needs $\mathbf{y}_{\text{id}_1^*}, \dots, \mathbf{y}_{\text{id}_d^*}$ for the different private keys, which seems hard for us to generate the \mathbf{y} , thus making our proof fail.

6 KDM Security for IBE

In this section, we present an efficient KDM-ID-CPA secure IBE scheme in the random oracle model, which can be used as a component of our generic construction of KDM-secure RIBE in Sect. 4.

6.1 KDM-IBE Scheme

The plaintext space is \mathbb{Z}_p . The secret key clique size of scheme is d , the parameter $w = n \lceil \log q \rceil$, and the random oracle is $H : U \setminus \{0\} \rightarrow \mathbb{Z}_q^n$, where $U \setminus \{0\}$ is the identity space defined as before. The concrete construction is as follows:

- $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^n, d)$: On input the security parameter n and secret key clique size d , perform the following steps:
 1. Sample $\mathbf{R} \leftarrow D_{\mathbb{Z}, \omega(\sqrt{\log n})}^{m \times w}$. Choose a uniform random matrix $\mathbf{A} \xleftarrow{r} \mathbb{Z}_q^{n \times m}$, and let $\tilde{\mathbf{A}} = -\mathbf{A}\mathbf{R} \in \mathbb{Z}_q^{n \times w}$.
 2. The public parameters is $\text{PP} = \{\mathbf{A}, \tilde{\mathbf{A}}\}$. The master secret key is $\text{MSK} = \mathbf{R}$.
- $\text{SK}_{\text{id}} \leftarrow \text{Ext}(\text{PP}, \text{MSK}, \text{id})$: On input the public parameters PP , identity $\text{id} \in U \setminus \{0\}$, and master secret key MSK , it performs the following steps:
 1. Set $\mathbf{A}_{\text{id}} = [\mathbf{A} | (\text{id})\mathbf{G} + \tilde{\mathbf{A}}] \in \mathbb{Z}_q^{n \times (m+w)}$ and $\mathbf{y}_{\text{id}} = H(\text{id}) \in \mathbb{Z}_q^n$.
 2. Sample $\mathbf{z}_0 \leftarrow D_{\mathbb{Z}, r}^n$, $\mathbf{z}_1 \leftarrow D_{\Lambda_{\mathbf{z}_0 - \mathbf{y}_{\text{id}}}^{\perp}(\mathbf{A}_{\text{id}}), \sigma}^{m+w}$ such that $\mathbf{y}_{\text{id}} = \mathbf{z}_0 - \mathbf{A}_{\text{id}}\mathbf{z}_1$.
 3. Output $\text{SK}_{\text{id}} := \mathbf{z}_1$.
- $\text{CT} \leftarrow \text{Enc}(\text{PP}, \text{id}, \mu)$: On input the public parameters PP , identity $\text{id} \in U \setminus \{0\}$ and message $\mu \in \mathbb{Z}_p$, perform the following steps:
 1. Let $\mathbf{A}_{\text{id}} = [\mathbf{A} | (\text{id})\mathbf{G} + \tilde{\mathbf{A}}]$ and $\mathbf{y}_{\text{id}} = H(\text{id}) \in \mathbb{Z}_q^n$.
 2. Choose $\mathbf{x}_0 \leftarrow D_{\mathbb{Z}, r}^n$, $\mathbf{x}_1 \leftarrow D_{\mathbb{Z}, r}^{m+w}$ and $x_2 \leftarrow D_{\mathbb{Z}, r}$. Compute $\mathbf{c}^t = \mathbf{x}_0^t [\mathbf{A}_{\text{id}} | \mathbf{y}_{\text{id}}] + [\mathbf{x}_1^t | x_2] + [\mathbf{0} | p \cdot \mu] \in \mathbb{Z}_q^{1 \times (m+w+1)}$.
 3. Output $\text{CT} := \mathbf{c}^t$.
- $\mu \leftarrow \text{Dec}(\text{PP}, \text{SK}_{\text{id}}, \text{CT})$: On input the public parameters PP , private key SK_{id} and ciphertext CT , perform the following steps:
 1. Compute $\mu' = \mathbf{c}^t \begin{bmatrix} \mathbf{z}_1 \\ 1 \end{bmatrix} \in \mathbb{Z}_q$.
 2. Output the $\mu \in \{0, \dots, p-1\} = \mathbb{Z}_p$ such that μ' is closest to $p \cdot \mu \pmod q$.

By Lemma 2, we set $m = \Theta(n \log q)$, and Gaussian parameter $r \geq 2\sqrt{n}$ to satisfy the reductions from LWE to worst-case lattice problems [17] and $\sigma \geq s_1(\mathbf{R}) \cdot \omega(\sqrt{\log n}) \geq O(\sqrt{m}) \cdot \omega(\sqrt{\log n})^2$ and set $\sigma = r \cdot O(\sqrt{m+w}) \cdot \omega(\sqrt{\log n})$. For correctness, we let message space size $p \geq \sigma^2(n+m+w) \cdot \omega(\sqrt{\log n})$ and $q = p^2$.

6.2 Security of KDM-IBE Scheme

We analyze the security in both the classical random oracle and quantum random oracle model.

Theorem 4. The IBE system described above is d -KDM-ID-CPA secure with the affine functions over \mathbb{Z}_p for the arbitrary constant d in the classical random oracle model under the LWE assumption.

In the quantum random oracle model (QROM), similar to [22], replacing the random oracle with a semi-constant distribution SC_λ ⁷, we can get the 1-KDM security in the QROM. However, when extending to the d -KDM security in the QROM, such technique fails. The main obstacle is how to embed d challenges simultaneously. When plugging d challenges to the random oracle queries like [22], the adversary may detect its non-randomness. In details, we can define a d -leveled semi-constant distribution $SC_{d,\lambda}$, which makes d patches with d constant values with some probability and others are random values. By Corollary 4.3 in [22], there is at most a distance $\frac{8}{3}dq_H^4\lambda^2$ between $SC_{d,\lambda}$ and the true random oracle. However, in the security proof, compared to the adversary's advantage $O(\lambda^d)$, this distance seems too large. Thus, it makes the original proof strategy fail. Besides, Katsumata et al. [13] provided a tighter security reduction for GPV-IBE in QROM by programming the random oracle the same way for all identities. However, it seems not easy to apply their techniques to our setting. We leave the d -KDM security in the quantum world as the further work.

Acknowledgement. The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported in part by National Natural Science Foundation of China (Nos. 61632020, 61472416, 61772520), Key Research Project of Zhejiang Province (No. 2017C01062), Fundamental Theory and Cutting-Edge Technology Research Program of Institute of Information Engineering, CAS (No. Y7Z0321102).

References

1. Alperin-Sheriff, J., Peikert, C.: Circular and KDM security for identity-based encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 334–352. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_20
2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_35

⁷ The definition of semi-constant distribution in [22] plays an essential role in the security proof of GPV IBE in the QROM, which makes a random value inserted into a small but significant fraction of oracle inputs. When replacing the oracle with the semi-constant distribution, the adversary can use the challenge with significant probability, which cannot be detected by the quantum adversary.

3. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36492-7_6
4. Boldyreva, A., Goyal, V., Kumar, A.: Identity-based encryption with efficient revocation. In: Ning, P., Syverson, P.E., Jha, S. (eds.) CCS 2008. ACM, New York (2008)
5. 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
6. 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
7. Brakerski, Z., Goldwasser, S., Kalai, Y.T.: Black-box circular-secure encryption beyond affine functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 201–218. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_13
8. Chen, J., Lim, H.W., Ling, S., Wang, H., Nguyen, K.: Revocable identity-based encryption from lattices. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 390–403. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31448-3_29
9. Chen, Y., Zhang, J., Deng, Y., Chang, J.: KDM security for identity-based encryption: Constructions and separations. IACR Cryptology ePrint Archive 2016, 1020 (2016). <http://eprint.iacr.org/2016/1020>
10. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM (2008)
11. Katsumata, S., Matsuda, T., Takayasu, A.: Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance. IACR Cryptology ePrint Archive 2018, 420 (2018). <https://eprint.iacr.org/2018/420>
12. Katsumata, S., Yamada, S.: Partitioning via non-linear polynomial functions: more compact IBEs from ideal lattices and bilinear maps. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 682–712. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_23
13. Katsumata, S., Yamada, S., Yamakawa, T.: Tighter security proofs for GPV-IBE in the quantum random oracle model. IACR Cryptology ePrint Archive 2018, 451 (2018). <https://eprint.iacr.org/2018/451>
14. Kitagawa, F., Tanaka, K.: Key dependent message security and receiver selective opening security for identity-based encryption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10769, pp. 32–61. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_2
15. 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
16. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: 45th Symposium on Foundations of Computer Science (FOCS 2004), pp. 372–381. IEEE Computer Society (2004)
17. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM (2005)

18. 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
19. 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
20. Takayasu, A., Watanabe, Y.: Lattice-based revocable identity-based encryption with bounded decryption key exposure resistance. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 184–204. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_10
21. Wee, H.: KDM-security via homomorphic smooth projective hashing. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 159–179. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_7
22. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_44

Full Paper Session IX: Verifiable Storage and Computing



Publicly Verifiable Data Transfer and Deletion Scheme for Cloud Storage

Changsong Yang¹(✉), Jianfeng Wang¹, Xiaoling Tao², and Xiaofeng Chen¹

¹ State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710071, China

cseyang020163.com, {jfwang,xfchen}@xidian.edu.cn

² Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin 541004, China

txl@guet.edu.cn

Abstract. As one of the most important services in the cloud computing, cloud storage service can provide boundless storage resources for clients. Because of its tremendous advantages, cloud storage service has attracted widespread attentions and plenty of clients. By employing cloud storage service, the data owners can migrate the heavy overhead of maintaining their data to the cloud server. However, upon uploading their data to the cloud server, the data owners will lose the direct control over their data, therefore, the computations over the data will be performed by the cloud server, for example, data transfer and deletion. However, during the data transfer and deletion processes, the cloud server may not perform these operations honestly for economic interests. That will make data transfer and deletion become new security challenges. In this paper, we propose a novel data transfer and deletion scheme that supporting public verifiability. In our construction, if the cloud server is malicious, the data owners can detect the cloud server's unfaithful data transfer or dishonest data deletion by verifying the returned proofs efficiently. Moreover, with the utilization of vector commitment, our novel scheme can realize public verifiability without any trusted third party.

Keywords: Cloud storage · Public verifiability
Data deletion · Data transfer · Vector commitment

1 Introduction

Cloud computing is a newly-developing computing paradigm, which connects large-scale storage resources, computing resources and network resources together through Internet [10, 26]. With its enormous resources, cloud computing can ubiquitously and conveniently offer on-demand self-service, such as verifiable databases service [4, 6] and outsourcing computing service [5, 7, 8]. Cloud storage service can provide unstinted storage resources for resource-constraint clients. Thanks to the attractive advantages, cloud storage service has been widely

accepted and applied by the public. For saving overhead of maintaining the data, an increasing number of clients, including individuals and corporations are willing to outsource their data to the cloud server.

However, how to transfer and delete data securely have become two problems in cloud storage. According to the investigation of Nskinc [15], 82% organizations profit from cloud storage service for saving human resources and IT resources. The report of Cisco [9] shows that there will be nearly 2 billion tenants embrace cloud storage service by the end of 2018, and the data traffic is predicted to grow 19.3 Exabytes per year. Particularly, 9% of the total cloud data traffic will be the traffic between different cloud servers by the end of 2018, up from about 7% at the end of 2013. To protect the data during the transfer process, Cloudsfer [16] utilized encryption technique to design an app. However, the app can only resist the external attackers during the data transfer, it cannot guarantee the integrity of the transferred data. To solve this problem, Ni et al. [17] proposed a provable data migration protocol with integrity verification. However, their scheme is not efficient because they utilize proxy-encryption to delete transferred data.

Besides, how to delete the cloud data permanently is a hot topic in both the academic community and industrial circle. Unlinking is a traditional data deletion method, which deletes the link of the file, and returns a one-bit outcome. This method can delete the file's link quickly, however, the file's contents still remain on the physical medium. The attackers can recover the deleted data by scanning the medium with a forensic tool [12]. Then plenty of researchers suggest that it should realize deletion by overwriting the physical medium with random data [1, 11, 13]. However, overwriting only makes the recovery more difficult, and the attackers can recover the overwritten data with the help of physical remanence. Moreover, overwriting is not efficient for real-world applications, especially for distributed storage system. To make the deletion operation more efficient, Boneh and Lipton [2] firstly proposed a cryptography-based solution. In their scheme, they destroy the decryption key to reach data deletion, and with plenty of follow-up works [18, 21, 22, 28]. Although deleting by cryptography is efficient, lots of the existing schemes do not support public verification.

Although a series of data transfer and deletion schemes have been proposed, most of them have some inherent limitations. Firstly, most of the existing schemes can not satisfy the property of verifiability, therefore, the data owner has to trust the returned result. Secondly, although some schemes give the data owner the ability of verifying the deletion result, most of these schemes introduce a trusted third party. However, it is particularly difficult to find such a trusted third party in real-world. To the best of our knowledge, it seems that there is no research work on efficient data transfer and deletion schemes that support public verification without any trusted third party in the malicious server model. Therefore, we put forward a vector commitment-based publicly verifiable data transfer and deletion scheme, which can reach public verification without any trusted third party.

1.1 Our Contribution

In this paper, we propose a vector commitment-based publicly verifiable data transfer and deletion scheme. The main contributions of our paper are as follows:

- We propose a novel vector commitment-based publicly verifiable data deletion scheme, which can realize provable data transfer simultaneously. During the data transfer and deletion processes, if the original cloud server S_1 does not migrate or delete the data honestly, our scheme can enable the data owner O to detect the malicious behaviors by verifying the returned proofs efficiently.
- We introduce the primitive of vector commitment to solve the problem of public verification in the secure data transfer and deletion scheme. Taking the advantages of vector commitment, the proposed scheme can realize public verification without any trusted third party, which is different from the previous schemes. Moreover, our new construction is also very efficient in computation as well as communication.

1.2 Related Work

The problem of verifiable data deletion has been studied for a long time, and a series of methods have been proposed, such as unlinking. Although unlinking is an efficient deletion method, the contents of the file still remain on the storage medium [12]. Therefore, it is significant to design more secure and efficient schemes for cloud data deletion.

In 2010, Paul and Saxena [19] proposed a verifiable data deletion scheme, which called “Proof of Erasability” (PoE). Besides, to delete data from the embedded devices, Perito and Tsudik [20] proposed a similar scheme, which called “Proofs of Secure Erasure” (PoSE-s). Both of the two schemes overwrite the disk with random patterns, and return the same patterns as a deletion proof. In 2016, Hao et al. [14] proposed a Trusted Platform Module (TPM)-based publicly verifiable data deletion scheme. They combine Chaum-Pedersen Zero Knowledge Proof with Diffie-Hellman encryption protocol to realize data confidentiality and data provable deletion. In 2018, Yang et al. [25] proposed a Blockchain-based publicly verifiable data deletion scheme. In their scheme, they use Blockchain to reach public verification without any trusted third party.

Besides, to realize verifiable data transfer and deletion simultaneously, Yu et al. [27] proposed a provable data transfer scheme. In their scheme, they delete the transferred data by revoking the decryption key, and verify the integrity of the transferred data on the new cloud through provable data possession scheme. In 2017, Xue et al. [24] proposed a provable data transfer protocol, which can enable the data owner to migrate the outsourced data between different cloud servers, and verify the data integrity on the new cloud. Finally, the original cloud server deletes the transferred data and returns a deletion proof. In 2018, Wang et al. [23] proposed a similar scheme for secure outsourced data transfer and deletion. They introduce homomorphic encryption and homomorphic authenticator to realize verifiable deletion and proof data possession.

1.3 Organization

The rest of this paper is organized as follows. In Sect. 2, some preliminaries are presented. We describe the system model, security threats and the design goals of our scheme in Sect. 3. Then we present the novel publicly verifiable data transfer and deletion scheme in Sect. 4. The analysis of the proposed scheme and the comparison with an existing scheme are given in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Preliminaries

In this section, we introduce some preliminaries, including bilinear pairings, the Computational Diffie-Hellman problem, and the vector commitment.

2.1 Bilinear Pairings

Let \mathbb{G}_1 and \mathbb{G}_2 be two cyclic multiplicative groups, whose orders both are the prime p , and g is a generator of \mathbb{G}_1 . A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following three properties:

- *Bilinear*: For all $P, Q \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_p^*$, $e(P^a, Q^b) = e(P, Q)^{ab}$.
- *Non-degenerate*: $e(g, g) \neq 1$.
- *Computable*: For all $P, Q \in \mathbb{G}_1$, there always exists an algorithm which can compute $e(P, Q)$ efficiently.

2.2 The Computational Diffie-Hellman Problem

The definitions of the Computational Diffie-Hellman (CDH) problem in \mathbb{G}_1 would be described as follows: on input a triple (g, g^x, g^y) and then output g^{xy} , where x and y are both chosen from \mathbb{Z}_p randomly. For all of the security parameter k , if for every probabilistic polynomial time (PPT) algorithm \mathcal{A} , there exists such a negligible function $negl(\cdot)$ that $Pr[\mathcal{A}(1^k, g, g^x, g^y) = g^{xy} \leq negl(k)]$, we say that the Computational Diffie-Hellman assumption holds.

2.3 Vector Commitments

Commitment is a fundamental primitive in cryptography, and it plays an important role in many security protocols, for example, digital voting, verifiable database, zero-knowledge proof, and so on. A commitment scheme can be seen as the digital equivalent of a sealed envelope: the sender puts a message m in the sealed envelope, and then sends it to the receiver. Subsequently, only the sender can open the sealed envelope to reveal the message m , which is called *hiding*. Besides, the sender can not change the committed message m anymore, which is called *binding*.

In 2013, Catalano and Fiore [3] presented a novel primitive of Vector Commitment (VC), which is very closely related to zero-knowledge set. Intuitively,

a vector commitment protocol can allow to commit to an ordered sequence of message (m_1, \dots, m_q) in a special way: upon committing, the committer is able to open the commitment at specific positions. Furthermore, no one can open the commitment to two different values at the same position, which is called *position binding*. Besides, for a given commitment, although have known some openings at the related positions, only the committer is able to distinguish the commitment is created to the message $m = (m_1, \dots, m_q)$ or to $m' = (m'_1, \dots, m'_q)$, which is called *hiding*. And we describe the formal definition of vector commitment as follows:

- $VC.KeyGen(1^k, q)$. The key generation algorithm takes the security parameter k and the size of the committed vector $q = poly(k)$ as inputs, and then outputs some public parameters pp , which also define the message space \mathbb{M} implicitly.
- $VC.Com_{pp}(m_1, \dots, m_q)$. On input the public parameters pp and a sequence of messages $(m_1, \dots, m_q) \in \mathbb{M}^q$, the committing algorithm outputs a commitment π and an auxiliary information aux .
- $VC.Open_{pp}(m, i, aux)$. The committer runs this algorithm to generate a proof λ_i , which can prove that m is the i th committed message.
- $VC.Ver_{pp}(\pi, m, i, \lambda_i)$. Only if λ_i is a valid proof that π is a commitment to the messages (m_1, \dots, m_q) such that $m = m_i$ the verification algorithm will output 1.
- $VC.Update_{pp}(\pi, m, i, m')$. The original committer runs this algorithm to update π by changing the i th message to m' . On input the old message m at the position i , the new message m' , the update algorithm outputs a new commitment π' and an update information U .
- $VC.ProofUpdate_{pp}(\pi, U, m', i, \lambda_i)$. Any user who holds a proof λ_i for the message at the position j w.r.t π can run this algorithm. It allows the user to compute an updated proof λ'_i (and the updated commitment π') such that λ'_i is valid w.r.t π' which contains m' as the new message at the position i . Basically, the value U contains the update information which is needed to compute such values.

3 Problem Statement

3.1 System Model

The system model of the verifiable data transfer and deletion involves three entities: the data owner O , an original cloud server S_1 , and a new target cloud server S_2 , as illustrated in Fig. 1.

- **Data owner** O . The data owner O is an entity who has limited resources. O is willing to outsource his data to the cloud server S_1 . Then, O may transfer his data from the cloud server S_1 to a new cloud server S_2 . After that, O wants to delete the transferred data from S_1 and verify the results.

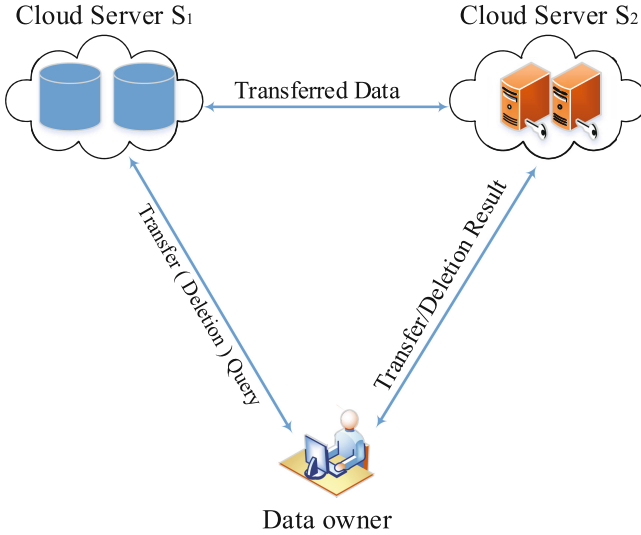


Fig. 1. The system model

- **Cloud Server S_1 .** The cloud server S_1 is an entity which has a lot of storage resources. And we define S_1 as the original cloud server, which may transfer the data to another cloud server S_2 , and then delete the transferred data. Finally, S_1 generates a proof to prove that it has performed honestly.
- **Cloud Server S_2 .** The cloud server S_2 is another entity which also has a good deal of storage space, and we define S_2 as the target cloud server, which may receive the transferred data from S_1 . After that, S_2 can generate a proof to persuade O that it has stored the transferred data honestly.

3.2 Security Threats

In our scheme, we mainly take the following three security threats into account.

- *Data privacy exposure.* Both the external attackers and the internal attackers may try to access the outsourced data to dig sensitive information. Besides, the cloud server may outsource the data to other subcontractors, or share the data with other corporators.
- *Dishonest data transfer.* During the data transfer process, S_1 may only migrate part of data for saving the bandwidth, or deliver some unrelated data to cheat O . Besides, the hacker may modify or delete some of the transferred data maliciously.
- *Malicious data reservation.* O wants to delete the transferred data from S_1 since they may contain some private information. However, S_1 may reserve the transferred data maliciously to dig the sensitive information.

3.3 Design Goals

In our scheme, we aim to migrate the data from S_1 to S_2 securely, and then delete the transferred data from S_1 permanently. Therefore, we should achieve the following three design goals.

- *Data confidentiality.* To protect the data confidentiality, both the external attackers and the internal attackers should be prevented from accessing to the outsourced data. That means, we should utilize cryptography tools to encrypt all the data before uploading them to the cloud server.
- *Provable data transfer.* To ensure the data can be transmitted from S_1 to S_2 intactly, O and S_2 should have the ability to verify the integrity of the transferred data. If the data are not intact, S_2 will detect and refuse to accept the transferred data, and then inform the data loss to the data owner.
- *Verifiable data deletion.* To guarantee all the transferred data have been deleted from S_1 , O should be able to verify the deletion result. If S_1 reserves the transferred data maliciously, it cannot forge an effective evidence to persuade O that the transferred data have been deleted.

4 Our Construction

4.1 The Concrete Construction

In the following, we describe our publicly verifiable data transfer and deletion scheme in detail. Firstly, we introduce some notations which are used later in our protocol. For simplicity, we can assume that O is a legal client of the cloud server S_1 and S_2 , and O has an identity number id_o which is only known by O . Then, let $H_1(\cdot)$ and $H_2(\cdot)$ be two secure one-way collision resistant hash functions. Moreover, we assume that each file's name is unique, and it is kept secret by O . In our scheme, we assume that O wants to outsource the file F , whose name is $fname$.

- *KeyGen.* Firstly, to generate the ECDSA key pairs (sk_o, pk_o) , (sk_{s_1}, pk_{s_1}) and (sk_{s_2}, pk_{s_2}) for O , S_1 and S_2 respectively. Secondly, O should run the algorithm $pp \leftarrow VC.KeyGen(1^k, q)$ to obtain the public parameters pp , where VC is a CDH assumption-based vector commitment scheme, and $pp = (g, \{h_i\}_{i \in [q]}, \{h\}_{i, j \in [q], i \neq j})$.
- *Encrypt.* To protect the privacy of the outsourced data, O encrypts F before uploading. Firstly, O divides F into q blocks (m_1, \dots, m_q) . Then for all $i = 1, \dots, q$, O encrypts the block m_i with the key k_i as $cip_i = Enc_{k_i}(m_i)$, where Enc is a symmetric encryption algorithm, and $k_i = H_1(fname, sk_o, i)$. Subsequently, O computes the file tag $tag_f = H_2(fname, sk_o)$ and the block tags $tag_{f_i} = H_2(cip_i, sk_o, i)$. Therefore, O obtains the final ciphertext $C_f = (tag_f, C_1, \dots, C_q)$, and then sends C_f to S_1 , where $C_i = (cip_i, tag_{f_i}, i)$. Finally, O sets $aux = (C_1, \dots, C_q)$, and runs $VC.Com_{pp}(C_1, \dots, C_q)$ to obtain a commitment π_f for F . Upon receiving the ciphertext C_f , S_1 stores the block C_i at the position i , and stores the file tag tag_f as the index of the ciphertext C_f .

- *StoreCheck*. After uploading, O wants to check the correctness of the storage. To make O verify conveniently, S_1 generates some storage proofs for O . For all $i = 1, \dots, q$, S_1 runs $\lambda_i \leftarrow VC.Open_{pp}(C_i, i, aux)$ to obtain the proofs $\lambda = (\lambda_1, \dots, \lambda_q)$, then S_1 sends λ to O . Upon receiving the proofs, O runs $x_i \leftarrow VC.Ver_{pp}(\pi_f, C_i, i, \lambda_i)$. If not all of the x_i are one, O thinks that S_1 does not store the data correctly; otherwise, O could believe that S_1 stores the data honestly. Subsequently, O deletes the local backups.
- *Decryption*. When O needs the file F , he should download the ciphertext from S_1 , and then decrypts it to obtain the plaintext. Therefore, the *Decryption* algorithm contains two steps: *Download* and *Decrypt*.
 - *Download*. To download the ciphertext blocks from S_1 , O firstly generates a set of block indices Φ , which identifies the blocks that are required to be downloaded. Subsequently, O computes a signature sig_d by $sig_d = Sig_{sk_o}(download, tag_f, T_d, \Phi)$, where sk_o is O 's private key, T_d is the current time, and Sig is an ECDSA signature algorithm. Finally, O generates a download request $R_d = (download, sig_d, tag_f, T_d, \Phi)$, and then O sends R_d to S_1 . Upon receiving R_d , S_1 parses it, if R_d is invalid, S_1 returns failure; otherwise, S_1 sends $\{C_i = (cip_i, tag_{fi}, i)\}_{i \in \Phi}$ to O .
 - *Decrypt*. Before decrypting, O should check the integrity of the ciphertext. For all $i \in \Phi$, O firstly checks that whether the equation $tag_{fi} \stackrel{?}{=} H_2(cip_i, sk_o, i)$ holds. If the equation does not hold, it means that the ciphertext cip_i is error, therefore, O requests S_1 to send C_i again; otherwise, O decrypts the ciphertext cip_i to obtain the plaintext m_i by $m_i = Dec_{k_i}(cip_i)$, where Dec is a symmetric decryption algorithm, and $k_i = H_1(fname, sk_o, i)$.
- *Transfer*. When O wants to change the cloud storage service provider, O will migrate some data blocks of the outsourced file F , or even the whole file from S_1 to S_2 .
 - *Migrate*. To transfer the data blocks, O should compute a transfer request R_t : O firstly generates a set of block indices Ψ , which defines the blocks that should be transferred. Subsequently, O computes a signature $sig_t = Sig_{sk_o}(transfer, tag_f, T_t, \Psi)$, where T_t is the timestamp. Finally, O generates the transfer request $R_t = (transfer, sig_t, tag_f, T_t, \Psi)$, and then O sends R_t to S_1 . Besides, O sends the commitment π_f to S_2 . Upon receiving R_t , S_1 verifies it, if R_t is invalid, S_1 returns failure; otherwise, S_1 computes a signature $sig_{ts} = Sig_{sk_{s_1}}(transfer, R_t)$ as the notarize of R_t . Then S_1 sends the signature sig_{ts} to S_2 , along with the transferred data blocks $\{C_i\}_{i \in \Psi}$ and the proofs $\{\lambda_i\}_{i \in \Psi}$.
 - *TranCheck*. After receiving the sig_{ts} and $\{C_i, \lambda_i\}_{i \in \Psi}$ from S_1 , S_2 verifies the signature sig_{ts} firstly. If the signature sig_{ts} is invalid, S_2 aborts and returns failure; otherwise, for all $i \in \Psi$, S_2 runs $x_i \leftarrow VC.Ver_{pp}(\pi_f, C_i, i, \lambda_i)$, if not all of the $x_i = 1$, S_2 aborts and returns failure; otherwise, S_2 prepares q storage spaces (b_1, \dots, b_q) . Subsequently, for all $i \in \Psi$, S_2 stores the block C_i at b_i , the other spaces $\{b_i\}_{i \notin \Psi}$ are set *NULL*. Then S_2 runs $VC.Com_{pp}(b_1, \dots, b_q)$ to obtain a new commitment

- π_{fs_2} , and sets the auxiliary information $aux_{s_2} = (b_1, \dots, b_q)$. Finally, S_2 returns π_{fs_2} to O , and informs O that the transfer is successful.
- *Deletion.* After the data blocks are transferred successfully, O is willing to delete the transferred data blocks from S_1 permanently, and then the transferred data blocks are stored on S_2 merely.
 - *DelRe.* To delete the transferred blocks, O should generate a deletion request R_e . O computes a signature $sig_e = Sig_{sk_o}(erasure, tag_f, T_e, \Psi)$ firstly, where T_e is the timestamp. Secondly, O generates a deletion request $R_e = (erasure, sig_e, tag_f, T_e, \Psi)$, and then O sends R_e to S_1 .
 - *Delete.* Upon receiving R_e , S_1 verifies it, if R_e is invalid, S_1 aborts and returns failure; otherwise, for all $i \in \Psi$, S_1 computes the signatures $sig_{ei} = Sig_{sk_{s_1}}(erasure, R_e, tag_f, i)$. After that, S_1 overwrites C_i with sig_{ei} to obtain a new auxiliary information aux' . Subsequently, S_1 takes aux' as input to generate a new commitment π'_f and proofs $\{\lambda'_i\}_{i \in \Psi}$. Finally, S_1 sends $\tau = (R_e, \pi'_f, \{i, \lambda'_i, sig_{ei}\}_{i \in \Psi}, \Psi)$ to S_2 .
 - *DelCheck.* Upon receipt of the evidence τ from S_1 , S_2 verifies R_e and $\{sig_{ei}, \lambda'_i\}_{i \in \Psi}$. If not all of the verifications pass, S_2 aborts and returns failure; otherwise, S_2 checks that whether the equation $\pi'_f \stackrel{?}{=} \pi_f \prod_{i \in \Psi} h_i^{sig_{ei} - C_i}$ holds. If the equation does not hold, S_2 aborts and returns failure; otherwise, S_2 returns $\tau = (R_e, \pi'_f, \{i, \lambda'_i, sig_{ei}\}_{i \in \Psi}, \Psi)$ to O and informs O that the deletion is performed successfully. Finally, anyone who has τ can verify the deletion result.

5 Analysis of Our Scheme

In the following section, we give an analysis of our scheme in detail. Firstly, we will analyze the security property of our scheme detailedly. Then we compare our scheme with a very recent work [14]. Finally, we would demonstrate the performance evaluation comparisons between our scheme and scheme [14].

5.1 Security Analysis

We give the security analysis of the proposed scheme in this subsection. As we described above, we assume that the original cloud server S_1 , the target cloud server S_2 and the data owner O all do not trust each other fully.

The Proposed Scheme Satisfies the Property of Correctness. If the two cloud servers S_1 and S_2 , and the data owner O are assumed to be honest, then the evidences are $\tau = (R_e, \pi'_f, \{i, \lambda'_i, sig_{ei}\}_{i \in \Psi}, \Psi)$. Firstly, note that the deletion request R_e contains a signature sig_e which is computed by O with sk_o as $sig_e = Sig_{sk_o}(erasure, tag_f, T_e, \Psi)$. Similarly, for all $i \in \Psi$, the signatures sig_{ei} are computed by S_1 as $sig_{ei} = Sig_{sk_{s_1}}(erasure, R_e, tag_f, i)$. Finally, the new commitment can be described as $\pi'_f = \pi_f \prod_{i \in \Psi} h_i^{sig_{ei} - C_i}$. As S_1 , S_2 and O all are assumed to be honest, therefore, the signatures sig_e and sig_{ei} , and the new

commitment π'_f all are valid. That is, in the *DelCheck* algorithm the evidences always can pass the verifications.

The Proposed Scheme Satisfies the Property of Data Integrity. In the *Decryption* phase, since O deletes all the local backups after uploading the file to S_1 . Therefore, O should download the ciphertext $C_i = (cip_i, tag_{fi}, i)$ from S_1 and decrypt it to obtain the plaintext. Upon receiving the ciphertext, O verifies that whether the equation $tag_{fi} \stackrel{?}{=} H_2(cip_i, sk_o, i)$ holds. Since sk_o is the private key of O , therefore, S_1 cannot falsify a cip'_i and a new tag'_{fi} to make the equation $tag'_{fi} = H_2(cip'_i, sk_o, i)$ hold. That is, if C_i is correct and integrated, the correct plaintext is always the output of the *Decryption* algorithm; otherwise, O can always detect the malicious operation of S_1 .

In the *Transfer* process, S_1 migrates $(sig_{ts}, \{C_i, \lambda_i\}_{i \in \Psi})$ to S_2 . On the one hand, since the vector commitment scheme satisfies the properties of *position binding* and *hiding*, therefore, S_1 cannot open the message C_i to another message C'_i at the position i . That is, the proofs $\{\lambda_i\}_{i \in \Psi}$ must be valid. Otherwise, the malicious behavior would be detected by S_2 . On the other hand, for all $i \in \Psi$, S_2 runs $x_i \leftarrow VC.Ver_{pp}(\pi_f, C_i, i, \lambda_i)$, and then checks x_i . Only if $\{C_i\}_{i \in \Psi}$ are intact and $\{\lambda_i\}_{i \in \Psi}$ are correct can the verifications pass. Otherwise, S_2 can detect the manipulation of the data blocks.

In a word, our verifiable data transfer and deletion scheme can guarantee the integrity of the outsourced data.

The Proposed Scheme Satisfies the Property of Public Verifiability. The proposed scheme can realize publicly verifiable cloud data deletion. Note that the original cloud server will generate the deletion evidences τ after deleting the data blocks from the physical medium, where $\tau = (R_e, \pi'_f, \{i, \lambda'_i, sig_{ei}\}_{i \in \Psi}, \Psi)$. Then anyone who owns the evidences τ (called verifier) can verify the result of the deletion operation conveniently and efficiently. Firstly, the verifier can check the deletion request $R_e = (erasure, sig_e, tag_f, T_e, \Psi)$. If the deletion request R_e is valid, it means that the data owner has required the original cloud server to delete the data blocks. Then for all $i \in \Psi$, the verifier can verify the signatures sig_{ei} which are computed by the original cloud server as $sig_{ei} = Sig_{sk_{s_1}}(erasure, R_e, tag_f, i)$. If the signatures $\{sig_{ei}\}_{i \in \Psi}$ are valid, the verifier will further verify the proofs $\{\lambda'_i\}_{i \in \Psi}$ by running the vector commitment verification algorithm as $x_i \leftarrow VC.Ver_{pp}(\pi'_f, sig_{ei}, i, \lambda'_i)$. Then the verifier checks that whether all the x_i are one. Only if all the verifications pass, the verifier trusts that the original cloud server performs the deletion operation honestly. Besides, as described above, all the verifications processes do not need any secret information. That is, anyone who owns τ can verify the deletion result. Therefore, the proposed scheme can realize publicly verifiable deletion.

The Proposed Scheme Satisfies the Property of Accountable Traceability. We analyze the accountable traceability in deletion phase when O is dishonest and S_1 is malicious respectively.

Dishonest Data Owner O. In the deletion phase, if O is dishonest, he may slander S_1 maliciously. On one hand, O has required S_1 to delete the data. However, O

denies his request and slanders that S_1 deletes the data arbitrarily. For this case, S_1 can present the deletion request $R_e = (erasure, sig_e, tag_f, T_e, \Psi)$, where sig_e is a signature which can be generated by O merely. Therefore, R_e can be seen as a proof that O has already asked S_1 to erase the data. On the other hand, O has not asked S_1 to delete the data. However, O slanders that he has required S_1 to delete the data and S_1 does not delete the data honestly. For this case, S_1 can ask O to present the signature sig_{es} , which is computed by S_1 with sk_{s_1} . This is, O cannot forge sig_{es} . Therefore, O cannot slander S_1 successfully.

Malicious Cloud Server S_1 . Similarly, S_1 may behave maliciously if it is dishonest. Firstly, O has never required to delete the data, while S_1 deletes the data arbitrarily. Then, S_1 declares that he executed the deletion as O 's command. In this scenario, O can ask S_1 to show the deletion request $R_e = (erasure, sig_e, tag_f, T_e, \Psi)$. However, S_1 does not have R_e . Besides, R_e contains a signature sig_e , which is computed by O with private key sk_o . Therefore, S_1 can not forge sig_e . That is, S_1 cannot present R_e to prove that O has required to delete the data. Secondly, S_1 may not execute the deletion honestly. Meanwhile, S_1 claims that O has never asked him to delete the data. Here, O can present the signature sig_{ei} , which can prove that not only O has asked S_1 to erase the data, but also S_1 has responded to the deletion command. Therefore, the malicious S_1 can not slander O successfully.

Table 1. Comparison between two schemes

Scheme	Hao et al. scheme [14]	Our scheme
Computational model	Amortized model	Amortized model
TTP	Yes	No
Public verifiability	Yes	Yes
Accountability	Yes	Yes
Computation (Encrypt)	$1\mathcal{M} + 2\mathcal{E} + 4\mathcal{H}$	$1\mathcal{E} + (2q + 1)\mathcal{H} + qE$
Computation (Decrypt)	$1\mathcal{E} + 1\mathcal{D} + 3\mathcal{H}$	$1\mathcal{S} + 1\mathcal{V} + 1\mathcal{D} + 2m\mathcal{H}$
Computation (Delete)	$1\mathcal{S} + 1\mathcal{V}$	$(l + 1)(\mathcal{S} + \mathcal{V}) + (q + l)E$

5.2 Comparison

We compare our novel data transfer and deletion scheme with a very recent data deletion scheme [14] in this subsection.

Firstly, both of our scheme and Hao et al. scheme [14] require some one-time computational efforts in the *KeyGen* phase. Secondly, although both of the two schemes can realize publicly verifiable data deletion, our novel scheme will not depend on any trusted third party, while Hao et al. scheme [14] needs a Trusted Platform Module (TPM). Although our scheme costs some more overhead to delete the data, the extra overhead is acceptable. Besides, our novel deletion scheme considers data transfer between different cloud servers. Furthermore, in

both of the two schemes, the most of the computations will be completed by the cloud server.

Table 1 presents the comparisons of our scheme and Hao et al. scheme [14]. We assume that the file is divided into q blocks in our scheme, and we denote by \mathcal{E} an *AES* encryption performance, resp., \mathcal{D} an *AES* decryption computation (particularly, other secure encryption algorithms can also be appropriate for our novel scheme), \mathcal{H} a hash calculation, M a multiplication in \mathbb{G}_1 (or \mathbb{G}_2), E an exponentiation in \mathbb{G}_1 , m the number of the downloaded data blocks in the *Decryption* phase, \mathcal{S} an ECDSA signature operation (resp., \mathcal{V} an operation for verifying an ECDSA signature). Besides, we define l the number of transferred blocks.

We can find that our scheme can realize publicly verifiable data transfer and deletion without any trusted third party. Compared with the scheme [14], our novel scheme is more efficient in *Encryption* process and *Decryption* phase. To delete l blocks, our scheme needs to generate $(l + 1)$ signatures and then verify these signatures, and it also needs to execute $(q + l)$ exponentiations. However, scheme [14] only needs to compute a signature and then verify the signature. Although our scheme needs a little more computation overhead in *Deletion*, it is a one-time operation, and the time cost in *Deletion* is completely acceptable. Therefore, our scheme is considerably efficient.

6 Conclusion

In this paper, we present a vector commitment-based publicly verifiable data deletion protocol, which supports secure data transfer between different cloud storage service providers simultaneously. In the cloud storage, the data owner O will not fully believe that the cloud server S_1 would transfer the data to the other cloud server S_2 and then delete the transferred data sincerely. In our scheme, we utilize vector commitment to enable the data owner O to become aware of the malicious behaviors when S_1 does not perform honestly during the transfer and deletion processes. Besides, the data owner O can verify the transfer result and the deletion outcome without any trusted third party.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (Nos. 61572382 and 61702401), China 111 Project (No. B16037), the National Cryptography Development Fund (No. MMJJ20180110), and the Natural Science Basic Research Plan in Shaanxi Province of China (Nos. 2016JZ021 and 2018JQ6001).

References

1. Bauer, S., Priyantha, N.: Secure data deletion for Linux file systems. In: Proceedings of the 10th Conference on USENIX Security Symposium, vol. 10, pp. 153–164. ACM, New York (2001)

2. Boneh, D., Lipton, R.: A revocable backup system. In: Proceedings of the 6th Conference on USENIX Security Symposium, vol. 6, pp. 91–96. ACM, New York (1996)
3. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_5
4. Chen, X.F., Li, J., Weng, J., Ma, J., Lou, W.: Verifiable computation over large database with incremental updates. *IEEE Trans. Comput.* **65**(10), 3184–3195 (2016)
5. Chen, X., Huang, X., Li, J., Ma, J., Lou, W., Wong, D.: New algorithms for secure outsourcing of large-scale systems of linear equations. *IEEE Trans. Inf. Forensics Secur.* **10**(1), 69–78 (2015)
6. Chen, X., Li, J., Huang, X., Ma, J., Lou, W.: New publicly verifiable databases with efficient updates. *IEEE Trans. Dependable Secure Comput.* **12**(5), 546–556 (2015)
7. Chen, X., Li, J., Huang, X., Li, J., Xiang, Y., Wong, D.: Secure outsourced attribute-based signatures. *IEEE Trans. Parallel Distrib. Syst.* **25**(12), 3285–3294 (2014)
8. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. *IEEE Trans. Parallel Distrib. Syst.* **25**(9), 2386–2396 (2014)
9. Cisco Global Cloud Index: Forecast and Methodology. <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/CloudIndexWhitePaper.html>
10. Dinh, H., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. *Wirel. Commun. Mob. Comput.* **13**(18), 1587–1611 (2013)
11. Diesburg, S., Wang, A.: A survey of confidential data storage and deletion methods. *ACM Comput. Surv.* **43**(1), 1–37 (2010)
12. Garfinkel, S., Shelat, A.: Remembrance of data passed: a study of disk sanitization practices. *IEEE Secur. Priv.* **99**(1), 17–27 (2003)
13. Hughes, G., Coughlin, T., Commins, D.: Disposal of disk and tape data by secure sanitization. *IEEE Secur. Priv.* **7**(4), 17–27 (2009)
14. Hao, F., Clarke, D., Zorzo, A.: Deleting secret data with public verifiability. *IEEE Trans. Dependable Secure Comput.* **13**(6), 617–629 (2016)
15. IT Consultants Insight on Business Technology, 7 Statistics You Didnt Know About Cloud Computing. <http://blog.nskinc.com/topic/Cloud-computing/IT-Services-Boston/7-Statistics-You-Didn-t-Know-About-Cloud-Computing>
16. Migrate & backup your files from any cloud to any cloud (2015)
17. Ni, J., Lin, X., Zhang, K., Yu, Y., Shen, X.: Secure outsourced data transfer with integrity verification in cloud storage. In: Proceedings of the 2016 IEEE/CIC International Conference on Communications in China, ICCIC 2016, pp. 1–6. IEEE Computer Society, Washington (2016)
18. Peterson, Z., Burns, R.: Ext3cow: a time-shifting file system for regulatory compliance. *ACM Trans. Storage* **1**(2), 190–212 (2005)
19. Paul, M., Saxena, A.: Proof of erasability for ensuring comprehensive data deletion in cloud computing. In: Meghanathan, N., Boumerdassi, S., Chaki, N., Nagamalai, D. (eds.) CNSA 2010. CCIS, vol. 89, pp. 340–348. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14478-3_35

20. Perito, D., Tsudik, G.: Secure code update for embedded devices via proofs of secure erasure. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 643–662. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15497-3_39
21. Reardon, J., Basin, D., Capkun, S.: SoK: secure data deletion. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP 2013, pp. 301–315. IEEE Computer Society, Washington (2013)
22. Tang, Y., Lee, P., Lui, J., Perlman, R.: Secure overlay cloud storage with access control and assured deletion. *IEEE Trans. Dependable Secure Comput.* **9**(6), 903–916 (2012)
23. Wang, Y., Tao, X., Ni, J., Yu, Y.: Data integrity checking with reliable data transfer for secure cloud storage. *Int. J. Web Grid Serv.* **14**(1), 106–121 (2018)
24. Xue, L., Ni, J., Li, Y., Shen, J.: Provable data transfer from provable data possession and deletion in cloud storage. *Comput. Stand. Interfaces* **54**, 46–54 (2017)
25. Yang, C., Chen, X., Xiang, Y.: Blockchain-based publicly verifiable data deletion scheme for cloud storage. *J. Netw. Comput. Appl.* **103**, 185–193 (2018)
26. Yang, C., Ye, J.: Secure and efficient fine-grained data access control scheme in cloud computing1. *J. High Speed Netw.* **21**(4), 259–271 (2015)
27. Yu, Y., Ni, J., Wu, W., Wang, Y.: Provable data possession supporting secure data transfer for cloud storage. In: Proceedings of the 10th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA 2015, pp. 38–42. IEEE Computer Society, Washington (2015)
28. Yuan, J., Yu, S.: Secure and constant cost public cloud storage auditing with deduplication. In: Proceedings of the 2013 IEEE Conference on Communications and Network Security, CNS 2013, pp. 145–153. IEEE Computer Society, Washington (2013)



Publicly Verifiable Static Proofs of Storage: A Novel Scheme and Efficiency Comparisons

Henning Kopp^(✉), Frank Kargl, and Christoph Bösch

Institute of Distributed Systems, Ulm University, Ulm, Germany
{henning.kopp, frank.kargl, christoph.boesch}@uni-ulm.de

Abstract. Proofs of storage are cryptographic primitives that enable a storage provider to prove that it honestly stores files of its users without tampering or deleting parts of them. The performance of publicly verifiable proofs of storage is not well understood and is mostly measured asymptotically in the literature. We propose and implement a novel publicly verifiable static proof of storage based on the RSA assumption, measure its computational performance, and compare it to other state of the art schemes. In our performance evaluation, our scheme outperforms existing schemes with similar security guarantees in the time taken to encode the file. In the other metrics its runtime is comparable to that of existing schemes. We consider our scheme together with our practical evaluations to be an important contribution to the application of cloud storage security mechanisms.

Keywords: Proof of storage · Remote data integrity check
Data outsourcing security · Applied cryptography

1 Introduction

A persistent trend in information technology is the outsourcing of storage capacities to external cloud providers. This creates the need to remotely check the integrity of stored files. Approaches which are used in local settings like the use of checksums fail when applied to a remote setting due to a potentially malicious storage provider. One mechanism to address this need for a remote file integrity check is a proof of storage. This allows a storage provider to cryptographically prove the possession of a stored file with sublinear complexity, i.e., without transferring the whole file. Types that exist are privately and publicly verifiable proofs of storage. In a privately verifiable proof of storage only the uploading party can verify the proof, whereas a publicly verifiable proof allows verification by any third party, e.g., an external auditor. In contrast to privately verifiable proofs of storage, publicly verifiable schemes have benefits in several use cases, due to the possibility of publishing the proof. Openly revealing publicly verifiable proofs of storage could improve the reputation of a storage provider,

since other parties could verify the proof and thus decide on its trustworthiness. Publicly verifiable proofs of storage can also be used to decentralize trust in a distributed file storage system, since each party can check if the others store files (cf. FileCoin [7] or KopperCoin [12]). Because of these unique applications we restrict our discussion to publicly verifiable proofs of storage.

Ateniese et al. [2] described how to obtain a publicly verifiable proof of storage protocol in the random oracle model where the communication complexity is independent of the file size using any homomorphic identification protocol. An identification protocol allows a prover to prove its identity to a verifier. The verifier \mathcal{V} in turn is unable to convince someone else that \mathcal{V} is the prover [6]. A *homomorphic* identification protocol additionally allows aggregation of multiple transcripts of different runs of the protocol without sacrificing security. Despite lots of research in identification protocols [6, 9, 16, 18, 21] there has not been much effort in applying these findings to construct publicly verifiable proofs of storage. A modification of the Shoup protocol [21] is used by Ateniese et al. [2] to construct a publicly verifiable proof of storage. There is also a publicly verifiable scheme by Shacham and Waters [19] which was proposed independently of the transformation but can be modified to fit in the framework of Ateniese et al., as we describe later. As a by-product, we obtain a novel unforgeable homomorphic identification protocol which may be of independent interest. In order to obtain a new publicly verifiable proof of storage we modify the Guillou-Quisquater protocol [9] to be secure in a stronger attacker model than the original one and apply the transformation by Ateniese et al. on it. Modifying the Guillou-Quisquater protocol is necessary, since the transformation of Ateniese et al. is only proven secure for identification protocols which are secure in the stronger attacker model. The result is a novel secure publicly verifiable proof of storage.

Recent work in proofs of storage mainly deals with privately verifiable proofs of storage since these use symmetric cryptography and thus are faster than publicly verifiable proofs of storage. Another line of work is to extend the features of proofs of storage. *Dynamic* proofs of storage allow for updating (insertion, modification, deletion) of chunks [4, 20] or even support revision control [25]. Research also covers *distributed* proofs of storage, where storage of multiple replicas of a file can be proven [5] or storage providers can prove that they cooperatively store a file, and each storage provider needs to store only parts of the file [26]. However, none of these is publicly verifiable.

One major shortcoming of the current state of proofs of storage is that practical performance is rarely compared. There are complexity discussions of privately verifiable proofs of storage by Xu and Chang [23] and Ateniese et al. [1], and of publicly verifiable proofs of storage [24]. But these are only asymptotic comparisons which do not show the real-world performance. Practical evaluations of proofs of storage [17, 22] only benchmark one scheme or variations of one scheme and thus are often incomparable due to differences in the hardware and the programming language used. Further, the schemes themselves are often incomparable, since they support different features like dynamic data or confidentiality of data with respect to the external auditor. Thus, we implement

and compare *only publicly verifiable static* proofs of storage. In particular, we examine our novel proof of storage scheme, together with the proof of storage based on the Shoup protocol [2], and the scheme by Shacham and Waters [19].

Contribution. Our contributions are as follows:

- We provide a modification of the Guillou-Quisquater (GQ) identification protocol and prove its unforgeability in a strictly stronger attacker model than the original GQ protocol.
- Based on our modified GQ identification protocol we propose a novel efficient publicly verifiable proof of storage scheme.
- We propose a novel homomorphic identification protocol based on the Diffie-Hellman assumption by reconstruction from a slightly modified version of the proof of storage by Shacham and Waters [19]. Additionally, a proof of security of the identification protocol is given. As a lemma we receive a proof of the modified scheme of Shacham and Waters.
- We implement the Shacham-Waters proof of storage [19], the proof of storage instantiated from the Shoup identification protocol as mentioned by Ateniese [2], and our novel proof of storage scheme. We provide comprehensive performance measurements for the three schemes we have implemented.

Roadmap. The next section provides the necessary definitions for our discussion. The main part of our article is Sect. 3 where we discuss the original GQ protocol, our modifications to it, and our novel proof of storage scheme. This section also contains a new identification scheme and discusses the other publicly verifiable proofs we have implemented and evaluated. Section 4 contains the benchmarks of our implementations. We conclude with Sect. 5.

Remark. In the literature there are the notions of proof of retrievability, proof of storage, as well as provable data possession which are used to describe similar but different concepts.

A proof of retrievability [10] is a challenge-response protocol that enables a cloud provider to demonstrate to a client that a file is retrievable, i.e., recoverable without any loss or corruption. Proofs of data possession [1] are related protocols that only detect a large amount of corruption in outsourced data. In a proof of data possession the existence of a knowledge extractor is required which can extract knowledge of the file, whereas a proof of storage additionally requires the knowledge extractor to be efficiently computable. In particular it needs to have expected polynomial runtime [2]. In the following we focus only on proofs of storage.

2 Preliminaries

In this section we introduce the notation used throughout the remainder of the paper as well as definitions for a homomorphic identification protocol and a proof of storage.

2.1 Notation

We write $a \leftarrow \mathcal{A}(x)$ to assign to a the output of running the randomized algorithm \mathcal{A} on input x . With $a \leftarrow \mathcal{A}(x; r)$ we denote the deterministic result of running \mathcal{A} on input x and the fixed randomness r . We say that an algorithm \mathcal{A} is ppt if it runs in probabilistic polynomial time. With \mathbb{Z}_p we denote the residue classes of the integers \mathbb{Z} modulo $p \in \mathbb{N}$. We write $x \in_R S$ if we choose an element x from a finite set S uniformly at random. Vectors are written in **bold** typeface.

We say that a function f is negligible if for all positive polynomials p there is a natural number $N \in \mathbb{N}$ such that for all $n > N$ it holds that $|f(n)| < 1/p(n)$. Throughout the text, $\mathbf{f} \in \mathbb{Z}_B^\ell$ with $B \in \mathbb{N}$ denotes a file viewed as a vector with chunks $f_1, \dots, f_\ell \in \mathbb{Z}_B$. The symbol φ denotes Euler's totient function, i.e., $\varphi(x) := |\mathbb{Z}_x^*|$.

2.2 Identification Protocol

The purpose of an identification protocol is that a prover \mathcal{P} who is in possession of a secret key sk can prove its identity to a verifier \mathcal{V} who knows the corresponding public key pk . The security property of an identification protocol is that a verifier \mathcal{V} is not able to prove to someone else that \mathcal{V} is the prover \mathcal{P} , i.e., that \mathcal{V} knows the private key sk [6].

The prover generates the first message α using his public key pk and a random value r . The verifier \mathcal{V} chooses a random challenge β in the challenge space of the identification protocol and gives it to \mathcal{P} . The prover \mathcal{P} computes a response γ by using the challenge β together with the private key sk and the randomness r used in generating the first message α . To verify the interaction \mathcal{V} needs to know the transcript α, β, γ , as well as the public key pk . We repeat the definitions of a homomorphic identification protocol by Ateniese et al. [2] in Definitions 1 and 2.

Definition 1 (Identification Protocol [2]). *An identification protocol is a three-move-protocol between a ppt prover \mathcal{P} and a ppt verifier \mathcal{V} . The protocol consists of four polynomial-time algorithms (Setup, Commit, Response, Verify) such that:*

1. $(pk, sk) \leftarrow \text{Setup}(1^k)$ is a probabilistic algorithm that takes as input the security parameter k and outputs a public and private key pair (pk, sk) .
2. $\alpha \leftarrow \text{Commit}(pk; r)$ is a probabilistic algorithm run by the prover \mathcal{P} to generate the first message. It takes as input the public key and random coins r , and outputs an initial message α . We stress that there is no need for the secret key sk .
3. $\gamma \leftarrow \text{Response}(pk, sk, r, \beta)$ is a probabilistic algorithm that is run by the prover \mathcal{P} to generate the third message. It takes as input the public key pk , the secret key sk , a random string r , and a challenge β from some associated challenge space, and outputs a response γ .

4. $b := \text{Verify}(pk, \alpha, \beta, \gamma)$ is a deterministic algorithm run by the verifier \mathcal{V} to decide whether to accept the interaction. It takes as input the public key pk , an initial message α , a challenge β , and a response γ . It outputs a single bit b , where ‘1’ indicates acceptance and ‘0’ indicates rejection.

We call an identification protocol correct if for all $k \in \mathbb{N}$, all (pk, sk) output by $\text{Setup}(1^k)$, all random coins r , and all β in the appropriate challenge space, it holds that

$$\text{Verify}\left(pk, \text{Commit}(pk; r), \beta, \text{Response}(pk, sk, r, \beta)\right) = 1.$$

For applying the transformation by Ateniese et al. on a proof of storage [2] we need homomorphic identification protocols. These are identification protocols where transcripts of several runs of the protocol can be aggregated. A verifier can then verify the aggregated transcripts. This amounts to batch verification of the transcripts of different runs of the protocol without sacrificing security.

Definition 2 (Homomorphic Identification Protocol [2]). An identification protocol $\Sigma = (\text{Setup}, \text{Commit}, \text{Response}, \text{Verify})$ is homomorphic if efficient functions Combine1 , Combine3 exist such that:

Completeness. For all (pk, sk) output by $\text{Setup}(1^k)$ and all coefficient vectors $\mathbf{s} \in \mathbb{Z}_{2^k}^n$, if transcripts $\{(\alpha_i, \beta_i, \gamma_i)\}_{1 \leq i \leq n}$ are such that $\text{Verify}(pk, \alpha_i, \beta_i, \gamma_i) = 1$ for all i , then:

$$\text{Verify}\left(pk, \text{Combine1}(\mathbf{s}, \boldsymbol{\alpha}), \sum_{i=1}^n s_i \beta_i, \text{Combine3}(\mathbf{s}, \boldsymbol{\gamma})\right) = 1$$

Unforgeability. Consider the following experiment involving an adversary \mathcal{A} :

1. The challenger computes $(pk, sk) \leftarrow \text{Setup}(1^k)$ and gives pk to \mathcal{A} .
2. The following is repeated a polynomial number of times:
 - \mathcal{A} outputs β' in the challenge space. The challenger chooses a random r , computes $\gamma \leftarrow \text{Response}(pk, sk, r, \beta')$ and gives (r, γ) to \mathcal{A} .
3. The adversary outputs an n -vector of challenges $\boldsymbol{\beta}$. Then for each i the challenger chooses r_i at random, sets $\alpha_i \leftarrow \text{Commit}(pk; r_i)$ and $\gamma_i \leftarrow \text{Response}(pk, sk, r_i, \beta_i)$, and gives $(\mathbf{r}, \boldsymbol{\gamma})$ to \mathcal{A} .
4. \mathcal{A} outputs a triple $(\mathbf{s}, \mu', \gamma')$, where $\mathbf{s} \in \mathbb{Z}_{2^k}^n$. The adversary succeeds if (1) $\mu' \neq \sum_i s_i \beta_i$ and (2) $\text{Verify}(pk, \text{Combine1}(\mathbf{s}, \boldsymbol{\alpha}), \mu', \gamma') = 1$.

Remark. The attacker has access to the randomness r underlying the messages and thus also to the messages since he can reconstruct $\boldsymbol{\alpha} = \text{Commit}(pk; r)$. This is a strictly stronger attacker model compared to an attacker who has only access to the messages. As an example, the well-known Schnorr identification protocol [18] is insecure in this attacker model but secure if the attacker has only access to the messages. In the Schnorr protocol the commit message consists of g^r and

thus the attacker is unable to know r only from the commit. Knowledge of the underlying randomness r allows to reconstruct the secret key using subsequent messages. Thus, the Schnorr identification protocol is insecure in the stronger attacker model of Ateniese et al. and cannot be used to construct a secure proof of storage. The same holds true for the Okamoto identification protocol [16]. It is difficult to find homomorphic identification protocols that satisfy this strictly stronger notion of unforgeability and consequently can be transformed to proof of storage schemes.

2.3 Proof of Storage

A proof of storage is a mechanism used by a prover \mathcal{P} to convince a verifier \mathcal{V} that \mathcal{P} stores a specific file \mathbf{f} . Since we consider only publicly verifiable proofs of storage the verifier \mathcal{V} and the user \mathcal{U} who is uploading the file \mathbf{f} may be distinct. First, \mathcal{U} encodes its file using its private key sk and sends the encoded file f' together with a state st which serves as an identifier for the file to the prover \mathcal{P} . As soon as the verifier \mathcal{V} wants to know if \mathcal{P} stores the file, \mathcal{V} generates a random challenge c and sends it to \mathcal{P} . \mathcal{P} generates a proof π by using the encoded file f' as well as the challenge c . To verify the proof the public key pk , the identifier of the file st , the challenge c , and the proof π itself is needed. Since the private key is not needed for the verification, the proof is called publicly verifiable.

Definition 3 (Proof of Storage [2]). A (publicly-verifiable) proof of storage is a tuple of four ppt algorithms (Gen, Encode, Prove, Verify) such that:

1. $(pk, sk) \leftarrow \text{Gen}(1^k)$ is a probabilistic algorithm that is run by the user \mathcal{U} to set up the scheme. It takes as input a security parameter k , and outputs a public and private key pair (pk, sk) . We assume that pk implicitly defines a positive integer B which determines the size of the file chunks.
2. $(f', st) \leftarrow \text{Encode}_{sk}(\mathbf{f})$ is a probabilistic algorithm that is run by the user in order to encode the file. It takes as input the secret key sk , and a file $\mathbf{f} \in \mathbb{Z}_B^\ell$ viewed as a vector of chunks with size B . It outputs an encoded file f' and state information st .
3. $\pi := \text{Prove}(pk, f', c)$ is a deterministic algorithm that takes as input the public key pk , an encoded file f' , and a challenge $c \in \{0, 1\}^k$. It outputs a proof π .
4. $b := \text{Verify}(pk, st, c, \pi)$ is a deterministic algorithm that takes as input the public key pk , the state st , a challenge $c \in \{0, 1\}^k$, and a proof π . It outputs a bit, where '1' indicates acceptance and '0' indicates rejection.

We require that for all $k \in \mathbb{N}$, all (pk, sk) output by $\text{Gen}(1^k)$, all $\mathbf{f} \in \mathbb{Z}_B^\ell$, all (f', st) output by $\text{Encode}_{sk}(\mathbf{f})$, and all $c \in \{0, 1\}^k$, it holds that

$$\text{Verify}(pk, st, c, \text{Prove}(pk, f', c)) = 1.$$

By using the transformation of Ateniese et al. [2] which transforms any homomorphic identification protocol to a proof of storage we can describe the intuition behind the algorithms more clearly. In $\text{Encode}_{sk}(\mathbf{f})$ the user \mathcal{U} splits the file into

ℓ chunks, each signed with a homomorphic signature scheme generated from the homomorphic identification protocol. We will refer to these signatures as authenticators. In the algorithm `Prove` the challenge c is expanded to a vector of dimension ℓ by setting $chal_i = \mathcal{H}_c(i)$, where \mathcal{H}_c is a pseudorandom function keyed with the challenge c . Each coefficient in the vector $chal$ corresponds to a chunk of the file. The proof consists of a linear combination τ of the chunks and a linear combination μ of the authenticators where the coefficients are from the expanded challenge $chal$. In particular, the size of the proof is constant and independent of the size of the file. The state st guarantees that the linear combination is computed over the correct file.

Remark. The proof of storage resulting from the transformation requires that ℓ , the number of chunks in the file is public. Otherwise it is not possible to expand the challenge c in the verification step to the correct length. In practice this is often undesired. We can remedy this by including the number of chunks ℓ in the state st as $st' = (st, \ell)$ and use st' instead of st as the key for the pseudorandom function. This way the verifier knows the number of chunks from st and is able to verify the proof without further knowledge. By including ℓ in the state st , the length ℓ cannot be modified since otherwise the authenticators on the chunks could not be verified.

A different approach to remedy that the number of chunks ℓ of the file needs to be public is to augment the proof of storage with a signature scheme. The algorithm `Gen`(1^k) additionally outputs a signing key pair. In the encoding step the number of chunks is signed by the user and included in the information needed to verify the proof of storage. In the verification step the signature is checked, in addition to the verification of the proof. If the signature does not validate it is rejected since the information of the number of chunks ℓ is not correct. This technique is used in the publicly verifiable proof of storage by Shacham and Waters [19].

3 Proofs of Storage

In this section we describe three proofs of storage: (i) our novel Guillou-Quisquater-based proof of storage, (ii) the Shoup proof of storage, and (iii) the Shacham-Waters proof of storage.

First, we modify the Guillou-Quisquater identification protocol to be secure in the stronger attacker model such that it can be used to construct a novel proof of storage. Next, we describe a proof of storage based on the Shoup protocol. Its existence was stated by Ateniese et al. [2] but the scheme was not explicitly described. We present this scheme to which we will refer as Shoup proof of storage in Sect. 3.2. The third and last scheme we explain is the scheme of Shacham and Waters [19] along with some modifications. These modifications enable a better comparison and allow us to reconstruct the underlying identification protocol which to our knowledge has not been described yet in literature and might be of independent interest.

3.1 Proof of Storage from Guillou-Quisquater

In 1988, Guillou and Quisquater (GQ) described an identification protocol based on the RSA problem. This identification protocol is proven secure under an attacker that has access to the messages but not to the underlying randomness r . In our attacker model from Definition 2 the protocol is insecure.

We have modified the original GQ protocol to be secure in our stronger attacker model as described in Definition 2. In particular we substituted the random r from the original protocol by r^d everywhere. Thus the commit in our modified scheme is only the randomness r instead of $r^e \in \mathbb{Z}_n$ as in the original protocol. The resulting scheme is shown in Fig. 1.

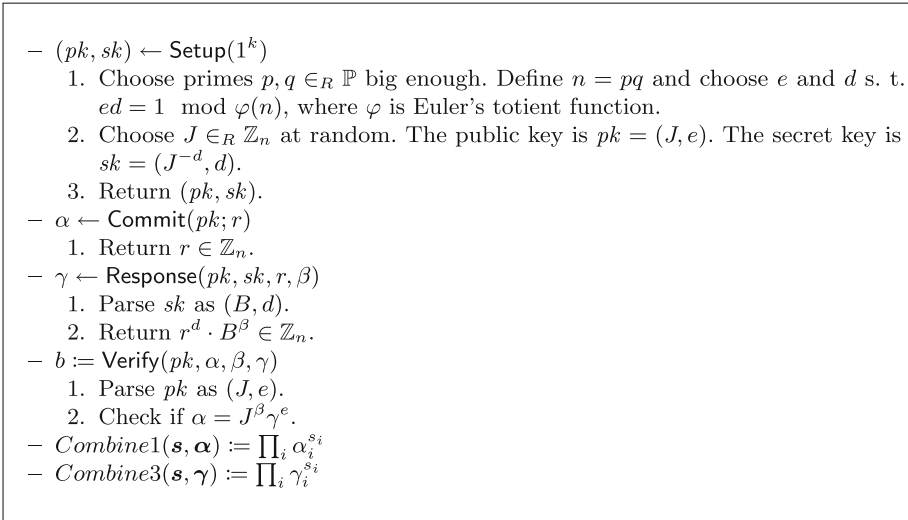


Fig. 1. Our modified Guillou-Quisquater identification protocol

The proof of security of the scheme can be found in Theorem 1 in the appendix. With the modified GQ identification protocol we are now able to construct a proof of storage using the transformation of Ateniese et al. [2]. The resulting scheme is depicted in Fig. 2.

Lemma 1. *The GQ proof of storage shown in Fig. 2 is secure under the RSA-assumption if \mathcal{H} is modelled as a random oracle.*

Proof. Follows directly from the security of the transformation (Theorems 1 and 2 in [2]) and the security of the modified GQ scheme in Theorem 1 in the appendix.

- $(pk, sk) \leftarrow \text{Gen}(1^k)$
 1. Choose primes $p, q \in_R \mathbb{P}$ big enough. Define $n = pq$ and choose e and d s. t. $ed = 1 \pmod{\varphi(n)}$, where φ is Euler's totient function.
 2. Choose $J \in_R \mathbb{Z}_n$ at random. The public key is $pk = (J, e)$. The secret key is $sk = (J^{-d}, d)$.
 3. Return (pk, sk) .
- $(f', st) \leftarrow \text{Encode}_{sk}(\mathbf{f})$
 1. Parse \mathbf{f} as f_1, \dots, f_ℓ with $f_i \in \mathbb{Z}_n$ for all $i = 1, \dots, \ell$.
 2. Parse sk as (J^{-d}, d) .
 3. Define the state $st = (s, \ell)$, where $s \in_R \{0, 1\}^k$ is chosen at random.
 4. Compute the authenticators for each chunk as $\sigma_i = \mathcal{H}_{st}(i)^d \cdot (J^{-d})^{f_i} \in \mathbb{Z}_n$ for $i = 1, \dots, \ell$, where $\mathcal{H}_{st}(i)$ is a pseudorandom function keyed with key st .
 5. Return $f' = (\mathbf{f}, \sigma)$ and st .
- $\pi := \text{Prove}(pk, f', c)$
 1. Expand the challenge c as $chal_i = \mathcal{H}_c(i)$ for $i = 1, \dots, n$.
 2. Parse f' as (\mathbf{f}, σ) and \mathbf{f} as f_1, \dots, f_ℓ .
 3. Aggregate the authenticators as $\tau = \prod_{i=1}^{\ell} \sigma_i^{chal_i}$.
 4. Aggregate the chunks as $\mu = \sum_{i=1}^{\ell} f_i \cdot chal_i$.
 5. Return $\pi = (\tau, \mu)$
- $b := \text{Verify}(pk, st, c, \pi)$
 1. Recover the number of chunks ℓ in the file from st .
 2. Parse the public key pk as (J, e) .
 3. Parse π as (τ, μ) .
 4. Check if $J^\mu \cdot \tau^e = \prod_{i=1}^{\ell} \mathcal{H}_{st}(i)^{\mathcal{H}_c(i)}$.

Fig. 2. Our GQ proof of storage

3.2 Shoup Proof of Storage

The Shoup identification protocol [21] is an unforgeable homomorphic identification protocol based on the factoring assumption for Blum integers. It is similar to the GQ protocol with the main difference that the secret key is a 2^{3m} -th root instead of an e -th root, where e is chosen such that it is invertible modulo $\varphi(n)$.

The proof of unforgeability in the homomorphic case is Theorem 3 in the paper of Ateniese et al. [2]. There, it is shown that this protocol can be used to obtain a proof of storage based on the factoring assumption. Since the resulting proof of storage from Shoup's identification protocol is not stated explicitly in their paper we present it in Fig. 3.

3.3 Shacham-Waters Proof of Storage

Shacham and Waters (SW) proposed two proof of storage schemes [19] where the resulting proofs are of constant size. One of the schemes provides private verifiability, whereas the other scheme is publicly verifiable. We are only interested in the scheme with public verifiability. Its proof of security can be found in their paper [19].

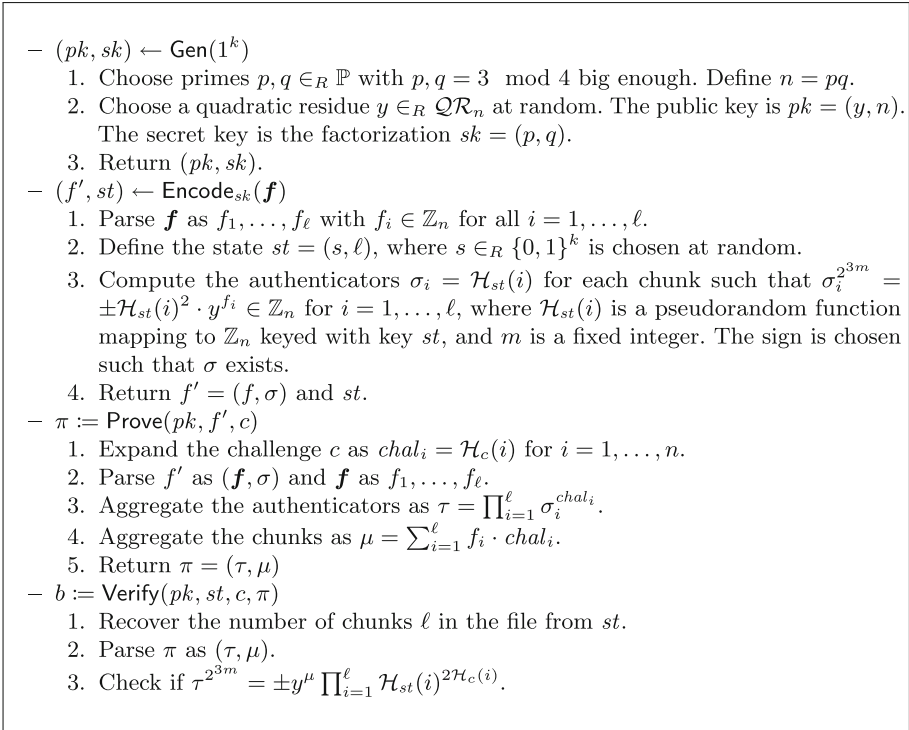


Fig. 3. Shoup’s proof of storage

The SW scheme supports a tuning parameter s which subdivides the chunks into smaller subchunks. In the terminology of Shacham and Waters these are called blocks and sectors. Each of the n blocks of the file f is s sectors long. If ℓ is the number of sectors, then $\ell = n \cdot s$. A larger value for s increases the size of the proofs by a factor of s . On the other hand this decreases the number of authenticators σ_i the prover \mathcal{P} needs to store by a factor of s .

The SW scheme was presented independently of the transformation from an identification protocol [2]. However as hinted by Ateniese et al. [2] it can be slightly modified to be a result of this transformation. We modified the original scheme in order to increase the comparability with the other schemes described, and in order to reconstruct the underlying identification protocol based on the Diffie-Hellman assumption. In particular we made the following changes:

- In the SW scheme, a second key pair is generated and used to sign the number of chunks in the file such that it cannot be tampered. Instead we include the number of chunks in the state st as described in the other schemes.
- SW introduce a tuning parameter s which subdivides the chunks into smaller subchunks. A larger value for s increases the size of the proofs linearly but

on the other hand decreases the number of authenticators σ_i , the prover \mathcal{P} needs to store, linearly.

We set $s = 1$ to not subdivide the chunks, so that the number of authenticators is the same as in the other schemes.

- In the SW scheme, there are public values $u_1, \dots, u_s \in G$ chosen at random from a group G . Since we set $s = 1$, this is only one value u , which we include in the public key, rather than choosing it per file.
- The challenge in the SW scheme is a set $Q = \{(i, \nu_i)\}$, where $1 \leq i \leq \ell, 1 \leq \nu_i \leq p\}$ of coefficients. We generate the challenge from a pseudorandom function $\nu_i = \mathcal{H}'_c(i)$ for $i = 1, \dots, \ell$, mapping to \mathbb{Z}_p and thus the size of the challenge is the same as in the other schemes we described.
- Further, we adapted the notation to match our earlier expositions.

A full description of the modified protocol is given in Fig. 4.

- $(pk, sk) \leftarrow \text{Gen}(1^k)$
 1. Take a group G of order p such that an efficient bilinear pairing $e : G \times G \rightarrow G_T$ exists. Let g be a generator of G .
 2. Choose a random secret key $sk \in_R \{1, \dots, p-1\}$ and compute the public key as $pk = (g^{sk}, u)$, where u is another generator of G .
 3. Return (pk, sk) .
- $(f', st) \leftarrow \text{Encode}_{sk}(f)$
 1. Parse f as f_1, \dots, f_ℓ with $f_i \in \mathbb{Z}_p$ for all $i = 1, \dots, \ell$.
 2. Define the state $st = (s, \ell)$, where $s \in_R \{0, 1\}^k$ is chosen at random.
 3. Compute the authenticators $\sigma_i = [\mathcal{H}_{st}(i) \cdot u^{f_i}]^{sk} \in G$ for $i = 1, \dots, \ell$, where \mathcal{H}_{st} is a pseudorandom function mapping to G keyed with key st .
 4. Return $f' = (f, \sigma)$ and st .
- $\pi := \text{Prove}(pk, f', c)$
 1. Expand the challenge c as $chal_i = \mathcal{H}'_c(i)$ for $i = 1, \dots, n$, where \mathcal{H}'_c is a pseudorandom function mapping to \mathbb{Z}_p keyed with key c .
 2. Parse f' as (f, σ) and f as f_1, \dots, f_ℓ .
 3. Aggregate the authenticators as $\tau = \prod_{i=1}^{\ell} \sigma_i^{chal_i}$.
 4. Aggregate the chunks as $\mu = \sum_{i=1}^{\ell} f_i \cdot chal_i$.
 5. Return $\pi = (\tau, \mu)$
- $b := \text{Verify}(pk, st, c, \pi)$
 1. Parse pk as (v, u) .
 2. Recover the number of chunks ℓ in the file from st .
 3. Parse π as (τ, μ) .
 4. Check if $e(\tau, g) = e\left(\prod_{i=1}^{\ell} \mathcal{H}_{st}(i)^{\mathcal{H}'_c(i)} \cdot u^\mu, v\right)$.

Fig. 4. Our modified Shacham-Waters proof of storage

These modifications allow us to extract the underlying identification protocol which is shown in Fig. 5 and might be of independent interest. It can be shown

that our modifications to the SW proof of storage scheme are secure by proving that the underlying identification protocol is indeed an unforgeable homomorphic identification protocol in our attacker model from Definition 2. This is done in Theorem 2 in the appendix.

Lemma 2. *The modified SW proof of storage shown in Fig. 4 is secure under the Diffie-Hellman assumption if \mathcal{H} is modelled as a random oracle.*

Proof. Follows directly from the security of the transformation (Theorems 1 and 2 in [2]) and the security of the SW identification scheme in Theorem 2 in the appendix.

- $(pk, sk) \leftarrow \text{Setup}(1^k)$
 1. Let G be a group of prime order $p \in \mathbb{P}$ such that an efficient bilinear pairing $e : G \times G \rightarrow G_T$ exists. Let g be a generator of G .
 2. Choose a random secret key $sk \in_R \{1, \dots, p-1\}$ and compute the public key as $pk = (g^{sk}, u)$, where u is another generator of G .
 3. Return (pk, sk) .
- $\alpha \leftarrow \text{Commit}(pk; r)$
 1. Return r .
- $\gamma \leftarrow \text{Response}(pk, sk, r, \beta)$
 1. Return $(r \cdot u^\beta)^{sk}$.
- $b := \text{Verify}(pk, \alpha, \beta, \gamma)$
 1. Check if $e(\gamma, g) = e(r \cdot u^\beta, pk)$.
- $\text{Combine1}(\mathbf{s}, \boldsymbol{\alpha}) := \prod_i \alpha_i^{s_i}$
- $\text{Combine3}(\mathbf{s}, \boldsymbol{\gamma}) := \prod_i \gamma_i^{s_i}$

Fig. 5. Our Shacham-Waters identification protocol

4 Evaluation

In the following we provide benchmarks for the three proof of storage schemes we described and implemented.

4.1 Method

We implemented the three discussed proof of storage schemes in Python 3.5.2 and evaluated their performance on a quad core Intel Xeon CPU with 3.10 GHz running Ubuntu 16.04.1. We did not use any parallelization though most of the algorithms are easily parallelizable. The implementations were written by the same developer, hence the coding style and quality is similar.

The choice of parameters to achieve equal security level, and consequently guarantee a fair comparison is summarized in Table 1. We use the symbol $|\cdot|$ to denote the bit length.

For the GQ and Shoup scheme we chose the order of the finite group to be 2048 bits. For these schemes, we used gmpy v1.17, which is a Python binding of the GNU multiple precision library GMPlib [8]. In the Shoup protocol we chose the parameter $k = 5$ thus the security depends on the inability to compute 2^{15} -th roots in a 2048 bit group which according to NIST [3, Section 5.6.1] is equivalent to a security level of 112 bits and is considered secure.

The pairing operations in the SW scheme were implemented with the Python bindings of the PBC Library v0.5.14 [14]. We decided to use a type F pairing for the implementation. On the one hand, type F pairing operations are slower than in other types of pairings, but the algorithm in the SW scheme only needs to compute one of them. On the other hand, elements in type F pairings are smaller than in other pairing types of comparable security, so the many exponentiations in our implementation are faster. The pairing was generated with the script `genfparam` bundled with libPBC. For a security level of 112 bits the size of the curve needs to be around 224–255 bits according to NIST [3, Section 5.6.1]. However, this does not take into account the recent improvements in computing discrete logarithms in finite fields by Kim and Barbulescu [11] which halve the security parameter. Thus, we chose to use a curve of size 448–510 bits. Hence, the security of the three implemented schemes is comparable. An overview of the choice of all parameters can be found in Table 1.

For \mathcal{H}_{st} , the pseudorandom function keyed with the state, where the state is a pair, we convert the state into JSON format and use the standard HMAC with SHA256 algorithm as described by RFC 2104 [13].

We measured the duration needed for encoding the file T_{Encode} , generating the proof of storage T_{Prove} , and verification time T_{Verify} for files up to 4000 kB in steps of 250 kB. Each measurement is the mean of the duration of 10 runs of the algorithm. For the time needed to generate the keys T_{Gen} we computed the mean over 160 runs.

Table 1. Overview of the parameters

Algorithm	Parameters
Our GQ PoS	$ n = 2048, e = 65537$
Shoup PoS [2]	$ n = 2048, k = 5$
SW PoS [19]	$ r = 445, q = 445$

4.2 Results and Discussion

As expected all durations were linearly dependent on the size of the file. The interpolated processing bandwidth for the different algorithms, as well as the time used for key generation can be found in Table 2. The interpolation was done by ordinary least squares regression.

Table 2. Overview of the results

Algorithm	T_{Gen}	B_{Encode}	B_{Prove}	B_{Verify}
Our GQ PoS	0.111s	56.1 kB/s	414 kB/s	62.7 kB/s
Shoup PoS [2]	0.187s	4.37 kB/s	416 kB/s	64.8 kB/s
SW PoS [19]	0.124s	5.20 kB/s	54.4 kB/s	16.9 kB/s

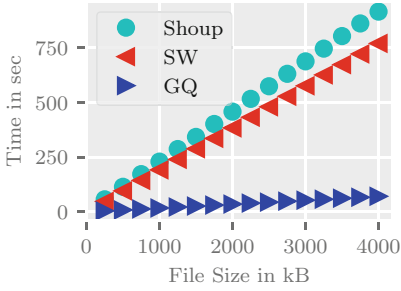
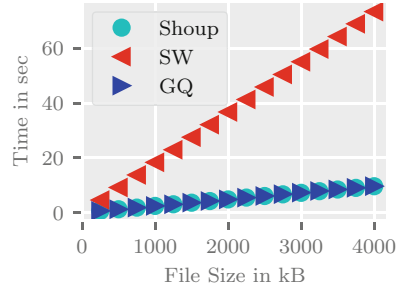
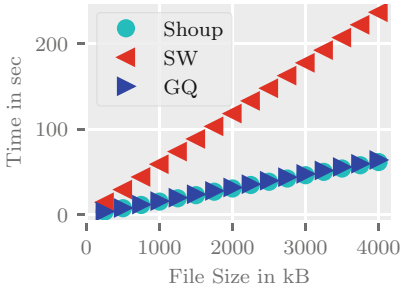
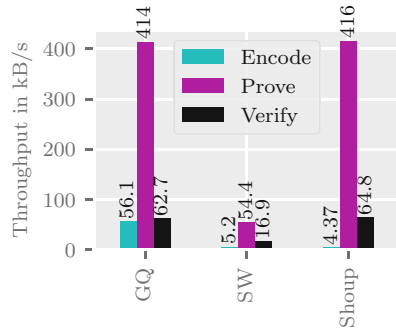
When encoding the file, the Shoup proof of storage was slower than our GQ proof of storage. This performance difference can be traced back to a performance difference in the underlying identification protocols. The main difference between the Shoup and the GQ identification protocol is that in the Shoup identification protocol the secret key is a 2^{3m} -th root instead of an e -th root, where e is chosen such that it is invertible modulo $\varphi(n)$. Thus the Shoup protocol has to be slower, since when computing an e -root in the GQ identification protocol we can simply exponentiate with the inverse d , where $de = 1 \pmod{\varphi(n)}$. For a 2^{3m} -th root in the Shoup protocol this does not work, since $2 \mid \gcd(2^{3m}, \varphi(n))$ and thus we cannot compute an inverse d . Instead we have to compute roots modulo the primes p and q by successively raising to the $\frac{p-1}{2}$ -th power, respectively the $\frac{q-1}{2}$ -th power and recombine the solution modulo n with the chinese remainder theorem. The performance of the algorithm `Encode` in the SW protocol was faster than the same procedure in the Shoup protocol, but slower than in the GQ protocol.

The Shoup and GQ scheme needed exactly the same time for generation of the proof, since the algorithm for generating the proof is the same. Generation of the proof in the SW protocol had the worst performance. We found out that this is due to expanding the challenge by computing $chal_i = \mathcal{H}_c(i)$ for $i = 1, \dots, \ell$. This was implemented by using the function `element_from_hash` of libPBC since the element needed to be a point on the elliptic curve. The exact algorithm can be found on page 19 in Lynn's thesis [15].

In the verification procedure the Shoup protocol was insignificantly faster than our GQ protocol. As in generation of the proof the SW proof of storage performed worst in verifying.

An interesting observation is that generation of the key in the SW scheme was fastest. This is explained by the fact that we used a type F pairing, so our underlying group is small and thus exponentiations are fast, as explained previously.

The full performance measurements of the schemes are given in Fig. 6a to d. The duration of the algorithms in seconds is shown on the y-axis. The size of the file of which storage was proven is shown on the x-axis.

(a) T_{Encode} for the schemes(b) T_{Prove} for the schemes(c) T_{Verify} for the schemes

(d) Regressions of the throughput of the schemes according to Table 2

Fig. 6. Comparison of runtimes of the schemes

5 Conclusion

In this paper we modified the GQ identification protocol such that it is secure in a stronger attacker model. This allowed us to construct a novel proof of storage based on the RSA assumption. These schemes allow a storage provider to provide a publicly verifiable proof that it has stored a special file. Additionally we introduced a novel identification protocol based on bilinear pairings. We carefully compared the performance of our scheme to other state of the art publicly verifiable proof of storage schemes. In our evaluation our new scheme greatly outperformed existing schemes with similar security level in the time taken to encode the file. However, all schemes were comparatively slow and additional progress needs to be made to use them in a real world scenario.

Nevertheless, our scheme is a new contribution to the growing ecosystem of cloud storage security mechanisms to close the gap between theory and practice regarding publicly verifiable proofs of storage.

Acknowledgments. This work was funded by the Baden-Württemberg Stiftung. Further we would like to thank Jonathan Katz for his helpful emails and his suggestion to include the file length in the state. We thank Stephan Kleber for discussions and proofreading.

A Proofs

Theorem 1. *Our modified GQ identification protocol as described in Fig. 1 is an unforgeable homomorphic identification protocol under the RSA-assumption.*

Proof. Completeness is clear. To prove unforgeability we construct an algorithm \mathcal{B} that can solve the RSA problem given access to an attacker \mathcal{A} . Let $n = pq$ be an RSA modulus with unknown primes p and q .

- \mathcal{B} is given the composite n , as well as an integer $J \in_R \mathbb{Z}_n$ and $e \in_R \mathbb{Z}$ which is coprime to n . We construct \mathcal{B} to return $sk^{-1} = J^d$, where $de = 1 \pmod{\varphi(n)}$, i.e., the e -th root of J . \mathcal{B} gives the public key $pk = (J, e)$ as an input to the algorithm \mathcal{A} .
- Whenever \mathcal{A} outputs β' in the challenge space, \mathcal{B} chooses a random $\gamma \in \mathbb{Z}_n$ and sets $r = J^{\beta'}\gamma^e$. The algorithm \mathcal{B} then invokes \mathcal{A} on the input (r, γ) .
- When algorithm \mathcal{A} outputs an n -vector of challenges β , for each i the algorithm \mathcal{B} computes (r, γ) as in the previous step, sets $\alpha_i \leftarrow \text{Commit}(pk; r) = r$ and gives the vectors $(\mathbf{r}, \boldsymbol{\gamma})$ to the algorithm \mathcal{A} .
- If \mathcal{A} outputs $(\mathbf{s}, \mu', \gamma')$ where $\mathbf{s} \in \mathbb{Z}_{2^k}^n$, and
 1. $\mu' \neq \sum_i s_i \beta_i$, and
 2. $\text{Verify}(pk, \text{Combine1}(\mathbf{s}, \boldsymbol{\alpha}), \mu', \gamma') = 1$
 we compute the e -th root of J , namely sk^{-1} , as follows:
 For ease of notation define $\alpha^* := \text{Combine1}(\mathbf{s}, \mathbf{r})$, $\gamma^* := \text{Combine3}(\mathbf{s}, \boldsymbol{\gamma})$, and $\mu^* := \sum_i s_i \beta_i$. We know that

$$J^{\mu^*} (\gamma^*)^e = \alpha^* = J^{\mu'} (\gamma')^e.$$

where the first equality follows from the condition on the output of \mathcal{A} and the second equality follows from completeness. Thus

$$J^{\mu^* - \mu'} (\gamma^* / \gamma')^e = 1.$$

Since $\text{gcd}(e, \mu^* - \mu') = 1$ by the choice of e , using the Euclidean algorithm we can find coefficients s, t such that $s \cdot e + t \cdot (\mu^* - \mu') = 1$. The e -th root of J is now $(\gamma' / \gamma^*)^t \cdot J^s$, since

$$\begin{aligned} ((\gamma' / \gamma^*)^t \cdot J^s)^e &= (\gamma' / \gamma^*)^{te} \cdot J^{se} \\ &= (\gamma' / \gamma^*)^{te} \cdot J^{1-t(\mu^* - \mu')} \\ &= J \left((\gamma' / \gamma^*)^e \cdot J^{-(\mu^* - \mu')} \right)^t \\ &= J. \end{aligned}$$

Since \mathcal{B} solves the RSA problem if \mathcal{A} succeeds we conclude that the success probability of \mathcal{A} is negligible.

Theorem 2. *Our reconstructed identification protocol shown in Fig. 5 is an unforgeable homomorphic identification protocol under the assumption that the computational Diffie-Hellman problem is intractable.*

Proof. Completeness is immediate. To prove unforgeability we construct a ppt algorithm \mathcal{B} that can solve the Diffie-Hellman problem, given access to an attacker algorithm \mathcal{A} .

- \mathcal{B} is given an element $g \in G$ as well as g^{sk} for some $sk \in \{1, \dots, p-1\}$ and an element $h \in G$. To solve the Diffie-Hellman problem \mathcal{B} needs to compute h^{sk} . It chooses random coefficients $s, t \neq 0$ and sets $u = g^s h^t$. Thus, the public key is $pk = (g^{sk}, u)$. Afterwards the algorithm \mathcal{B} executes \mathcal{A} .
- Whenever \mathcal{A} outputs β' in the challenge space \mathcal{B} chooses a random $r \in_R \{1, \dots, p-1\}$ and computes $\gamma = pk^r$. One can easily verify that this is indeed a valid transcript. \mathcal{B} gives (r, γ) to \mathcal{A} .
- When \mathcal{A} outputs an n -vector of challenges β , then for each i the algorithm \mathcal{B} computes (r, γ) as in the previous step and sets $\alpha = g^r u^{-\beta}$. It gives the vectors $(\mathbf{r}, \boldsymbol{\gamma})$ to the algorithm \mathcal{A} .
- If \mathcal{A} outputs $(\mathbf{s}, \mu', \gamma')$ where $\mathbf{s} \in \mathbb{Z}_{2^k}^n$, and
 1. $\mu' \neq \sum_i s_i \beta_i$, and
 2. $\text{Verify}(pk, \text{Combine1}(\mathbf{s}, \boldsymbol{\alpha}), \mu', \gamma') = 1$
 we solve the discrete logarithm problem by computing h^{sk} as follows: Define $\alpha^* := \text{Combine1}(\mathbf{s}, \mathbf{r})$, $\gamma^* := \text{Combine3}(\mathbf{s}, \boldsymbol{\gamma})$, and $\mu^* := \sum_i s_i \beta_i$. Because of the second condition on $(\mathbf{s}, \mu', \gamma')$ we know that

$$e(\gamma', g) = e(\alpha^* u^{\mu'}, g^{sk}).$$

Since $\alpha^*, \mu^*, \gamma^*$ is also a valid transcript we know that

$$e(\gamma^*, g) = e(\alpha^* u^{\mu^*}, g^{sk}).$$

Dividing the two equations yields

$$e(\gamma'/\gamma^*, g) = e(u^{\mu' - \mu^*}, g^{sk}).$$

Substituting $u = g^s h^t$ results in

$$e(\gamma'/\gamma^*, g) = e(g^{s(\mu' - \mu^*)} h^{t(\mu' - \mu^*)}, g^{sk}).$$

We can now rearrange the terms as

$$e(\gamma'/\gamma^* \cdot g^{s(\mu^* - \mu')}, g) = e(h, g^{sk})^{t(\mu' - \mu^*)}$$

and solve the discrete logarithm problem by computing h^{sk} as follows:

$$h^{sk} = \left(\gamma'/\gamma^* \cdot g^{s(\mu^* - \mu')} \right)^{\frac{1}{t(\mu' - \mu^*)}}.$$

This fails if and only if the denominator is zero. But this will not happen, as we have chosen t to be nonzero and $\mu' - \mu^* \neq 0$ holds due to the first condition on the output of the attacker.

Since \mathcal{B} solves the discrete logarithm problem whenever \mathcal{A} succeeds we conclude that the success probability of the attacker \mathcal{A} is negligible.

References

1. Ateniese, G., et al.: Remote data checking using provable data possession. *ACM Trans. Inf. Syst. Secur.* **14**(1), 12:1–12:34 (2011)
2. Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: Matsui, M. (ed.) *ASIACRYPT 2009*. LNCS, vol. 5912, pp. 319–333. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_19
3. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management-part 1, revision 4. In: NIST special publication 800-857 (2016)
4. Cash, D., Kupcu, A., Wichs, D.: Dynamic proofs of retrievability via oblivious ram. *Cryptology ePrint Archive*, Report 2012/550 (2012)
5. Curtmola, R., Khan, O., Burns, R., Ateniese, G.: MR-PDP: multiple-replica provable data possession. In: *ICDCS 2008*, pp. 411–420. IEEE (2008)
6. 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
7. filecoin.io: Filecoin: A Cryptocurrency Operated File Storage Network (2014). <https://filecoin.io/filecoin-jul-2014.pdf>
8. GNU Project: GMPlib. <https://gmplib.org/>
9. Guillou, L.C., Quisquater, J.-J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Barstow, D., et al. (eds.) *EUROCRYPT 1988*. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-45961-8_11
10. Juels, A., Kaliski Jr, B.S.: PORs: proofs of retrievability for large files. In: *Conference on Computer and Communications Security, CCS 2007*, pp. 584–597. ACM (2007)
11. Kim, T., Barbulescu, R.: Extended tower number field sieve: a new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) *CRYPTO 2016*. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_20
12. Kopp, H., Bösch, C., Kargl, F.: KopperCoin – a distributed file storage with financial incentives. In: Bao, F., Chen, L., Deng, R.H., Wang, G. (eds.) *ISPEC 2016*. LNCS, vol. 10060, pp. 79–93. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49151-6_6
13. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: keyed-hashing for message authentication. RFC 2104 (1997). <http://www.ietf.org/rfc/rfc2104.txt>
14. Lynn, B.: PBC Library. <https://crypto.stanford.edu/pbc/>
15. Lynn, B.: On the implementation of pairing-based cryptography. Ph.D. thesis, Department of Computer Science, Stanford University (2007)
16. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_3
17. Ren, Z., Wang, L., Wang, Q., Xu, M.: Dynamic proofs of retrievability for coded cloud storage systems. *IEEE Trans. Serv. Comput.* **11**, 685–698 (2015)

18. Schnorr, C.P.: Efficient signature generation by smart cards. *J. Cryptol.* **4**(3), 161–174 (1991)
19. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_7
20. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: Conference on Computer and Communications Security, pp. 325–336. ACM (2013)
21. Shoup, V.: On the security of a practical identification scheme. *J. Cryptol.* **12**(4), 247–260 (1999)
22. Wang, C., Chow, S.S.M., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **62**(2), 362–375 (2013)
23. Xu, J., Chang, E.C.: Towards efficient proofs of retrievability. In: ASIACCS 2012, pp. 79–80. ACM (2012)
24. Xu, J., Yang, A., Zhou, J., Wong, D.S.: Lightweight and privacy-preserving delegatable proofs of storage. *IACR Cryptology ePrint Archive* 2014, 395 (2014)
25. Zhang, Y., Blanton, M.: Efficient dynamic provable possession of remote data via balanced update trees. In: ASIACCS 2013, pp. 183–194. ACM (2013)
26. Zhu, Y., Hu, H., Ahn, G.J., Yu, M.: Cooperative provable data possession for integrity verification in multicloud storage. *IEEE Trans. Parallel Distrib. Syst.* **23**(12), 2231–2244 (2012)



Verifiable Single-Server Private Information Retrieval

Xingfeng Wang¹ and Liang Zhao^{1,2}(✉)

¹ College of Cybersecurity, Sichuan University, Chengdu, China
xingfengw@yeah.net, zhaoliangjapan@scu.edu.cn

² HIFIVE Lennon Laboratory, Chengdu HiFive Technology Co., Ltd.,
Chengdu, China

Abstract. Single-server Private Information Retrieval (SPIR) allows a client to privately retrieve some data from a database stored on a server. While many SPIR schemes exist, these previous SPIR schemes are generally under the honest-but-curious server model. This model however is not suitable for many real world scenarios such as involving the untrusted cloud server. In this paper, we first propose an SPIR scheme that is based on the learning with (binary) errors assumption under the honest-but-curious server model. Specifically, compared with some previous SPIR schemes, our proposal provides a low communication complexity. Then, according to the above warm-up scheme, we introduce a Verifiable SPIR (VSPiR) scheme under the malicious server model where the server may provide some fraudulent answers. To the best of our knowledge, our scheme is the first practical VSPiR scheme that employs the probabilistic verification process. Finally, for our proposal, we present the theoretical analyses of the properties (i.e., correctness, privacy and security), and give the detailed implementation results.

Keywords: Learning with errors
Single-server private information retrieval
Probabilistic verification process

1 Introduction

1.1 Background

In the age of Internet accessing remote database is common and information is the most sought after and costliest commodity. In such a situation it is very important not only to protect information but also to protect the identity of the information that a user is interested in [11]. Private Information Retrieval (PIR) schemes are cryptographic schemes that enable users to retrieve records from public databases while keeping private the identity of the retrieved records [2]. In PIR schemes, a client is allowed to retrieve an entry from a server in possession of a database without revealing which entry is retrieved. The concept of PIR was first proposed in 1995 by Chor et al. [4]. There is a trivial solution consisting

in sending the entire database regardless of the query. This solution has a high communication complexity of the database's size tb (at least $\log tb$ bits). Later, some schemes [3, 16, 18] that send less data have been proposed. Specifically, the Fully Homomorphic Encryption (FHE) and even the SomeWhat Homomorphic Encryption (SWHE) proposed by Gentry is known to imply the PIR scheme [3].

Moreover, in some practical scenarios, the server may provide the incorrect answers due to malicious behaviors or accidental failures. These scenarios can be defined as the malicious server model. Under this model, a PIR scheme can work effectively if the client should be able to identify the incorrect answers with overwhelming probability. This implies that how to verify the returned answers is a significant problem for a PIR scheme. Actually, for the honest-but-curious server model used in the previous work, it is assumed that the server is honest, which means that he follows the predefined scheme. From this point, this model is not very practical compared with the malicious server model. Then, constructing a PIR scheme that is secure in the malicious server model is well motivated and has been put forth by Beimel [2].

1.2 Related Work

In [18], Zhang and Safavi-Naini gave a verifiable multi-server PIR scheme where the servers may be malicious and provide some fraudulent answers. This scheme is an unconditionally t -private and computationally secure k -server verifiable PIR scheme in the honest-but-curious server setting. The drawback of this scheme is that it is too complicated to implement practically. Moreover, this PIR scheme does not work when all colluding servers host the database, which can be seen as the single malicious server setting.

In Sect. 5 of [3], the SWHE scheme is used to construct an asymptotically efficient Single-server PIR (SPIR) scheme based on the Learning With Errors (LWE) assumption. Specifically, this scheme employs some symmetric encryption scheme in the retrieval procedure. Using the most efficient symmetric scheme with the respect to the communication, the corresponding complexity of this scheme is $\mathcal{O}((\log n) + \kappa \text{poly} \log(\kappa))$ (n is the database size and κ is the security parameter).

In [16], Vannet and Kunihiro proposed an SPIR scheme under the honest-but-curious server model relying on the unrelated Approximate GCD (AGCD) assumption. Assume the size of database is tb , which can be split into nb blocks of mw words of bb bits each, such that $nb \cdot mw \cdot bb = tb$. When nb cannot be decomposed in this way, pad the database with several bits. The database is denoted by a 2-dimensional array of words where each word is marked by two coordinates. Now use the set $\{b_{i,j} | 1 \leq i \leq nb, 1 \leq j \leq mw\}$ to denote the database, and write block u as $\{b_{u,j} | 1 \leq j \leq mw\}$. The security of this scheme is based on the AGCD assumption introduced in [6]. The assumption is said that given a random distribution of values $pq + \epsilon$ where $\epsilon \ll p$, the q has φ_q bits. Sample a set of this distribution, output p . In the single bit scheme, assume that the client wants to retrieve the block u consisting of $\{b_{u,j}\}_{1 \leq j \leq mw}$. The client samples a large random odd number p , and saves it as the secret key. He picks nb

random numbers q_i and ϵ_i , and computes $Q_i = pq_i + 2\epsilon_i + \delta_{i,u}$ ($\delta_{i,u}$ is the index vector where $\delta_{i,j} = 1$ if $i = j$, 0 otherwise). For each Q_i that the server received, compute $R_j = \sum_{i=1}^{nb} b_{i,j}Q_i$ and send it back to the client. On receiving R_j , the client decodes that $(R_j \bmod p) \bmod 2 = (\sum_{i=1}^{nb} b_{i,j}(pq_i + 2\epsilon_i + \delta_{i,u}) \bmod q) \bmod 2 = \sum_{i=1}^{nb} b_{i,j}(2\epsilon_i + \delta_{i,u}) \bmod 2 = b_{u,j}$. In this scheme, p and q should be two large integers, which can guarantee that the scheme holds the security property. However, this scheme works under the honest-but-curious server model but not the malicious server model.

1.3 Open Problem

The previous work [9,16] related to SPIR is generally under the honest-but-curious server model. This model however is not suitable for many real-world scenarios such as involving the untrusted cloud server. From Zhang and Safavi-Naini's work [18], although a verifiable multi-server PIR scheme has been presented, constructing a Verifiable SPIR constructing a Verifiable SPIR (VSPiR) scheme under the malicious server model seems to be a difficult task. This is due to the fact that the protection of the input index depends on the heavy FHE scheme, which implies that the computational complexity is very high. To the best of our knowledge, there has not been a practical VSPiR scheme. Therefore how to construct a simple and practical VSPiR is still an open problem.

1.4 Our Contributions

In this work, we present two main contributions. The first warmup one is to introduce an SPIR scheme based on the decision-LWE with binary error assumption under the honest-but-curious server model. Then, according to this scheme, we construct a VSPiR scheme under the malicious server model.

- **The SPIR scheme based on the decision-LWE assumption.** In our proposed SPIR, we use the database defined in [16]. We assume that a client wants to obtain the block u without revealing any information about u . The client uses a special variant of the encryption scheme with additive homomorphism in [7] to encrypt the query vector where the u -th elements is 1 and others are 0, then compute the query messages $\{Q_i\}$. A server computes $R_j = \sum_i b_{i,j}Q_i$ where R_i is equivalent to the encrypted $b_{u,j}$. Then the server sends R_j to the client. For each R_j , the client runs the homomorphic decryption scheme to recover the block u . Thus the client can get the real block. The privacy of our SPIR scheme is based on the hardness of LWE with binary error problem.
- **The VSPiR scheme using the probabilistic verification process** (see Fig. 1). Based on our proposed SPIR scheme, we use a probabilistic verification process [5] to construct our VSPiR scheme. The main idea of the probabilistic verification is very simple: a client samples a random input r and precomputes a specific function $F(r)$. He sends an input pair (x, r) to a

server in a random order, and wants to receive both $F(x)$ and $F(r)$ from the server. When receiving the answers from the server, the client checks the correctness of the response value $F(r)$; if it is the same as the precomputed $F(r)$, then the client accepts the response $F(x)$, and rejects otherwise. Because both x and r are independent and distributed identically, no malicious adversary can distinguish the real input x from the random input r and deceive with probability greater than $1/2$. In our proposed VSPIR scheme, we do the similar process: the client generates a random vector $\mathbf{r} \in \{0, 1\}^m$ to replace the u -th row in a matrix. Encrypt this matrix, then send the query message to the server and decode the responses R_j from the server. If the elements of the random vector corresponding to the index are the same, the elements of u -th row are the same. Then, the client accepts the received responses.

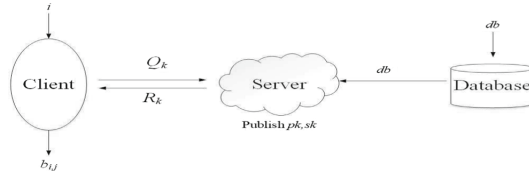


Fig. 1. Verifiable single-server PIR

For showing the merits of our proposal, we list the differences between our scheme and some other related scheme in Table 1. Specifically, our construction is essentially different from the SPIR construction proposed by Brakerski and Vaikuntanathan [3]. In our proposal, the client uses the encryption scheme with the additively homomorphic property to encrypt the index directly and the server responds the answer using the additive homomorphically evaluate the database access function. However, in [3], the client encrypts the symmetric key using the FHE or SWHE scheme, then uses the encrypted symmetric key to encrypt the index, which can convert the symmetric ciphertexts into homomorphic ciphertexts. The server uses the homomorphic ciphertexts homomorphically evaluate the database access function to retrieve an encryption of the answer. Moreover, in our SPIR construction, since using the encryption scheme based on the LWE with binary error assumption, the matrix multiplication operation in encryption scheme is equivalent to some matrix addition operation. The computational complexity can be slowed down to be $\mathcal{O}(\sqrt{tb})$.

1.5 Outline of Our Paper

The rest of this paper is organized as follows: in Sect. 2, after finishing notations used in this paper, we introduce the LWE problem and some definitions related to the VSPIR. In Sect. 3, we detail our proposed constructions for the SPIR scheme and VSPIR scheme, and then in Sect. 4 we analyze their performances.

In Sect. 5, we present some computer simulations for our proposals. Finally, in Sect. 6, we make some concluding remarks.

Table 1. Comparisons between the proposed scheme and some other related schemes.

	Zhang and Safavi-Naini [18]	Brakerski and Vaikuntanathan [3]	Vannet and Kunihiro [16]	Our VSPiR
Single-server	No	Yes	Yes	Yes
Verifiability	Yes	No	No	Yes
Assumption	d -SBDH	LWE	AGCD	LWE with binary error
Communication complexity	$\mathcal{O}(\kappa n^{1/\lceil(2k-1)/t\rceil})$	$\mathcal{O}(\log n + \kappa \text{poly} \log(\kappa))$	$\mathcal{O}(\sqrt{n} \log n)$	$\mathcal{O}(c\sqrt{n})$
Computational complexity	$\mathcal{O}(\kappa n^{2/\lceil(2k-1)/t\rceil})$	$\mathcal{O}(\kappa^3)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(c\sqrt{n})$

n : database size; k : server number related parameter; κ : security parameter; t : number of servers

2 Preliminaries

2.1 Notations

Before we present our scheme, we give some notations used in this paper. In this work, We denote vectors by bold lower-case letters ($\mathbf{x}, \mathbf{y}, \dots$), matrices by bold upper-case letters ($\mathbf{X}, \mathbf{Y}, \dots$). We denote a security parameter by $\kappa \in \mathbb{N}_+$. We denote the class of polynomial functions in κ by $\text{poly}(\kappa)$, some fixed polynomial functions q in κ by $q = q(\kappa)$, and some unspecified negligible function in κ by $\text{negl}(\kappa)$. We denote the transpose of \mathbf{x} by \mathbf{x}^T . We consider the operation $x \xleftarrow{\$} \Psi$ as choosing x uniformly at random in a set Ψ . We use \mathcal{D} to indicate a distribution over some finite set \mathcal{S} . We denote $x \xleftarrow{\$} \mathcal{D}$ that x is generated at random from the distribution.

2.2 Learning with Errors

The LWE problem was first introduced by Regev [15]. The formal definition can be as follows:

Definition 1 (LWE Problem [3]). For security parameter κ , $n = n(\kappa)$, let $q = q(n)$ be an integer and error distribution $\chi = \chi(n)$ over \mathbb{Z}_q . Let $A_{s,\chi}$ be the distribution obtained by choosing a vector \mathbf{a} from \mathbb{Z}_q^n and an error term e from χ uniformly at random, and outputting $(\mathbf{a}, \langle \mathbf{a}, s \rangle + e) \in (\mathbb{Z}_q^n \times \mathbb{Z}_q)$. The learning with errors problem $LWE_{n,m,q,\chi}$ defined as follows: Given m independent instances from $A_{s,\chi}$, output \mathbf{s} with non-negligible probability.

The decision variant of the LWE problem, denoted $\text{decision-LWE}_{n,m,q,\chi}$ is to distinguish the following two distributions: One is that sampling m instances

$(\mathbf{a}_i, \mathbf{b}_i)$ uniformly from \mathbb{Z}_q^{n+1} . The other one is that sampling m instances sampled according to $A_{\mathbf{s}, \chi}$. The **decision-LWE** $_{n,m,q,\chi}$ assumption is that the **decision-LWE** $_{n,\kappa,q,\chi}$ problem is computationally infeasible.

Regev proved in [15] that given certain module q and Gaussian error distribution χ , **LWE** $_{n,\kappa,q,\chi}$ problem is as long as certain worst-case lattice problems which are hard to solve using a quantum algorithm. These reductions take χ to be the discretized versions of the Gaussian distribution which is B -bounded for an appropriate value B .

Definition 2 (B-Bounded Distributions [3]). A distribution ensemble $\{\chi_n\}_{n \in \mathbb{N}}$, supported over the integers, is called B -bounded if

$$\Pr_{e \leftarrow \chi_n} [|e| > B] \leq \text{negl}(n).$$

The following theorem is the Regev’s worst-case to average-case reduction for **LWE**:

Theorem 1 ([15]). For $q = q(n) \in \mathbb{N}$ be a product of $q = \prod q_i$ such that for all i , $q_i = \text{poly}(n)$, and let $B \geq n$. There exists an efficiently sampleable B -bounded distribution χ such that if there is an efficient algorithm that solves the **decision-LWE** $_{n,q,\chi}$ problem, then there is an efficient quantum algorithm for solving $\tilde{O}(qn^{1.5}/B)$ -approximate worst-case **SIVP** and **gapSVP**.

We refer the readers to [14, 15] for the detailed and formal definitions of these lattice problems.

Definition 3 (LWE with Binary Error Problem [10]). Let n, q be positive integers, χ be a uniform distribution on $\{0, 1\}$ and $\mathbf{s} \xleftarrow{\$} \chi^n$ be a secret vector in $\{0, 1\}^n$. Let $A'_{\mathbf{s}, \chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

LWE with binary error problem is to recover \mathbf{s} from m samples $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

The **decision variant** of the **LWE with binary error problem** is to distinguish with non-negligible advantage m samples chosen according to $A'_{\mathbf{s}, \chi}$, from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Theorem 2 ([13]). For any integers n and $m = n \cdot (1 + \Omega(1/\log n))$, and all sufficiently large polynomially bounded prime modulus $q \geq n^{O(1)}$, solving **LWE** $_{n,m,q}$ with uniformly random binary errors (i.e. in $\{0, 1\}$) is at least as hard as approximating lattice in the worst case on $\Theta(n/\log n)$ -dimensional lattices within a factor $\gamma = \tilde{O}(\sqrt{n} \cdot q)$.

Theorem 2 shows that for the **LWE** problem, it remains hard even when the errors are small (e.g. uniformly random from $\{0, 1\}$). Most cryptographic constructions are based on the **LWE** problem where secret and error are identically distributed [10]. Using the search-to-decision reduction of [13], Peikert et al. proved that **decision-LWE** $_{n,m,q}$ with binary error has the similar hardness of **LWE** $_{n,m,q}$ with binary error.

2.3 The GHV-Type Encryption Scheme

The basis of the GHV scheme [7] is a trapdoor sampling algorithm [8]. The trapdoor sampling procedure generates a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ (that is within negligible statistical distance of uniform), together with an invertible matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ with small entries such that $\mathbf{T} \cdot \mathbf{A} = 0(\text{mod } q)$.

The trapdoor can be used to solve the LWE problem relative to \mathbf{A} . This is done as follows: $\mathbf{T}\mathbf{y} = \mathbf{T}(\mathbf{A}\mathbf{s} + \mathbf{x}) = \mathbf{T}\mathbf{x}(\text{mod } q)$. Multiplying \mathbf{T}^{-1} gives us \mathbf{x} . There is a probabilistic polynomial-time (PPT) algorithm `TrapDoor` that, on input 1^κ , positive integer $q \geq 2$, and a $\text{poly}(n)$ -bounded positive integer $m \geq 5n \log q$, output matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{T} \in \mathbb{Z}^{m \times m}$ where the Euclidean norm of each rows is at least $20n \log q$ [1].

The GHV-type encryption scheme [7] is defined by a triple PPT algorithm $\text{GHV} = (\text{GHV.KeyGen}, \text{GHV.Enc}, \text{GHV.Dec})$:

- $\text{GHV.KeyGen}(1^\kappa) \rightarrow (pk, sk)$: On input the 1^κ , let $n = \kappa$, run the trapdoor sampling algorithm to obtain a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ together with a trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$, i.e., $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapDoor}(1^n, q, m)$. The public key pk is \mathbf{A} and the secret key sk is \mathbf{T} .
- $\text{GHV.Enc}_{pk}(\mathbf{M}) \rightarrow \mathbf{C}$: To encrypt the binary message $\mathbf{M} \in \{0, 1\}^{m \times m}$, choose a uniformly random matrix $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and an “error matrix” $\mathbf{X} \xleftarrow{\$} \chi^{m \times m}$. Output the ciphertext $\mathbf{C} \leftarrow \mathbf{A}\mathbf{S} + 2\mathbf{X} + \mathbf{M}(\text{mod } q)$ where $2\mathbf{X}$ means multiplying each entry of the matrix \mathbf{X} by 2.
- $\text{GHV.Dec}_{sk}(\mathbf{C}) \rightarrow \mathbf{M}$: Set $\mathbf{E} \leftarrow \mathbf{T}\mathbf{C}\mathbf{T}^T(\text{mod } q)$, and then output $\mathbf{B} \leftarrow \mathbf{T}^{-1}\mathbf{E}(\mathbf{T}^T)^{-1} \text{mod } 2$.

2.4 Formal Definitions About VSPIR

Definition 4 (VSPIR). *The VSPIR scheme consists of a database owner server S , and a client C . S has the database $db = (db_1, \dots, db_{tb})$. C owns an index $i \in [tb]$ and wants to recover the db_i from the clouds, keeping the i secret. The VSPIR scheme is defined by five PPT algorithms $\text{VSPIR} = (\text{VSP.Setup}, \text{VSP.Query}, \text{VSP.Challenge}, \text{VSP.Response}, \text{VSP.Verify})$:*

1. $\text{VSP.Setup}(1^\kappa) \rightarrow (pk, sk)$: On input 1^κ , output the public key pk and secret key sk .
2. $\text{VSP.Query}_{sk}(i) \rightarrow (Q, aux)$: On input a private key sk and an index $i \in [tb]$, output a query Q along with auxiliary information aux .
3. $\text{VSP.Challenge}_{sk}(i, \kappa) \rightarrow L$: On input κ , the index i and sk output the challenge message L .
4. $\text{VSP.Response}_{pk}(Q, db, L) \rightarrow R$: On input a public key pk , the query Q , database db and the challenge L . Output the response message R .
5. $\text{VSP.Verify}_{sk}(Q, R, aux) \rightarrow \{db_i, \perp\}$: On input sk , Q , the response message R , and the auxiliary aux . Output the db_i or \perp .

The server S who owns the database is responsible to set up the system. To set up the system, S runs `VSP.Setup` to obtain (pk, sk) in the off-line stage.

Then pk is published or sent to server, the database db is given to the cloud, and the sk is kept private by client. To retrieve db_i , C runs VSP.Query to compute (Q, aux) and sends the query message Q to the cloud S . Upon receiving Q , S runs VSP.Response and replies with the responses message R . To verify the responses, C runs VSP.Challenge to generate a challenge message L and runs VSP.Verify to verify the responding messages and compute db_i if the algorithm VSP.Verify does not output the failure message.

Now, we present some formal properties of VSPiR, these definitions are based on the previous work [3, 16, 18].

Definition 5 (Correctness). *The VSPiR scheme is convinced to be correct if the verify algorithm always computes the correct value of db_i when the server gives the correct response. Formally, for κ , database db , let $\text{VSP.Setup}(1^\kappa) \rightarrow (pk, sk)$, for any query index $i \in [tb]$, let $\text{VSP.Query}_{sk}(i) \rightarrow (Q, aux)$ and $\text{VSP.Response}_{pk}(Q, db, L) \rightarrow R_i$, it holds that*

$$\Pr[\text{VSP.Verify}_{sk}(Q, R, aux) \neq db_i] \leq \text{negl}(\kappa)$$

if the verify algorithm does not compute the failure message.

Definition 6 (Privacy). *The scheme VSPiR is convinced to be private if the adversary can not learn any information about i . Namely, for two queries $i_1, i_2 \in [tb]$ it can computationally distinguish $\text{VSP.Query}_{sk}(i_1)$ from $\text{VSP.Query}_{sk}(i_2)$ with negligible probability. Formally, let κ be a security parameter for an adversary \mathcal{A} running in polynomial time and asking polynomially many queries, it holds that*

$$\Pr[\mathcal{A}(\text{VSP.Query}_{sk}(i_1))] - \Pr[\mathcal{A}(\text{VSP.Query}_{sk}(i_2))] \leq \text{negl}(\kappa).$$

Definition 7 (Security). *The scheme VSPiR is convinced to be secure if PPT adversary can deceive the client into obtaining an incorrect value of db_i with negligible probability. We can consider the behavior of \mathcal{A} in a number of Game_0 , Game_1 , Game_2 as defined below:*

1. Game_0 . The challenger generates $\text{VSP.Setup}(1^\kappa) \rightarrow (pk, sk)$ and then publishes pk to \mathcal{A} . \mathcal{A} owns the database db and every time \mathcal{A} chooses an index i the challenger will reply corresponding query message Q .
2. Game_1 . \mathcal{A} picks a specific index i and sends it to the challenger, the challenger responses $\text{VSP.Query}_{sk}(i) \rightarrow (Q, aux)$. To verify the db_i , challenger runs $\text{VSP.Challenge}_{sk}(i, \kappa) \rightarrow C$. Then \mathcal{A} runs $\text{VSP.Respon- se}_{pk}(Q, db, C) \rightarrow R$ to response the challenger.
3. Game_2 . \mathcal{A} wins if $\text{VSP.Verify}_{sk}(Q, R, aux) \notin \{db_i, \perp\}$.

In the security Game_2 , \mathcal{A} can deceive the client into reconstructing an incorrect value of db_i even if it can choose the index of database freely with negligible probability. Thus, the security of a VSPiR scheme defined above allows the client to recover the correct block that he wants to obtain from the database

Definition 8 (Communication Complexity). *The communication complexity of a scheme is defined as the number of bits being exchanged to transfer a single database element excluding the setup phase.*

Definition 9 (Index Mapping Function). *We define an index mapping function which maps the index u to an vector matrix. It takes as input an index u in some scope and output an index vector: $\delta_{i,u} \leftarrow E(u)$. where the u -th element of the vector is 1, the others are 0.*

3 Our Constructions

In this section, we demonstrate our scheme in a gradual manner. We first present our variant of the GHV-type encryption scheme and an SPIR using this variant. After that, based on the proposed SPIR, we give a VSPiR construction under the malicious server model.

3.1 A Variant of the GHV-Type Encryption Scheme

In GHV scheme [7], it can encrypt a matrix of m^2 elements in time $\tilde{O}(m^3)$. To reduce the computational complexity, we consider the LWE with binary error assumption. Luckily, previous work [10, 13] has proved the hardness of the LWE with binary error problem.

For ease of presentation, we focus below on the case of encrypting binary vectors for better use in our SPIR scheme. The extension for encrypting matrices with lower computational complexity comparable to GHV scheme is straightforward. Our variant of the GHV-type encryption scheme $\text{VGHV} = (\text{VG.KeyGen}, \text{VG.Enc}, \text{VG.Dec})$ is a triple of PPT algorithms as follows:

- $\text{VG.KeyGen}(1^\kappa) \rightarrow (pk, sk)$: The algorithm is the same as the algorithm in GHV scheme. The public key $pk = \mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and the secret key $sk = \mathbf{T} \in \mathbb{Z}^{m \times m}$.
- $\text{VG.Enc}_{pk}(\mathbf{m}) \rightarrow \mathbf{c}$: To encrypt $\mathbf{m} \in \{0, 1\}^m$, choose a uniformly random vector $\mathbf{s} \in \{0, 1\}^n$ and a uniformly random error vector $\mathbf{x} \in \{0, 1\}^m$. Output the ciphertext $\mathbf{c} \leftarrow \mathbf{A}\mathbf{s} + 2\mathbf{x} + \mathbf{m} \pmod q$ where $2\mathbf{x}$ means multiplying each entry of the vector \mathbf{x} by 2.
- $\text{VG.Dec}_{sk}(\mathbf{c}) \rightarrow \mathbf{m}$: Set $\mathbf{e} \leftarrow \mathbf{T}\mathbf{c} \pmod q$, and then output $\mathbf{m} \leftarrow \mathbf{T}^{-1} \mathbf{e} \pmod 2$.

For the decryption algorithm, recall that $\mathbf{T} \cdot \mathbf{A} = 0 \pmod q$ and therefore $\mathbf{T}\mathbf{c} = \mathbf{T}(2\mathbf{x} + \mathbf{m}) \pmod q$. If in addition all the entries of $\mathbf{T}(2\mathbf{x} + \mathbf{m})$ are smaller than q then we also have the equality over the integers $\mathbf{e} = (\mathbf{T}\mathbf{c} \pmod q) = \mathbf{T}(2\mathbf{x} + \mathbf{m}) \pmod q$, and hence $\mathbf{T}^{-1}\mathbf{e} = \mathbf{m} \pmod 2$. We have the correct decryption when all the entries of $\mathbf{T}(2\mathbf{x} + \mathbf{m})$ are smaller than $2/q$.

Additional Homomorphic Operation. Given two ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ that decrypt to $\mathbf{m}_1, \mathbf{m}_2$. Let $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_2$. For addition, we have $\mathbf{c} = \mathbf{A}(\mathbf{s}_1 + \mathbf{s}_2) + 2(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{m}_1 + \mathbf{m}_2$ which can be decrypted as $\mathbf{m}_1 + \mathbf{m}_2$ when all entries in $\mathbf{T}(2(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{m}_1 + \mathbf{m}_2)$ are smaller than $q/2$.

Theorem 3 *Any distinguishing algorithm with advantage ϵ against the CPA privacy¹ of the scheme can be converted to a distinguisher against decision-LWE $_{m,n,q}$ with binary error with roughly the same advantage at least $\epsilon/2m$.*

Proof. See Appendix D for the proof.

We can use the above variant of the GHV scheme to encrypt the binary matrices by setting uniformly random $\mathbf{S} \in \{0,1\}^{n \times m}$ and uniformly random “error matrix” $\mathbf{X} \in \{0,1\}^{m \times m}$. Specifically, we call this variant of the GHV scheme as MVGHV. Note that, our MVGHV is more efficient than the original GHV scheme: MVGHV takes time $\mathcal{O}(m \cdot n)$ to encrypt a matrix of m^2 elements comparing with $\tilde{\mathcal{O}}(m^3)$ in GHV scheme. The CPA privacy of MVGHV scheme is based on the LWE with binary error using the proof algorithm in [7].

Theorem 4. *For the parameter $n = n(\kappa)$ and $c = c(n) > 0$, let $q > 8n^{3c}$, $m = \lfloor 5n \log q \rfloor$, then the encryption scheme from above with parameters n, m, q supports n^c additions.*

Proof. See Appendix A for the proof.

Theorem 4 shows that the number of LWE with binary error samples $m = \mathcal{O}(n)$ is linear. For selection about m , it can be satisfied by taking $m = 5n \log q$ for fixed q .

3.2 Our SPIR Scheme

Now we introduce our SPIR scheme. We redefine the database (db_1, \dots, db_{tb}) , $db_i \in \{0,1\}$: assume the database can be split into nb blocks of mw words of bb bits each, such that $nb \cdot mw \cdot bb = tb$ (tb is the size of database). Namely, the nb is the count of blocks, and mw for words per block and bb for bits per word. If nb cannot be decomposed in this way, pad the database with several extra bits to make it that. We denote the database by a 2-dimensional array of words where each word is marked by two coordinates. Then we obtain the database $b = \{b_{i,j} | 1 \leq i \leq nb, 1 \leq j \leq mw\}$, the total bit size of the database is tb . First we assume that every word is a single bit, clearly $mw = 1$.

Assume that the client C wants to recover the block u that is consisted of $b_{u,j}$. We present the SPIR with four PPT algorithms $\text{SPIR} = (\text{SP.Setup}, \text{SP.Query}, \text{SP.Response}, \text{SP.Dec})$:

- $\text{SP.Setup}(1^\kappa) \rightarrow (pk, sk)$: This algorithm is to set up the system and generate the public key pk and the secret key sk . On input κ , Run the VG.KeyGen to obtain the public key $pk = \mathbf{A} \in \mathbb{Z}_q^{m \times n}$, and the secret key $sk = \mathbf{T} \in \mathbb{Z}^{m \times m}$. The pk is published to S , and the sk is kept secretly in C .

¹ The notation of CPA privacy is equivalent to the formal notation of CPA security.

- $\text{SP.Query}_{pk}(u) \rightarrow Q$: This algorithm for C is to obtain the query string. On input the public key pk and index u , compute the function $E(u)$ to obtain the index vector $\delta_{i,u} \in \{0, 1\}^{nb}$, and then spit it to $m_c = \lceil nb/m \rceil$ vectors, if δ cannot be decomposed in this way, pad the last vector with several 0 elements. Encrypt these vectors in order. Run the algorithm $\text{VGHV.Enc}_{pk}(\mathbf{m}_c) \rightarrow \mathbf{c}_c$, $c \in [m_c]$. The query message Q is these ordered vectors \mathbf{c}_c .
- $\text{SP.Response}(b, Q) \rightarrow R$: This algorithm for S is to compute the responses. On input the query message Q and the database b , compute the responses for every j from 1 to mw : $\mathbf{r}_j = \sum_{c=1}^{m_c} b_{m(c-1)+i,j} \mathbf{c}_c$ for i from 1 to m . According to the homomorphism, multiply $b_{i,j}$ by \mathbf{c} corresponds to the multiplication of $b_{i,j}$ and δ . Thus \mathbf{r}_j is the homomorphic sum of $b_{i,j} \mathbf{m}_c$. The element of vector $\delta_{i,u}$ is 1 where $i = u$, otherwise 0. The \mathbf{r}_j is the ciphertext of block b_u . The response message R is these vectors \mathbf{r}_j .
- $\text{SP.Dec}_{sk}(R) \rightarrow b_{u,j}$: This algorithm for S is to recover the block that C wants. On input the secret key sk and responses R , run $\text{VG.Dec}_{sk}(\mathbf{r}_j)$ to obtain the block u .

Multi-Words SPIR Scheme. In the above scheme, we assume that the word is a single bit. We can easily modify it to recover multi-words scheme. Instead of computing $\mathbf{A}\mathbf{s} + \mathbf{2e}$, we compute $\mathbf{A}\mathbf{s} + 2^{mw} \mathbf{e}$, $b_{u,j} \mathbf{e}_j \leftarrow b_{u,j} \mathbf{T}_j(2\mathbf{x}_j + \delta_j) \bmod q$, hence $b_{u,j} \mathbf{T}_j^{-1} \mathbf{e} \bmod 2^{mw} = b_{u,j} \delta_j$. Since q has to be large for security reasons and the noise only progresses linearly when processing the database, we can afford to start with a fairly large noise. We can utilize the same trick to obtain the “multi-words matrix retrieval”.

3.3 Our VSPIR Scheme

Before introducing our VSPIR scheme, we show the probabilistic verification process used in the work [5]. C delegates some computation F to an untrusted server S , and C wants to verify the response from S . Assume that C can precompute $F(x)$, we can define it as three procedures:

- *Setup.* Input κ , the delegated computation function $F : \{0, 1\}^n \rightarrow \{0, 1\}^m$, and the value $x \in \{0, 1\}^n$.
- *Precomputation.* C samples a random input r , computes $w = F(r)$, and stores (r, w) as secret state.
- *Delegation and Verification.* C has an input $m \in \{0, 1\}^n$. C sets $r_0 = r$ and $r_1 = m$, then samples a random bit $b \in \{0, 1\}$, and sends the pair (r_b, r_{1-b}) to S . S computes and sends $(z_0, z_1) = (F(r_0), F(r_1))$ to C . Then C accepts and recovers the response z_{1-b} if $w = z_b$.

From the work in [5], we can find that since x and r are independent and identically distributed, no malicious adversary can cheat C successfully with non-negligible probability. Our VSPIR scheme employs the similar probabilistic verification procedures as above. The main difference of probabilistic verification procedure between our VSPIR and the proposal in [5] is that we use a random

vector to mark the result that we want instead of precomputating. Now we detail the description of our VSPiR scheme $\text{VSPiR} = (\text{VSP.Setup}, \text{VSP.Challenge}, \text{VSP.Response}, \text{VSP.Verify})$:

- $\text{VSP.Setup}(1^\kappa) \rightarrow (sk, pk)$. On input 1^κ , run SP.Setup and output (sk, pk) .
- $\text{VSP.Challenge}_{sk}(u) \rightarrow Q$. On input the index u and sk , pick up a random vector $\mathbf{v} \in \{0, 1\}^m$, if the d -th element of $\mathbf{v}v_d$ is 1, then run index mapping function $\mathbf{v}_{i,u} \leftarrow E(u)$ otherwise generate a 0 vector with the same dimension. Then combine these vectors into a matrix \mathbf{I} . Run the encryption algorithm in MVGHV to encrypt the matrix using the SP.Query way to obtain the query Q .
- $\text{VSP.Response}(b, Q) \rightarrow R_k$. On input database b and query messages Q , compute $\text{SP.Response}(b, Q)$ to obtain the responses R_j .
- $\text{VSP.Verify}_{sk}(Q, R_j) \rightarrow \{b_u, \perp\}$. On input Q and the responses messages R_j , run the decryption algorithm in MVGHV and make sure whether the decryption result is our expectation. Accept and output the recovers if when the elements of the random vector's corresponding index are the same, the elements of u -th row in the matrices $b_{u,j}\mathbf{I}$ are the same as $b_{u,j}$. Otherwise, reject and output \perp .

4 Performance Analysis

In this section, we first analyze correctness, privacy and security of the proposed SPIR and VSPiR scheme. After that, we present communication complexity and computational complexity of our proposals.

4.1 Correctness, Privacy and Security

Theorem 5 (Correctness). *If the proposed MVGHV encryption scheme holds the homomorphic property for supporting polynomially many additions, then our VSPiR scheme can compute the correct retrieval information when server gives the correct response.*

Proof. See Appendix B for the proof.

Theorem 6 (Privacy). *If any distinguishing algorithm can distinguish two queries for distinct bits of database with probability at least $1/2 + \epsilon/2$, then the distinguisher can break the privacy of our VSPiR with probability $(1 + \epsilon)/2$.*

Proof. See Appendix C for the proof.

Theorem 7 (Security). *Our VSPiR scheme is convinced to be secure if no PPT adversary can deceive the client into obtaining an incorrect value responded from the server with non-negligible probability.*

Proof. See Appendix D for the proof.

4.2 Complexity

In this section, we analyze the communication complexity and computational complexity of our protocols. Recall our schemes, the public key is sent only once, it is independent of the database and the query, and it can be used for many

queries. Therefore it is customary to analyze such schemes in the public key model where sending the public key does not count towards the communication complexity.

For SPIR scheme, to encrypt the query vectors, the size of ciphertext in our encryption algorithm is composed of $\lceil nb/m \rceil$ vectors whose size is $\lceil nb/m \rceil m \log q$. Then the response size is $m \log q$. When $nb = mw = \sqrt{tb}$, for fixed security parameter, the communication complexity is $\mathcal{O}(c' \sqrt{tb})$. For VSPiR scheme, the total size of one round transform messages comes to $(m^2 + \lceil nb/m \rceil m^2) \log q$ and the communication complexity is $\mathcal{O}(c\sqrt{tb})$ (c and c' are constant).

The communication complexity is not changed in the process for multiple bits recovery, but we now can retrieve a block of mw bb bits. Furthermore, when $nb = mw = \sqrt{tb/bb}$ the communication complexity is $\mathcal{O}(c\sqrt{tb/bb})$ per an index recovered.

Now let us look at the computational complexity. To set up the system, the key generation algorithm is executed once and takes time $\mathcal{O}(1)$. If not considering any optimization algorithm, to encrypt the vector index it takes about $\lceil nb/m \rceil m$ operations and to encrypt the matrix index it takes about $\lceil nb/m \rceil m \cdot n$. When $nb = mw = \sqrt{tb}$, for fixed security parameter, the computational complexity is $\mathcal{O}(\sqrt{tb})$ and $\mathcal{O}(c\sqrt{tb})$, respectively.

5 Computer Implementations

In this section, we made a straightforward implementation of our VSPiR scheme without aiming for high levels of optimization. The timings were performed on a

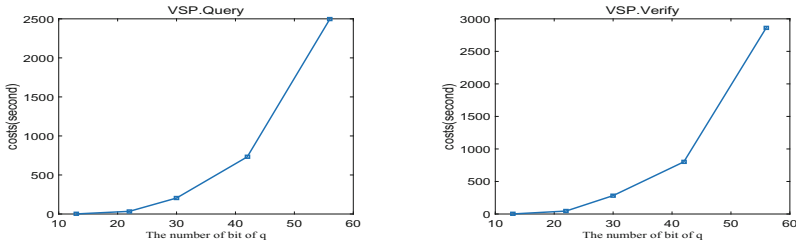


Fig. 2. The trends of the client’s costs in our VSPiR scheme

Table 2. The client’s costs in our VSPiR scheme (second).

(ψ, n)	RVSP.Query	RVSP.Verify
(13, 8)	1.012	1.010
(22, 51)	34.236	45.236
(30, 269)	204.415	282.754
(42, 531)	731.950	801.481
(56, 1563)	2494.514	2864.157

ψ : the number of bit of q

2013 ASUS (Intel(R) Core(TM) i5-3230M, 2 hyperthreaded cores at 2.60 GHz, 8 GB RAM at 1.600 GHz), on Windows (Windows 10 Home, x64 64). Our implementations are single-threaded. We used NTL for operations over \mathbb{Z}_q , matrix operations, and big number operations. We implement the algorithm VSP.Query and the decryption part of VSP.Verify. To simplify the operation, we focus on a matrix. For showing the cost of the proposed VSPiR scheme, we list the time costs in Table 2 when choosing different parameters and draw the trends of the proposed VSPiR in the Fig. 2. Note that, second can be denoted by s.

Now we consider some real scenarios. As showed in [17], the maximum bandwidth of common Internet access technologies such as the Wireless 802.11 g is 54Mbit/s, Fast Ethernet is 100Mbit/s, OC12 is 622Mbit/s. In these scenarios, the communication complexity of the server and the client can be asymmetrically negligible. Let the database be 1G bits, we set the bit of $q = 30$, $n = 269$, $nb = \sqrt{tb}$. Assume that the upload speed and download speed are all 20Mbit/s, now we can roughly compute the time cost of one round query and the response is about 31 s. The probabilistic verification process can increase the time costs slightly.

6 Conclusions

In this paper, we have proposed an efficient SPIR scheme based on the LWE with binary error assumption and a VSPiR scheme using the probabilistic verification that can work under the malicious server model. Compared with previous works, our scheme is the first practical VSPiR scheme under the malicious server model that we know of. Specifically, our VSPiR scheme has communication complexity $\mathcal{O}(c\sqrt{tb})$ that is smaller than the communication complexity of the proposal in [18].

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant 61302161, in part by the Doctoral Fund, Ministry of Education, China, under Grant 20130181120076. The corresponding author of this research Liang Zhao is supported by the International Visiting Program for Excellent Young Scholars of SCU.

A PROOF OF THEOREM 3

Proof (sketch). Let \mathcal{A} be a CPA-adversary that distinguishes between encryptions of messages of its choice with advantage ϵ . First, We construct a distinguisher \mathcal{D} with advantage at least $\epsilon/2$ between the two distributions: $\{(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{x}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}, \mathbf{s} \in \{0, 1\}^n, \mathbf{x} \in \{0, 1\}^m\}$ and $\{\text{Unif}(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)\}$. The distinguisher \mathcal{D} takes as input (\mathbf{A}, \mathbf{c}) , and runs the adversary \mathcal{A} with \mathbf{A} as the public key. Upon receiving message $\mathbf{m}_0, \mathbf{m}_1$ from the adversary, \mathcal{D} chooses at random $i \in \{0, 1\}$, returns the challenge ciphertext $2\mathbf{c} + \mathbf{m}_i \pmod q$, then outputs 1 if the adversary \mathcal{A} guesses the right i , and 0 otherwise.

On the one hand, if \mathbf{c} is uniformly random matrix then the challenge ciphertext is also uniformly random, regardless of the choice of i . Hence in this case \mathcal{D} outputs 1 with probability at most $1/2$. On the other hand, if $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{x} \pmod q$,

then the challenge ciphertext is $2\mathbf{c} + \mathbf{m} = \mathbf{A}\mathbf{s}' + 2\mathbf{x} + \mathbf{m} \pmod{q}$. By assumption \mathcal{A} will guess the right i with probability $(1 + \epsilon/2)$. Finally, a standard hybrid argument can be used to convert the distinguisher \mathcal{D} from above to a decision-LWE $_{m,n,q}$ with binary error distinguisher with advantage $\epsilon/2m$.

B PROOF OF THEOREM 4

Proof. Let vector $\mathbf{c} = \sum_{i=1}^{\ell} (\mathbf{A}\mathbf{s}_i + 2\mathbf{x}_i + \mathbf{m}_i)$ be obtained by adding $\ell \leq n^c$ ciphertexts. Recall that every row of \mathbf{T} has Euclidean norm at most $20n \log q$. Then every entry of $\sum_{i=1}^{\ell} \mathbf{T}\mathbf{x}_i$ is at most $20\ell n \log q$. All the \mathbf{m}'_i 's are binary so each entry of $\mathbf{T}\mathbf{m}$ is at most $20\ell n \log q$. Hence the each entry in $\mathbf{T}(2\mathbf{x} + \mathbf{m})$ is bounded by $40\ell n \log q < 4n^{3c} < q/2$ for some q . Now we can decrypt \mathbf{c} to recover the correct value.

C PROOF OF THEOREM 5

Proof. On the one hand, our proposed similar probabilistic verification procedures provide the correctness of the response message from server. On the other hand, as introduced in the preliminaries section, the GHV scheme is correct regard to additive homomorphic operation, our MVGHV scheme follows the same property. Then our VSPiR using the MVGHV scheme has the correctness when the server provides the correct response.

D PROOF OF THEOREM 6

Proof (sketch). Let \mathcal{A} be a CPA-adversary that distinguishes between encryptions of queries of its choice with advantage ϵ , we first construct a distinguisher \mathcal{D} with advantage at least $\epsilon/2$ between two queries: the query the client wants and a query chosen randomly. The distinguisher \mathcal{D} takes as input a pair of matrices (\mathbf{A}, \mathbf{C}) , and runs the adversary \mathcal{A} with \mathbf{A} as the public key. Upon receiving message $\mathbf{B}_0, \mathbf{B}_1$ from adversary, \mathcal{D} chooses at random $i \in \{0, 1\}$, returns the challenge ciphertext encrypted by MVGHV scheme, then outputs 1 if the adversary \mathcal{A} guesses the right i , and 0 otherwise. On the one hand, if \mathbf{C} is the random query ciphertext, then \mathbf{C} is equivalent to be a uniformly distribution. In this case \mathcal{D} outputs 1 with probability at most $1/2$. On the other hand, If \mathbf{C} encrypts the query the client wants, by assumption \mathcal{A} will guess the right i with probability $(1+\epsilon)/2$. Then the privacy of our VSPiR is based on the decision-LWE with binary error assumption.

E PROOF OF THEOREM 7

Proof. To verify the correctness of the response, we sample a random vector replace the index matrix. If an algorithm can break the privacy of our scheme with probability $\text{negl}(\kappa)$, the algorithm can distinguish queries for block X_1 and block X_2 . Then the adversary can deceive the client into obtaining an incorrect value with probability $\text{negl}(\kappa)/2^m$.

References

1. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS, pp. 75–86 (2009)
2. Beimel, A. Ishai, Y.: Private information retrieval: a primer. www.cs.bgu.ac.il/beimel/Papers/PIRsurvey.ps
3. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2001)
4. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: FOCS, pp. 41–50 (1995)
5. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_26
6. 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
7. Gentry, C., Halevi, S., Vaikuntanathan, V.: A simple BGN-type cryptosystem from LWE. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 506–522. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_26
8. Gentry, C. Peikert, C. Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206. ACM (2008)
9. Goldberg, I.: Improving the robustness of private information retrieval. In: 2007 IEEE Symposium on Security and Privacy, pp. 131–148 (2007)
10. Johannes, A., Buchmann, F.G., Rachel, P., Thomas W.: On the hardness of LWE with binary error: revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In: AFRICACRYPT, pp. 24–43 (2016)
11. Kumar, M., S., Sarkar, P.: Symmetrically private information. In: INDOCRYPT, pp. 225–236 (2000)
12. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_26
13. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_2
14. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC, pp. 333–342 (2009)
15. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34 (2009). Preliminary version in STOC 2005
16. Vannet, T., Kunihiro, N.: Private information retrieval with preprocessing based on the approximate GCD problem. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 227–240. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_14
17. Wikipedia. <https://en.wikipedia.org/wiki/Bandwidthx>
18. Zhang, L.F., Safavi-Naini, R.: Verifiable multi-server private information retrieval. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 62–79. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07536-5_5

Full Paper Session X: Applied Cryptography



Cryptographic Password Obfuscation

Giovanni Di Crescenzo^(✉), Lisa Bahler, and Brian Coan

Perspecta Labs, Basking Ridge, NJ 07920, USA
{gdicrescenzo,lbahler,bcoan}@perspectalabs.com

Abstract. We consider cryptographic program obfuscation of point functions (i.e., functions parameterized by a secret s that, on input a string x , return 1 if $x = s$ and 0 otherwise). We achieve the following notable results: (1) the first *efficient* point function obfuscator for arbitrarily large as well as very short secrets, provable *without random oracle assumptions*; (2) the first efficient and provably-secure (under the existence of one-way permutations or block ciphers that have no theoretical attack faster than exhaustive key search) real-life applications built on top of these obfuscators, such as: (a) entity authentication via password verification; (b) entity authentication via passphrase verification; and (c) password management for multi-site entity authentication.

Keywords: Program obfuscation · Password authentication

1 Introduction

Program obfuscation is the problem of modifying a computer program so to hide sensitive details of its code without changing its input/output behavior. While this problem has been known for several years in computer science, only in the last 15 years, researchers have considered the problem of provable program obfuscation; that is, the problem of program obfuscation, where sensitive code details are proved to remain hidden under a widely accepted intractability assumption (such as those often used in applied cryptography). Early results in the area implied the likely impossibility of constructing a single program capable of obfuscating an arbitrary polynomial-time program into a virtual black box [3]. Moreover, most recent results show the possibility of constructing different obfuscators for restricted families of functions, such as point functions (and extensions of them), under more or less accepted hardness assumptions (see, e.g., [2, 5, 10, 12, 16, 19]). Point functions can be seen as functions that return 1 if the

Supported by the Defense Advanced Research Projects Agency (DARPA) via U.S. Army Research Office (ARO), contract number W911NF-15-C-0233. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, ARO or the U.S. Government.

input value is equal to a secret value stored in the program, and 0 otherwise. Although conceptually simple, point functions come with surprisingly interesting applicability to real-life problems. As often suggested in the literature (see, e.g., [16, 19]), point function obfuscation might be applicable to the password verification function in very commonly used login/password authentication methods.

In this paper we carry out an exploration of this suggestion. Our main result is a *practical* method for cryptographic password obfuscation *under standard cryptographic assumptions* (and, specifically, without using a random oracle assumption or a heuristic construction for a multilinear map), such as the existence of a one-way permutation or, of a block cipher for which there are no theoretical attacks faster than exhaustive key search. Known practical methods include the well-known password hashing method (i.e., at registration, server stores the cryptographic hash of the user's password; at authentication, server checks that hash of the provided password is equal to the stored hash). This method was analyzed in a cryptographic program obfuscation context by [16], but is however only proved secure under the random oracle assumption. (Note that this assumption, although often accepted by practitioners, has been declared as almost certainly false in its generality [9], and is especially troublesome in light of the less and more recent attacks to widely considered or used hash functions such as MD4, MD5 and SHA1). By now, there are several known obfuscators for point functions that do not make the random oracle assumption, but they all assume secrets much longer than typical passwords. The only exception and the closer result in the literature to ours is an elegant construction of a perfectly one-way function from [10], which could be used to construct a point function obfuscator under the assumption of claw-free permutations. We note that this construction is not practical as it is estimated 4–5 orders of magnitude less efficient than the one we propose.

As all point function obfuscators in the literature use secrets of length about equal to the factoring-type security parameter (e.g., 2048), to increase capability to commonly used passwords and passphrases as well as secrets of arbitrary lengths, we have designed two new methods: (a) a new hash function that transforms these point function obfuscators in the literature so that they can work with *arbitrarily longer secrets*; (b) a new (multi-bit-output) point function obfuscator, which can work with secrets *as short as the symmetric cryptography security parameter* (e.g., 128). Our obfuscators satisfy a computational notion of functional correctness (i.e., no efficient adversary can find an input on which the obfuscated program differs from the original program), and a rather strong notion of obfuscation security (i.e., the obfuscated program is efficiently simulatable). An underlying technical contribution is the construction of an *efficient second-preimage-resistant extractor* that is simultaneously a second-preimage-resistant hash function, a pairwise almost-independent hash function, and has efficient instantiations from a single efficient cryptographic primitive. Our efficiency claims on this extractor and its resulting obfuscators are substantiated by implementations and performance results on commodity computing resources. Finally, we demonstrate that program obfuscators for point functions are usable in the following real-life applications: *password verification* (obfuscating a server's

algorithm verifying if a client’s input string is equal to the client’s previously registered password), *passphrase verification* (as for password verification, but with the variant that the client has registered a passphrase containing only structured text); and *password manager* (obfuscating a server’s verification and retrieval algorithms that verify the client’s master password or passphrase, and retrieve a client’s previously registered password for a specific server).

In particular, we have modified code on an open-source password manager (Pass, based on gpg2) to accommodate our (multi-bit-output) point function obfuscator instead of their current cryptographic solution (whose obfuscation properties can at best be proved using a random oracle assumption). The overall resulting runtime of a specific password retrieval on the modified application is less than 3% slower than the same operation on unmodified Pass. Solving these password and passphrase obfuscation problems without using a random oracle assumption are natural problems that have remained unsolved for decades. Details of our real-life application results are discussed in Appendix A.

2 Definitions and Preliminaries

In this section we first recall basic definitions and slightly modify the theory-oriented definition of program obfuscators into a practice-oriented definition that better fits a large class of obfuscator implementations (including ours for point functions) and where the correctness property only holds in a computational sense (i.e., even against a possibly malicious polynomial-time adversary). Finally, we discuss security notions for point function obfuscators.

Security Parameters. In our constructions and concrete security analysis, will use two types of *security parameters*, described below:

1. the ‘*factoring-based*’ *security parameter*, a global parameter λ_f , currently set to 2048, that is typically used to determine key lengths in asymmetric cryptographic primitives (e.g., public-key encryption) proved secure under the hardness of number theoretic problems related to factoring and/or discrete logarithm problem; and
2. the ‘*symmetric-cryptography*’ *security parameter*, a global parameter λ_s , currently set to 128, that is most typically used to determine key/output lengths in several symmetric cryptographic primitives (e.g., block ciphers).

Point Functions. We consider *families of functions* as families of maps from a domain to a range, where maps are parameterized by some values chosen according to some distribution on a parameter set. Let pF be a family of functions $f_{par} : Dom \rightarrow \{0, 1\}$, where $Dom = \{0, 1\}^n$, and each function is parameterized by value par from a parameter set $Par = \{0, 1\}^n$, for some *length parameter* n . We say that pF is the *family of point functions* if on input $x \in Dom$, and *secret value* $y \in Par$, the point function f_y returns 1 if $x = y$ and 0 otherwise. When we assume an efficiently samplable distribution of secret values $y \in Par$, we define the (rounded) *min entropy parameter* of pF as the largest integer t

such that each $y \in Par$ is sampled with probability $\leq 2^{-t}$. The *family mbpF of multi-bit-output point functions* is defined as follows: on input $x \in Dom$, secret value $y \in Par$, and output value z , the function f_y returns z if $x = y$ and 0 otherwise.

Program Obfuscators: Formal Definitions. To define a cryptographic program obfuscator for a class of functions F , we consider a pair of efficient algorithms with the following syntax. On input function parameters $fpar$, including a description $desc(f)$ of function $f \in F$, the *obfuscation generator* $genO$ returns generator output $gpar$. On input a description $desc(f)$ of function $f \in F$, generator output $gpar$, and evaluator input x , the *obfuscation evaluator* $evalO$ returns evaluator output y .

Informally, we would like to define an obfuscator for the class pF of point functions as any such pair of algorithms satisfying some *functionality correctness* property (i.e., the obfuscated program computes the same function as the original program), some *efficiency* property (i.e., the obfuscated program is not much slower than the original program), and some *obfuscation security* property (i.e., the obfuscated program hides any sensitive information about the original program which is not computable by program evaluation). Here, we actually consider a slightly relaxed notion of the functionality correctness property, according to which the obfuscated program can return an output different from the original program for some of the inputs; however, these inputs are hard to find, even to an efficient algorithm that has access to the program's secret value. Furthermore, we discuss some of the security notions in the literature, and eventually formally define the strongest known notion (implicit in [3] and saying, informally speaking, that any efficient adversary's view of the obfuscated program can be efficiently simulated and thus the adversary learns nothing more than an upper bound on the program size), specialized to the class of point functions pF with secret distributions having high min-entropy. We now proceed more formally.

We say that the pair $(genO, evalO)$ is a *cryptographic program obfuscator* for the class pF of point functions if it satisfies the following:

1. (Computational correctness): For any f_s in pF , with function parameters $fpar = (s, desc(pF))$, and any efficient algorithm A , the event $f_s(x') \neq y$ holds with probability δ , for some negligible (or very small) δ , where x', y are generated by the following probabilistic steps:
 - $gpar \leftarrow genO(fpar)$,
 - $x' \leftarrow A(gpar, fpar)$,
 - $y \leftarrow evalO(gpar, x')$.
2. (Polynomial Blowup Efficiency): There exists a polynomial p such that for all f_s in pF , the running time of $evalO(gpar, \cdot)$ is $\leq p(|f_s|)$, where $|f_s|$ denotes the size of the (smallest) boolean circuit computing f_s .
3. (Adversary view simulation security): Given any high min-entropy distribution D returning an n -bit secret, there exists a polynomial-time algorithm Sim such that for any function f_s , $|s| = n$, in the class pF of point functions, with black-box access to f_s such that for all f_s in pF with parameters $fpar$, the distributions D_{view} and D_{sim} are computationally indistinguishable, where

- $D_{view} = \{s \leftarrow D; gout \leftarrow genO(s, desc(pF)) : gout\}$,
- $D_{sim} = \{s \leftarrow D; gout \leftarrow Sim(1^{|s|}, desc(pF)) : gout\}$.

Other security notions considered in the literature include *adversary output black-box simulation* (where the simulator has also access to a black-box computing the program [3] and targets simulating the adversary's output bit), *real-vs-random indistinguishability* (where no efficient adversary can distinguish the obfuscation of the function f from an obfuscation of a random function in the class F) [5], and *indistinguishability obfuscation* (where no efficient adversary can distinguish the obfuscation of any two circuits computing the same function f) [3]. We note that an obfuscator satisfying the adversary view black-box simulation security notion also satisfies these latter 3 security notions.

Known Point Function Obfuscators. The obfuscator in [16] for the family of point functions satisfies adversary view black-box simulation under the random oracle assumption. This obfuscator essentially consists of computing a cryptographic hash of the secret, similarly as typically done for passwords in real-life systems. A previous result of [7], although formulated as an oracle hashing scheme, can be restated as an obfuscator satisfying a strong variant of real-vs-random indistinguishability under the Decisional Diffie Hellman assumption. The obfuscator in [19] satisfies adversary output black-box simulation under the existence of a strong type of one-way permutations. Moreover, one of the obfuscators in [5], based on any deterministic encryption scheme, satisfies real-vs-random indistinguishability, and has several instantiations. This follows as deterministic encryption schemes can be built using the hardness of the learning with rounding problem [20] or the existence of lossy trapdoor functions [6], and the latter have been built using any one of many group-theoretic assumptions (see, e.g., [13]). Some of the resulting obfuscators have efficient implementations [12]. Finally, an obfuscator was given in [2] using the hardness of the learning with error problem.

All results mentioned so far either make the random oracle assumption or work for secret distributions not significantly different than uniform. The only obfuscator working for arbitrary secret distributions of high min-entropy can be obtained using a result from [10] on perfectly one-way functions, constructed assuming the existence of claw-free permutations. This result is far from having an efficient implementation.

Our goal in the rest of this paper is to show an obfuscator for point functions that works for arbitrary secret distributions of high min-entropy, without making the random oracle assumption, and resulting in an efficient implementation.

Families of One-Way α -Permutations. The term *efficient* is used for running time in both a practical and theoretical sense, as needed. We say that a family of functions $\{F\}$ is *efficiently samplable* if there exists an efficient algorithm randomly choosing a function F from the family, and is *efficiently computable* if there exists an efficient algorithm that evaluates any function F from the family. We say that a family of functions $\{F\}$, is a family of *α -permutations* if the probability that, for a randomly chosen x , $F(x)$ has > 1 preimages, is $< \alpha$.

Families of Pairwise-Independent Hash Functions. We say that a family of hash functions $\{H_{m,n}\}$, where $H_{m,n} : \{0,1\}^m \rightarrow \{0,1\}^n$ is *pairwise δ -independent* if for any $x_0 \neq x_1 \in \{0,1\}^m$, and any $y_0, y_1 \in \{0,1\}^n$, it holds that $\text{Prob}[H(x_0) = y_0 \wedge H(x_1) = y_1] \leq \delta + 2^{-2n}$. We say that family $\{H_{m,n}\}$ is *pairwise independent* if it is pairwise δ -independent, for $\delta = 0$. Constructions for pairwise-independent hash functions include a random one-degree polynomial in a Galois field or a random one-degree polynomial modulo a prime [11], where by a random polynomial we denote a polynomial with coefficients randomly chosen in their domain set. All these constructions are efficiently sampleable and efficiently computable.

Families of Second-Preimage-Resistant Hash Functions. This cryptographic primitive was introduced in [17], under the name of universal one-way hash functions, and have also been called target-collision-resistant hash functions since [4] or second-preimage-resistant hash functions. We say that a family of functions $\{h \mid h : \{0,1\}^a \rightarrow \{0,1\}^b\}$ is *second-preimage-resistant* over $\{0,1\}^a$ if it satisfies the following three properties: (1) h is efficiently sampleable from its family; (2) every h in the family is efficiently computable; and (3) no efficient adversary can win, except with very small probability, in the following game: first, the adversary picks an input z , then a random function h is sampled from its family; finally, the adversary, given $h(z)$, wins the game if it finds an input x such that $h(x) = h(z)$. The first constructions for such families of functions were proposed in [17], based on families of one-way permutations with varying domain sizes and any family of pairwise-independent hash functions. Later, more practical constructions were proposed in [4,18], based on collision-intractable hash functions. Generally speaking, second-preimage-resistant hash functions may or may not satisfy pairwise-independence properties.

Randomness Extractors. The *statistical distance* between two distributions D_1, D_2 over the same space S is defined as $sd(D_1, D_2) = \frac{1}{2} \sum_{x \in S} |\text{Prob}[x \leftarrow D_1] - \text{Prob}[x \leftarrow D_2]|$. We say that distributions D_1, D_2 are *δ -close* if it holds that $sd(D_1, D_2) \leq \delta$. We say that a distribution D is *δ -close to uniform*, or, briefly, *δ -uniform*, if it holds that $sd(D, U) \leq \delta$, where U denotes the uniform distribution over the same space S . The *min-entropy* of a distribution D is defined as $H_\infty(D) = \min_x \{-\log_2(\text{Prob}[x \leftarrow D])\}$. A function $\text{Ext}: \{0,1\}^a \times \{0,1\}^b \rightarrow \{0,1\}^c$ is called a (k, ϵ) -*extractor* if for any distribution D on $\{0,1\}^a$ with min-entropy at least k , the distribution $N(D)$ is ϵ -uniform, where $N(D) = \{x \leftarrow D; e \leftarrow \{0,1\}^b; y = \text{Ext}(x, e) : (e, y)\}$. The *leftover hash lemma* [14] says that if $\{H_{m,n}\}$ is a family of pairwise-independent hash functions, value x is drawn according to a distribution D such that $H_\infty(D) \geq k$, and $n \geq k - 2 \log(1/\epsilon)$, then the function $\text{Ext}(x, H_{m,n})$ defined as $y = \text{Ext}(x, H_{m,n}) = H_{m,n}(x)$ is a (k, ϵ) -extractor. By inspection of the proof in [15], we see that it can be directly extended to families of pairwise δ -independent hash functions, as follows.

Lemma 1. *For any $\delta > 0$, if $\{H_{m,n}\}$ is a family of pairwise δ -independent hash functions, value x is drawn according to a distribution D such that $H_\infty(D) \geq k$, and $n \leq k - 2 \log(1/\epsilon)$, for some $\epsilon \leq (1/2) \log(1/\delta)$, then the function $\text{Ext}(x, H_{m,n})$ defined as $y = \text{Ext}(x, H_{m,n}) = H_{m,n}(x)$ is a (k, ϵ) -extractor.*

We say that a function $\text{Ext}: \{0, 1\}^a \times \{0, 1\}^b \rightarrow \{0, 1\}^c$ is a *second-preimage-resistant (k, ϵ) -extractor* if it is both a second-preimage-resistant hash function over $\{0, 1\}^a$ and a (k, ϵ) -extractor.

3 An Efficient Second-Preimage-Resistant Extractor

In this section we construct an efficient second-preimage-resistant extractor, or actually a family of hash functions which satisfies the following desirable combination of functionality, efficiency and security properties:

1. it achieves arbitrarily large compression, in that it maps an arbitrarily-long input string to a fixed-length output string;
2. it is an almost pairwise-independent hash function;
3. it is a one-way function with second-preimage resistance;
4. in addition to elementary operations, it only uses, as a black-box, a hash function satisfying above properties 2 and 3, and achieving small and fixed compression (specifically: it maps a fixed-length input string to a fixed-length output string, where the difference between the input string's length and the output string's length can be any small constant ≥ 1).

Properties 1 and 4 (resp., 2 and 3) are used to satisfy functionality correctness and efficiency (resp., security) requirements. The closest constructions to ours from the literature only satisfy 3 out of 4 of the listed properties, as follows: two constructions in [17] missed properties 1 or 4, and a construction from [4, 18] missed property 2.

Formally, we achieve the following

Theorem 1. *Let $t_{F, \text{sample}}$ (resp., $t_{F, \text{eval}}$) denotes the running time to sample (resp., evaluate) a function F . Let $\{aF \mid aF : \{0, 1\}^b \rightarrow \{0, 1\}^b\}$ be a family of one-way α -permutations, and let $\{piH \mid piH : \{0, 1\}^a \rightarrow \{0, 1\}^b\}$ be a family of pairwise δ -independent hash functions. There exists (constructively) a family $\{sprH \mid sprH : \{0, 1\}^{\ell(a-b)} \rightarrow \{0, 1\}^b\}$ of second-preimage-resistant (k, ϵ) -extractors such that*

- $b \leq k - 2 \log(1 + \epsilon)$ and $\epsilon \leq (1/2) \log(1/\delta')$, for $\delta' = \ell(\delta + \alpha)$
- $t_{sprH, \text{sample}} = O(\ell(t_{piH, \text{sample}}) + t_{aF, \text{sample}})$, and
- $t_{sprH, \text{eval}} = O(\ell(t_{aF, \text{eval}} + t_{piH, \text{eval}} + t_{piH, \text{sample}}) + t_{aF, \text{sample}})$.

The function $sprH$ obtained in the proof of Theorem 1 will be applied to obtain the following two important new results: (1) in Sect. 4, it will be used in combination with the obfuscators from [5–7, 13], and [19], to design efficient obfuscators for point functions with secret length higher than the factoring-type security

parameter (e.g., 2048); (2) in Sect. 5 it will be used to design an efficient obfuscator for multi-bit-output point functions with secret length greater than or equal to the symmetric-cryptography security parameter (e.g., 128). The rest of this section is devoted to proving Theorem 1.

Informal Description of Function $sprH$: Our goal is to define a family of functions, denoted as $sprH$, that satisfies the above properties 1–4. One higher-level view of our construction looks similar to the linear hash construction in [4, 18], and its lower-level component looks similar to a function from [17]. However, some technical differences with these papers actually allow us to achieve all 4 desired properties; most importantly:

1. $sprH$ processes an arbitrarily long input by repeatedly applying an inner function with the same domain and codomain sizes (instead, in [17] domain and codomain sizes vary). This approach is important to achieve properties 1 and 4.
2. in $sprH$ the inner function used at each iteration is both a second-preimage-resistant function and a pairwise almost-independent hash function (as opposed to only a collision-intractable hash function, as in [4, 18]). This approach is important to achieve properties 2 and 3.

Formal Description: Let $desc(F)$ denote a conventional encoding of function F , and let a, b denote positive integers such that $a > b$ and $a - b \geq 1$ is a small constant. The construction for $sprH$ uses a pairwise δ -independent hash functions $piH : \{0, 1\}^a \rightarrow \{0, 1\}^b$, and a one-way α -permutations $aP : \{0, 1\}^b \rightarrow \{0, 1\}^b$. We define function $sprH : \{0, 1\}^* \rightarrow \{0, 1\}^b$, as follows.

Input to $sprH$: string $x = x_1 | \dots | x_\ell$, where $x_i \in \{0, 1\}^{a-b}$, for $i = 1, \dots, \ell$.

Instructions for $sprH$:

1. Set $u_0 = 0^b$
2. Randomly sample a one-way α -permutation aP
3. For $i = 1, \dots, \ell$,
 - randomly sample a pairwise δ -independent hash function piH_i
 - compute $v_i = aP(u_{i-1} | x_i)$ and $u_i = piH_i(v_i)$
4. Return: $(u_\ell, desc(aP), desc(piH_1), \dots, desc(piH_\ell))$.

The running times $t_{sprH, sample}$ and $t_{sprH, eval}$ claimed in Theorem 1 are verified by algorithm inspection, observing that $sprH$ can compress arbitrarily long inputs into b -bit outputs, and that it invokes ℓ times a single function $piH(aP(\cdot))$ compressing a -bit outputs to b -bit outputs. In what follows, we show that $sprH$ is a second-preimage-resistant (k, ϵ, δ') -extractor with the parameters in Theorem 1, by showing that it is both a second-preimage-resistant hash function over $\{0, 1\}^a$ and a pairwise δ' -independent hash function.

The proof that $sprH$ is a second-preimage-resistant hash function directly follows by applying results in [4, 17, 18], as follows. First, we observe that the function obtained by cascading a one-way α -permutation aP with a pairwise δ -independent hash function piH , is a second-preimage-resistant hash function.

This follows directly by Lemma 2.2 in [17], which proves the exact same result when $\alpha = 0, \delta = 0$ and $a - b = 1$. We observe that no technical difficulty is encountered in extending this proof to values of α, δ that are negligible or very small and a value of $a - b$ that is a small constant (or even logarithmic in the security parameter). Because of this observation, we note that *sprH* can be considered as the linear hash iterated application of a second-preimage-resistant hash function, as in the linear hash construction from [4, 18]. In particular, we can apply Theorem 5.3 from [4] which proves our desired statement; i.e., the linear hash construction transforms a second-preimage-resistant hash function from a -bit strings to b -bit strings into a second-preimage-resistant hash function from arbitrary-length strings to b -bit strings.

The proof that *sprH* is a pairwise δ' -independent hash function is obtained by induction over ℓ . The base case directly follows by observing that the assumptions that function $\text{pi}H_1$ is pairwise δ -independent and that function aP is an α -permutation imply that the composed function $\text{pi}H_1(aP(\cdot))$ is a pairwise δ' -independent hash function, for $\delta' = \alpha + \delta$. The inductive case follows by combining the induction hypothesis with the fact that at the ℓ -th iteration, function *sprH* computes u_ℓ using function $\text{pi}H_\ell(aP(\cdot))$ for an independently chosen pairwise δ -independent hash function $\text{pi}H_\ell$.

Implementation: Primitive Setting. Families of pairwise-independent hash functions $\text{pi}H_i$ can be implemented as in Sect. 2. Function aP can be instantiated in 3 ways:

1. setting $n = 2048$, and using exponentiation modulo a prime; that is, $aP_{g,p}(x) = g^x \bmod p$, where publicly available parameters p, g are as follows: p is an $(n + 1)$ -bit prime and g is a generator of \mathbb{Z}_p^* ;
2. using a length-preserving collision-intractable hash function $\text{c}ih_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ for which no theoretical attacks (faster than birthday attacks) are known, and assuming such a function is a one-way α -permutation, for a value α negligible in n or very small; that is, $aP_{\text{c}ih,k}(x) = \text{c}ih_k(x)$;
3. as a block cipher $bc : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ for which no theoretical attacks (faster than exhaustive search attacks) are known, to be run on a fixed, but randomly chosen, input block r , and assuming that the resulting function $bc(\cdot, r)$ is a one-way α -permutation over the set of block cipher keys, for α negligible in n or very small; that is, $aP_{bc,r}(x) = bc(x, r)$.

In our implementation, we used the 3rd option for efficiency reasons, and based on the observation that function $aP_{bc,r}(x)$, mapping the set of keys of the block cipher to the cipher's output, is indeed expected to be a one-way α -permutation. This observation is based on the fact that if function $aP_{bc,r}(x)$ were not close to a one-way α -permutation, a theoretical attack exhaustively searching for any of the colliding keys would be possible. Note that such an attack would be faster than exhaustive key-search, thus giving a theoretical break of the block cipher.

4 Obfuscators for Point Functions with Larger Secrets

In this section we show how to obtain point function obfuscators where the obfuscated secret value can have length and min entropy parameters arbitrarily greater than the factoring-type security parameter, starting from a point function obfuscator where the obfuscated secret value has fixed length and min-entropy parameter, which we already know how to build. Formally, we obtain the following

Theorem 2. *Let $\ell_a, e_a, \ell_u, \epsilon$ be integers such that $\ell_u + 2\epsilon \leq e_a \leq \ell_a$ and $\epsilon \geq \lambda_s$, let $\text{spr}H$ be a second-preimage-resistant (ℓ_a, ϵ) -extractor, and let $(\text{gen}O_u, \text{eval}O_u)$ be a cryptographic program obfuscator for the family of point functions with ϵ -uniformly distributed ℓ_u -bit secret values. Then there exists (constructively) a cryptographic program obfuscator $(\text{gen}O_a, \text{eval}O_a)$ for the family of point functions with respect to ℓ_a -bit secrets drawn from any distribution of min-entropy e_a .*

An important consequence of Theorem 2 is that any one of the point function obfuscators in [7], [5, 6, 13], or [19] can be extended to obtain a point function obfuscator that works for secret values with arbitrarily larger length and drawn from arbitrary distributions of min entropy larger than the factoring-type security parameter.

Informal and Formal Descriptions: The basic idea of the transformation underlying Theorem 2 follows a ‘hash-and-obfuscate’ paradigm, analogously to the much studied ‘hash-and-sign’ paradigm used for the design of digital signature schemes for large messages. This paradigm goes through two steps: first, the input is hashed using a second-preimage-resistant extractor, which we will implement using the construction $\text{spr}H$ from Sect. 3; then, the extractor’s output is processed through the obfuscator with fixed length parameter, which can be instantiated using any one of the schemes from the literature (e.g., [5, 5–7, 13, 20], [10] or [19].) The resulting scheme satisfies computational functionality correctness, and the same adversary view simulation obfuscation notion as the used obfuscator for fixed-length secrets. We now proceed more formally. The construction for $(\text{gen}O_a, \text{eval}O_a)$ uses the family of efficiently samplable functions $\text{spr}H : \{0, 1\}^* \rightarrow \{0, 1\}^b$ from Sect. 3, which are simultaneously second-preimage-resistant hash functions and pairwise δ' -independent extractors, and an obfuscator $(\text{gen}O_u, \text{eval}O_u)$ for the family of point functions with length parameter ℓ_u and secret values with almost uniform distribution.

Input to $\text{gen}O_a$: parameters $1^{e_u}, 1^{\ell_u}, \epsilon$, secret value $s \in \{0, 1\}^{\ell_a}$

Instructions for $\text{gen}O_a$:

1. Randomly sample function $\text{spr}H : \{0, 1\}^{\ell_a} \rightarrow \{0, 1\}^{\ell_u}$
2. Compute $v = \text{spr}H(s)$
3. Compute $\text{out}_u = \text{gen}O_u(v)$
4. Return: $\text{out}_a = (\text{desc}(\text{spr}H), \text{out}_u)$.

Input to $evalO_a$: input value $x \in \{0, 1\}^{\ell_a}$ and the output from $genO_a$, containing the description $desc(sprH)$ of function $sprH$ and the output out_u from $genO_u$.

Instructions for $evalO_a$:

1. Compute $v' = sprH(x)$
2. Return: $evalO_u(v')$.

Proofs are omitted due to space restrictions.

5 Obfuscators for Multi-bit-output Point Functions With Shorter Secrets

In this section we describe an obfuscator, denoted as $(genO_{mb}, evalO_{mb})$, for the family of multi-bit-output point functions, where secrets can have a shorter length parameter than in our previous implementations, which implies applicability to the obfuscation of passphrases and even passwords. More specifically, this obfuscator differs from analogue results in the literature and in previous sections, in the following properties:

1. it works for a generalized type of point functions: multi-bit-output point functions, whose output can be a long string, instead of a bit;
2. it works for a length parameter that can be arbitrarily chosen as \geq the symmetric-cryptography security parameter (i.e., 128), instead of the factoring-type security parameter (i.e., 2048);
3. its obfuscation property can be based on the security of a symmetric cryptography primitive (i.e., a block cipher or a cryptographic hash function), instead of a number theory problem typically applied to construct an asymmetric cryptography primitive.

Formally, we achieve the following

Theorem 3. *Let $\ell_o, \ell_s, k, \epsilon$ be integers such that $k \leq \ell_s$. Also, let $sprH$ be a second-preimage-resistant (k, ϵ) -extractor and let $(KeyGen, Enc, Dec)$ be a secure symmetric encryption scheme. Then there exists (constructively) a cryptographic program obfuscator $(genO_{mb}, evalO_{mb})$ for the family of multi-bit output point functions $mbpF$ with ℓ_o -bit outputs and ℓ_s -bit secrets drawn from any distribution of min-entropy k .*

We note that in the above theorem we are trading off some slightly, but not significantly, reduced confidence in the security assumptions (as indicated in item 3 of the above list), to achieve increased functionality power (as indicated in items 1 and especially item 2 of the above list). Indeed the property in item 1 can be obtained without resorting to symmetric cryptography primitives (see, e.g., [8]), but this comes with decreased obfuscator's efficiency. The (most interesting) property in item 2 was unknown and is the one that allows applications to passphrase and password obfuscation, as further detailed in Appendix A.

Formal Description: Let $|$ denote string concatenation, and let $sprH$ denote a second-preimage-resistant (k, ϵ) -extractor (such as the one constructed in Sect. 3). Also, let $(KeyGen, Enc, Dec)$ be a symmetric encryption scheme with the following syntax: on input a unary string 1^n denoting the symmetric encryption security parameter, $KeyGen$ returns an n -bit random key ; on input key and message m , encryption algorithm Enc returns ciphertext c ; on input key and ciphertext c , decryption algorithm Dec returns message m . Our construction of $(genO_{mb}, evalO_{mb})$ combines the extractor $sprH$ with the encryption scheme $(KeyGen, Enc, Dec)$, as follows.

Input to $genO_{mb}$: security parameters $1^n, 1^{n_0}, 1^\epsilon$, entropy parameter k , secret value $s \in \{0, 1\}^{\ell_s}$, output value $w \in \{0, 1\}^{\ell_o}$

Instructions for $genO_{mb}$:

1. uniformly and independently choose $r \in \{0, 1\}^{n_0}$
2. compute $key = sprH(r|s)$, where $key \in \{0, 1\}^n$
3. compute $v = Enc(key, w|0^{n_0})$
4. set $gpar = (r, v)$ and return: $gpar$.

Input to $evalO_{mb}$: security parameters $1^n, 1^{n_0}, 1^\epsilon$, entropy parameter k , the pair (r, v) returned by $genO_{mb}$, and input value $x \in \{0, 1\}^\ell$

Instructions for $evalO_{mb}$:

1. compute $key' = sprH(r|x)$, where $key' \in \{0, 1\}^n$
2. compute $(w'|w'') = Dec(key', v)$
3. if $w'' = 0^{n_0}$ return w' else return 0

Proofs and performance analysis are omitted due to space restrictions.

6 Conclusions

We showed for the first time how to *efficiently* obfuscate passwords, passphrases and password managers, *without a random oracle assumption*. Our obfuscator can work with passwords and passphrases of practical lengths. Even if we expect practitioners to continue using the simpler to implement construction based on cryptographic hashing of a password, our construction gives confidence that the impact of any future attacks to cryptographic hash functions can be significantly limited by a simple protocol design change.

A Applications: How to Obfuscate Password Verification, Passphrase Verification and Password Managers

In this section we show how to use the obfuscators from previous sections to obfuscate, using standard cryptographic assumptions (and specifically not using the random oracle assumption), software applications commonly used in real-life, such as password verification, passphrase verification, and password managers.

Obfuscation of Password Verification. Entity authentication based on shared secrets is one important class of system applications that seem to significantly rely on obfuscation, as we now explain. Consider the typical scenario of a client and a server who use a secret to let the server successfully register and later authenticate a client, as follows:

1. registration: the client gives an obfuscated version of the secret verification program to the server, thus not revealing the secret to any server intruder;
2. authentication: the client securely sends the secret to the server, which runs the obfuscated version of the secret verification program to verify that the received secret is the same as the one in the obfuscated verification program.

One important case is when such secret is the client's password. Indeed, *entity authentication via password verification*, has often been used as an application motivating the design of program obfuscators for point functions. Note that in this case, the verification program computes precisely a point function, where the secret value is the client's password stored during the registration phase, and the obfuscated program's input is the client's string entered during the authentication phase. As mentioned before, point function obfuscators in the literature are either proved secure under the random oracle assumption or for secret points of length about equal to the factoring-type security parameter (i.e., 2048 bits, which is much more than the length of real-life passwords). When using passwords with ASCII characters including lowercase letters, uppercase letters, numbers and special symbols (for a total of 96 characters), a password of 20 uniformly and independently chosen characters will contain just below 132 bits of entropy. Note that passwords of 20 or more characters have already widespread usage, for instance, in WiFi access to residential networks in private homes. Our solution, which works only when passwords are chosen by the user, can be defined as the hashing-based scheme from [16], with the only difference that the cryptographic hash function H , which could be implemented as SHA2 or SHA3, is actually replaced by the hash function $sprH$ from Sect. 3, where the one-way α -permutation is instantiated using a block cipher like AES, as already described there. With this instantiation, the provable properties of hash function $sprH$ only depend on the (arguably reasonable) assumption that a block cipher like AES (when parameterized by a random input block x and seen as a function $F(\cdot, x)$ mapping the key k to an output $y = F(k, x)$) is a one-way α -permutation, for some small α . Following the discussion at the end of Sect. 3, this assumption is also supported by the lack of a known theoretical attack faster than exhaustive key-search for AES.

Obfuscation of Passphrase Verification. Our solution for password verification is directly applicable to passphrase verification, where a meaningful English sentence, with lower entropy per character, is used as a passphrase. Various techniques have been proposed in the literature to estimate the average number of entropy bits per character in an English passphrase (typically, a number between 0.5 and 3), and may vary depending on the specific assumptions made on the used character sequences. After estimating the average number v of entropy

bits per character in a passphrase taken from a desired set of sequences, a system designer could augment passphrase choice requirements by requiring that a passphrase has length at least q , satisfying $qv \geq 128$.

Obfuscation of a Password Manager. Our solution for password verification is also directly applicable to password managers, another pervasive real-life authentication application, as today the number of password-based services used by the average computer user has increased dramatically. We have evaluated some password manager packages for suitability for our experiments with point function obfuscators, and chosen Pass, a well-known, open-source, password manager [1]. The cryptography currently used in Pass can be seen as a natural extension of the obfuscator from [16] for point functions, to an obfuscator for the password manager’s password derivation program, under the assumption that the used collision-resistant hash function behaves like a random oracle. We have augmented this password manager with an option that obfuscates the password manager’s password derivation program without making a random oracle assumption. Our option processes the user’s passphrase or password using our multi-bit-output point function obfuscator from Sect. 5, which is based on the hash function from Sect. 3, and only makes the previously discussed assumption on a block cipher with unknown key. Recall that our multi-bit-output obfuscator from Sect. 5 could work in two modes: a symmetric or an asymmetric mode, depending on whether the multi-bit output string was encrypted using symmetric or asymmetric encryption. Because Pass already encrypts the website passwords using asymmetric encryption, we could use our multi-bit-output obfuscator in asymmetric mode, and were able to significantly reduce code complexity by focusing our software production on augmenting Pass with the use of our hash function from Sect. 3.

Performance Analysis. We used (here and in the rest of the paper) a Dell 2950 processor (Intel(R) Xeon(R) 8 cores: CPU E5405 @ 2.00GHz, 16 GB RAM), without parallelism. We performed two types of performance analysis. In Table 1 we compare the running time of the cryptographic hash function SHA1 implemented under Pass against the running time of our proposed replacement: i.e.,

Table 1. Performance of SHA1 against our hash function $sprH$.

Input length	Output length	SHA1 running time	$sprH$ running time
64	128	0.0000 s	0.0001 s
128	128	0.0000 s	0.0001 s
192	128	0.0001 s	0.0004 s
256	128	0.0001 s	0.0007 s
320	128	0.0001 s	0.0010 s
384	128	0.0001 s	0.0014 s
448	128	0.0001 s	0.0017 s
512	128	0.0001 s	0.0020 s

our cryptographic hash function $sprH$. In Table 2 we compare the running times of Pass procedures against the running time of our newly proposed option: i.e., Pass with our multi-bit-output point function obfuscator.

Our second set of performance results is about a metric, denoted as *time ratio*, and defined as the ratio of the running time of Pass while using SHA1 to the running time of Pass while using our hash function. Because Pass is essentially performing not much computation other than running calls to `gpg2`, we performed our measurements directly on `gpg2` calls.

Table 2. Performance of Pass with SHA1 against Pass with our hash function $sprH$

Type of master secret	Estimated entropy	Time ratio on <code>gpg2 -X -gen-key</code>	Time ratio on <code>pass insert <site></code>	Time ratio on <code>pass -X show <site></code>
Password	132	0.8756	1	0.9783
Passphrase	100	0.9606	1	0.9363

In Table 2, we consider the three main commands that can be run with Pass: key generation, website password insertion and website password recovery. In the first row, we used a master password of 20 characters uniformly and independently chosen from a set of 96 ASCII characters. In the second row, we used a master passphrase of 56 characters from a meaningful English sentence, which was roughly estimated to have about 100 bits of entropy. As clear from the last 3 columns of the table, our modified version of Pass only slows down Pass by very small percentages, while offering a password manager obfuscation without a random oracle assumption.

References

1. <https://www.passwordstore.org/>
2. Bahler, L., Di Crescenzo, G., Polyakov, Y., Rohloff, K., Cousins, D.B.: Practical implementations of lattice-based program obfuscators for point functions. In: International Conference on High Performance Computing & Simulation, HPCS (2017)
3. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
4. Bellare, M., Rogaway, P.: Collision-resistant hashing: towards making UOWHF's practical. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052256>
5. Bellare, M., Stepanovs, I.: Point-function obfuscation: a framework and generic constructions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 565–594. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_21

6. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_19
7. Canetti, R.: Towards realizing random oracles: hash functions that hide all partial information. Proc. CRYPTO 97, 455–469 (1997)
8. Canetti, R., Dakdouk, R.R.: Obfuscating point functions with multibit output. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 489–508. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_28
9. Cannetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: Proceedings of 30th ACM STOC, pp. 209–218 (1998)
10. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions (preliminary version). In: Proceedings of 30th ACM STOC, pp. 131–140 (1998)
11. Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. **18**(2), 143–154 (1979)
12. Di Crescenzo, G., Bahler, L., Coan, B., Polyakov, Y., Rohloff, K., Cousins, D.B.: Practical implementations of program obfuscators for point functions. In: International Conference on High Performance Computing & Simulation, HPCS (2016)
13. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 279–295. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_17
14. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999)
15. Impagliazzo, R., Zuckerman, D.: How to recycle random bits. In: 30th IEEE FOCS 1989, pp. 248–253 (1989)
16. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_2
17. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Proceedings of 21st ACM STOC 1989, pp. 33–43 (1989)
18. Shoup, V.: A composition theorem for universal one-way hash functions. In: Peneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_32
19. Wee, H.: On obfuscating point functions. In: Proceedings of 37th ACM STOC 2005, pp. 523–532 (2005)
20. Xie, X., Xue, R., Zhang, R.: Deterministic public key encryption and identity-based encryption from lattices in the auxiliary-input setting. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 1–18. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32928-9_1



CCA Secure Multi-recipient KEM from LPN

Haitao Cheng^{1,2}, Xiangxue Li^{1,3(✉)}, Haifeng Qian¹, and Di Yan⁴

¹ Department of Computer Science and Technology, East China Normal University,
Shanghai, China

{xxli,hfqian}@cs.ecnu.edu.cn

² National Engineering Laboratory for Wireless Security,
Xi'an University of Posts and Telecommunications, Xi'an, China

³ Westone Cryptologic Research Center, Beijing, China

⁴ Department of Computer Science and Engineering, Shanghai Jiaotong University,
Shanghai, China

Abstract. We propose a novel multiple-recipient key-encapsulation mechanism (mKEM) scheme which takes multiple public keys as input and outputs a single key shared by corresponding recipients. We construct our scheme in the random oracle model based on low-noise LPN assumption which is a post-quantum problem. In the game simulation of security proof, a variant of Extended Knapsack LPN (which can be proved equivalent to standard LPN) is used to handle the decapsulation queries. The property of LPN problem provides randomness reuse property to shorten the length of the ciphertext compared with traditional way.

Keywords: Post quantum cryptography · Low-noise LPN
Multi-recipient KEM

1 Introduction

Multi-recipient key-encapsulation mechanism (mKEM) is a popular public key cryptographic primitive that allows a sender to transfer a session key to multiple recipients who can get the session key by decrypting the ciphertext with their own private key. A hybrid public key encryption based on a data encapsulation mechanism (DEM) keyed by a KEM is a typical scenario that allows one to send a large amount of data [13]. In such case, one could share a session key K with n parties via sending the underlying session key K n times and then send the data by using any fast symmetric scheme with that key. However, this may be quite inefficient in either size of ciphertext or computation cost. In this context, the mKEM scheme is motivated to encapsulate a single session key to multiple recipients. Running one mKEM other than multiple KEM instances benefits to the ciphertext size to be linearly proportional to the number of receivers while the latter one is many times more than it. For example, Alice now wants to encrypt an email to Bob and Charlie, or encrypt a file on her system such

that Bob or Charlie can decrypt, which is a typical scenario of one-to-many model. One way is to simply encrypts the data twice, once for Bob and once for Charlie, using their respective public key schemes. Another way is to encrypt the data once with a symmetric encryption key K and then encrypt this key under Bob and Charlie's public keys. The former one is clearly wasteful especially if the data to be encrypted is large and the latter one will become very expensive for a large number of users. It is apparently to see that mKEM is more efficient in such cases.

Multi-recipient PKE is a close cryptographic primitive to mKEM. Kurosawa [21] proposed randomness reuse technique across single-recipient PKE schemes, which could yield a shortened ciphertext multi-recipient PKE schemes. In 2003, a new notion named reproducibility is introduced by Bellare et al. [6]. Additionally, a reproducibility test is given to determine whether a single-recipient PKE scheme allows to transform to multi-recipient PKE (mPKE) through such randomness reuse technique. However, a recent concrete mPKE introduced by Wei et al. [26] showed that passing the reproducibility test is not the determining factor for obtaining a mPKE. In 2009, Hiwatari et al. [17] proposed two concrete IND-CCA secure mKEM schemes that do not pass the reproducibility test.

Most of the mKEM schemes are based on the discrete logarithm problem. Taking the scheme proposed by Boneh, Ma and Waters (BMW scheme) [27] and its variants [8] as examples, the security of those protocols are based on decisional bilinear Diffie-Hellman assumption. Since post quantum cryptography has become a hot topic, to build a post-quantum mKEM is also an exciting thing to do. A promising candidate could be Learning Parity with Noise (LPN) problem which is well-known in cryptography and learning theory. The LPN problem states that the distribution $\{(\mathbf{A}, (\mathbf{A}\mathbf{s} + \mathbf{e})) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}, \mathbf{e} \xleftarrow{\$} \mathcal{B}_\mu^m, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n\}$ is indistinguishable from uniform where \mathcal{B}_μ is the Bernoulli distribution with parameter μ , i.e., $\Pr[x = 1 : x \leftarrow \mathcal{B}_\mu] = \mu$. While in standard LPN the Bernoulli parameter μ is constant (with noise parameter $0 < \mu < 1/2$), in low-noise LPN we have $\mu = O(n^{-c})$ (typically constant $c = 1/2$), where n is the dimension of the LPN secret.

There are two versions of LPN, the decisional one is given above and the search one requires that it is computational infeasible to find out the random secret binary vector $\mathbf{s} \in \{0, 1\}^n$ from those noisy linear samples. Those two versions have been proved polynomially equivalent [2, 19]. The computational LPN problem is deemed as a well-known NP-complete problem "decoding random linear codes" [9], which makes LPN be a promising candidate for post-quantum cryptography. Under low-noise rate, the best solving algorithm needs $2^{O(n^{-c})}$ time when given $O(n)$ samples [5, 10, 24].

1.1 Related Work

The first formalized security notion for mKEM is given by Smart [25], where the IND-CCA2 security is defined in a similar manner as the Replayable CCA security introduced in [11]. In this paper, we use IND-CCA to represent the

same meaning as IND-CCA2 in that model. Smart’s model imposes a restriction that the decapsulation oracle queries can be only allowed if the resulting key is different from the key encapsulated by challenge ciphertext and the adversary can choose arbitrary honest and uncorrupted ciphertext.

Barbosa et al. [4] constructed multi-recipient encryption scheme supporting the reuse of randomness based on the work of Bellare et al. [6]. A new notion called weak reproducibility is proposed to construct a single-message multi-recipient scheme from single-recipient PKE. An mKEM provably secure in the standard model is proposed.

In 2006, Bellare et al. [7] proposed a framework that allows for very efficient encryption operations regarding stateful encryption. Some DHIES [1] and Kurosawa-Desmedt [22] schemes are proposed in their stateful setting. But a drawback for this is that it requires multiple distinct encryption instances to share a secret state which may cause other security risks.

In 2009, Hiwatari et al. [17] proposed two concrete mKEM schemes satisfying IND-CCA security and IND-CCCA security [18], respectively by applying the proof simulation techniques in [14] and [16]. Matsuda et al. [23] introduced a new framework to cover the efficient constructions for mKEM in PKC’13. However, those schemes were proved in a model that the challenged parties are fixed.

As in [3, 12], mKEM under the indentity-based setting can be found. We here concentrate on traditional public key infrastructure but depend on LPN hardness.

1.2 Our Contribution

We propose a post-quantum IND-CCA secure mKEM scheme under the LPN assumption. Firstly, based on the Extended Knapsack LPN and its variant, we get an IND-CPA secure PKE. Then with the random oracle to randomize the key encapsulation from a random message \mathbf{m} , the output can be ensured to be uniformly distributed. In our scheme we apply the randomness reuse to shorten the ciphertext. Technically, the LPN problem itself makes it feasible to generate multiple ciphertexts with a common matrix \mathbf{A} (as a part of the public key shared by all recipients), which is still secure for instantiating with different secret key as LPN instances. Different from previous mKEM scheme’s proof, we give it by game simulation in Smart’s model.

2 Preliminaries

2.1 Notation and Definitions

We use capital letters (e.g., X, Y) for random variables and distributions, standard letters (e.g., x, y) for values. Vectors are used in the column form and denoted by bold lower-case letters (e.g., \mathbf{a}). We treat matrices as the sets of its column vectors and denote them by bold capital letters (e.g., \mathbf{A}). The support of a random variable X , denoted by $\text{Supp}(X)$, refers to the set of values on which

X takes with non-zero probability, i.e., $\{x : \Pr[X = x] > 0\}$. For a binary string x , $|x|$ refers to the Hamming weight of x . We use \mathcal{B}_μ to denote the Bernoulli distribution with parameter μ , i.e., $\Pr[\mathcal{B}_\mu = 1] = \mu$, $\Pr[\mathcal{B}_\mu = 0] = 1 - \mu$, while \mathcal{B}_μ^n denotes the concatenation of n independent copies of \mathcal{B}_μ . For $n, \ell \in \mathbb{N}$, U_n (resp., $U_{\ell \times n}$) denotes the uniform distribution over $\{0, 1\}^n$ (resp., $\{0, 1\}^{\ell \times n}$) and independent of any other random variables in consideration. $X \sim D$ denotes that random variable X follows distribution D . We use $s \leftarrow S$ to denote sampling an element s according to distribution S .

INDISTINGUISHABILITY. We define the (t, ε) -computational distance between random variables X and Y , denoted by $X \underset{(t, \varepsilon)}{\sim} Y$, if for every probabilistic distinguisher \mathcal{D} of running time t it holds that

$$| \Pr[\mathcal{D}(X) = 1] - \Pr[\mathcal{D}(Y) = 1] | \leq \varepsilon$$

COMPUTATIONAL INDISTINGUISHABILITY. Is defined with respect to distribution ensembles (indexed by a security parameter). For example, $X \stackrel{\text{def}}{=} \{X_n\}_{n \in \mathbb{N}}$ and $Y \stackrel{\text{def}}{=} \{Y_n\}_{n \in \mathbb{N}}$ are computationally indistinguishable, denoted by $X \underset{(t, \varepsilon)}{\sim} Y$, if for every $t = \text{poly}(n)$ there exists $\varepsilon = \text{negl}(n)$ such that $X \underset{(t, \varepsilon)}{\sim} Y$.

ENTROPY NOTIONS. For a random variable X and any $x \in \text{Supp}(X)$, the sample-entropy of x with respect to X is defined as

$$\mathbf{H}_X(\mu) \stackrel{\text{def}}{=} \log(1/\Pr[X = x])$$

from which we define the Shannon entropy, Rényi entropy and min-entropy of X respectively, i.e.,

$$\mathbf{H}_1(X) \stackrel{\text{def}}{=} \mathbb{E}_{x \leftarrow X} [\mathbf{H}_X(x)], \mathbf{H}_2 \stackrel{\text{def}}{=} -\log \sum_{x \in \text{Supp}(X)} 2^{-2\mathbf{H}_X(x)}, \mathbf{H}_\infty \stackrel{\text{def}}{=} \min_{x \in \text{Supp}(X)} \mathbf{H}_X(x).$$

For $0 < \mu < 1/2$, let $\mathbf{H}(\mu) \stackrel{\text{def}}{=} \mu \log(1/\mu) + (1 - \mu) \log(1/(1 - \mu))$ be the binary entropy function so that $\mathbf{H}_\mu = \mathbf{H}_1(\mathcal{B}_\mu)$. We know that $\mathbf{H}_1(X) \geq \mathbf{H}_2(X) \geq \mathbf{H}_\infty(X)$ with equality when X is uniformly distributed. A random variable X of length n is called an (n, λ) -Rényi entropy (resp., min-entropy) source if $\mathbf{H}_2(X) \geq \lambda$ (resp., $\mathbf{H}_\infty(X) \geq \lambda$). The statistical distance between X and Y , denoted by $\text{SD}(X, Y)$, is defined by

$$\text{SD}(X, Y) \stackrel{\text{def}}{=} \frac{1}{2} \sum_x |\Pr[X = x] - \Pr[Y = x]|$$

We use $\text{SD}(X, Y|Z)$ as a shorthand for $\text{SD}((X, Z), (Y, Z))$.

Definition 1 (pairwise independent hashing). A function family $\mathcal{H} = \{h_a : \{0, 1\}^n \leftarrow \{0, 1\}^m, a \in \{0, 1\}^l\}$ is pairwise independent if for any $x_1 \neq x_2 \in \{0, 1\}^n$ and any $v \in \{0, 1\}^{2m}$ it holds that

$$\Pr_{a \leftarrow \{0, 1\}^l} [(h_a(x_1), h_a(x_2)) = v] = 2^{-2m}.$$

2.2 Learning Parity with Noise

Definition 2 (Learning Parity with Noise). The *decisional* $\text{LPN}_{n,\ell,\mu}$ problem is hard if for every $\ell = \text{poly}(n)$ we have

$$(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b})$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{s} \sim U_n$, $\mathbf{e} \sim \mathcal{B}_\mu^\ell$ and $\mathbf{b} \sim U_\ell$ while the secret length is n and the noise rate is $0 < \mu < 1/2$. The *computational* $\text{LPN}_{n,\ell,\mu}$ problem is hard if for every $\ell = \text{poly}(n)$ and every PPT algorithm \mathcal{D} we have

$$\Pr[\mathcal{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) = \mathbf{s}] = \text{negl}(n)$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{s} \sim U_n$ and $\mathbf{e} \sim \mathcal{B}_\mu^\ell$.

Definition 3 (Knapsack LPN). The knapsack LPN problem is hard if for $\ell > n$ samples we have

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b})$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{s} \sim \mathcal{B}_\mu^\ell$, $\mathbf{b} \sim U_n$.

With a standard hybrid argument technique, we have results on the r -fold LPN and r -fold KLPN that

$$(\mathbf{A}, \mathbf{A}\mathbf{S} + \mathbf{E}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{B}_1)$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{S} \sim U_{n \times r}$, $\mathbf{E} \sim \mathcal{B}_\mu^{\ell \times r}$ and $\mathbf{B}_1 \sim U_{\ell \times r}$.

$$(\mathbf{A}, \mathbf{S}^\top \mathbf{A}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{B}_2)$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{S} \sim \mathcal{B}_\mu^{\ell \times r}$ and $\mathbf{B}_2 \sim U_{r \times n}$.

Next, we introduce extended knapsack LPN problem which is still hard even if the information of errors is leaked to the adversary partially.

Definition 4 (Extended Knapsack LPN-EKLPN). The *Extended Knapsack LPN* problem is hard if for $\ell > n$ samples we have

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s}, \mathbf{e}, \mathbf{s}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b}, \mathbf{e}, \mathbf{s}^\top \mathbf{e})$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{b} \sim U_n$, $\mathbf{s}, \mathbf{e} \sim \mathcal{B}_\mu^\ell$.

Lemma 1. Assume that the Extended Knapsack LPN problem is hard then we have

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s}, \mathbf{e}, \mathbf{s}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{A}^\top \mathbf{s}', \mathbf{e}, \mathbf{s}'^\top \mathbf{e})$$

Proof. From Definition 4 we have

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s}, \mathbf{e}, \mathbf{s}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b}, \mathbf{e}, \mathbf{s}^\top \mathbf{e}) \tag{1}$$

From Definition 3 we have

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s}') \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b}) \tag{2}$$

where $\mathbf{A} \sim U_{\ell \times n}$, $\mathbf{s}, \mathbf{s}' \sim \mathcal{B}_\mu^\ell$.

By combining (1) with (2), we immediately obtain

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s}, \mathbf{e}, \mathbf{s}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{A}^\top \mathbf{s}', \mathbf{e}, \mathbf{s}^\top \mathbf{e})$$

The Extended Knapsack LPN problem to standard LPN problem reduction has been given in [15].

3 mKEM from LPN

3.1 Multi-recipient Key-Encapsulation Mechanism

Let the variable $k \in \mathbb{N}$ denote the maximum number of honest recipients in an mKEM scheme. An mKEM can take m ($1 \leq m \leq k$) receivers' public keys as input and output a key $K \in \mathcal{K}_{\text{mKEM}}$ and a ciphertext generated based on all the m public keys. The mKEM can be formally defined by the following algorithms:

mKEM.KeyGen(1^n): This is a probabilistic algorithm that takes as input the security parameter 1^n and outputs a pair of public and secret key (pk, sk) .

mKEM.EnCap(\mathcal{P}): This is a probabilistic algorithm that takes as input a set of public keys of m receivers $\mathcal{P} = (pk_1, \dots, pk_m)$ and outputs a key $K \in \mathcal{K}_{\text{mKEM}}$ and a ciphertext C of K .

mKEM.DeCap(i, \mathcal{P}, C, sk_i): This is a deterministic algorithm that takes as input an index i ($i \in [\#\mathcal{P}]$) indicating the secret key sk_i of pk_i in the parameter $\mathcal{P} = (pk_1, \dots, pk_m)$, and a ciphertext C as input, and outputs a key K if C is valid.

Definition 5 (Correctness). We say that a mKEM scheme is correct if for an arbitrary secret key sk_i ($i \in [m]$) corresponding to $pk_i \in \mathcal{P} = \{pk_1, \dots, pk_m\}$, $(K, C) = \text{mKEM.EnCap}(\mathcal{P})$ and $\text{mKEM.DeCap}(i, \mathcal{P}, C, sk_i) = K$ holds with overwhelming probability.

3.2 Smart's CCA Model

Definition 6. A multi-recipient key-encapsulation mechanism scheme mKEM is IND-CCA secure if for any integers m and k with $m \leq k$, the advantage of the adversary $\text{Adv}_{\text{mKEM}, \mathcal{A}}^{\text{ind-cca}}(n) = |\Pr[\text{EXP}_{\text{mKEM}, \mathcal{A}}^{\text{ind-cca}}(n, k) = 1] - 1/2|$ is negligible in n for all probabilistic polynomial time (PPT) adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where the security experiment $\text{EXP}_{\text{mKEM}, \mathcal{A}}^{\text{ind-cca}}(n, k)$ is defined as in Fig. 1:

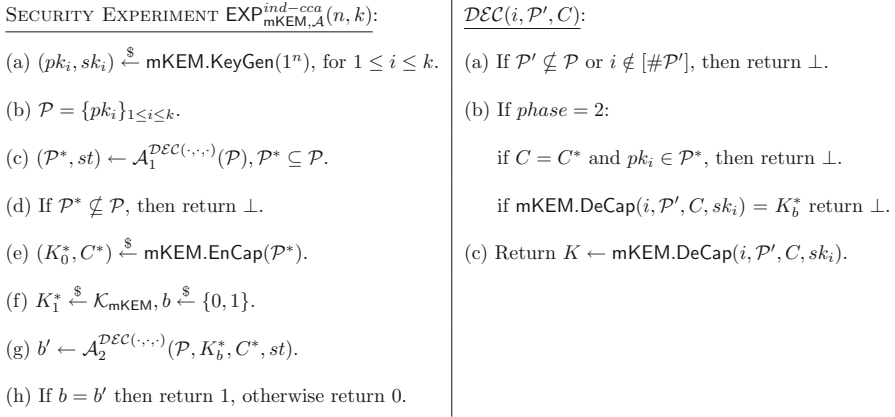


Fig. 1. Security experiment $\text{EXP}_{\text{mKEM}, \mathcal{A}}^{\text{ind-cca}}(n, k)$.

3.3 The Construction

Our mKEM scheme uses the following parameters and building blocks.

- Let n be the security parameter and let $\ell \geq 2n$ be an integer, constants $0 < c < \frac{1}{6}$ and α satisfy that $6c < \alpha < 1$.
- Let $\mu = \sqrt{c/\ell}$, $\beta = 2\sqrt{c\ell}$ to check consistency during decapsulation.
- $\text{ECC} : \{0, 1\}^p \rightarrow \{0, 1\}^\ell$ be an error correction code with an efficient decoding algorithm ECC^{-1} , which can correct at least an α -fraction of errors with α mentioned before.
- Two hash functions $H_1 : \{0, 1\}^p \rightarrow \mathcal{B}_\mu^\ell \times \mathcal{B}_\mu^\ell$ and $H_2 : \{0, 1\}^\ell \rightarrow \{0, 1\}^{\ell'}$, the requirement for p (length of messages \mathbf{m}) is discussed below and ℓ' could be $\ell' \ll \ell$ (say 128 bit typically).

Note that when H_1 is used as a random oracle, the mapping result to two Bernoulli vectors is reasonable. We give a sketch about Bernoulli randomness extraction in Fig. 2 below which applies i.i.d pairwise independent hash functions

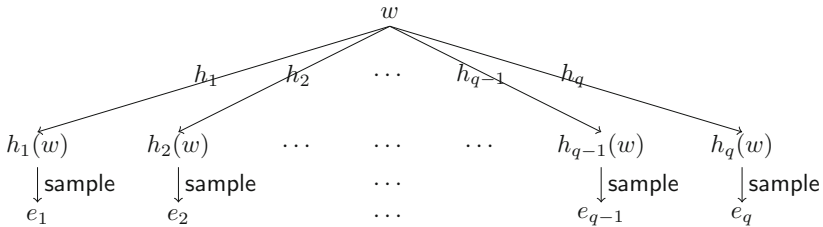


Fig. 2. An illustration of the Bernoulli randomness extractor.

h_1, \dots, h_q to a p -bit uniform random string w and then uses `sample` on the bits extracted to get a vector distributed to \mathcal{B}_μ^q where q can be up to $\Theta(p/\mathbf{H}_1(\mathcal{B}_\mu))$ (where $\mathbf{H}_1(\mathcal{B}_\mu) = \Theta(\mu \log(1/\mu))$) [28]. Thus, to get a \mathcal{B}_μ^ℓ vector mapping result, p should be $\Theta(\ell \cdot \mathbf{H}_1(\mathcal{B}_\mu))$.

The following lemma states that the proposed Bernoulli randomness extractor extracts almost all entropy from a Rényi entropy (or min-entropy) source. We use \vec{H} to denote a vector in the lemma.

Lemma 2 (Bernoulli randomness extraction [28]). *For any $m, v \in \mathbb{N}$ and $0 < \mu \leq 1/2$, let $W \in \mathcal{W}$ be any $(\lceil \log |\mathcal{W}| \rceil, m)$ -Rényi entropy source, let \mathcal{H} be a family of pairwise independent hash functions mapping from \mathcal{W} to $\{0, 1\}^v$, let $\vec{H} = (H_1, \dots, H_q)$ be a vector of i.i.d. random variables such that each H_i is uniformly distributed over \mathcal{H} , let `sample` : $\{0, 1\}^v \rightarrow \{0, 1\}$ be any Boolean function such that `sample`(U_v) $\sim \mathcal{B}_\mu$. Then, for any constant $0 < \Delta \leq 1$ it holds that*

$$\text{SD}(\mathcal{B}_\mu^q, \text{sample}(\vec{H}(W)) | \vec{H}) \leq 2^{((1+\Delta)q\mathbf{H}(\mu)-m)/2} + \exp^{-\frac{\Delta^2 \mu q}{3}},$$

where `sample`($\vec{H}(W)$) $\stackrel{\text{def}}{=} (\text{sample}(H_1(W)), \dots, \text{sample}(H_q(W)))$.

A CCA SECURE mKEM. We present the construction of our mKEM scheme `mKEM = (mKEM.KeyGen, mKEM.EnCap, mKEM.DeCap)` as follows.

`mKEM.KeyGen`(1^n): On input a security parameter 1^n , it first chooses a common matrix $\mathbf{A} \xleftarrow{\$} U_{\ell \times n}$, then for each recipient, it samples a square matrix $\mathbf{S}_i \xleftarrow{\$} \mathcal{B}_\mu^{\ell \times \ell}$ and computes $\mathbf{B}_i = \mathbf{S}_i \mathbf{A} \in \{0, 1\}^{\ell \times n}$. The output is $(pk_i, sk_i)_{1 \leq i \leq m} = ((\mathbf{A}, \mathbf{B}_i), \mathbf{S}_i)_{1 \leq i \leq m}$.

`mKEM.EnCap`($\mathcal{P} = \{pk_i\}_{1 \leq i \leq m}$): It chooses a uniform $\mathbf{m} \in \{0, 1\}^p$, computes $(\mathbf{s}, \mathbf{e}) := H_1(\mathbf{m})$ and calculates $\mathbf{c}_0 = \mathbf{A}\mathbf{s} + \mathbf{e} \in \{0, 1\}^\ell$ then on each input the public key $pk_i = (\mathbf{A}, \mathbf{B}_i)$, it chooses a matrix $\mathbf{S}'_i \xleftarrow{\$} \mathcal{B}_\mu^{\ell \times \ell}$ for $1 \leq i \leq m$, and computes

$$\begin{aligned} \mathbf{c}_i &= \mathbf{B}_i \mathbf{s} + \mathbf{S}'_i \mathbf{e} + \text{ECC}(\mathbf{m}) \in \{0, 1\}^\ell \\ \mathbf{k} &= H_2(\mathbf{m}) \in \{0, 1\}^{\ell'}. \end{aligned}$$

It outputs \mathbf{k} as the encapsulated session key and $\mathbf{C} = (\mathbf{c}_0, \{\mathbf{c}_i\}_{1 \leq i \leq m})$ as the ciphertext.

`mKEM.DeCap`($i, \mathcal{P}, \mathbf{C}, sk_i$): On input the set of all recipients \mathcal{P} , an encapsulation \mathbf{C} and the recipient i 's secret key sk_i , this algorithm does the following steps:

1. Reject if $\#\mathcal{P} \neq m$;
2. Compute

$$\tilde{\mathbf{c}} = \mathbf{c}_i - \mathbf{S}_i \mathbf{c}_0 = \text{ECC}(\mathbf{m}) + (\mathbf{S}'_i - \mathbf{S}_i) \mathbf{e}$$

and then compute $\mathbf{m}' = \text{ECC}^{-1}(\tilde{\mathbf{c}})$ and $(\mathbf{s}', \mathbf{e}') := H_1(\mathbf{m}')$. If $\mathbf{c}_0 = \mathbf{A}\mathbf{s}' + \mathbf{e}'$ and $|\mathbf{c}_i - \mathbf{B}_i \mathbf{s}' - \text{ECC}(\mathbf{m}')| \leq \beta$ hold, output $\mathbf{k} := H_2(\mathbf{m}')$; otherwise, output \perp .

3.4 Correctness

Lemma 3 (Chernoff Bound [20, 29]). *For any $0 < \mu < 1$ and any $\delta > 0$, we have*

$$\Pr[|\mathcal{B}_\mu^\ell| > (1 + \delta)\mu\ell] < e^{-\frac{\min(\delta, \delta^2)}{3}\mu\ell}, \quad (3)$$

in particular, for $\delta = 1$

$$\Pr[|\mathcal{B}_\mu^\ell| > 2\mu\ell] < e^{-\mu\ell/3}. \quad (4)$$

Obviously, for the chosen $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^\ell$, the Chernoff Bound (2) yields:

$$\Pr[|\mathbf{e}| > \underbrace{\beta}_{=2\mu\ell}] < e^{-\mu\ell/3} = 2^{-\Theta(\sqrt{\ell})}. \quad (5)$$

Theorem 1 (Correctness). *With overwhelming probability over the choice of the public and secret keys $\text{mKEM.DeCap}(i, \mathcal{P}, \mathbf{C}, sk_i)$ outputs \mathbf{k} correctly over $\mathbf{C} \leftarrow \text{mKEM.EnCap}(\mathcal{P} = \{pk_i\}_{1 \leq i \leq m})$.*

Proof. The scheme's correctness requires the following:

1. $|(\mathbf{S}'_i - \mathbf{S}_i)\mathbf{e}| \leq \alpha\ell$ (to let ECC^{-1} reconstruct \mathbf{m}' from $\tilde{\mathbf{c}} = \mathbf{c}_i - \mathbf{S}_i\mathbf{c}_0$).
2. When $\mathbf{m}' = \text{ECC}^{-1}(\tilde{\mathbf{c}})$ and $(\mathbf{s}', \mathbf{e}') := H_1(\mathbf{m}')$, $\mathbf{c}_0 = \mathbf{A}\mathbf{s}' + \mathbf{e}'$ should hold.
3. $|\mathbf{c}_i - \mathbf{B}_i\mathbf{s}' - \text{ECC}(\mathbf{m}')| \leq \beta$.

For the decapsulation algorithm we require that the Hamming weight for the inner-product of a matrix $\mathbf{S} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^{\ell \times \ell}$ and a vector $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^\ell$ is upper bounded by $\frac{1}{3}\alpha\ell$ with overwhelming probability. We firstly analyze the inner-product of a vector $\mathbf{s} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^\ell$ and the vector $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^\ell$ whose Hamming weight is at most β described as above. Since $|\mathbf{e}| \leq \beta$, a necessary condition for $\mathbf{s}^\top \mathbf{e} = 1$ is that $\mathbf{s}[i] = 1$ for at least one of the i 's where $\mathbf{e}[i] = 1$. By a simple XOR-Lemma, it holds that

$$\mu' = \Pr[\mathbf{s}^\top \mathbf{e} = 1] \leq \beta\mu = 2c.$$

By the Chernoff Bound (1) and with $\delta = \alpha/(3\mu') - 1$ (where $\mu' \leq 2c < \alpha/3$)

$$\Pr\left[|\mathbf{S}\mathbf{e}| > \frac{1}{3}\alpha\ell\right] = \Pr[|\mathbf{S}\mathbf{e}| > (1 + \delta)\mu'\ell] < e^{-\frac{\min(\delta, \delta^2)}{3}\mu'\ell}.$$

Since $\delta\mu' = \alpha/3 - \mu' \geq \alpha/3 - 2c > 0$ and $\delta = \alpha/(3\mu') - 1 \geq \alpha/(6c) - 1 > 0$ are lower bounded by constants and therefore

$$\Pr\left[|\mathbf{S}\mathbf{e}| > \frac{1}{3}\alpha\ell\right] < e^{-\frac{\min(\delta, \delta^2)}{3}\mu'\ell} = 2^{-\Theta(\ell)}. \quad (6)$$

Finally, in the ciphertext of our construction we have

$$|\mathbf{c}_i - \mathbf{B}_i\mathbf{s} - \text{ECC}(\mathbf{m})| = |\mathbf{S}'_i\mathbf{e}| \leq \beta$$

holds with overwhelming probability $1 - 2^{-\Theta(\sqrt{m})}$ by (3) and (4).

In the decapsulation operation,

$$\begin{aligned} \tilde{\mathbf{c}} &= \mathbf{c}_i - \mathbf{S}_i \mathbf{c}_0 \\ &= \text{ECC}(\mathbf{m}) + (\mathbf{S}'_i - \mathbf{S}_i) \mathbf{e} \end{aligned}$$

it is sufficient to bound the error item $|(\mathbf{S}'_i - \mathbf{S}_i) \mathbf{e}|$. It holds that

$$|(\mathbf{S}'_i - \mathbf{S}_i) \mathbf{e}| \leq |\mathbf{S}'_i \mathbf{e}| + |\mathbf{S}_i \mathbf{e}| \leq \frac{2}{3} \alpha \ell < \alpha \ell.$$

In all, the message \mathbf{m} can be decrypted with overwhelming probability thus the encapsulated key \mathbf{k} can be correctly outputed.

3.5 Security

Theorem 2. *Assume that the LPN problem is hard and both hash functions H_1 and H_2 are modeled as random oracles then the above mKEM scheme is IND-CCA secure.*

Proof. Let \mathcal{A} be any PPT adversary that can attack our scheme mKEM with advantage ε . We show that ε must be negligible in n . We continue the proof by using a sequence of games, where the first game is the real game, while the last is a random game. Since \mathcal{A} 's advantage in a random game is exactly 0, the security of mKEM can be established by showing that \mathcal{A} 's advantage in any two consecutive games are negligibly close.

Game 1. This is the experiment $\text{EXP}_{\text{mKEM}, \mathcal{A}}^{\text{ind-cca}}(n, k)$. The challenger \mathcal{C} honestly runs the adversary \mathcal{A} with the security parameter n . Then, it simulates the security experiment $\text{EXP}_{\text{mKEM}, \mathcal{A}}^{\text{ind-cca}}(n, k)$ as follows:

KeyGen. First \mathcal{C} uniformly chooses a matrix $\mathbf{A} \xleftarrow{\$} U_{\ell \times n}$. Then for each $i \in [k]$, it chooses a square matrix $\mathbf{S}_i \xleftarrow{\$} \mathcal{B}_{\mu}^{\ell \times \ell}$ and computes $\mathbf{B}_i = \mathbf{S}_i \mathbf{A}$. Finally, \mathcal{C} sends $pk_i = (\mathbf{A}, \mathbf{B}_i)$ to the adversary \mathcal{A} and keeps $sk_i = \mathbf{S}_i$ to itself. \mathcal{P} is used to denote the set of all the public keys $\{pk_i\}_{1 \leq i \leq k}$.

Phase 1. While receiving a decapsulation query $(i, \mathcal{P}', \mathbf{C} = (\mathbf{c}_0, \{\mathbf{c}_t\}_{1 \leq t \leq m}))$ from adversary \mathcal{A} , the challenger \mathcal{C} does the following thing:

(a) If $\mathcal{P}' \not\subseteq \mathcal{P}$ or $i \notin [\#\mathcal{P}']$, then return \perp .

If (a) is passed, it computes

$$\tilde{\mathbf{c}} = \mathbf{c}_i - \mathbf{S}_i \mathbf{c}_0 = \text{ECC}(\mathbf{m}) + (\mathbf{S}'_i - \mathbf{S}_i) \mathbf{e}$$

and then computes $\mathbf{m}' = \text{ECC}^{-1}(\tilde{\mathbf{c}})$ and $(\mathbf{s}', \mathbf{e}') := H_1(\mathbf{m}')$. If $\mathbf{c}_0 = \mathbf{A} \mathbf{s}' + \mathbf{e}'$ and $|\mathbf{c}_i - \mathbf{B}_i \mathbf{s}' - \text{ECC}(\mathbf{m}')| \leq \beta$ hold, it outputs $\mathbf{k} := H_2(\mathbf{m}')$ and returns \mathbf{k} to \mathcal{A} ; otherwise, return \perp to \mathcal{A} . After that, \mathcal{A} chooses a set $\mathcal{P}^* \subseteq \mathcal{P}$ and sends it to the challenger \mathcal{C} .

Challenge. After receiving the challenge public key set \mathcal{P}^* from \mathcal{A} , \mathcal{C} randomly chooses a uniform $\mathbf{m} \in \{0, 1\}^p$, computes $(\mathbf{s}, \mathbf{e}) := H_1(\mathbf{m})$ and calculates $\mathbf{c}_0^* = \mathbf{A}\mathbf{s} + \mathbf{e} \in \{0, 1\}^\ell$ then it chooses a random bit $b \xleftarrow{\$} \{0, 1\}$, on each public key $pk_i = (\mathbf{A}, \mathbf{B}_i)$ it chooses a matrix $\mathbf{S}'_i \xleftarrow{\$} \mathcal{B}_\mu^{\ell \times \ell}$, computes

$$\begin{aligned} \mathbf{c}_i^* &= \mathbf{B}_i \mathbf{s} + \mathbf{S}'_i \mathbf{e} + \text{ECC}(\mathbf{m}) \in \{0, 1\}^\ell \\ \mathbf{k}_b^* &= H_2(\mathbf{m}) \in \{0, 1\}^{\ell'} \end{aligned}$$

and chooses a $\mathbf{k}_{1-b}^* \xleftarrow{\$} \{0, 1\}^{\ell'}$. Finally, \mathcal{C} sends $(\mathbf{C}^*, \mathbf{k}_b^*, \mathbf{k}_{1-b}^*)$ to \mathcal{A} .

Phase 2. While receiving a decapsulation query $(i, \mathcal{P}', \mathbf{C} = (\mathbf{C}_0, \{\mathbf{c}_t\}_{1 \leq t \leq m}))$ from adversary \mathcal{A} , the challenger \mathcal{C} does the following thing:

- (a) If $\mathcal{P}' \not\subseteq \mathcal{P}$ or $i \notin [\#\mathcal{P}']$, then return \perp .
- (b) If $C = C^*$ and $pk_i \in \mathcal{P}^*$, then return \perp .
- (c) If $\text{mKEM.DeCap}(i, \mathcal{P}', C, sk_i) = \mathbf{k}_b^*$ return \perp .

If (a), (b), (c) are passed, it computes

$$\tilde{\mathbf{c}} = \mathbf{c}_i - \mathbf{S}_i \mathbf{c}_0 = \text{ECC}(\mathbf{m}) + (\mathbf{S}'_i - \mathbf{S}_i) \mathbf{e}$$

and then compute $\mathbf{m}' = \text{ECC}^{-1}(\tilde{\mathbf{c}})$ and $(\mathbf{s}', \mathbf{e}') := H_1(\mathbf{m}')$. If $\mathbf{c}_0 = \mathbf{A}\mathbf{s}' + \mathbf{e}'$ and $|\mathbf{c}_i - \mathbf{B}_i \mathbf{s}' - \text{ECC}(\mathbf{m}')| \leq \beta$ hold, it returns $\mathbf{k} := H_2(\mathbf{m}')$ to \mathcal{A} ; otherwise, it returns \perp to \mathcal{A} .

Guess. Finally, \mathcal{A} outputs a guess $b' \in \{0, 1\}$. If $b' = b$, the challenger \mathcal{C} outputs 1, else outputs 0.

Let W_i be the event \mathcal{C} outputs 1 in Game i for i in $\{1, 2, 3\}$.

Game 2. This Game is identical to Game 1 except that the challenge phase is changed as follows:

Challenge. After receiving the challenge public key set \mathcal{P}^* from \mathcal{A} , \mathcal{C} randomly chooses a uniform $\mathbf{m} \in \{0, 1\}^p$, computes $(\mathbf{s}, \mathbf{e}) := H_1(\mathbf{m})$ and calculates $\mathbf{c}_0^* = \mathbf{A}\mathbf{s} + \mathbf{e} \in \{0, 1\}^\ell$ then on each public key $pk_i = (\mathbf{A}, \mathbf{B}_i)$ it chooses a random bit $b \xleftarrow{\$} \{0, 1\}$, computes

$$\begin{aligned} \mathbf{c}_i^* &= \mathbf{B}_i \mathbf{s} + \mathbf{S}_i \mathbf{e} + \text{ECC}(\mathbf{m}) \in \{0, 1\}^\ell \\ \mathbf{k}_b^* &= H_2(\mathbf{m}) \in \{0, 1\}^{\ell'} \end{aligned}$$

and chooses a $\mathbf{k}_{1-b}^* \xleftarrow{\$} \{0, 1\}^{\ell'}$. Finally, \mathcal{C} sends $(\mathbf{C}^*, \mathbf{k}_b^*, \mathbf{k}_{1-b}^*)$ to \mathcal{A} .

Lemma 4. *If the EKLPN problem is hard, then we have $|\Pr[W_1] - \Pr[W_2]| \leq \text{negl}(n)$.*

Proof. The only difference between Game 1 and Game 2 is that \mathcal{C} replaces each $\mathbf{c}_i^* := \mathbf{B}_i \mathbf{s} + \mathbf{S}'_i \mathbf{e} + \text{ECC}(\mathbf{m})$ in Game 1 with $\mathbf{c}_i^* := \mathbf{B}_i \mathbf{s} + \mathbf{S}_i \mathbf{e} + \text{ECC}(\mathbf{m})$ in Game 2 for $1 \leq i \leq m$. Next, we introduce a sequence of games $\{\text{Game}_{1,i,j}\}_{1 \leq j \leq \ell}^{1 \leq i \leq m}$

between Game 1 and Game 2 to replace each \mathbf{S}'_i in the \mathbf{c}_i^* row by row. Firstly, we define

$$\mathbf{S}_i = \begin{pmatrix} \mathbf{s}_{1_i}^\top \\ \vdots \\ \mathbf{s}_{\ell_i}^\top \end{pmatrix}, \mathbf{S}'_i = \begin{pmatrix} \mathbf{s}'_{1_i} \\ \vdots \\ \mathbf{s}'_{\ell_i} \end{pmatrix}.$$

- $\text{Game}_{1_i,j}$, $i \in [m], j \in [\ell]$. This game is a hybrid of Game 1 and Game 2: the challenger \mathcal{C} replaces \mathbf{s}'_{j_i} with $\mathbf{s}_{j_i}^\top$ in \mathbf{c}_i^* during the challenge phase and keeps the remaining rows as in $\text{Game}_{1_i,j-1}$. Thus there are $m \times \ell$ games. For example, for $i = 1$, there are ℓ games: from $\text{Game}_{11,1}$ to $\text{Game}_{11,\ell}$. Besides i is from 1 to m . Let $\text{Game}_{11,0}$ be Game 1. Obviously, $\text{Game}_{1_m,\ell}$ is identical to Game 2.

It suffices to show that $|\Pr[W_{1_i,j}] - \Pr[W_{1_i,j-1}]| \leq \text{negl}(n)$ for any $i \in [m], j \in [\ell]$.

The hardness of the EKLPN problem ensures that the probability for adversary \mathcal{A} to distinguish $\text{Game}_{1_i,j}$ from $\text{Game}_{1_i,j-1}$ is negligible. Otherwise we can construct an algorithm \mathcal{B} to solve EKLPN problem. Precisely, \mathcal{B} is constructed by simulating $\text{Game}_{1_i,j}$ or $\text{Game}_{1_i,j-1}$ for \mathcal{A} . \mathcal{B} is given a quadruple $(\mathbf{A}, (\bar{\mathbf{s}}_{j_i}^\top \mathbf{A})^\top, \mathbf{e}, \bar{z}_{j_i})$, where \bar{z}_{j_i} is either $\bar{\mathbf{s}}_{j_i}^\top \mathbf{e}$ or $\bar{\mathbf{s}}'_{j_i}^\top \mathbf{e}$. \mathcal{B} 's behavior is as follows.

KEYGEN. \mathcal{B} picks all $\mathbf{S}_t \xleftarrow{\$} \mathcal{B}_\mu^{\ell \times \ell}$ and sets $\mathbf{B}_t = \mathbf{S}_t \mathbf{A}$ for $t \in [m]$ except that the j th row in secret \mathbf{S}_i and public key \mathbf{B}_i is chosen and set as follows

$$\mathbf{S}_i = \begin{pmatrix} \mathbf{s}_{1_i}^\top \\ \vdots \\ \mathbf{r}_{j_i}^\top \\ \vdots \\ \mathbf{s}_{\ell_i}^\top \end{pmatrix} \xleftarrow{\$} \mathcal{B}_\mu^{\ell \times \ell} \text{ and sets } \mathbf{B}_i = \begin{pmatrix} \mathbf{s}_{1_i}^\top \mathbf{A} \\ \vdots \\ \boxed{\bar{\mathbf{s}}_{j_i}^\top \mathbf{A}} \\ \vdots \\ \mathbf{s}_{\ell_i}^\top \mathbf{A} \end{pmatrix}$$

Finally, \mathcal{B} sends $\mathcal{P} = \{pk_t = (\mathbf{A}, \mathbf{B}_t)\}_{1 \leq t \leq m}$ to the adversary \mathcal{A} , and keeps all $sk_t = \mathbf{S}_t$ to itself. Note that the j^{th} row in \mathbf{S}_i is chosen randomly and the j^{th} row in \mathbf{B}_i is independent of it.

PHASE 1. While receiving a decapsulation query $(q, \mathcal{P}', \mathbf{C} = (\mathbf{c}_0, \{\mathbf{c}_t\}_{1 \leq t \leq m}))$ from adversary \mathcal{A} , the \mathcal{B} does the following thing:

- (a) If $\mathcal{P}' \not\subseteq \mathcal{P}$ or $q \notin \{\#\mathcal{P}'\}$, then return \perp .
- If (a) is passed and $q \neq i$, it computes

$$\tilde{\mathbf{c}} = \mathbf{c}_q - \mathbf{S}_q \cdot \mathbf{c}_0 = \text{ECC}(\mathbf{m}) + (\mathbf{S}'_q - \mathbf{S}_q)\mathbf{e}$$

and then compute $\mathbf{m}' = \text{ECC}^{-1}(\tilde{\mathbf{c}})$ and $(\mathbf{s}', \mathbf{e}') := H_1(\mathbf{m})$. If $\mathbf{c}_0 = \mathbf{A}\mathbf{s}' + \mathbf{e}'$ and $|\mathbf{c}_q - \mathbf{B}_q \mathbf{s}' - \text{ECC}(\mathbf{m}')| \leq \beta$ hold, it outputs $\mathbf{k} := H_2(\mathbf{m}')$; otherwise, outputs \perp .

- If (a) is passed and $q = i$, it computes

$$\begin{aligned} \tilde{\mathbf{c}} &= \mathbf{c}_i - \mathbf{S}_i \cdot \mathbf{c}_0 \\ &= \mathbf{B}_i \mathbf{s} + \mathbf{S}_i \mathbf{e} - \mathbf{S}_i (\mathbf{A} \mathbf{s} + \mathbf{e}) + \text{ECC}(\mathbf{m}) \\ &= \underbrace{\begin{pmatrix} 0 \\ \vdots \\ (\bar{\mathbf{s}}_{j_i}^\top - \mathbf{r}_{j_i}^\top) \mathbf{A} \mathbf{s} \\ \vdots \\ 0 \end{pmatrix}}_{\Delta_i} + \begin{pmatrix} (\mathbf{s}'_{1_i}{}^\top - \mathbf{s}_{1_i}^\top) \mathbf{e} \\ \vdots \\ (\mathbf{s}'_{j_i}{}^\top - \mathbf{r}_{j_i}^\top) \mathbf{e} \\ \vdots \\ (\mathbf{s}'_{\ell_i}{}^\top - \mathbf{s}_{\ell_i}^\top) \mathbf{e} \end{pmatrix} + \text{ECC}(\mathbf{m}) \end{aligned}$$

Let $\tilde{\mathbf{c}} = \text{ECC}(\mathbf{m}) + \Delta_i$, where $|\Delta_i| \leq \frac{2}{3}\alpha\ell + 1 < \alpha\ell$, and $\mathbf{m}' = \text{ECC}^{-1}(\tilde{\mathbf{c}})$ can also be computed correctly. Then compute $(\mathbf{s}', \mathbf{e}') := H_1(\mathbf{m})$. If $\mathbf{c}_0 = \mathbf{A} \mathbf{s}' + \mathbf{e}'$ and $|\mathbf{c}_i - \mathbf{B}_i \mathbf{s}' - \text{ECC}(\mathbf{m}')| \leq \beta$ hold, it outputs $\mathbf{k} := H_2(\mathbf{m}')$; otherwise, outputs \perp .

After that, \mathcal{A} chooses a set $\mathcal{P}^* \subseteq \mathcal{P}$ and sends it to the challenger \mathcal{C} .

CHALLENGE. After receiving the challenge public key set \mathcal{P}^* from \mathcal{A} , \mathcal{C} randomly chooses a uniform $\mathbf{m} \in \{0, 1\}^p$, computes $(\mathbf{s}, \mathbf{e}) := H_1(\mathbf{m})$ and calculates $\mathbf{c}_0^* = \mathbf{A} \mathbf{s} + \mathbf{e} \in \{0, 1\}^\ell$, it chooses a random bit $b \stackrel{\$}{\leftarrow} \{0, 1\}$ then on each public key $pk_t = (\mathbf{A}, \mathbf{B}_t)$, for $1 \leq t \leq i - 1$ it computes

$$\mathbf{c}_i^* = \mathbf{B}_i \mathbf{s} + \mathbf{S}_i \mathbf{e} + \text{ECC}(\mathbf{m}) \in \{0, 1\}^\ell$$

and the i^{th} ciphertext is computed as:

$$\mathbf{c}_i^* = \mathbf{B}_i \mathbf{s} + \begin{pmatrix} \mathbf{s}_{1_i}^\top \mathbf{e} \\ \vdots \\ \mathbf{s}_{(j-1)_i}^\top \mathbf{e} \\ \boxed{\bar{z}_{j_i}} \\ \mathbf{s}_{(j+1)_i}^\top \mathbf{e} \\ \vdots \\ \mathbf{s}'_{\ell_i}{}^\top \mathbf{e} \end{pmatrix} + \text{ECC}(\mathbf{m}) \in \{0, 1\}^\ell$$

with the encapsulated key computed as: $\mathbf{k}_b^* = H_2(\mathbf{m}) \in \{0, 1\}^{\ell'}$ and chooses a $\mathbf{k}_{1-b}^* \stackrel{\$}{\leftarrow} \{0, 1\}^{\ell'}$. Finally, \mathcal{C} sends $(\mathbf{C}^*, \mathbf{k}_b^*, \mathbf{k}_{1-b}^*)$ to \mathcal{A} .

PHASE 2. While receiving a decapsulation query $(q, \mathcal{P}', \mathbf{C} = (\mathbf{c}_0, \{\mathbf{c}_t\}_{1 \leq t \leq m}))$ from adversary \mathcal{A} , the challenger \mathcal{C} does the following thing:

- If $\mathcal{P}' \not\subseteq \mathcal{P}$ or $q \notin [\#\mathcal{P}']$, then return \perp .
- If $\mathbf{C} = \mathbf{C}^*$ and $pk_q \in \mathcal{P}^*$, then return \perp .
- If $\text{mKEM.DeCap}(q, \mathcal{P}', \mathbf{C}, sk_q) = \mathbf{k}_b^*$ return \perp .

If (a), (b), (c) are passed, it responds as in Phase 1.

GUESS. Finally, \mathcal{A} outputs a guess $b \in \{0, 1\}$. If $b = b^*$, \mathcal{B} outputs 1, else outputs 0.

If $\bar{z}_{j_i} = \bar{s}'_{j_i} \mathbf{e}$, then \mathcal{B} simulates the behavior of the challenger in $\text{Game}_{1_i, j-1}$ exactly. Hence, $\Pr[W_{1_i, j-1}] = \Pr[\mathcal{B}(\mathbf{A}, (\bar{s}'_i \mathbf{A})^\top, \mathbf{e}, \bar{s}'_{j_i} \mathbf{e}) = 1]$.

If $\bar{z}_{j_i} = \bar{s}^\top_{j_i} \mathbf{e}$, then \mathcal{B} simulates the behavior of the challenger in $\text{Game}_{1_i, j}$ exactly. Hence, $\Pr[W_{1_i, j}] = \Pr[\mathcal{B}(\mathbf{A}, (\bar{s}^\top_{j_i} \mathbf{A})^\top, \mathbf{e}, \bar{s}^\top_{j_i} \mathbf{e}) = 1]$.

Therefore, for $i \in [m], j \in [\ell]$, we have $|\Pr[W_{1_i, j-1}] - \Pr[W_{1_i, j}]| \leq \text{negl}(n)$.

Game 3. This Game is identical to Game 2 except that the challenger \mathcal{C} changes the challenge phase as follows:

Challenge. After receiving the challenge public key set \mathcal{P}^* from \mathcal{A} , \mathcal{C} randomly chooses a uniform $\mathbf{m} \in \{0, 1\}^\ell$, chooses $\mathbf{u} \xleftarrow{\$} U_\ell$, it chooses a random bit $b \xleftarrow{\$} \{0, 1\}$ then on each public key $pk_i = (\mathbf{A}, \mathbf{B}_i)$, it and defines

$$\begin{aligned} \mathbf{c}_0^* &:= \mathbf{u} && \in \{0, 1\}^\ell \\ \mathbf{c}_i^* &:= \mathbf{S}_i \mathbf{c}_0^* + \text{ECC}(\mathbf{m}) && \in \{0, 1\}^\ell \\ \mathbf{k}_b^* &= H_2(\mathbf{m}) && \in \{0, 1\}^{\ell'} \end{aligned}$$

and chooses a $\mathbf{k}_{1-b}^* \xleftarrow{\$} \{0, 1\}^{\ell'}$. Finally, \mathcal{C} sends $(\mathbf{C}^*, \mathbf{k}_b^*, \mathbf{k}_{1-b}^*)$ to \mathcal{A} .

Note that in Game 2, $\mathbf{c}_i^* = (\mathbf{B}_i)\mathbf{s} + \mathbf{S}_i\mathbf{e} + \text{ECC}(\mathbf{m}) = \mathbf{S}_i(\mathbf{A}\mathbf{s} + \mathbf{e}) + \text{ECC}(\mathbf{m})$.

Lemma 5. $|\Pr[W_3] - \Pr[W_2]| \leq \text{negl}(n)$.

Proof. Since the only difference between Game 2 and Game 3 is that \mathcal{C} replaces $\mathbf{c}_0^* = \mathbf{A}\mathbf{s} + \mathbf{e} \in \{0, 1\}^\ell$ in Game 2 with $\mathbf{c}_0^* = \mathbf{u} \in \{0, 1\}^\ell$ in Game 3, we can construct a distinguisher \mathcal{D} that distinguishes the distributions $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ and (\mathbf{A}, \mathbf{u}) (where $\mathbf{u} \xleftarrow{\$} U_\ell$) with advantage $\text{adv}(n)$ (assuming that \mathcal{A} distinguishes 2 and Game 3 with non-negligible $\text{adv}(n)$), contradicting the assumption. The input of \mathcal{D} is an instance $(\mathbf{A}^\dagger, \mathbf{c}_0^\dagger)$. \mathcal{D} simulates the interaction with \mathcal{A} in the same way Game 2 does, except for KeyGen step. It sets $\mathbf{A} = \mathbf{A}^\dagger$ and in the challenge step it sets $\mathbf{c}_0^* = \mathbf{c}_0^\dagger$. At last, \mathcal{D} outputs whatever \mathcal{A} outputs. When $(\mathbf{A}^\dagger, \mathbf{c}_0^\dagger)$ is chosen according to $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$, then \mathcal{A} 's view in \mathcal{D} 's simulation is identically distributed as in Game 2. If $(\mathbf{A}^\dagger, \mathbf{c}_0^\dagger)$ is chosen according to (\mathbf{A}, \mathbf{u}) , then from \mathcal{A} 's view the distribution in \mathcal{D} 's simulation is identical to that in Game 3. Thus we have $|\Pr[\mathcal{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})] - \Pr[\mathcal{D}(\mathbf{A}, \mathbf{u})]| = |\Pr[W_2] - \Pr[W_3]| = \text{adv}(n)$, which contradicts the assumption. This means that we have $|\Pr[W_2] - \Pr[W_3]| \leq \text{negl}(n)$.

Lemma 6. $\Pr[W_3] = \frac{1}{2} + \text{negl}(n)$.

Proof. This lemma follows from that the challenge ciphertext $(\mathbf{c}_0^*, \mathbf{c}_i^*)$ in Game 3 is uniformly distributed. Since the message \mathbf{m} is chosen uniformly and $H_1(\cdot), H_2(\cdot)$ is modeled as a random oracle from \mathcal{A} 's view, all components in the challenge ciphertext are randomly distributed which ensures that the advantage of the adversary \mathcal{A} is negligible.

In all, we have $\Pr[W_1] = \frac{1}{2} + \text{negl}(n)$, such that $\varepsilon = \text{negl}(n)$. Thus we complete the proof.

4 Conclusion

In this work we constructed the IND-CCA secure mKEM based on the hardness of low-noise LPN assumption. We apply randomness reuse technique to shorten the length of ciphertext. On contrast to previous work, our new scheme is quantum attack resistant in the random oracle model. How to build a mKEM based on LPN without the random oracle model or transfer different session keys to multiple recipients are worth further studying.

Acknowledgements. The work was supported by the National Cryptography Development Fund (Grant No. MMJJ20180106).

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_12
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography with constant input locality. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 92–110. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_6
3. Barbosa, M., Farshim, P.: Efficient identity-based key encapsulation to multiple parties. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 428–441. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_28
4. Barbosa, M., Farshim, P.: Randomness reuse: extensions and improvements. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 257–276. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77272-9_16
5. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes $2^{n/20}$: how $1 + 1 = 0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_31
6. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_7
7. Bellare, M., Kohno, T., Shoup, V.: Stateful public-key cryptosystems: how to encrypt with one 160-bit exponentiation. In: Juels I.A., Wright R.N., di Vimercati S.D. (eds.) CCS 2006, pp. 380–389. ACM (2006)
8. Bellare, M., Paterson, K.G., Thomson, S.: RKA security beyond the linear barrier: IBE, encryption and signatures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 331–348. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_21
9. Berlekamp, E., McEliece, R.J., van Tilborg, H.: On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theor.* **24**(3), 384–386 (1978)
10. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 743–760. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_42
11. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_33

12. Chatterjee, S., Sarkar, P.: Multi-receiver identity-based key encapsulation with shortened ciphertext. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 394–408. Springer, Heidelberg (2006). https://doi.org/10.1007/11941378_28
13. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**, 167–226 (2003)
14. Cash, D., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman problem and applications. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_8
15. Döttling, N.: Low noise LPN: KDM secure public key encryption and sample amplification. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 604–626. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_27
16. Hanaoka, G., Kurosawa, K.: Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 308–325. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_19
17. Hiwatari, H., Tanaka, K., Asano, T., Sakumoto, K.: Multi-recipient public-key encryption from simulators in security proofs. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 293–308. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02620-1_21
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
19. Katz, J., Shin, J.S.: Parallel and concurrent security of the HB and HB⁺ protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_6
20. Kiltz, E., Masny, D., Pietrzak, K.: Simple chosen-ciphertext security from low-noise LPN. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 1–18. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_1
21. Kurosawa, K.: Multi-recipient public-key encryption with shortened ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45664-3_4
22. 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
23. Matsuda, T., Hanaoka, G.: Key encapsulation mechanisms from extractable hash proof systems, revisited. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 332–351. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_21
24. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989). <https://doi.org/10.1007/BFb0019850>
25. Smart, N.P.: Efficient key encapsulation to multiple parties. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 208–219. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30598-9_15
26. Wei, P., Zheng, Y., Wang, W.: Multi-recipient encryption in heterogeneous setting. In: Huang, X., Zhou, J. (eds.) ISPEC 2014. LNCS, vol. 8434, pp. 462–480. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06320-1_34

27. Xavier, B., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Meadows, V.C., Juels, A. (eds.) CCS, pp. 320-329. ACM (2005)
28. Yu, Y., Steinberger, J.: Pseudorandom functions in almost constant depth from low-noise LPN. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 154–183. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_6
29. Yu, Y., Zhang, J.: Cryptography with auxiliary input and trapdoor from constant-noise LPN. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 214–243. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_9



Witness-Indistinguishable Arguments with Σ -Protocols for Bundled Witness Spaces and Its Application to Global Identities

Hiroaki Anada^{1(✉)} and Seiko Arita²

¹ Department of Information Security, University of Nagasaki, W408, 1-1-1, Manabino, Nagayo-cho, Nishisonogi-gun, Nagasaki 851-2195, Japan
anada@sun.ac.jp

² Graduate School of Information Security, Institute of Information Security, 509, 2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama 221-0835, Japan
arita@iisec.ac.jp

Abstract. We propose a generic construction of a Σ -protocol of commit-and-prove type, which is an AND-composition of Σ -protocols on the statements that include a common commitment. Our protocol enables a prover to convince a verifier that the prover knows a bundle of witnesses that have a common component which we call a base witness point. When the component Σ -protocols are of witness-indistinguishable argument systems, our Σ -protocol is also a witness-indistinguishable argument system as a whole. As an application, we propose a decentralized multi-authority anonymous authentication scheme. We first define a syntax and security notions of the scheme. Then we give a generic construction of a decentralized multi-authority anonymous authentication scheme. There a witness is a bundle of witnesses each of which decomposes into a common global identity string and a digital signature on it. We mention an instantiation of the generic scheme in the setting of bilinear groups.

Keywords: Interactive argument · Sigma protocol
Witness indistinguishability · Decentralized · Collusion resistance

1 Introduction

Global identities such as Passport Numbers (PNs) or Social Security Numbers (SSNs) in each country are currently common for identification. They are used not only for governmental identification but also for commercial services; that is, when we want to use a commercial service, we often ask the service administration authority to issue an attribute certificate at the registration stage. In the stage, the authority confirms our identities by the global identity string such as PN or SSN. Once the attribute certificate is issued, we become to be accepted at

the authentication stage of the service. Hence the global identity strings work for us to be issued our attribute certificates. It is notable that recently multi-factor authentication schemes are utilized to prevent misauthentication. In the scheme a user of a service is granted access only after presenting several separate pieces of evidence. Actually the multi-factor authentication of using both a laptop PC, which is connected to the internet by a service provider, and a smartphone, which is activated by a cellular carrier, is getting usual. Thus, there is a compound model that involves independent administration authorities for us to be authenticated and receive benefit of a service.

Privacy protection is a function to be pursued in the authentication, especially recently. The growth of the internet of things and related big data analysis have protecting privacy more critical to involved users. For the purpose, an authentication framework of identity strings and passwords should be evolved into a framework where anonymity is guaranteed at the authentication stage. For example, when a smart household machine generates a report about the situation of a house via the internet as a query for a useful suggestion (such as air conditioning or cooking recipes), the identity information is often unnecessary. A further example is a connected-to-the-internet vehicle which uses a combination of plural services like local traffic information system and the passenger's web-scheduler. The identity information should not be leaked even when the memberships are needed in the registration stages. In this example a user should be authenticated by the service providers at the same time in the authentication stages, anonymously. This is an authentication framework in which plural attributes of a single user are authenticated. However, there is a threat on anonymous authentication frameworks; *the collusion attack*. A malicious user collects private attribute keys from honest users who have different identities, and tries to make a verifier accept anonymously by using the merged attribute keys. Here the vary anonymity is a critical potential drawback from the view point of the collusion attacks.

1.1 Related Work and Our Contribution

A decentralized multi-authority attribute-based signature scheme (DMA-ABS) [11] is an ABS scheme with decentralized key-issuing authorities. In an ABS scheme, a signer has credentials (i.e. private secret keys) on her attributes, and a message has a signing policy expressed as a boolean formula on attributes. The signer is able to sign it if and only if her attributes satisfies the boolean formula. There are assignment patterns and the attribute privacy of an ABS scheme should assure that the signatures do not leak any information on the satisfying pattern. We note that this property also requires the anonymity of the signer's identity. A non-trivial task in constructing an ABS scheme is to assure *both* the collusion resistance and the attribute privacy. On the other hand, allowing decentralized multi-authorities is to have independent issuers each of which generates each private secret attribute-key to the user.

In this paper, we propose a new notion; a witness-indistinguishable argument system (WIA) with Σ -protocols for a *bundled witness space*. It is known that

WIA is a natural building block to achieve anonymity in cryptographic primitives [9]. However, there is no previous work for the multi-prover setting executed by a *hidden single prover* who is able to convince a verifier that she is certainly a single prover. We construct the kind of WIA by employing a commitment scheme as one of the building blocks.

As an application, we give a generic construction of a decentralized multi-authority anonymous authentication scheme, which can be converted into a DMA-ABS scheme by the Fiat-Shamir transform [8]. Actually, if a prover chooses a monotone boolean formula instead of an all-AND formula (as in this paper), and if we apply the Fiat-Shamir transform to the Σ -protocol in our authentication scheme, then we obtain a DMA-ABS scheme.

2 Preliminaries

The security parameter is denoted by λ . The bit length of a string a is denoted by $|a|$. The number of elements of a set S is denoted by $|S|$. A uniform random sampling of an element a from a set S is denoted as $a \in_R S$. The expression $a =_? b$ returns a boolean 1 (TRUE) when $a = b$, and otherwise 0 (FALSE). The expression $a \in_? S$ returns a boolean 1 when $a \in S$, and otherwise 0. When an algorithm A with input a returns z , we denote it as $z \leftarrow A(a)$, or, $A(a) \rightarrow z$. When a probabilistic polynomial-time (PPT, for short) algorithm A with input a and a randomness r on a random tape returns z , we denote it as $z \leftarrow A(a; r)$. When an algorithm A with input a and an algorithm B with input b interact with each other and return z , we denote it as $z \leftarrow \langle A(a), B(b) \rangle$. The transcript of all the messages of the interaction is denoted by $transc(A(a), B(b))$. When an algorithm A accesses an oracle \mathbf{O} , we denote it by $A^{\mathbf{O}}$. When A accesses n oracles $\mathbf{O}_1, \dots, \mathbf{O}_n$ concurrently, i.e. in arbitrarily interleaved order of messages, we denote it by $A^{\mathbf{O}_i}_{i=1}^n$. The probability of an event E is denoted by $\Pr[E]$. The conditional probability of an event E given events F_1, \dots, F_n in this order is denoted by $\Pr[E|F_1, \dots, F_n]$. The distribution of a random variable X is denoted by $dist(X)$. The distribution of a random variable X whose probability is given by a joint probability of random variables X, Y_1, \dots, Y_n is denoted by $dist(X|X, Y_1, \dots, Y_n)$. We say that a probability p is negligible in λ if it is upper-bounded by the inverse of any polynomial $\text{poly}(\lambda)$ of positive coefficients (i.e. $p < 1/\text{poly}(\lambda)$). We say that a probability p is overwhelming in λ if it is lower-bounded by $1 -$ (the inverse of any fixed polynomial $\text{poly}(\lambda)$ of positive coefficients) (i.e. $p > 1 - 1/\text{poly}(\lambda)$).

2.1 Interactive Argument System, Σ -Protocol and Witness-Indistinguishability

Suppose that there exists a predicate Φ that defines the membership of a binary relation R ; i.e., Φ maps $(x, w) \in (\{0, 1\}^*)^2$ to TRUE or FALSE. The relation R is defined as $R \stackrel{\text{def}}{=} \{(x, w) \in (\{0, 1\}^*)^2 | \Phi(x, w) = \text{TRUE}\}$. We say that R is polynomially bounded if there exists a polynomial $\ell(\cdot)$ such that $|w| \leq \ell(|x|)$ for

any $(x, w) \in R$. We say that R is an NP relation if R is polynomially bounded and Φ is computable within polynomial-time in $|x|$ as an algorithm. For a pair $(x, w) \in R$ we call x a statement and w a witness of x . We call R the witness relation, and $\Phi(\cdot, \cdot)$ the predicate of the witness relation R . When a set of public parameter values PP are needed to define the predicate (for example, to set up algebraic operations), we denote it as Φ_{pp} . An NP language L for an NP relation R is defined as the set of all possible statements: $L \stackrel{\text{def}}{=} \{x \in \{0, 1\}^*; \exists w \in \{0, 1\}^*, (x, w) \in R\}$. We denote the set of witnesses of a statement x by $W(x)$: $W(x) \stackrel{\text{def}}{=} \{w \in \{0, 1\}^* \mid (x, w) \in R\}$. We call the union W of all the sets $W(x)$ for $x \in L$ the *witness space* of L : $W \stackrel{\text{def}}{=} \bigcup_{x \in L} W(x)$. We denote an interactive proof system on an NP relation R [1, 10] by $\Pi = (\Pi.\text{Setup}, \text{P}, \text{V})$, where $\Pi.\text{Setup}$ is a set up algorithm for a set of public parameter values PP , and P and V are a pair of interactive algorithms. P , which is called a prover, is probabilistic and unbounded, and V , which is called a verifier, is probabilistic polynomial-time (PPT). If P is also limited to PPT, then Π is called an interactive *argument* system.

Σ -protocol [4, 5]. Let R be an NP relation. A Σ -protocol Σ on the relation R is a 3-move public-coin protocol of an interactive argument system $\Pi = (\Pi.\text{Setup}, \text{P}, \text{V})$ [4, 5]. We introduce six PPT algorithms for a Σ -protocol: $\Sigma = (\Sigma_{\text{com}}, \Sigma_{\text{cha}}, \Sigma_{\text{res}}, \Sigma_{\text{vrf}}, \Sigma_{\text{ext}}, \Sigma_{\text{sim}})$. The first algorithm Σ_{com} is executed by P . On input a pair of a statement and a witness $(x, w) \in R$, it generates a commitment message COM and outputs its inner state St . It returns them as $\Sigma_{\text{com}}(x, w) \rightarrow (\text{COM}, St)$. The second algorithm Σ_{cha} is executed by V . On input the statement x , it reads out the size of the security parameter as 1^λ and chooses a challenge message $\text{CHA} \in_R \text{CHASP}(1^\lambda)$ from the challenge space $\text{CHASP}(1^\lambda) := \{0, 1\}^{\omega(\lambda)}$, where $\omega(\cdot)$ is a super-log function [2]. It returns the message as $\Sigma_{\text{cha}}(x) \rightarrow \text{CHA}$. The third algorithm Σ_{res} is executed by P . On input the state St and the challenge message CHA , it generates a response message RES . It returns the message as $\Sigma_{\text{res}}(St, \text{CHA}) \rightarrow \text{RES}$. The fourth algorithm Σ_{vrf} is executed by V . On input the statement x and the messages COM , CHA and RES , it computes a boolean decision d . It returns the decision as $\Sigma_{\text{vrf}}(x, \text{COM}, \text{CHA}, \text{RES}) \rightarrow d$. If $d = 1$, then we say that P is accepted by V on x . Otherwise, we say that P is rejected by V on x . The vector of all the messages $(\text{COM}, \text{CHA}, \text{RES})$ is called a transcript of the interaction on x .

These four algorithms $(\Sigma_{\text{com}}, \Sigma_{\text{cha}}, \Sigma_{\text{res}}, \Sigma_{\text{vrf}})$ must satisfy the following property.

Completeness. For any $(x, w) \in R$, a prover $\text{P}(x, w)$ has a verifier $\text{V}(x)$ accept with probability 1: $\Pr[\Sigma_{\text{vrf}}(x, \text{COM}, \text{CHA}, \text{RES}) = 1 \mid \Sigma_{\text{com}}(x, w) \rightarrow (\text{COM}, St), \Sigma_{\text{cha}}(x) \rightarrow \text{CHA}, \Sigma_{\text{res}}(St, \text{CHA}) \rightarrow \text{RES}]$.

The fifth algorithm Σ_{ext} concerns with the following property.

Special Soundness. There is a PPT algorithm Σ_{ext} called a *knowledge extractor*, which, on input a statement x and two accepting transcripts with a common commitment message and different challenge messages, $(\text{COM}, \text{CHA}, \text{RES})$ and

$(\text{COM}, \text{CHA}', \text{RES}')$, $\text{CHA} \neq \text{CHA}'$, computes a witness \hat{w} satisfying $(x, \hat{w}) \in R$ with an overwhelming probability in $|x|$: $\hat{w} \leftarrow \Sigma_{\text{ext}}(x, \text{COM}, \text{CHA}, \text{RES}, \text{CHA}', \text{RES}')$.

The sixth algorithm Σ_{sim} concerns with the following property.

Honest-Verifier Zero-Knowledge. There is a PPT algorithm called a *simulator* Σ_{sim} , which, on input a statement x , computes an accepting transcript on x : $(\text{C}\hat{\text{O}}\text{M}, \text{C}\hat{\text{H}}\text{A}, \text{R}\hat{\text{E}}\text{S}) \leftarrow \Sigma_{\text{sim}}(x)$, where the distribution of the simulated transcripts $\text{dist}(\text{C}\hat{\text{O}}\text{M}, \text{C}\hat{\text{H}}\text{A}, \text{R}\hat{\text{E}}\text{S})$ is identical to the distribution of the real accepting transcripts $\text{dist}(\text{COM}, \text{CHA}, \text{RES})$.

Note 1: Our Use Case. In a Σ -protocol the challenge message CHA is a public coin. This property enables us in this paper to use the following variant of the simulator $\Sigma_{\text{sim}}(x)$: On input a simulated challenge message $\text{C}\hat{\text{H}}\text{A}$ that is chosen uniformly at random, the variant generates a commitment $\text{C}\hat{\text{O}}\text{M}$ and a response $\text{R}\hat{\text{E}}\text{S}$: $\text{C}\hat{\text{H}}\text{A} \in_R \text{CHASP}(1^\lambda)$, $(\text{C}\hat{\text{O}}\text{M}, \text{R}\hat{\text{E}}\text{S}) \leftarrow \Sigma_{\text{sim}}(x, \text{C}\hat{\text{H}}\text{A})$.

Witness-Indistinguishability [7,9]. Let R be an NP relation. Suppose that an interactive argument system $\Pi = (\Pi.\text{Setup}, \text{P}, \text{V})$ with a Σ -protocol Σ on the relation R is given. In this paper we focus on the following property.

Perfect Witness Indistinguishability. For any PPT algorithm V^* , any sequences of witnesses $\mathbf{w} = (w_x)_{x \in L}$ and $\mathbf{w}' = (w'_x)_{x \in L}$ s.t. $w_x, w'_x \in W(x)$, any string $x \in L$ and any string $z \in \{0, 1\}^*$, the two distributions $\text{dist}(x, z, \text{transc}(\text{P}(x, w_x), \text{V}^*(x, z)))$ and $\text{dist}(x, z, \text{transc}(\text{P}(x, w'_x), \text{V}^*(x, z)))$ are identical.

2.2 Commit-and-Prove Scheme [3,6]

A commit-and-prove scheme CmtPrv consists of five PPT algorithms: $\text{CmtPrv} = (\text{CmtPrv}.\text{Setup}, \text{Cmt} = (\text{Cmt}.\text{Com}, \text{Cmt}.\text{Vrf}), \Pi = (\text{P}, \text{V}))$.

$\text{CmtPrv}.\text{Setup}(1^\lambda) \rightarrow \text{PP}$. On input the security parameter 1^λ , it generates a set of public parameter values PP . It returns PP .

$\text{Cmt}.\text{Com}(\text{PP}, m) \rightarrow (c, \kappa)$. On input the set of public parameter values PP , a message m in the message space $\text{Msg}(1^\lambda)$, this PPT algorithm generates a commitment c . It also generates an opening key κ . It returns (c, κ) .

$\text{Cmt}.\text{Vrf}(\text{PP}, c, m, \kappa) \rightarrow d$. On input the set of public parameter values PP , a commitment c , a message m and an opening key κ , this deterministic algorithm generates a boolean decision d . It returns d .

The correctness should hold for the commitment part Cmt of the scheme: For any security parameter 1^λ , any set of public parameter values PP and any message $m \in \text{Msg}(1^\lambda)$, $\Pr[d = 1 \mid (c, \kappa) \leftarrow \text{Cmt}.\text{Com}(\text{PP}, m), d \leftarrow \text{Cmt}.\text{Vrf}(\text{PP}, c, m, \kappa)] = 1$.

We denote by Φ_{PP} a predicate that returns the boolean decision: $\Phi_{\text{PP}}(c, (m, \kappa)) \stackrel{\text{def}}{=} (\text{Cmt}.\text{Vrf}(\text{PP}, c, m, \kappa))$. In the scheme there is an interactive argument system $\Pi = (\text{P}, \text{V})$ for the following relation R :

$$R := \{(c, (m, \kappa)) \in \{0, 1\}^* \times (\{0, 1\}^*)^2 \mid \Phi_{\text{PP}}(c, (m, \kappa)) = \text{TRUE}\}.$$

In this paper we focus on the following properties for the commitment part Cmt .

Perfectly Hiding. For any security parameter 1^λ , any set of public parameter values PP and any two messages $m, m' \in \text{Msg}(1^\lambda)$, the two distributions $\text{dist}(c \mid (c, \kappa) \leftarrow \text{Cmt.Com}(\text{PP}, m))$ and $\text{dist}(c \mid (c, \kappa) \leftarrow \text{Cmt.Com}(\text{PP}, m'))$ are identical.

Computationally Binding. The attack of breaking binding property of Cmt by an algorithm \mathbf{A} is defined by the following experiment.

$$\begin{aligned} \text{Exp}_{\text{Cmt}, \mathbf{A}}^{\text{bind}}(1^\lambda) : & \text{PP} \leftarrow \text{CmtPrv.Setup}(1^\lambda), (c, m, \kappa, m', \kappa') \leftarrow \mathbf{A}(\text{PP}) \\ & \text{If } \text{Cmt.Vrf}(\text{PP}, c, m, \kappa) = \text{Cmt.Vrf}(\text{PP}, c, m', \kappa') = 1 \wedge m \neq m', \\ & \text{then Return WIN else Return LOSE} \end{aligned}$$

The advantage of \mathbf{A} over Cmt is defined as $\text{Adv}_{\text{Cmt}, \mathbf{A}}^{\text{bind}}(\lambda) := \Pr[\text{Exp}_{\text{Cmt}, \mathbf{A}}^{\text{bind}}(1^\lambda)$ returns WIN]. The commitment scheme Cmt is said to be *computationally binding* if for any set of public parameter values PP and any PPT algorithm \mathbf{A} , the advantage $\text{Adv}_{\text{Cmt}, \mathbf{A}}^{\text{bind}}(\lambda)$ is negligible in λ .

Note 2: Our Use Case. The commitment generation algorithm Cmt.Com uses random tapes [9]. In this paper we are in the case that a randomness $r \in \{0, 1\}^\lambda$ is used to generate a commitment c , and the opening key κ is the randomness: $\kappa := r$. That is, $\text{Cmt.Com}(\text{PP}, m; r) \rightarrow (c, r)$.

2.3 Digital Signature Scheme [8]

A digital signature scheme Sig consists of four PPT algorithms: $\text{Sig} = (\text{Sig.Setup}, \text{Sig.KG}, \text{Sig.Sign}, \text{Sig.Vrf})$.

$\text{Sig.Setup}(1^\lambda) \rightarrow \text{PP}$. On input the security parameter 1^λ , it generates a set of public parameter values PP . It returns PP .

$\text{Sig.KG}(\text{PP}) \rightarrow (\text{PK}, \text{SK})$. On input the set of public parameter values PP , this PPT algorithm generates a signing key SK and the corresponding public key PK . It returns (PK, SK) .

$\text{Sig.Sign}(\text{PP}, \text{PK}, \text{SK}, m) \rightarrow \sigma$. On input the set of public parameter values PP , the public key PK , the secret key SK and a message m in the message space $\text{Msg}(1^\lambda)$, this PPT algorithm generates a signature σ . It returns σ .

$\text{Sig.Vrf}(\text{PP}, \text{PK}, m, \sigma) \rightarrow d$. On input the public key PK , a message m and a signature σ , it returns a boolean d .

The correctness should hold for the scheme Sig : For any security parameter 1^λ and any message $m \in \text{Msg}(1^\lambda)$, $\Pr[d = 1 \mid \text{PP} \leftarrow \text{Sig.Setup}(1^\lambda), (\text{PK}, \text{SK}) \leftarrow \text{Sig.KG}(\text{PP}), \sigma \leftarrow \text{Sig.Sign}(\text{PP}, \text{PK}, \text{SK}, m), d \leftarrow \text{Sig.Vrf}(\text{PP}, \text{PK}, m, \sigma)] = 1$.

An adaptive chosen-message attack on the scheme Sig by a forger algorithm \mathbf{F} is defined by the following experiment.

$$\begin{aligned} \text{Exp}_{\text{Sig}, \mathbf{F}}^{\text{euf-cma}}(1^\lambda) : & \text{PP} \leftarrow \text{Sig.Setup}(1^\lambda), (\text{PK}, \text{SK}) \leftarrow \text{Sig.KG}(\text{PP}) \\ & (m^*, \sigma^*) \leftarrow \mathbf{F}^{\text{Sig.O}(\text{PP}, \text{PK}, \text{SK}, \cdot)}(\text{PP}, \text{PK}) \\ & \text{If } m^* \notin \{m_j\}_{1 \leq j \leq q_s} \text{ and } \text{Sig.Vrf}(\text{PK}, m^*, \sigma^*) = 1, \\ & \text{then Return WIN else Return LOSE} \end{aligned}$$

In the experiment, \mathbf{F} issues a signing query to its signing oracle $\mathbf{SignO}(\text{pp}, \text{PK}, \text{SK}, \cdot)$ by sending a message m_j at most q_s times ($1 \leq j \leq q_s$). As a reply, \mathbf{F} receives a valid signature σ_j on m_j . After receiving replies, \mathbf{F} returns a message and a signature (m^*, σ^*) . A restriction is imposed on the algorithm \mathbf{F} : The set of queried messages $\{m_j\}_{1 \leq j \leq q_s}$ should not contain the message m^* . The advantage of \mathbf{F} over \mathbf{Sig} is defined as $\mathbf{Adv}_{\mathbf{Sig}, \mathbf{F}}^{\text{euf-cma}}(\lambda) := \Pr[\mathbf{Exp}_{\mathbf{Sig}, \mathbf{F}}^{\text{euf-cma}}(1^\lambda) \text{ returns WIN}]$. The digital signature scheme \mathbf{Sig} is said to be *existentially unforgeable against adaptive chosen-message attacks* if for any given PPT algorithm \mathbf{F} , the advantage $\mathbf{Adv}_{\mathbf{Sig}, \mathbf{F}}^{\text{euf-cma}}(\lambda)$ is negligible in λ .

3 Witness-Indistinguishable Arguments with Σ -Protocols for Bundled Witness Space

In this section, we propose a generic construction of an interactive argument system that is a witness-indistinguishable argument system for a newly introduced *bundled witness space*. Our protocol of the interactive argument system is an AND-composition of Σ -protocols together with a commitment scheme, which is to prove the knowledge of witness pairs each of which consists of two components; one is a common component (such as a global identity string) and the other is an individual component (such as a digital signature issued by an individual authority on the global identity). We prove that our protocol is certainly a Σ -protocol. Finally, we prove that our interactive argument system with the protocol is perfectly witness-indistinguishable under the condition that the employed commitment scheme is perfectly hiding and the component Σ -protocols are perfectly witness-indistinguishable.

3.1 Building Blocks

Component Interactive Argument Systems with Σ -Protocols. For a polynomially bounded integer n , let A be the set of indices: $A := \{1, \dots, n\}$. We start with an efficiently computable predicate Φ_{pp}^a for each index $a \in A$, which determines an NP witness relation R^a :

$$R^a = \{(x^a, w^a) \in \{0, 1\}^* \times \{0, 1\}^* \mid \Phi_{\text{pp}}^a(x^a, w^a) = \text{TRUE}\}, a \in A. \tag{1}$$

We suppose for each $a \in A$ that there is an interactive argument system $\Pi^a = (\Pi.\text{Setup}, \text{P}^a, \text{V}^a)$ which is executed in accordance with a Σ -protocol for the relation R^a :

$$\Sigma^a = (\Sigma_{\text{com}}^a, \Sigma_{\text{cha}}^a, \Sigma_{\text{res}}^a, \Sigma_{\text{vrf}}^a, \Sigma_{\text{ext}}^a, \Sigma_{\text{sim}}^a). \tag{2}$$

We suppose further that the witness space W^a decomposes into two components $W^a = W_0^a \times W_1^a$ for each $a \in A$. *In this paper, our interest is in the case that all the 0th components $W_0^a, a \in A$, are equal, which we denote by W_0 .* We call the equal set W_0 the *base witness space* of the witness spaces $W^a, a \in A$, and

an element $w_0 \in W_0$ a *base witness point*. Then a witness $w^a \in W^a$ consists of w_0 and w_1^a . That is, $W^a = W_0 \times W_1^a \ni (w_0, w_1^a) = w^a$.

Commit-and-Prove Scheme with Σ -Protocol. We employ a commit-and-prove scheme with a Σ -protocol: $\text{CmtPrv} = (\text{CmtPrv.Setup}, \text{Cmt} = (\text{Cmt.Com}, \text{Cmt.Vrf}), \Pi_0 = (\text{P}_0, \text{V}_0))$, where the predicate $\Phi_{0,\text{pp}}$ and the relation R_0 is defined as follows, and Π_0 is executed in accordance with a Σ -protocol Σ_0 :

$$\begin{aligned} \Phi_{0,\text{pp}}(c_0, (w_0, r_0)) &\stackrel{\text{def}}{=} (\text{Cmt.Com}(\text{PP}_0, w_0; r_0) =? (c_0, r_0)), \\ R_0 &\stackrel{\text{def}}{=} \{(c_0, (w_0, r_0)) \in \{0, 1\}^* \times (\{0, 1\}^*)^2 \mid \Phi_{0,\text{pp}}(c_0, (w_0, r_0)) = \text{TRUE}\}, \quad (3) \\ \Sigma_0 &= (\Sigma_{0,\text{com}}, \Sigma_{0,\text{cha}}, \Sigma_{0,\text{res}}, \Sigma_{0,\text{vrf}}, \Sigma_{0,\text{ext}}, \Sigma_{0,\text{sim}}). \quad (4) \end{aligned}$$

Note that a message m to be committed is a base witness point w_0 .

3.2 On the Existence of a Σ -Protocol for Simultaneous Satisfiability

We introduce for each index $a \in A$ the following composed relation determined by the two predicates Φ_{pp}^a and $\Phi_{0,\text{pp}}$. That is, the relation R_0^a is for *simultaneous satisfiability* of Φ_{pp}^a and $\Phi_{0,\text{pp}}$ on the base witness point w_0 : For each $a \in A$,

$$R_0^a := \left\{ (x_0^a = (x^a, c_0), w_0^a = (w_0, w_1^a, r_0)) \mid \left\{ \begin{array}{l} \Phi_{\text{pp}}^a(x^a, (w_0, w_1^a)) = \text{TRUE} \\ \Phi_{0,\text{pp}}(c_0, (w_0, r_0)) = \text{TRUE} \end{array} \right. \right\}. \quad (5)$$

We require here that the Σ -protocols Σ^a and Σ_0 can be merged into a single Σ -protocol Σ_0^a of an interactive argument system $\Pi_0^a = (\Pi.\text{Setup}, \text{CmtPrv.Setup}, \text{P}_0^a, \text{V}_0^a)$ for the above relation R_0^a :

$$\Sigma_0^a = (\Sigma_{0,\text{com}}^a, \Sigma_{0,\text{cha}}^a, \Sigma_{0,\text{res}}^a, \Sigma_{0,\text{vrf}}^a, \Sigma_{0,\text{ext}}^a, \Sigma_{0,\text{sim}}^a). \quad (6)$$

- $\Sigma_{0,\text{com}}^a(x_0^a, w_0^a) \rightarrow (\text{COM}^a, \text{COM}_{a,0}, St_0^a)$. This PPT algorithm is executed by P_0^a . On input a statement $x_0^a = (x^a, c_0)$ and a witness $w_0^a = (w_0, w_1^a, r_0)$, it runs the algorithms $\Sigma_{\text{com}}^a(x^a, (w_0, w_1^a))$ and $\Sigma_{0,\text{com}}(c_0, (w_0, r_0))$ to obtain the commitment messages and the inner states, (COM^a, St^a) and $(\text{COM}_{a,0}, St_{a,0})$, respectively, with a constraint that the knowledge extractor $\Sigma_{0,\text{ext}}^a$ should return a witness which simultaneously satisfies the two predicates Φ^a and Φ_0 on the base witness point w_0 . It sets the state as $St_0^a := (St^a, St_{a,0})$. It returns $(\text{COM}^a, \text{COM}_{a,0}, St_0^a)$. P_0^a sends $(\text{COM}^a, \text{COM}_{a,0})$ to V_0^a as a commitment message, and keeps the state St_0^a .
- $\Sigma_{0,\text{cha}}^a(x_0^a) \rightarrow \text{CHA}$. This PPT algorithm is executed by V_0^a . On input the statement x_0^a , it reads out the size of the security parameter as 1^λ and chooses a challenge message $\text{CHA} \in_R \text{CHASP}(1^\lambda)$. It returns CHA . V_0^a sends CHA to P_0^a as a challenge message.
- $\Sigma_{0,\text{res}}^a(St_0^a, \text{CHA}) \rightarrow (\text{RES}^a, \text{RES}_{a,0})$. This PPT algorithm is executed by P_0^a . On input the state St_0^a and the challenge message CHA , it runs the algorithms $\Sigma_{\text{res}}^a(St^a, \text{CHA})$ and $\Sigma_{0,\text{res}}(St_{a,0}, \text{CHA})$ to obtain the response messages RES^a and $\text{RES}_{a,0}$, respectively, with the constraint that the knowledge extractor

$\Sigma_{0,\text{ext}}^a$ should return a witness which simultaneously satisfies Φ^a and Φ_0 on w_0 . It returns $(\text{RES}^a, \text{RES}_{a,0})$. P_0^a sends $(\text{RES}^a, \text{RES}_{a,0})$ to V_0^a as a response message.

- $\Sigma_{0,\text{vrf}}^a(x_0^a, (\text{COM}^a, \text{COM}_{a,0}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0})) \rightarrow d$. This deterministic algorithm is executed by V_0^a . On input the statement $x_0^a = (x^a, c_0)$ and all the messages $(\text{COM}^a, \text{COM}_{a,0}), \text{CHA}$ and $(\text{RES}^a, \text{RES}_{a,0})$, it runs the algorithms $\Sigma_{\text{vrf}}^a(x^a, \text{COM}^a, \text{CHA}, \text{RES}^a)$ and $\Sigma_{0,\text{vrf}}(c_0, \text{COM}_{a,0}, \text{CHA}, \text{RES}_{a,0})$ to obtain two boolean decisions d^a and $d_{a,0}$. If the both d^a and $d_{a,0}$ are 1, then it returns $d := 1$, and otherwise $d := 0$. V_0^a returns d as the decision of the interactive protocol on x_0^a .
- $\Sigma_{0,\text{ext}}^a(x_0^a, (\text{COM}^a, \text{COM}_{a,0}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0}), \text{CHA}', (\text{RES}^{a'}, \text{RES}_{a,0}')) \rightarrow (\hat{w}_0^a, \hat{w}_1^a, \hat{r}_{a,0})$. This PPT algorithm is for knowledge extraction. On input the statement $x_0^a = (x^a, c_0)$ and two accepting transcripts with a common commitment message and different challenge messages, $((\text{COM}^a, \text{COM}_{a,0}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0}))$ and $((\text{COM}^a, \text{COM}_{a,0}), \text{CHA}', (\text{RES}^{a'}, \text{RES}_{a,0}'))$, $\text{CHA} \neq \text{CHA}'$, it runs the algorithms $\Sigma_{\text{ext}}^a(x^a, \text{COM}^a, \text{CHA}, \text{RES}^a, \text{CHA}', \text{RES}^{a'})$ and $\Sigma_{0,\text{ext}}(c_0, \text{COM}_{a,0}, \text{CHA}, \text{RES}_{a,0}, \text{CHA}', \text{RES}_{a,0}')$ to obtain witnesses $(\hat{w}_0^a, \hat{w}_1^a)$ and $(\hat{w}_{a,0}, \hat{r}_{a,0})$ satisfying $(x^a, (\hat{w}_0^a, \hat{w}_1^a)) \in R^a$ and $(c_0, (\hat{w}_{a,0}, \hat{r}_{a,0})) \in R_0$ with an overwhelming probability in $|x^a|$ and $|c_0|$, respectively. Here the simultaneous satisfiability on w_0 should assure the following equality:

$$\hat{w}_0^a = \hat{w}_{a,0} \text{ with probability one.} \tag{7}$$

It returns $(\hat{w}_0^a, \hat{w}_1^a, \hat{r}_{a,0})$.

- $\Sigma_{0,\text{sim}}^a(x_0^a, \text{C}\tilde{\text{H}}\text{A}) \rightarrow ((\text{C}\tilde{\text{O}}\text{M}^a, \text{C}\tilde{\text{O}}\text{M}_{a,0}), (\text{R}\tilde{\text{E}}\text{S}^a, \text{R}\tilde{\text{E}}\text{S}_{a,0}))$. This PPT algorithm is for the simulation of an accepting transcript. On input a statement $x_0^a = (x^a, c_0)$ and a uniform random string $\text{C}\tilde{\text{H}}\text{A} \in_R \text{CHASp}(1^\lambda)$, it runs the algorithms $\Sigma_{\text{sim}}^a(x^a, \text{C}\tilde{\text{H}}\text{A})$ and $\Sigma_{0,\text{sim}}(c_0, \text{C}\tilde{\text{H}}\text{A})$ to obtain the remaining part of the transcripts $(\text{C}\tilde{\text{O}}\text{M}^a, \text{R}\tilde{\text{E}}\text{S}^a)$ and $(\text{C}\tilde{\text{O}}\text{M}_{a,0}, \text{R}\tilde{\text{E}}\text{S}_{a,0})$, respectively. The simulated messages $((\text{C}\tilde{\text{O}}\text{M}^a, \text{C}\tilde{\text{O}}\text{M}_{a,0}), \text{C}\tilde{\text{H}}\text{A}, (\text{R}\tilde{\text{E}}\text{S}^a, \text{R}\tilde{\text{E}}\text{S}_{a,0}))$ should form $\text{dist}((\text{C}\tilde{\text{O}}\text{M}^a, \text{C}\tilde{\text{O}}\text{M}_{a,0}), \text{C}\tilde{\text{H}}\text{A}, (\text{R}\tilde{\text{E}}\text{S}^a, \text{R}\tilde{\text{E}}\text{S}_{a,0}) \mid \text{gen. by CHASp}(1^\lambda), \Sigma_{0,\text{sim}}^a(x_0^a, \text{C}\tilde{\text{H}}\text{A}))$ which is identical to $\text{dist}((\text{COM}^a, \text{COM}_{a,0}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0}) \mid \text{real accepting})$.

Remark. To construct the algorithm $\Sigma_{0,\text{com}}^a$ of commitment message and the algorithm $\Sigma_{0,\text{res}}^a$ of response message is a non-trivial task. That is, we have to construct $\Sigma_{0,\text{com}}^a$ and $\Sigma_{0,\text{res}}^a$ so that the knowledge extractor $\Sigma_{0,\text{ext}}^a$ returns a witness which *simultaneously* satisfies Φ^a and Φ_0 on a base witness point w_0 . The idea of the construction is to use a common random tape to generate commitment messages COM^a and $\text{COM}_{a,0}$, but we do not describe the inner treatment of the random tapes in $\Sigma_{0,\text{com}}^a$ and $\Sigma_{0,\text{res}}^a$ for generality. Hence our approach is to show the construction when we instantiate the Σ -protocol Σ_0^a .

3.3 Bundled Witness Space

We now introduce an NP witness relation for our *bundled witness space*. We first fix the base witness point w_0 in the base witness space W_0 and consider a subset

$R_{w_0}^a$ for each NP witness relation $R^a, a \in A$:

$$R_{w_0}^a := \{(x^a, w^a) \in R^a \mid w^a = (w_0, w_1^a) \text{ for some } w_1^a \in R^a, a \in A. \quad (8)$$

Then we run the base witness point w_0 to claim the following property.

Claim 1. *For a polynomially bounded integer n , let A be the set of indices $\{1, \dots, n\}$. Then we have:*

$$\bigcup_{w_0 \in W_0} \left(\prod_{a \in A} R_{w_0}^a \right) \subset \prod_{a \in A} \left(\bigcup_{w_0 \in W_0} R_{w_0}^a \right) = \prod_{a \in A} R^a. \quad (9)$$

Proof. The equality of the right-hand side is because $\bigcup_{w_0 \in W_0} R_{w_0}^a = R^a$. An element of the left hand side is of the form $(x^1, (w_0, w_1^1)), \dots, (x^n, (w_0, w_1^n))$ where $w_0 \in W_0$ and $(x^a, (w_0, w_0^a)) \in R^a$ for $a \in A$. This is an element of $\prod_{a \in A} R^a$, and hence the inclusion follows. \square

Deleting the redundancy, we obtain the following one-to-one correspondence:

$$R_{\text{bund}}^{a \in A} \stackrel{\text{def}}{=} \{((x^a)^{a \in A}, w_0, (w_1^a)^{a \in A}) \mid (x^a, (w_0, w_1^a)) \in R^a, a \in A\} \simeq \bigcup_{w_0 \in W_0} \left(\prod_{a \in A} R_{w_0}^a \right).$$

Claim 2. *For a polynomially bounded integer n , let A be the set of indices $\{1, \dots, n\}$. Then the relation $R_{\text{bund}}^{a \in A}$ is an NP relation.*

Proof. Omitted. (will appear in the full version).

Definition 1 (Relation for Bundled Witness Space). *For a polynomially bounded integer n , an NP witness relation for the bundled witness spaces is defined as $R_{\text{bund}}^{a \in A}$.*

Definition 2 (Bundled Witness Space). *For a polynomially bounded integer n , let A be the set of indices $\{1, \dots, n\}$. Let $R^a, a \in A$ be NP witness relations where each witness space decomposes $W^a = W_0 \times W_1^a, a \in A$. Then the bundled witness space is defined as follows.*

$$W_{\text{bund}}^{a \in A} \stackrel{\text{def}}{=} W_0 \times (W_1^a)^{a \in A}. \quad (10)$$

3.4 Generic Construction of Σ -Protocol for Bundled Witness Space

By using the above Σ -protocols $(\Sigma_0^a)^{a \in A}$ and a commitment generation algorithm **Cmt.Com**, we construct an interactive argument system $\Pi_{\text{bund}}^{a \in A} = (\text{P}, \text{V})$ for the witness relation $R_{\text{bund}}^{a \in A}$ with a protocol $\Sigma_{\text{bund}}^{a \in A}$. $\Sigma_{\text{bund}}^{a \in A}$ is actually a Σ -protocol, which consists of the six PPT algorithms described below (see also Fig. 1):

$$\Sigma_{\text{bund}}^{a \in A} = (\Sigma_{\text{bund}, \text{com}}^{a \in A}, \Sigma_{\text{bund}, \text{cha}}^{a \in A}, \Sigma_{\text{bund}, \text{res}}^{a \in A}, \Sigma_{\text{bund}, \text{vrf}}^{a \in A}, \Sigma_{\text{bund}, \text{ext}}^{a \in A}, \Sigma_{\text{bund}, \text{sim}}^{a \in A}). \quad (11)$$

- $\Sigma_{\text{bnd,com}}^{a \in A}((x^a)^{a \in A}, (w_0, (w_1^a)^{a \in A})) \rightarrow (c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A}, St)$. This PPT algorithm is executed by P. On input a statement that is a vector $(x^a)^{a \in A}$ and a witness that is a vector $(w_0, (w_1^a)^{a \in A})$, it computes a commitment c_0 to the base witness point w_0 with a randomness $r_0 \in_R \{0, 1\}^\lambda$ by running the commitment generation algorithm of **Cmt**: $(c_0, r_0) \leftarrow \text{Cmt.Com}(w_0; r_0)$. It sets the extended statement as $x_0^a := (x^a, c_0)$ and the extended witness as $w_0^a := (w_0, w_1^a, r_0)$ for each $a \in A$. It runs the algorithms $\Sigma_{0,\text{com}}^a(x_0^a, w_0^a)$ to obtain $(\text{COM}^a, \text{COM}_{a,0}, St_0^a)$ for each $a \in A$. It sets the state as $St := (St_0^a)^{a \in A}$. It returns $(c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A}, St)$. P sends $(c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A})$ to V as a commitment message, and keeps the state St .
- $\Sigma_{\text{bnd,cha}}^{a \in A}((x^a)^{a \in A}) \rightarrow \text{CHA}$. This PPT algorithm is executed by V. On input the statement $(x^a)^{a \in A}$, it reads out the size of the security parameter as 1^λ and chooses a challenge message $\text{CHA} \in_R \text{CHASp}(1^\lambda)$. It returns CHA. V_0^a sends CHA to P_0^a as a challenge message.
- $\Sigma_{\text{bnd,res}}^{a \in A}(St, \text{CHA}) \rightarrow (\text{RES}^a, \text{RES}_{a,0})^{a \in A}$. This PPT algorithm is executed by P. On input the state St and the challenge message CHA, it runs the algorithms $\Sigma_{0,\text{res}}^a(St_0^a, \text{CHA})$ to obtain $(\text{RES}^a, \text{RES}_{a,0})$ for each $a \in A$. It returns $(\text{RES}^a, \text{RES}_{a,0})$. P sends $(\text{RES}^a, \text{RES}_{a,0})^{a \in A}$ to V as a response message.
- $\Sigma_{\text{bnd,vrf}}^{a \in A}((x^a)^{a \in A}) \rightarrow d$. This deterministic algorithm is executed by V. On input the statement $(x^a)^{a \in A}$ and all the messages $(c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A})$, CHA and $(\text{RES}^a, \text{RES}_{a,0})^{a \in A}$, it first sets the extended statement as $x_0^a := (x^a, c_0)$ for each $a \in A$. Then it runs the algorithms $\Sigma_{0,\text{vrf}}^a(x_0^a, \text{COM}^a, \text{COM}_{a,0}, \text{CHA}, \text{RES}^a, \text{RES}_{a,0})$ to obtain boolean decisions, for each $a \in A$. If all the decisions are 1, then V returns 1, and otherwise, 0.

These four algorithms $(\Sigma_{\text{bnd,com}}^{a \in A}, \Sigma_{\text{bnd,cha}}^{a \in A}, \Sigma_{\text{bnd,res}}^{a \in A}, \Sigma_{\text{bnd,vrf}}^{a \in A})$ must satisfy the following property.

Proposition 1 (Completeness). *If Cmt is correct, and if Σ_0^a is complete for $a \in A$, then our $\Sigma_{\text{bnd}}^{a \in A}$ is complete.*

Proof. The completeness of our $\Pi_{\text{bnd}}^{a \in A}$ comes from the correctness of Cmt and the completeness of Π_0^a for each $a \in A$. □

- $\Sigma_{\text{bnd,ext}}^{a \in A}((x^a)^{a \in A}, (c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0})^{a \in A}, \text{CHA}', ((\text{RES}^a)', (\text{RES}_{a,0}')^{a \in A})) \rightarrow (\hat{w}_0, (\hat{w}_1^a)^{a \in A})$. This PPT algorithm is for knowledge extraction. On input the statement $(x^a)^{a \in A}$ and two accepting transcripts with a common commitment message and different challenge messages, $((c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0})^{a \in A})$ and $((c_0, (\text{COM}^a, \text{COM}_{a,0})^{a \in A}), \text{CHA}', (\text{RES}^a', \text{RES}_{a,0}')^{a \in A})$, $\text{CHA} \neq \text{CHA}'$, it first sets the extended statement as $x_0^a := (x^a, c_0)$ for each $a \in A$. Then it runs the algorithms $\Sigma_{0,\text{ext}}^a(x_0^a, (\text{COM}^a, \text{COM}_{a,0}), \text{CHA}, (\text{RES}^a, \text{RES}_{a,0}), \text{CHA}', (\text{RES}^a', \text{RES}_{a,0}'))$ to obtain $(\hat{w}_0^a, \hat{w}_1^a, \hat{r}_0^a)$ for each $a \in A$. If this event does not occur (i.e. at least at one a $\Sigma_{0,\text{ext}}^a$ fails to extract a witness), then it returns \perp . Otherwise, if $\hat{w}_0^a = \hat{w}_0^{a'}$ for any $a, a' \in A$, then it sets the common value $\hat{w}_0 := \hat{w}_0^a$ and returns $(\hat{w}_0, (\hat{w}_1^a)^{a \in A})$. Otherwise it returns \perp^* . The binding

property of the commitment scheme \mathbf{Cmt} assures that the former case holds with an overwhelming probability, as claimed in the following proposition.

Proposition 2. (Special Soundness). *If \mathbf{Cmt} is correct and computationally binding, and if Σ_0^a has the special soundness for $a \in A$, then our $\Sigma_{\text{bind}}^{a \in A}$ has the special soundness.*

Proof. Omitted. (will appear in the full version).

Note 3. For simplicity of the later discussion, we hereafter assume that, for all $a \in A$, $\Pr[\Sigma_{0,\text{ext}}^a \text{ returns a witness}] = 1$. That is, we assume that $\Pr[\Sigma_{0,\text{ext}}^a \text{ returns } \perp] = 0$ for each $a \in A$.

- $\Sigma_{\text{bind},\text{sim}}^{a \in A}((x^a)^{a \in A}, \text{C}\check{\text{H}}\text{A}) \rightarrow ((\tilde{c}_0, (\text{C}\check{\text{O}}\text{M}^a, \text{C}\check{\text{O}}\text{M}_0^a)^{a \in A}), (\text{R}\check{\text{E}}\text{S}^a, \text{R}\check{\text{E}}\text{S}_0^a)^{a \in A})$. This PPT algorithm is for the simulation of an accepting transcript. On input a statement $(x^a)^{a \in A}$ and a uniform random string $\text{C}\check{\text{H}}\text{A} \in_R \text{CHASp}(1^\lambda)$, it first chooses a base witness point $\tilde{w}_0 \in_R W_0$ uniformly at random, and runs the commitment generation algorithm with a randomness \tilde{r}_0 , $\mathbf{Cmt.Com}(\tilde{w}_0; \tilde{r}_0) \rightarrow (\tilde{c}_0, \tilde{r}_0)$, to obtain a commitment \tilde{c}_0 . Then it sets the extended statement as $x_0^a := (x^a, \tilde{c}_0)$ for each $a \in A$. Then, it runs the algorithms $\Sigma_{0,\text{sim}}^a(x_0^a, \text{C}\check{\text{H}}\text{A})$ to obtain $((\text{C}\check{\text{O}}\text{M}^a, \text{C}\check{\text{O}}\text{M}_{a,0}), (\text{R}\check{\text{E}}\text{S}^a, \text{R}\check{\text{E}}\text{S}_{a,0}))$ for each $a \in A$. It returns $((\tilde{c}_0, (\text{C}\check{\text{O}}\text{M}^a, \text{C}\check{\text{O}}\text{M}_{a,0})^{a \in A}), (\text{R}\check{\text{E}}\text{S}^a, \text{R}\check{\text{E}}\text{S}_{a,0})^{a \in A})$.

Proposition 3. (Honest-Verifier Zero-Knowledge). *If \mathbf{Cmt} is perfectly hiding, and if Σ_0^a is honest-verifier zero-knowledge for $a \in A$, then our $\Sigma_{\text{bind}}^{a \in A}$ is honest-verifier zero-knowledge.*

Proof. Omitted. (will appear in the full version).

Theorem 1. *If \mathbf{Cmt} is correct, computationally binding and perfectly hiding, and if Σ_0^a is a Σ -protocol for $a \in A$, then our protocol $\Sigma_{\text{bind}}^{a \in A}$ is a Σ -protocol.*

Proof. Propositions 1, 2 and 3 deduces that $\Sigma_{\text{bind}}^{a \in A}$ is a Σ -protocol. \square

Theorem 2. *If the component interactive proof system Π_0^a with Σ_0^a is perfectly witness-indistinguishable for each $a \in A$, and if \mathbf{Cmt} is perfectly hiding, then our interactive argument system $\Pi_{\text{bind}}^{a \in A}$ with $\Sigma_{\text{bind}}^{a \in A}$ is perfectly witness-indistinguishable.*

Proof. Omitted. (will appear in the full version).

4 Decentralized Multi-authority Anonymous Authentication Scheme

In this section, we give a syntax and security definitions of an interactive anonymous authentication scheme $\mathbf{a-auth}$ in a decentralized multi-authority setting on key generation.

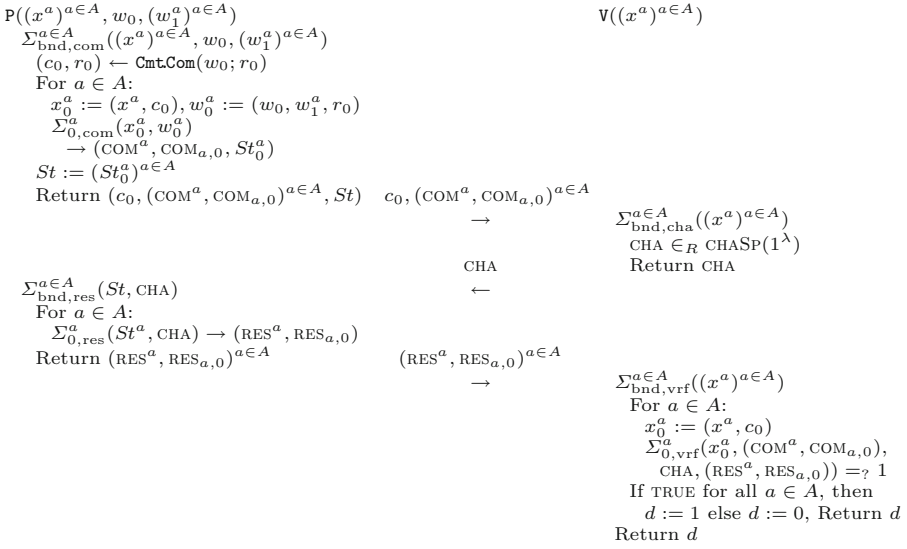


Fig. 1. The protocol $\Sigma_{\text{bnd}}^{a \in A}$ of our proof system $\Pi_{\text{bnd}}^{a \in A}$ for the NP witness relation $R_{\text{bnd}}^{a \in A}$.

4.1 Syntax and Security Definitions

Our **a-auth** consists of five PPT algorithms, (**Setup**, **AuthKG**, **PrivKG**, **P**, **V**).

- **Setup**(1^λ) \rightarrow PP. This PPT algorithm is needed to generate a set of public parameter values PP. On input the security parameter 1^λ , it generates the set of values PP. It returns PP.
- **AuthKG**(PP, a) \rightarrow ($\text{PK}^a, \text{MSK}^a$). This PPT algorithm is executed by a key-issuing authority indexed by a positive integer a . On input the set of public parameter values PP and the authority index a , it generates the a -th public key PK^a of the authority and the corresponding a -th master secret key MSK^a . It returns ($\text{PK}^a, \text{MSK}^a$).
- **PrivKG**(PP, $\text{PK}^a, \text{MSK}^a, \text{gid}$) \rightarrow sk_{gid}^a . This PPT algorithm is executed by the a -th key-issuing authority. On input the set of public parameter values PP, the a -th public and master secret keys ($\text{PK}^a, \text{MSK}^a$) and a string **gid** of a prover (a global identity string), it generates a private secret key sk_{gid}^a of a prover. It returns sk_{gid}^a .
- $\langle \text{P}(\text{PP}, (\text{PK}^a, \text{sk}_{\text{gid}}^a)^{a \in A'}), \text{V}(\text{PP}, (\text{PK}^a)^{a \in A'}) \rangle \rightarrow d$. These two interactive PPT algorithms are a prover who is to be authenticated, and a verifier who confirms that the prover certainly knows the secret keys for indices $a \in A'$, respectively, where A' denotes a subset of all indices at which the prover is issued her private secret keys by authorities. On input the set of public parameter values PP and the public keys $(\text{PK}^a)^{a \in A'}$ to P and V and the corresponding private secret keys $(\text{sk}_{\text{gid}}^a)^{a \in A'}$ to P, P and V interact with each other. After at most polynomially many (in λ) moves of messages between P and V, V returns $d := 1$ (“accept”) or $d := 0$ (“reject”).

We discuss two security notions for our authentication scheme **a-auth**.

Security Against Concurrent and Collusion Attack of Misauthentication. One of the possible attacks to cause misauthentication is the concurrent and collusion attack on our **a-auth**. For a formal treatment we define the following experiment on **a-auth** and an adversary algorithm **A**.

$$\begin{aligned} \mathbf{Expr}_{\mathbf{a-auth}, \mathbf{A}}^{\text{conc-coll}}(1^\lambda) : & q_A \leftarrow \mathbf{A}(1^\lambda), A := \{1, \dots, q_A\}, \text{PP} \leftarrow \text{Setup}(1^\lambda) \\ & \text{For } a \in A : (\text{PK}^a, \text{MSK}^a) \leftarrow \text{AuthKG}(\text{PP}, a) \\ & q_I \leftarrow \mathbf{A}(\text{PP}, (\text{PK}^a)^{a \in A}), I := \{1, \dots, q_I\}, \text{For } i \in I : \text{gid}_i \in_R \{0, 1\}^\lambda \\ & \text{For } a \in A : \text{For } i \in I : \text{sk}_{\text{gid}_i}^a \leftarrow \text{PrivKG}(\text{PP}, \text{PK}^a, \text{MSK}^a, \text{gid}_i) \\ & (A^*, St^*) \leftarrow \mathbf{A}^{\text{P}(\text{PP}, (\text{PK}^a, \text{sk}_{\text{gid}_i}^a)^{a \in A})}_{|i \in I}, \text{PrivKO}(\text{PP}, \text{PK}^{\cdot}, \text{MSK}^{\cdot}, \cdot)}(\text{PP}, (\text{PK}^a)^{a \in A}) \\ & \langle \mathbf{A}(St^*), \mathbf{V}(\text{PP}, (\text{PK}^a)^{a \in A^*}) \rangle \rightarrow d, \text{If } d = 1 \text{ then Return WIN else Return LOSE} \end{aligned}$$

Intuitively, the above experiment describes the attack as follows. The adversary algorithm **A**, on input the security parameter 1^λ , first outputs the number q_A of key-issuing authorities. Then, on input the set of public parameter values PP and the issued public keys $(\text{PK}^a)^{a \in A}$, **A** outputs the number q_I of provers with which **A** interacts concurrently (i.e. in arbitrarily interleaved order of messages). In addition, **A** collects at most q_{sk} private secret keys by issuing queries to the private secret key oracle $\text{PrivKO}(\text{PP}, \text{PK}^{\cdot}, \text{MSK}^{\cdot}, \cdot)$ with an authority index $a \in A$ and a global identity string $\text{gid}_j \in \{0, 1\}^\lambda$ for $j = q_I + 1, \dots, q_I + q_{\text{sk}}$. We denote by A_j the set of authority indices for which the queries with the global identity string gid_j were issued. That is, $A_j := \{a \in A \mid \mathbf{A} \text{ receives } \text{sk}_{\text{gid}_j}^a\}$, $j = q_I + 1, \dots, q_I + q_{\text{sk}}$. We here require that the numbers q_A , q_I and q_{sk} are bounded by a polynomial in λ . At the last of this “learning phase”, **A** outputs a target set of authority indices A^* and its inner state St^* . Next, in the “attacking phase”, on input the inner state St^* , the adversary **A** interacts with the verifier $\mathbf{V}(\text{PP}, (\text{PK}^a)^{a \in A^*})$. If the decision d of \mathbf{V} is 1, then the experiment returns WIN and otherwise, returns LOSE. A restriction is imposed on the adversary **A**: The target set of authority indices A^* should not be a subset of any single set A_j : $A^* \not\subseteq A_j$, $j = q_I + 1, \dots, q_I + q_{\text{sk}}$. This restriction is because, otherwise, **A** is given private secret keys for A^* on a single gid_{i^*} for some i^* , $q_I < i^* \leq q_I + q_{\text{sk}}$, and then **A** can trivially be accepted in the attacking phase.

The advantage of an adversary **A** over our authentication scheme **a-auth** in the experiment is defined as: $\text{Adv}_{\mathbf{a-auth}, \mathbf{A}}^{\text{conc-coll}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Expr}_{\mathbf{a-auth}, \mathbf{A}}^{\text{conc-coll}}(1^\lambda) = \text{WIN}]$. An authentication scheme **a-auth** is called secure against concurrent and collusion attacks of misauthentication if, for any given PPT algorithm **A**, the advantage $\text{Adv}_{\mathbf{a-auth}, \mathbf{A}}^{\text{conc-coll}}(\lambda)$ is negligible in λ .

Anonymity. A critical feature to be attained is provers’ anonymity on global identities when the provers are authenticated. For a formal treatment we define

the following experiment on **a-auth** and an adversary algorithm **A**.

Expr_{**a-auth**,**A**}^{ano}(1^λ) : $q_A \leftarrow \mathbf{A}(1^\lambda), A := \{1, \dots, q_A\}, \text{PP} \leftarrow \text{Setup}(1^\lambda)$
 For $a \in A$: $(\text{PK}^a, \text{MSK}^a) \leftarrow \text{AuthKG}(\text{PP}, a)$
 $\text{gid}_0, \text{gid}_1 \leftarrow \mathbf{A}(\text{PP}, (\text{PK}^a)^{a \in A})$
 For $a \in A$: For $i \in \{0, 1\}$: $\text{sk}_{\text{gid}_i}^a \leftarrow \text{PrivKG}(\text{PP}, \text{PK}^a, \text{MSK}^a, \text{gid}_i)$
 $b \in_R \{0, 1\}, b^* \leftarrow \mathbf{A}^{\text{P}(\text{PP}, (\text{PK}^a, \text{sk}_{\text{gid}_b}^a)^{a \in A})}(\text{PP}, (\text{PK}^a, \text{sk}_{\text{gid}_0}^a, \text{sk}_{\text{gid}_1}^a)^{a \in A})$
 If $b = b^*$, then Return WIN, else Return LOSE

Intuitively, the above experiment describes the attack as follows. The adversary algorithm **A**, on input the security parameter 1^λ, first outputs the number q_A of key-issuing authorities. Then, on input the issued public keys $(\text{PK}^a)^{a \in A}$, **A** designates two identity strings **gid**₀ and **gid**₁ (as is usual in the indistinguishability games). Next, **A** interacts with a prover **P** on input even the private secret keys $(\text{sk}_{\text{gid}_b}^a)^{a \in A}$, where the index b is chosen uniformly at random. If the decision b^* of **A** is equal to b , then the experiment returns WIN and otherwise, returns LOSE.

The advantage of an adversary **A** over our authentication scheme **a-auth** in the experiment is defined as: $\text{Adv}_{\mathbf{a-auth}, \mathbf{A}}^{\text{ano}}(\lambda) \stackrel{\text{def}}{=} |\Pr[\text{Expr}_{\mathbf{a-auth}, \mathbf{A}}^{\text{ano}}(1^\lambda) = \text{WIN}] - (1/2)|$. An authentication scheme **a-auth** is called to have anonymity if, for any PPT algorithm **A**, the advantage $\text{Adv}_{\mathbf{a-auth}, \mathbf{A}}^{\text{ano}}(\lambda)$ is negligible in λ .

4.2 Generic Construction

We give a generic construction of our authentication scheme **a-auth**. The building blocks are the interactive proof system $\Pi_{\text{bnd}}^{a \in A}$ with our Σ -protocol $\Sigma_{\text{bnd}}^{a \in A}$ and a digital signature scheme **Sig**. We note that a commit-and-prove scheme **CmtPrv** is employed in $\Sigma_{\text{bnd}}^{a \in A}$.

- **Setup**(1^λ) → PP. On input the security parameter 1^λ, this PPT algorithm generates a set of public parameter values by running the setup algorithms **Sig.Setup**(1^λ), Π .**Setup**(1^λ) and **CmtPrv.Setup**(1^λ). These algorithms are for the digital signature scheme **Sig**, the interactive argument systems $(\Pi_0^a)^{a \in A}$, and the commitment generation algorithm **Cmt.Com**. They generate $\text{PP}_{\text{Sig}}, \text{PP}_\Pi$ and PP_{Cmt} , respectively. It merges them as $\text{PP} := (\text{PP}_{\text{Sig}}, \text{PP}_\Pi, \text{PP}_{\text{Cmt}})$. It returns PP.
- **AuthKG**(PP, a) → (PK^{*a*}, MSK^{*a*}). On input the set of public parameter values PP and an authority index a , this PPT algorithm executes the key generation algorithm **Sig.KG**(PP_{Sig}) to obtain a signing key SK and the corresponding public key PK. It sets the master secret key as $\text{MSK}^a := \text{SK}$ and the corresponding public key as $\text{PK}^a := \text{PK}$. It returns (PK^{*a*}, MSK^{*a*}).
- **PrivKG**(PP, PK^{*a*}, MSK^{*a*}, **gid**) → sk^{*a*}_{**gid**}. On input the set of public parameter values PP, a public key PK^{*a*}, the corresponding master secret key MSK^{*a*} and a string **gid**, this PPT algorithm executes the signing algorithm

$\text{Setup}(1^\lambda)$ $\text{PP}_{\text{Sig}} \leftarrow \text{SigSetup}(1^\lambda)$ $\text{PP}_{\Pi} \leftarrow \Pi.\text{Setup}(1^\lambda)$ $\text{PP}_{\text{CmtPrv}} \leftarrow \text{CmtPrvSetup}(1^\lambda)$ $\text{PP} := (\text{PP}_{\Pi}, \text{PP}_{\text{CmtPrv}}, \text{PP}_{\text{Sig}})$ Return PP	$\text{AuthKG}(\text{PP}, a)$ $(\text{SK}, \text{PK}) \leftarrow \text{SigKG}(\text{PP}_{\text{Sig}})$ $\text{PK}^a := \text{PK}, \text{MSK}^a := \text{SK}$ Return $(\text{PK}^a, \text{MSK}^a)$	$\text{PrivKG}(\text{PP}, \text{PK}^a, \text{MSK}^a, \text{gid})$ $\sigma_{\text{gid}}^a \leftarrow \text{SigSign}(\text{PP}_{\text{Sig}}, \text{PK}^a, \text{MSK}^a, \text{gid})$ $\text{sk}_{\text{gid}}^a := \sigma_{\text{gid}}^a$ Return sk_{gid}^a
$\text{P}(\text{PP}, (\text{PK}^a)^{a \in A}, (\text{sk}_{\text{gid}}^a)^{a \in A})$ For $a \in A$: $x^a := \text{PK}^a, w_1^a := \text{sk}_{\text{gid}}^a$ $w_0 := \text{gid}$	(Execute $\Sigma_{\text{bnd}}^{a \in A}$)	
		$\text{V}(\text{PP}, (\text{PK}^a)^{a \in A})$ For $a \in A$: $x^a := \text{PK}^a$ Return $(d \leftarrow \Sigma_{\text{bnd}, \text{vrf}}^{a \in A})$

Fig. 2. Generic construction of our decentralized multi-authority anonymous authentication scheme **a-auth**.

$\text{Sig.Sign}(\text{PP}_{\text{Sig}}, \text{PK}^a, \text{MSK}^a, \text{gid})$ to obtain a digital signature σ_{gid}^a on the message gid . It puts a private secret key sk_{gid}^a as $\text{sk}_{\text{gid}}^a := \sigma_{\text{gid}}^a$. It returns sk_{gid}^a .

- $\text{P}(\text{PP}, (\text{PK}^a)^{a \in A}, (\text{sk}_{\text{gid}}^a)^{a \in A})$ and $\text{V}(\text{PP}, (\text{PK}^a)^{a \in A})$. On input the set of public parameter values PP and the public keys $(\text{PK}^a)^{a \in A}$ to the prover P and the verifier V, and the corresponding private secret keys $(\text{sk}_{\text{gid}}^a)^{a \in A}$ to P, PPT algorithms P and V first set the statements as $x^a := \text{PK}^a$ for $a \in A$ and P sets the witness as $w_0 := \text{gid}$ and $w_1^a := \text{sk}_{\text{gid}}^a$ for $a \in A$. The witness spaces $W^a, a \in A$ are described as follows: $W^a = W_0 \times W_1^a, W_0 = \{\text{gid} \mid \text{string of length } \lambda\} = \{0, 1\}^\lambda, W_1^a = \{\sigma_{\text{gid}}^a \mid \sigma_{\text{gid}}^a \leftarrow \text{Sig.Sign}(\text{PP}_{\text{Sig}}, \text{PK}^a, \text{MSK}^a, \text{gid}) \text{ for some } \text{gid} \in W_0\}$. P and V execute the Σ protocol $\Sigma_{\text{bnd}}^{a \in A}$. V returns the returned boolean d of the verifier algorithm $\Sigma_{\text{bnd}, \text{vrf}}^{a \in A}$.

4.3 Properties

Theorem 3. *If the component proof system Π_0^a is perfectly witness-indistinguishable for each $a \in A$, if the commitment scheme Cmt is perfectly hiding and computationally binding, and if the digital signature scheme Sig is existentially unforgeable against adaptive chosen-message attacks, then our **a-auth** is secure against concurrent and collusion attacks. More precisely, let q_A denote the maximum number of authorities. For any given PPT algorithm **A** that executes a concurrent and collusion attack on our **a-auth** in accordance with the experiment $\text{Expr}_{\text{a-auth}, \mathbf{A}}^{\text{conc-coll}}(1^\lambda)$, there exists a PPT algorithm **F** that generates an existential forgery on Sig in accordance with the experiment $\text{Exp}_{\text{Sig}, \mathbf{F}}^{\text{euf-cma}}(1^\lambda)$ and there exists a PPT algorithm **B** that breaks the bandaging property of Cmt in accordance with the experiment $\text{Exp}_{\text{Cmt}, \mathbf{B}}^{\text{bind}}(1^\lambda)$ satisfying the following inequality.*

$$\text{Adv}_{\text{a-auth}, \mathbf{A}}^{\text{conc-coll}}(\lambda) \leq \frac{1}{|\text{CHASp}(1^\lambda)|} + \sqrt{\frac{2^\lambda}{2^\lambda - 1} \cdot q_A \cdot \text{Adv}_{\text{Sig}, \mathbf{F}}^{\text{euf-cma}}(\lambda) + \text{Adv}_{\text{Cmt}, \mathbf{B}}^{\text{bind}}(\lambda)}.$$

Proof. Omitted. (will appear in the full version).

Theorem 4. *If the component proof system Π_0^a is perfectly witness-indistinguishable for each $a \in A$, and if the commitment scheme Cmt is perfectly hiding, then our $\mathbf{a}\text{-auth}$ has anonymity. More precisely, for any given PPT algorithm \mathbf{A} that executes the anonymity game on our $\mathbf{a}\text{-auth}$ in accordance with the experiment $\text{Expr}_{\mathbf{a}\text{-auth}, \mathbf{A}}^{\text{ano}}(1^\lambda)$, the following equality holds.*

$$\text{Adv}_{\mathbf{a}\text{-auth}, \mathbf{A}}^{\text{ano}}(\lambda) = 0.$$

Proof. Omitted. (will appear in the full version).

5 On Instantiation and Implementation

In this section, we briefly discuss instantiation and implementation of our generic authentication scheme $\mathbf{a}\text{-auth}$ in Sect. 4.

Basically, we can employ any three building blocks that satisfy the requirements stated in Sect. 4. We here briefly mention an instantiation in the setting of bilinear groups. The three building blocks are the pairing version of the Camenisch-Lysyanskaya digital signature scheme Sig^{CL} by Sudarsono-Nakanishi-Funabiki [14] and Teranishi-Furukawa [15], the pairing version of the Camenisch-Lysyanskaya perfectly witness-indistinguishable argument of knowledge system Π^{CL} by [14, 15], and the Pedersen-Okamoto commit-and-prove scheme $\text{CmtPrv}^{\text{PO}}$ [12, 13].

As for implementation, we expect a similar result to the result found in [14] because the execution of the Pedersen-Okamoto commit-and-prove is fast. When the number of authorities involved in our authentication is 3, the expected times for proof-generation and verification are both under 0.5 seconds except the communication time. (See Sect. 5.2 of [14] “the total number of string attribute types”.)

6 Conclusion

We proposed a generic construction of a Σ -protocol of commit-and-prove type, which is an AND-composition of Σ -protocols on the statements that include a common commitment. When the component Σ -protocols are of witness-indistinguishable argument systems, our Σ -protocol is also a witness-indistinguishable argument system as a whole. As an application, we gave a generic construction of a decentralized multi-authority anonymous authentication scheme. There a witness is a bundle of witnesses each of which decomposes into a fixed global identity string and a digital signature on it. We mentioned an instantiation of the scheme in the setting of bilinear groups. A post-quantum instantiation should be our future work.

References

1. Babai, L.: Trading group theory for randomness. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing, 6–8 May 1985, Providence, Rhode Island, USA, pp. 421–429 (1985). <https://doi.org/10.1145/22145.22192>
2. Bellare, M., Palacio, A.: GQ and schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_11
3. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, 19–21 May 2002, Montréal, Québec, Canada, pp. 494–503 (2002). <https://doi.org/10.1145/509907.509980>
4. Cramer, R.: Modular designs of secure, yet practical cryptographic protocols. Ph.D. thesis, University of Amsterdam, Amsterdam, The Netherlands (1996)
5. Damgård, I.: On σ -protocols. Course Notes (2010). <http://cs.au.dk/ivan/CPT.html>
6. Escala, A., Groth, J.: Fine-tuning groth-sahai proofs. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 630–649. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_36
7. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 13–17 May 1990, Baltimore, Maryland, USA, pp. 416–426 (1990). <https://doi.org/10.1145/100216.100272>
8. 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
9. Goldreich, O.: The Foundations of Cryptography: Basic Techniques, vol. 1. Cambridge University Press, Cambridge (2001)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, STOC 1985, pp. 291–304. ACM, New York (1985). <https://doi.org/10.1145/22145.22178>
11. Okamoto, T., Takashima, K.: Decentralized attribute-based signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 125–142. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_9
12. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_3
13. 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
14. Sudarsono, A., Nakanishi, T., Funabiki, N.: A pairing-based anonymous credential system with efficient attribute proofs. JIP **20**(3), 774–784 (2012). <https://doi.org/10.2197/ipsjip.20.774>
15. Teranishi, I., Furukawa, J.: Anonymous credential with attributes certification after registration. IEICE Trans. **95-A**(1), 125–137 (2012). http://search.ieice.org/bin/summary.php?id=e95-a_1_125

Full Paper Session XI: Supporting Techniques



Automatic Identification of Industrial Control Network Protocol Field Boundary Using Memory Propagation Tree

Chen Kai¹(✉), Zhang Ning², Wang Liming¹, and Xu Zhen¹

¹ State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
E-park C1, No. 65 Xingshikou Road, Haidian District, Beijing 100195, China
chenk@iie.ac.cn

² State Grid Jibei Electric Company, Hebei, China

Abstract. The knowledge of protocol specification, especially protocol field boundary, is invaluable for addressing many security problems, such as intrusion detection. But many industrial control network (ICN) protocols are closed. Closed protocol reverse engineering has often been a time-consuming, tedious and error-prone process. Some solutions have recently been proposed to allow for automatic protocol reverse engineering. But their prerequisites, e.g. assuming the existence of keywords or delimiters in protocol messages, limit the scope of their efforts to parse ICN protocol messages. In this paper, we present AutoBoundary that aims at automatically identifying field boundaries in an ICN protocol message. By instrumenting and monitoring program execution, AutoBoundary can obtain the execution context information, and build a memory propagation (MP) tree for each message byte. Based on the similarity between MP trees, AutoBoundary can identify protocol field boundaries, automatically. The intuition behind AutoBoundary makes it suitable for ICN protocols, which have the characteristics of no delimiter, no keyword, and no complex hierarchical structure in the message. We have implemented a prototype of AutoBoundary and evaluated it with 62 ICN protocol messages from 4 real-world ICN protocols. Our experimental results show that, for the ICN protocols whose fields are byte-aligned, AutoBoundary can identify field boundaries with high accuracy (100% for Modbus/TCP, 100% for Siemens S7, and 94.7% for ISO 9506).

Keywords: Industrial control network
Protocol field boundary identification · Memory propagation tree

1 Introduction

For industrial control network (ICN), the knowledge of application-level protocol specifications is invaluable for addressing many security problems. Protocol specifications are often required for intrusion detection and firewall systems to

perform deep packet inspection [3, 17]. For ICN management, protocol specifications can be used to identify protocols and analyze network traffic. They also allow the automatic generation of protocol fuzzers when performing the black-box testing [16]. Of course, many ICN protocol specifications can be obtained from authority documents directly, such as Modbus. But many ICN protocols are unknown, undocumented or proprietary, such as Siemens S7. For the closed protocol, protocol reverse engineered manually always means time-consuming and error-prone. In this case, the specification could only be specified through automatic protocol reverse engineering.

A key work of protocol reverse engineering is to identify the field boundaries. Some solutions have recently been proposed to allow for automatic protocol reverse engineering, including identifying field boundaries. Most of them are effective and efficient, when the prerequisites are met. The prerequisites include: (1) the existence of keywords or delimiters in protocol messages; (2) utilizing loops and comparison operations to parse protocol messages within the software binary; (3) getting key information ahead of time, e.g., IP address or host name, and so on. On the other hand, the syntactic structures of many ICN protocols have the characteristics of no delimiter, no keyword, and no hierarchical structure, such as Modbus. In this paper, we present AutoBoundary, a new approach that aims at automatically identifying field boundaries in an ICN protocol message. AutoBoundary is based on the key observation that *bytes belonging to the same protocol field of a message have the same propagation traces in the memory, due to they are typically handled together*. The intuition behind AutoBoundary does not depend on delimiter, keyword, or hierarchical structure. So it is more suitable for ICN protocols. By dynamically analyzing program execution, we record the address for a message byte once it propagates from one place to another. At last, all address records of a message byte compose a memory propagation tree. A n -byte message results n memory propagation trees. Through comparing between memory propagation trees, we can decide whether two message bytes belong to the same protocol field or not. Further, based on the similarity between memory propagation trees, we can identify the field boundaries of a protocol message. We have implemented a proof-of-concept prototype and evaluated it with 62 ICN protocol message from 4 ICN protocols.

The contributions of this paper are the following: (1) We present a novel approach to analyze the movement trace of message bytes in the memory. We use memory propagation (MP) tree as storage structure to record all movement traces of a message byte, and describe the detail way how to compare between MP trees. The comparison result embodies the similarity between MP trees. (2) We present AutoBoundary, an MP-tree-based approach to identify the field boundaries in an ICS protocol message. (3) We applied our techniques to a set of real-world applications that implement ICN protocols such as Modbus/TCP, IEC 60870-5-104, ISO 9506, and Siemens S7. Our results show that AutoBoundary can automatically identify field boundaries.

2 Goal and Assumption

There are three essential components in an application-level protocol specification: protocol syntax, protocol FSM and protocol semantics. Inferring protocol syntax, including identifying field boundary, lays the foundation for automatic protocol reverse engineering. In this paper, we focus on identifying the field boundaries. Our goal is to design an algorithm that, given one message of an ICN protocol and an application that can process this message, recovers the boundaries of fields.

As mentioned above, we assume that the application, which can parse the protocol message, could be obtained. Though these appliances that run on very “special” hardware seem to be hardly obtained, it is not always an unsolvable problem. We believe that some cutting-edge technologies (e.g., softPLC, virtualization, and digital twins) would make it possible.

3 System Design

The intuition behind AutoBoundary is simple but effective: “Bytes belonging to the same protocol field of a message have the same propagation traces in the memory, due to they are typically handled together.” As such, the propagation traces of each message byte can be compared to uncover field boundaries. AutoBoundary is interested in how memory propagation information can be collected and analyzed to identify field boundaries. It has three processing stages: (1) execution monitor, (2) MP tree generation, and (3) field boundary identification.

3.1 Stage 1: Execution Monitor

In the execution monitor stage, an ICN protocol message is sent to an application that “understands” the protocol that we are interested in, such as a server program implementing a particular ICN protocol. By monitoring application execution, we can intercept the network-related system calls (e.g., `sys_socket`), and mark the message received as tainted data. Moreover, throughout the message processing life-time, we instrument all instructions that operate on the tainted data to record propagation traces. More specially, for a data movement instruction, we check whether the source operand is tainted. If yes, we will mark the destination operand, which can be a register or a memory location, as tainted data; If no, we will simply unmark the destination operand. At the same time, we record the instruction address and the addresses of both the source operand and destination operand, with the format: “ $addr_{ins} : addr_{src} \rightarrow addr_{dst}$ ”. If an instruction has two source operands, we will union of their marks, and record with the format: “ $addr_{ins} : addr_{src1} + addr_{src2} \rightarrow addr_{dst}$ ”. Similar to previous systems that use dynamic taint analysis, we establish a relationship between a particular message byte and a location in memory (or a register). We reference interested readers to related literature such as [4, 10, 18].

As a result of the monitoring process, memory propagation records are produced for each ICN protocol message. They contain all operations that have one tainted operands at least.

3.2 Stage 2: Memory Propagation Tree Generation

To organize memory propagation records efficiently and make it possible for mathematical calculation, we define a new structure and name it as memory propagation tree, which is described as Definition 1. Each byte from an ICN protocol message has one, and only one, corresponding MP tree. In the rest of this section, we discuss the way how to build MP trees from memory propagation records, calculate branch contribution for an MP tree, and compress an MP tree, within stage 2.

Definition 1. *A Memory Propagation (MP) tree is a data structure made up of nodes and edges without having any cycle. An MP tree consists of a root node and potentially many levels of additional nodes that form a hierarchy. The root node represents the initial memory location of a byte from an ICN protocol message. The intermediate and leaf nodes represent the locations where this byte has appeared during its propagation process. An edge means propagating from a location (i.e., parent node) to another (i.e., child node).*

MP Tree Construction. By monitoring the network-related system calls, the buffer that contains the received protocol message is determined. For each byte in the buffer, AutoBoundary creates a blank MP tree, adds a root node into the MP tree, and sets two properties on the root node. One is location property that includes the byte address. The other is growth property that indicates whether a new branch can grow from this node. Then, as shown in Algorithm 1, AutoBoundary repeatedly reads a record from the memory propagation records generated in Sect. 3.1, and tries to insert a new child node and a new edge that is from $record.addr_{src}$ to $record.addr_{dst}$ into the MP tree mpt_i , by using the function $MPtree-InsertEdge()$. If there is an existing node $node_i$, whose location property value is equal with $record.addr_{src}$ and growth property value is “enable”, the node and edge can be insert into mpt_i . In this case, AutoBoundary creates a new node $node'$, sets its location property value to $record.addr_{dst}$, and sets its growth property values to “enable”. A new edge that is from $node_i$ to $node'$ is inserted into mpt_i . On the other hand, if there is no such node, AutoBoundary checks whether this memory propagation record would have impact on mpt_i . If $record.addr_{dst}$ is the same with any nodes' location property value, AutoBoundary modifies the growth property values of these nodes to “disable”. The value “disable” means that a new branch cannot grow from such nodes, namely, they cannot be a parent of a new node. It's important to note that, in the real-world application, after a tainted memory or register is overwritten by other non-tainted data, we should no longer keep tracks of its propagation. The growth property attached to the node helps us to decide whether to keep tracks of the node's propagation in the future.

Algorithm 1. MPtree-Gen

Input. $[buf]$: the buffer contains the protocol message
 $[records]$: the memory propagation records generated in Sect. 3.1

Output. $[mpt_1, mpt_2, \dots, mpt_n]$: MP trees, n means the length of buf

- 1: **for** each $byte_i$ in buf **do**
- 2: create a blank MP tree mpt_i for $byte_i$
- 3: create root node $rootNode_i$
- 4: set $rootNode_i.locationProperty = getAddress(byte_i)$
- 5: set $rootNode_i.growthProperty = enable$
- 6: insert $rootNode_i$ into mpt_i
- 7: **for** read a $record$ from $records$ **do**
- 8: MPtree-InsertEdge($record.addr_{src}, record.addr_{dst}, mpt_i$)
- 9: **end for**
- 10: **end for**
- 11: Return $(mpt_1, mpt_2, \dots, mpt_n)$;

Branch Contribution Calculation. The node that has no child is a leaf node. A branch consists of a root node, a leaf node, and all of nodes and edges between them. The branch length is defined in Definition 2.

Definition 2. *The length of a branch is the number of nodes, which belong to the branch. Term “longer” has the same meaning as “with more nodes”.*

Definition 3. *Branch contribution is used to quantify weigh values of branches with different length. It embodies the contribution degree of a branch during the process of identifying field boundaries. It is a decimal number between 0 and 1. 0 means that the branch has no contribution to filed identification procedure. In contrast, being close to 1 means that the branch has much impact on filed identification procedure.*

For an MP tree, are the weight values of branches with different length the same? Of course not. The longer a branch is, the wider a byte propagates. A wider propagation always means that the byte has been processed by more instructions, while each instruction can partially reflect characteristics of the byte. Therefore, a longer branch usually provides more help to identify field boundaries. To quantify weight values for branches with different length, we introduce the definition of branch contribution as Definition 3. For the i th branch, its contribution $cont_i$ can be calculated by (1), where n is the total number of branches, and the function $len()$ gets the length of a branch. The contribution degree is attached to every branch as an additional property.

$$cont_i = \frac{len(branch_i)}{\sum_{j=1}^n len(branch_j)}, 1 \leq i \leq n \quad (1)$$

MP Tree Compression. If the length of two branches are equal, and the nodes residing in the same level have the same properties (i.e., location and

growth), we say that these two branches are the same. AutoBoundary does not forbid establishing multiple same branches during the process of MP tree generation. The redundancy branches reduce the efficiency of compare operation in the next stage. To mitigate the problem, AutoBoundary compresses the MP tree by merging the redundancy branches. The compressing approach used by AutoBoundary is simple and easy, but efficient and effective. For the redundancy branches, first rip out all but one, and then add the contribution of discarded branches into the remaining one.

3.3 Stage 3: Field Boundary Identification

Our field boundary identification relies on clustering message bytes. To cluster bytes, we need to invoke both branch comparison and tree similarity calculation. In this section, we first explain these procedures before describing how to divide clusters.

Branch Comparison. To find similar branches across different MP trees, we need a proper approach to compare branches. As Sect. 3.2 describes, each branch consists of many nodes. Therefore, to compare two branches, we align their nodes by using a customized version of sequence alignment algorithm. And the score gotten by aligned nodes represents the result of branch comparison.

We refer to our approach for aligning nodes as *node-based sequence alignment algorithm*. The key observation behind our approach is that, while sequence alignment algorithm [14] cannot be used for comparing branches directly, it can be used to align nodes by leveraging the node’s properties (i.e., location and growth property) generated in the MP tree construction phase. In the node-based sequence alignment algorithm, we claim two aligned nodes are matched if they have the same growth property and the distance between their location is smaller than the size of a variable of type char (i.e., 1 byte). For instance, knowing a growth-enabled node N of a branch is placed in a particular location necessitates that its counterpart N' of another branch is also growth-enabled and next to N for these two nodes to be considered a match. We allow gaps in the node-based sequence alignment algorithm. In addition to using gap penalties to control gaps, we introduce extra constraints to make it more suitable for branch comparison. First, a node placed in registers is allowed to align with gaps. This constrain is for handling the case of coalescing multiple registers (i.e., EAX and EDX) to perform one computation. Second, a node placed in memory is allowed to align with gaps, but it must be imposed heavy penalty – the gap penalty is typically double. This constraint is for handling the case that string functions are used to parse fields. For example, when using the function “strncmp” to parse fields, the front part of the field may be handled more times than the rear part, which results in longer branches for the front part.

After aligning nodes, the node-based sequence alignment algorithm outputs a score for a pair of branches. This score quantifies the result of branch comparison. Since AutoBoundary does not focus on the absolute value of the score, it is insensitive to the scoring system (sub-scores for match, mismatch and gap) used by sequence alignment algorithm.

MP Tree Similarity Calculation. Let two target MP tree as mpt_1 and mpt_2 . To calculate the similarity between them, we need to find the matched branches and accumulate contribution degrees for each tree, respectively.

Definition 4. For a given branch from an MP tree, comparing it with all branches from another MP tree (by using branch comparison approach), the corresponding branch is defined as which one outscores others.

Definition 5. For a given branch and its corresponding branch, if the score for branch comparison exceeds a certain threshold, we claim they are matched.

$$\begin{aligned} threshold &= \overline{len} \times fac \times sScore_{ma} + \overline{len} \times (1 - fac) \times sScore_{mi}, \\ \overline{len} &= \frac{len(branch) + len(branch')}{2} \end{aligned} \quad (2)$$

Firstly, we deal with mpt_1 . Travel every branch br_i from mpt_1 , we search its corresponding branch br'_i in mpt_2 . We give the definition of corresponding branch in Definition 4. And then, check whether the branch br_i and its corresponding branch br'_i are matched. The definition of matched branch is given in Definition 5. As what described in it, a threshold is required to decide whether they are matched. This threshold can be obtained by (2), where \overline{len} is the mean value of the branch length, $sScore_{ma}$ and $sScore_{mi}$ are the sub-scores for matched and mismatched nodes respectively, and fac is an adjustment factor whose range is from 0.5 to 1 – the value of fac means the least percentage of matched nodes (for Modbus/TCP, IEC 60870-5-104, ISO 9506 and S7, we suggest setting fac to 0.75). After that, if br_i and br'_i are matched, accumulate the contribution degree for mpt_1 . Algorithm 2 describes this process in detail. It traverses all branches in mpt_1 , to find the corresponding branch in mpt_2 . If the comparison score between branch br_i and its corresponding branch br'_i is larger than the threshold, namely branches are matched, accumulate the contribution degree of br_i into the total contribution degree $matchContribution$. At the end, the accumulated contribution degree of mpt_1 is obtained.

Secondly, dispose mpt_2 with the same procedure as mpt_1 , but reverse roles of two MP trees. Travel all branches of mpt_2 , search the corresponding branch in mpt_1 , and accumulate the contribution degree for mpt_2 .

Algorithm 2. AutoBoundary-AccumulateContribution

Input. [mpt_1]: the first MP tree; [mpt_2]: the second MP tree

Output. [$matchContribution$]: accumulated contribution degrees for mpt_1

- 1: **for** each branch br_i in mpt_1 **do**
 - 2: $(br'_i, score) = \text{AutoBoundary-FindCorrespondingBranch}(br_i, mpt_2)$
 - 3: **if** $score > \text{getThreshold}(br_i, br'_i)$ **then**
 - 4: $matchContribution += \text{getContribution}(br_i)$
 - 5: **end if**
 - 6: **end for**
 - 7: **Return** ($matchContribution$);
-

Algorithm 3. AutoBoundary-CalculateTreeSimilarity

Input. $[mpt_1]$: the first MP tree; $[mpt_2]$: the second MP tree**Output.** $[similarity]$: the similarity between input MP trees

- 1: $matchCont_1 = \text{AutoBoundary-AccumulateContribution}(mpt_1, mpt_2)$
 - 2: $matchCont_2 = \text{AutoBoundary-AccumulateContribution}(mpt_2, mpt_1)$
 - 3: $similarity = (matchCont_1 \times \text{getLength}(mpt_1) + matchCont_2 \times \text{getLength}(mpt_2))$
 - 4: $similarity = similarity / (\text{getLength}(mpt_1) + \text{getLength}(mpt_2))$
 - 5: Return ($similarity$);
-

At last, calculate the similarity between two MP trees, based on their contribution degrees. As Algorithm 3 described, we obtain the result similarity through merging two accumulated contribution degrees.

Message Byte Clustering. Message byte clustering is an iterative process, from the first message byte to the last one. As shown in Algorithm 4, each iteration handles two adjacent message bytes. AutoBoundary finds MP trees for these two bytes, and then calculates the similarity between MP trees through the approach described above.

Algorithm 4. AutoBoundary-ClusterByte

Input. $[message]$: the ICN protocol message, including n bytes $[mpt_1, mpt_2, \dots, mpt_n]$: MP trees, one tree mapping one byte**Output.** $[cluster]$: the result clusters

- 1: **for** i from 1 to n **do**
 - 2: $mpt_i = \text{getMPTree}(byte_i)$ // $byte_i$ means i th byte in $message$
 - 3: $mpt_{i+1} = \text{getMPTree}(byte_{i+1})$
 - 4: $similarity_i = \text{AutoBoundary-CalculateTreeSimilarity}(mpt_i, mpt_{i+1})$
 - 5: **end for**
 - 6: **for** i from 1 to $n - 1$ **do**
 - 7: $k = \text{sizeof}(\text{short})$
 - 8: $pre = (1 > i - k/2) ? 1 : (i - k/2)$
 - 9: $post = (i + k/2 > n - 1) ? (n - 1) : (i + k/2)$
 - 10: **for** j from pre to $post$ **do**
 - 11: $neighbourMean += similarity_j$
 - 12: **end for**
 - 13: $neighbourMean = neighbourMean / (post - pre + 1)$
 - 14: // check whether $similarity_i$ is larger than the mean value of k neighbors
 - 15: **if** $similarity_i \geq neighbourMean$ **then**
 - 16: divide $byte_i$ and $byte_{i+1}$ into a cluster
 - 17: **end if**
 - 18: **end for**
 - 19: Return (clusters);
-

After that, $n - 1$ similarities are obtained. According to these similarities, AutoBoundary divides message bytes into clusters. To evaluate whether a similarity is high enough, we need a reference value. Therefore, for each one out of $n - 1$ similarities, AutoBoundary finds its k neighbors, and calculates the mean value. If a similarity is larger than its k -neighbor mean value, two message bytes related to the similarity are divided into a cluster. At last, every message byte is covered by one and only one cluster. While a cluster is identified as a protocol field, field boundaries is set between clusters.

4 Evaluation

We have implemented an AutoBoundary prototype in 20,500 lines of source code on Linux 3.16 (Debian 8.5.0). The execution monitor module extends the instrumentation tool Pin [12] (version 2.14-71313). However, we note that our design is not tightly coupled with Pin, and can be implemented using other instrumentation tools, e.g., Valgrind [15]. The MP tree generation module takes a memory propagation record file that is the outcome of execution monitor module as input, and outputs MP trees. Based on MP trees, the field boundary identification module infers message formats.

We will present two sets of experiments. The first set of experiments involves 20 kinds of prototype messages from 3 known ICN protocols, including *Modbus/TCP*, *IEC 60870-5-104* and *ISO 9506*. The second set of experiments involves 10 protocol messages in a closed ICN protocol used by Siemens PLCs, namely *S7*. These messages are either for conveying commands from the engineer station or for retrieving I/O data from the controller.

In the first set of experiments with known ICN protocols, we can quantitatively evaluate the effectiveness of AutoBoundary. We compare our results with the results from a popular network protocol analyzer – Wireshark. We present the set of message fields as F , and the number of F as $|F|$. We count $|F|$ in both Wireshark and AutoBoundary results. We also count the number of fields in protocol specifications, which will be taken as benchmarking. Because both Wireshark and AutoBoundary may consolidate multiple protocol fields as one coarse-grained field, we count the total number of coarse-grained fields as $|E_c|$. On the other hand, they may divide a protocol field into multiple overly-fine-grained fields. We count the number of overly-fine-grained fields as $|E_o|$. Table 1 reports the results. In the following, we describe our experiments in greater detail.

4.1 Modbus TCP Request

In this experiment, we monitor the execution of a Modbus/TCP server implemented by libmodbus v3.0.6, and trace 20 Modbus/TCP messages (8 types). The results in Table 1 show that AutoBoundary identifies all protocol fields, as there is one overly-fine-grained field discovered by Wireshark. For the error ratio, AutoBoundary performs better. As a detailed example, for the “Write Single Coil” sub-messages, Wireshark reports $|E_c| = 0$ and $|E_o| = 1$ while AutoBoundary

Table 1. Protocol field comparison between Wireshark and AutoBoundary

Protocol	Message type	Specification	Wireshark			AutoBoundary		
			$ F $	$ E_c $	$ E_o $	$ F $	$ E_c $	$ E_o $
Modbus/TCP	Write Single Coil	7	8	0	1	7	0	0
	Write Multiple Coils	9	9	0	0	9	0	0
	Write Single Register	7	7	0	0	7	0	0
	Write Multiple Registers	9	9	0	0	9	0	0
	Read Coils	7	7	0	0	7	0	0
	Read Discrete Inputs	7	7	0	0	7	0	0
	Read Holding Registers	7	7	0	0	7	0	0
	Read Input Registers	7	7	0	0	7	0	0
IEC 60870-5-104	U format STARTDT	5	4	1	0	4	1	0
	U format STOPDT	5	4	1	0	4	1	0
	I format Interrogation	16	15	1	0	11	5	0
	I format C_SC_NA_1	19	17	2	0	12	7	0
	I format C_DC_NA_1	18	17	1	0	11	7	0
	I format C_SE_NB_1	22	17	5	0	12	10	0
	I format C_RD_NA_1	15	14	1	0	10	5	0
	I format C_CS_NA_1	28	23	5	0	13	15	0
ISO 9506	getNameList	16	4	12	0	11	4	0
	Read	23	9	14	0	23	0	0
	Write	26	11	15	0	28	0	2
	defineNamedVariableList	29	12	17	0	31	0	2

shows $|E_c| = 0$ and $|E_o| = 0$, comparing with the Modbus/TCP specification. The reason for having an overly-fine-grained field in Wireshark result is that the low byte of “output value” field is erroneously identified as a padding of Ethernet frame. And if a Modbus/TCP server does not use the default port 502 (e.g., the example program of libmodbus uses 1502 as the default port), Wireshark cannot identify any Modbus/TCP fields. AutoBoundary works well no matter which port number is used by the Modbus/TCP server. Therefore we believe that AutoBoundary outperforms Wireshark, when confronting Modbus/TCP.

4.2 IEC 60870-5-104 Request

In this experiment, we monitor the execution of an IEC 60870-5-104 server implemented by lib60870 v0.9.4, and trace 20 messages (8 types) in control direction. Table 1 shows the existence of coarse-grained fields both in the Wireshark and AutoBoundary results. More specifically, Wireshark identifies 86.7% of protocol fields, while AutoBoundary only discovers 60.1% of them. To find out the root cause, we make in-deep analysis against “U format – STARTDT” sub-messages and “I format – Interrogation” sub-messages.

For the “U format – STARTDT” sub-messages, Wireshark reports $|E_c| = 1$ and $|E_o| = 0$ while AutoBoundary shows $|E_c| = 1$ and $|E_o| = 0$. According to IEC 60870-5-104 specification, a “STARTDT” message has 1 start field, 1 length field and 4 control fields. The first control field is divided into two parts: bit 1–2 are always 1 (format type), bit 3–8 represents one function (TESTFR, STOPDT or STARTDT). Therefore, the first control field should be treated as 2 different fields. Because the last 3 control fields are always 0, they are combined as 1 field. Wireshark does not identify the last field, i.e. 3-byte 0x00. AutoBoundary treats the first control field as 1 field. It does not discover the format type field (bit 1–2 of the first control field) and function field (bit 3–8 of the first control field).

For the “I format – Interrogation” sub-messages, Wireshark reports $|E_c| = 1$ and $|E_o| = 0$ while AutoBoundary shows $|E_c| = 5$ and $|E_o| = 0$. An “Interrogation” message has 16 protocol fields, including start field, length field, two format type fields (bit 1 of CFO¹ 1 and bit 1 of CFO 3), send sequence number field, receive sequence number field, type identification field, SQ field (bit 8 of VSQ²), number field, test flag field (bit 8 of COT³), P/N field (bit 7 of COT), cause field, originator address field, common address field, information object address field and qualifier of interrogation field. Wireshark only discovers one format type field, while AutoBoundary does not identify SQ, test flag, P/N and format type fields.

There is a main reason behind the coarse-grained field: for AutoBoundary, the granularity of dynamic taint analysis is byte but not bit. So it cannot identify field boundaries which are not byte-aligned, such as function field in the “STARTDT” message, SQ field in the “Interrogation” message and so on. It is easy to modify the granularity of dynamic taint analysis from byte to bit, but the resource consumption will be increasing exponentially. To balance out the costs and benefits, we keep the byte-granularity.

4.3 ISO 9506 Request

In this experiment, we monitor the execution of an ISO 9506 (MMS) server implemented by libiec61850 v1.0.1, and trace 12 messages (4 types) in control direction. Table 1 shows that, there are coarse-grained fields both in the Wireshark results and AutoBoundary results. Wireshark only identifies 39.3% of protocol fields. ISO 9506 (MMS) uses Abstract Syntax Notation One (ASN.1) to encode request PDUs. The Basic Encoding Rules (BER) of ASN.1 has three parts: identifier, length and content. Wireshark only identifies the content fields. This is the root cause for the poor result. On the other hand, AutoBoundary discovers 94.7% of protocol fields. As a detailed example, for “getNameList” sub-messages, AutoBoundary reports $|F| = 11$ and $|E_c| = 4$. Through static analysis against the source code, we find that the implementation of the protocol ignores specific messages fields. So these fields cannot be inferred by AutoBoundary.

¹ Control Field Octet.

² Variable Structure Qualifier.

³ Cause Of Transmission.

For “write” sub-messages, AutoBoundary reports $|F| = 28$ and $|E_o| = 2$. The reason behind overly-fine-grained fields is that the “itemID” field consists of multiple parts – a logical node name “LLN0”, a functional constraint “ST”, and a data name “Health”. The implementation of the protocol parses them respectively. Therefore, AutoBoundary regards one “itemID” field as three different fields. The same thing happens to “defineNamedVariableList” sub-messages.

Table 2. Protocol field comparison between S7 Wireshark dissector and AutoBoundary

Protocol	Message function code	S7 Wireshark dissector	AutoBoundary		
		$ F $	$ F $	$ E_c $	$ E_o $
Siemens S7 Protocol	Setup communication	11	11	0	0
	Upload	16	16	0	0
	PLC Stop	10	10	0	0
	Write Variable	20	20	0	0
	(Multiple) Read Variable	48	48	0	0

4.4 Siemens S7 Messages

We present our second set of experiments showing that AutoBoundary can uncover the field boundaries of a closed ICN protocol (S7) message used by Siemens PLCs. To verify the AutoBoundary results, we use a great project – S7 Wireshark dissector, and compare results between AutoBoundary and S7 Wireshark dissector.

We monitor the execution of an S7 server implemented by Snap7, and trace 10 messages (5 types) in control direction. As shown in Table 2, for each field boundary identified by S7 Wireshark dissector, there is an identical field boundary automatically discovered by AutoBoundary.

5 Limitations and Future Work

The first limitation of AutoBoundary is the granularity of dynamic taint analysis. To balance out the costs and benefits, we choose 1-byte as the minimum unit when tracing taint data. But some ICN protocol fields are not byte-aligned, such as IEC 60870-5-104 requests mentioned in Sect. 4.2. In other words, if a protocol field is not byte-aligned, AutoBoundary cannot infer the field boundary accurately. Secondly, AutoBoundary is the dynamic trace dependency. If the implementation of an ICN protocol ignores some message fields, AutoBoundary cannot discover the boundaries of these fields, just like what happened for ISO 9506 requests mentioned in Sect. 4.3. Fixing the above problems is a part of our future work. And we plan to extend execution monitoring process to the protocol client (e.g., HMI application), which is easier to be obtained. In addition, identifying entire structure of a message is another part of our future work.

6 Related Works

The methods of automatic protocol reverse engineering can be classified into network trace analysis and dynamic analysis approaches.

The network trace analysis approaches for protocol reverse engineering take as input a network capture and use clustering techniques to determine protocol information. *Protocol Informatics* project [2] aims to employ Smith Waterman algorithm to infer protocol formats from a set of protocol network packets. *Discoverer* [7] leverages recursive clustering and type-based sequence alignment to infer message formats. *RolePalyer* [8] can mimic both the server side and the client side of the session for application protocols. *Biprominer* [19] and *ProDecoder* [20] are two automatic protocol reverse engineering tools, which use statistical methods to find keywords and probable keyword sequences. *AutoReEngine* [13] adopts a similar method but measuring keyword location from the beginning of a message as well as the end. *ReverX* [1] uses a speech recognition algorithm to identify delimiters, and then finds keywords within protocol messages by identifying the frequency of byte sequences.

The dynamic analysis approaches monitor the execution of a software binary that implements the communication protocol to identify the protocol message fields. *Polyglot* [6] depends on the existence of loops in which tainted data is iteratively compared to a constant value. *Dispatcher* [5] targets transmitted messages. To extract the message format of sent messages, it leverages the intuition that the structure of the output buffer represents the inverse of the structure of the sent message. *AutoFormat* [11] treats consecutive bytes that are run in the same execution context as message fields, and then exposes a tree of hierarchal fields. *Tupni* [9] can reverse engineer an input format with a rich set of information. Its key property is that it identifies arbitrary record sequences by analyzing loops in a program. *Wondracek et al.* [21] presented a approach to extract information about the fields of individual messages, and aggregate this information to determine a general specification of the message format.

7 Conclusion

We have proposed MP tree to analyze the movement trace of message bytes, and described the way how to build, compress, and compare MP trees. Based on MP tree, we have presented AutoBoundary, a system for automatic protocol field boundary identification. We have implemented a prototype of AutoBoundary and evaluated it with a variety of ICN protocol messages. Our experimental results show that AutoBoundary achieves high accuracy in ICN protocol field boundary identification.

Acknowledgments. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0801900). We would like to thank all anonymous reviewers for helping us make this paper better.

References

1. Antunes, J., Neves, N., Verissimo, P.: Reverse engineering of protocols from network traces. In: Working Conference on Reverse Engineering, WCRE 2011, pp. 169–178. IEEE, Limerick (2011)
2. Beddoe, M.A.: Network protocol analysis using bioinformatics algorithms (2012). <http://www.4tphi.net/~awalters/PI/pi.pdf>. Accessed 31 Aug 2016
3. Borisov, N., Brumley, D.J., Wang, H.J., Dunagan, J., Joshi, P., Guo, C.: A generic application-level protocol analyzer and its language. In: 14th Symposium on Network and Distributed System Security, NDSS 2014. The Internet Society, San Diego, February 2007
4. Brumley, D., Caballero, J., Liang, Z., Newsome, J., Song, D.: Towards automatic discovery of deviations in binary implementations with applications to error detection and fingerprint generation. In: 16th USENIX Security Symposium, pp. 213–228. USENIX Association, Boston, August 2007
5. Caballero, J., Poosankam, P., Kreibich, C., Song, D.: Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering. In: ACM Conference on Computer and Communications Security, CCS 2009, pp. 621–634. ACM, Chicago (2009)
6. Caballero, J., Yin, H., Liang, Z., Song, D.: Polyglot: automatic extraction of protocol message format using dynamic binary analysis. In: ACM Conference on Computer and Communications Security, CCS 2007, pp. 317–329. ACM, Alexandria (2007)
7. Cui, W., Kannan, J., Wang, H.J.: Discoverer: automatic protocol reverse engineering from network traces. In: 16th USENIX Security Symposium, pp. 199–212. USENIX Association, Berkeley, August 2007
8. Cui, W., Paxson, V., Weaver, N.C., Katz, R.H.: Protocol-independent adaptive replay of application dialog. In: Network and Distributed System Security Symposium, NDSS 2006, pp. 487–490. The Internet Society, San Diego (2006)
9. Cui, W., Peinado, M., Chen, K., Wang, H.J., Irun-Briz, L.: Tupni: automatic reverse engineering of input formats. In: ACM Conference on Computer and Communications Security, CCS 2008, pp. 391–402. ACM, Alexandria (2008)
10. Egele, M., Kruegel, C., Kirda, E., Yin, H., Song, D.: Dynamic spyware analysis. In: 2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference, ATC 2007, pp. 18:1–18:14. USENIX Association, Santa Clara (2007). <http://dl.acm.org/citation.cfm?id=1364385.1364403>
11. Lin, Z., Jiang, X., Xu, D., Zhang, X.: Automatic protocol format reverse engineering through context-aware monitored execution. In: Network and Distributed System Security Symposium, NDSS 2008. The Internet Society, San Diego, February 2008
12. Luk, C.K., et al.: Pin: building customized program analysis tools with dynamic instrumentation. In: ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 190–200. ACM, Chicago, June 2005
13. Luo, J.Z., Yu, S.Z.: Position-based automatic reverse engineering of network protocols. *J. Netw. Comput. Appl.* **36**(3), 1070–1077 (2013)
14. Needleman, S.B., Wunsch, C.D.: A general method applicable to search for similarities in amino acid sequence of 2 proteins. *J. Mol. Biol.* **48**(3), 443–453 (1970)
15. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. *ACM SIGPLAN Not.* **42**(6), 89–100 (2007)

16. Oehlert, P.: Violating assumptions with fuzzing. *IEEE Secur. Priv.* **3**(2), 58–62 (2005)
17. Pang, R., Paxson, V., Sommer, R., Peterson, L.: Binpac: a yacc for writing application protocol parsers. In: *ACM SIGCOMM Conference on Internet Measurement*, pp. 289–300. ACM, Rio De Janeiro, October 2006
18. Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Krgel, C., Vigna, G.: Cross site scripting prevention with dynamic data tainting and static analysis. In: *Network and Distributed System Security Symposium, NDSS 2007*. The Internet Society, San Diego, February–March 2007
19. Wang, Y., Li, X., Meng, J., Zhao, Y., Zhang, Z., Guo, L.: Biprominer: automatic mining of binary protocol features. In: *12th International Conference on Parallel and Distributed Computing. Applications and Technologies, PDCAT 2011*, pp. 179–184. IEEE, Gwangju (2011)
20. Wang, Y., et al.: A semantics aware approach to automated reverse engineering unknown protocols. In: *20th IEEE International Conference on Network Protocols, ICNP 2012*, pp. 1–10. IEEE, Austin (2012)
21. Wondracek, G., Comparetti, P.M., Krgel, C., Kirda, E.: Automatic network protocol analysis. In: *Network and Distributed System Security Symposium, NDSS 2008*. The Internet Society, San Diego, February 2008



PCA: Page Correlation Aggregation for Memory Deduplication in Virtualized Environments

Min Zhu^{1,2}, Kun Zhang^{1,2(✉)}, and Bibo Tu^{1,2}

¹ Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhumin,zhangkun,tubibo}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

Abstract. To intelligently share limited memory across VMs in IaaS cloud, content-based page sharing (CBPS), like KSM, is utilized to greatly reduce the memory footprint of VMs. CBPS merges same-content pages into a single copy. However, it introduces some serious cross-VM covert channel threats. Besides, it has heavy overhead due to vast otiose operations, such as page comparisons and checksum calculations, when detecting page sharing opportunities. In this paper, we propose a novel memory deduplication approach called page correlation aggregation (PCA), which can efficiently reduce otiose operations. Meanwhile defends covert channels. One key idea of PCA is to divide VMs' pages into several sets, since pages with similar attributes have the greatest possibility with the same content. In PCA, the pages of VMs are firstly divided into different groups according to VMs' attributes. In each group pages are further separated into different classifications based on their access permissions. Thus page comparisons are restricted to the same classification for sharing. The other is that PCA introduces a dedicated cache to mitigate the latency of COW (Copy- On-Write) used for conducting covert channels. We have conducted a prototype on KSM, one popular CBPS technique. Our experimental results show that PCA reduces otiose operations about 40%, and can effectively resist covert channels.

Keywords: Covert channel · Secure memory deduplication
Page classification · KSM

1 Introduction

In IaaS (Infrastructure as a Service) clouds, the available memory size has become one of major bottlenecks to run more virtual machines (VMs) on a single machine. In this scenario, however, there is plenty of redundant data, resulting in lower utilization and higher hardware costs. To alleviate it, memory deduplication is proposed to detect and eliminate redundant memory pages.

A typical representative is content-based page sharing (CBPS) [2–4] that is transparently implemented in hypervisor layer, such as VMware vSphere, Xen and KVM etc. If multiple memory pages have the same content, CBPS only reserve a single copy for these so-called deduplicated pages in a copy-on-write (COW) way. When a deduplicated page is written, a new page is re-created with a copy. Many work [7, 13] has shown virtualized environments have a large amount of memory can be condensed, especially for VMs running similar applications or OSes.

However, CBPS introduces a new security threat: covert channels, which can extract and transmit data from co-located VMs exploiting additional access delays caused by COW for deduplicated pages. In this way, standard security measures, such as access control and data audit, are bypassed. Covert channels indeed pose realistic and serious threats to information security in the cloud [23–32]. For example, a hundred bytes of credit card can be secretly stolen less than 30s, even a thousand bytes of a file can be stealthily trafficked within 3 min [31]. However, covert channels cause negligible CPU and memory utilization, so it's hard to be detected in a cloud of heavy workloads. Moreover, it is robust against environmental noises than cache based covert channel [21, 22]. Therefore, covert channels become ideal choices for secret data transmissions.

CBPS also has performance interference [11, 12]. Since CBPS manages memory pages of VMs indiscriminately and globally, for each candidate page it needs to be compared with a large number of otiose pages repeatedly to detect its sharing opportunities, which induces plenty of otiose page comparisons and checksum calculations (so-called otiose operations), which take up the main run-time consumption of CBPS (over 60%). The induced CPU overhead will degrade the performance of VMs. As the number of comparisons increases, the CPU overhead increases correspondingly. And as the increasing capacity of mergeable memory, the comparisons expand proportionally, which makes the situation worse.

In this paper, we propose an efficient and secure memory deduplication approach called page correlation aggregation (PCA), which highly aggregates VM's pages based on correlative page properties. In PCA, VMs are divided into disparate groups based on their attributes (e.g. OS types), because inter-VM sharing is very low with different VM attributes. Further, according to page access permissions, pages of each group are divided into different classifications, since these pages have a higher possibility with the same content. PCA depends on VM introspection (VMI) to obtain VM attributes and access permission. Thus, a candidate page is only compared with pages in its classification and group, which reduces otiose operations. In addition, this feature prohibits covert channels if the receiver and sender are separated into different groups. Memory contents of one group are securely protected from another group. To prevent covert channels within the same group, an intuitive approach would be to add noise to obfuscate the specific covert-channel information. Based on this idea, PCA provides a dedicated cache to reserve deduplicated pages, so as to mitigate the write latency when occurring COW on deduplicated pages. By doing this, the receiver may consider the received bit as '1' due to low write latency, whereas the actual transmitted bit is '0'.

We have implemented a prototype in Kernel Samepage Merging (KSM), one widely used implementation of CBPS. PCA is guest-transparent due to the use of VMI. We evaluated it through several benchmarks. The results show that PCA is efficient and practical. Although our current implementation is performed on KSM, PCA can be applied to any CBPS instances due to its generality. Actually, PCA also works for deduplicating native processes.

Overall, we have made the following contributions:

- We propose a novel memory deduplication approach for covert channels defense and otiose operations reduction assisted by VMI.
- We employ the inter-VM grouping and intra-group classification mechanism to efficiently reduce otiose comparisons. Page sharing is performed just in the same classification of the same group.
- We provide a dedicated cache for writable deduplicated pages to mitigate the write latency by adding timing noise when occurring COW.
- We implement PCA based on KSM in a real experimental system. The experimental results show that PCA is able to reach its effectiveness.

The rest of this paper is structured as follows. Section 2 presents the necessary background. Section 3 details its design and implementation. Section 4 presents the analysis of PCA in terms of security and efficiency. Section 5 surveys related work of page sharing and covert channel. Section 6 concludes this paper.

2 Background and Threat Model

In this section, we describe some necessary background including the basic principles of KSM, and CBPS-based covert channel, which will help motivate our solution. And then the threat model and assumptions are discussed.

KSM: KSM, one implementation of CBPS in KVM, exists as a kernel thread in the host OS (Operating System) and periodically scans advised anonymous pages to merge identical pages. When a VM starts, Qemu invokes *madvise* call to advise its memory as mergeable. KSM manages pages by two red-black trees: **stable tree** and **unstable tree**. The stable tree stores already shared pages with write-protected, also named KSM pages. The unstable tree records the candidate pages that don't change frequently.

The content of pages is the index of the two trees for node search and insert. As shown in Fig. 1, in each scan round, a candidate page is firstly compared with pages in the stable tree. If there is a match, the candidate page is merged with it. Otherwise, its checksum will be recalculated before searching in the unstable tree, because KSM does not merge frequently changed pages. If the calculated checksum differs from the previous recorded one, KSM updates the checksum and continues with the next candidate page. Otherwise, if there is a match in the unstable tree, the candidate page is merged and added into the stable tree, while the matched page is purged from the unstable tree. If no match is found, the candidate page is only inserted into the unstable tree. Wherever a match

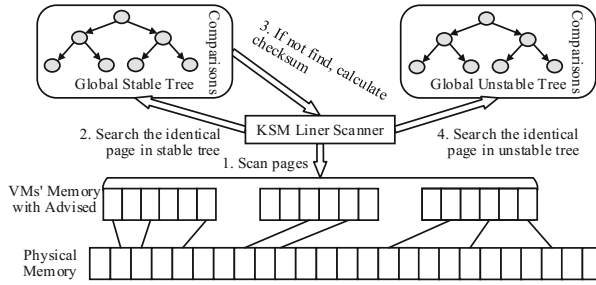


Fig. 1. The KSM overview for memory sharing.

is found, the page table entries of deduplicated pages are replaced by the KSM page with write-protected. After a full scan, the unstable tree is rebuilt since its pages content may be changed during the scan.

Covert Channel Attack: CBPS can be used to perform cross-VM covert channels to observe or transmit data because a COW page fault on deduplicated pages incurs measurable access latency than no COW. To transmit data, a program is required to run on the victim as the data sender. However, this requirement does not reduce their usefulness, since current OS is fragile and vulnerable, attackers are able to compromise the victim.

A successful covert channel attack needs four steps, as shown in Fig. 2. First, the sender and receiver load a certain amount of memory pages with identical content, for example reading a same file with memory alignment. Note, in this step, self-reflection should be avoided. Next, the sender encodes the information, e.g., writing certain pages. We make each page represent one bit of data. For instance, an unmodified page indicates bit 0 and a modified one denotes bit 1. If we want to transmit 10110010, the sender should modify the 1st, 3rd, 4th and 7th pages. Then, the sender and receiver need to wait for the eight pages being merged. Finally, the receiver writes all the eight pages and records their write access latency. At the receiver side, a long access time indicates 0, otherwise 1. The receiver can easily infer the transmitted data is 10110010.

Threat Model: We assume the attacker and victim are separate VMs co-resident on the same server. The attacker can compromise the target VM through multiple attack vectors. Thus she can stealthily place the ‘sender’ in the victim VM, which may be a spy program for filling pages with data that are expected to find in the victim’s memory. We also assume an active adversary model, in which the sender and receiver are cahoots. We assume the hypervisor, VMI tools and hardware are trusted. The introspected data structures cannot be modified by attackers, which is common to most existing VMI-based solutions [5,6].

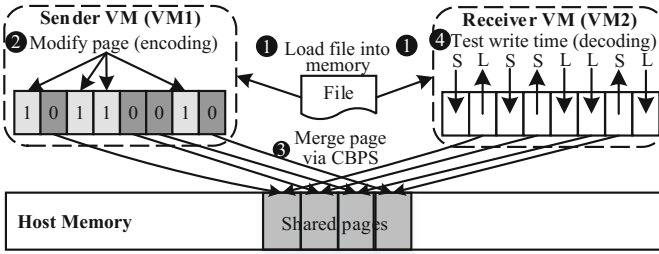


Fig. 2. To build a covert channel attack

3 Design and Implementation

In this section, we firstly introduce a highlight overview of PCA. Then we discuss how PCA classifies the pages of VMs, maintains the deduplicated pages to counter covert channels, and processes deduplication hints.

3.1 Overview

We have designed PCA with KSM as an example to clarify our approach. PCA includes two main mechanisms. One is “VM grouping and page classification”, which places pages that have much higher probability with the same content together, while pages with different content are divided. This mechanism is used to reduce the KSM overhead. Incidentally, covert channels of inter-group VMs are prevented. The other is “KSM cache”, which adds system noise for covert channels between intra-group VMs.

Figure 3 shows its architecture, which consists of four main components: (1) Page Permission Collector (PPC), (2) KSM Cache, (3) Grouping Manager (GM), and (4) Classification Manager (CM). PPC is used to capture the access permission of each candidate page from the guest OS by VMI-based interfaces, especially write access permission. Because pages with write access permission not only affect page sharing opportunities, but also may be used by covert channels. KSM cache maintains the private information of each deduplicated page: once a page is merged, it is inserted into the KSM cache. While the reconstructed KSM includes two parts: the GM and CM. The GM is in charge of dividing VMs into groups according to the VMs’ attributes. The duty of CM is to aggregate pages into different classifications based on page access permissions collected by PPC. Once system startup, the components are enforced and complement each other, which will be discussed in the following sections.

3.2 VM Grouping and Page Classification

Most of previous work is focus on how to detect more sharing opportunities, but they have overlooked the KSM own loss properties due to otiose operations in global trees. To solve this problem, the pages of VMs should be divided into

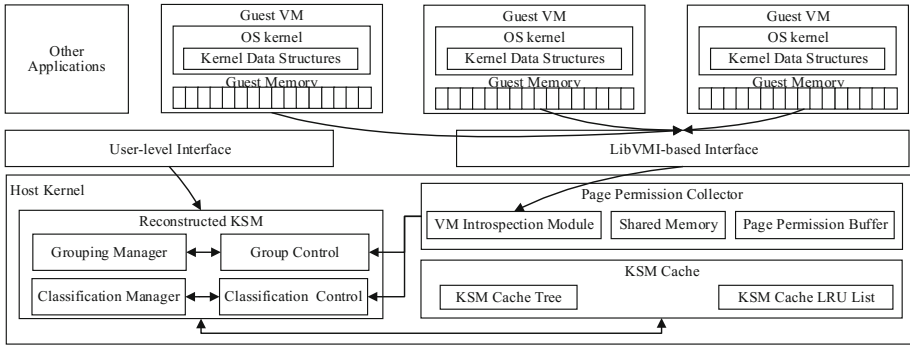


Fig. 3. The PCA architecture implemented with KSM

different sets to shrink looking up scope. While the division should meet the following two conditions: (1) pages with high probability to have same content should be divided into the same set, and vice versa. (2) the distribution of sharing potentials among sets should be balanced, because unbalanced division may still include a lot of otiose operations.

Grouping the VMs. The reason for grouping the VMs is the amount of redundant pages of inter-VM can be as low as 5%, but as high as 60% according to instances of the guest OS type and workloads [7]. For example, if several VMs run the same guest OS with same workloads their memory pages have a high possibility with equal content. Based on this fact, VMs can be divided into several groups. Accordingly, each global red-black tree is divided into multiple small trees, called G-trees. Thus, each group has a dedicated stable G-tree and unstable G-tree. A candidate page is only searched and compared within its G-trees. As the amount of tree nodes decreases, otiose operations are greatly reduced, while page sharing opportunities only have a slight variation. Besides, since the runtime of identification is saved, PCA can detect page sharing efficiently.

To support VM grouping, we mainly provide two ways. First, some options (user-level interfaces) are provided to customers. When renting VMs, they can use these interfaces to specify their VM’s workloads or security level. To achieve this, we extend the Qemu parameters for VM creation. This symbiotic manner can explicitly assist the provider to group VMs more accurate, and is fit for confidential scenarios, in which each VM has a security level. Another way is grouping the VMs based on their potential sharing opportunities. In this way, VMs are proactively grouped during their startup by analyzing their disk image or obtaining their guest kernel structures via VMI-based detection module.

To support multiple groups, we modify the KSM algorithm to break the global trees into multiple G-trees, and extend some additional structures. During startup, each VM is initialized with a group tag (called GID) according to its attributes. In the current prototype, the GID is attached to the *mm_struct* structure of each VM process. For each group, we defined a *group_node*

Table 1. The page classifications in each group

Classification	Classification ID	Description
Unused pages	0x000	The pages is not used by anyone
Kernel read-only pages	0x001	Pages is only readable in kernel space
Kernel read-write pages	0x010	Pages is writable in kernel space
User read-only pages	0x011	Pages mapped only readable in user space
User read-write pages	0x100	Pages is mapped writable in user space

structure, which obtains this group’s private information, such as GID, *ksm_scan* variable, tree root of G-trees and our introduced KSM cache etc. When function *ksm_madvise()* is invoked, the GID is retrieved for obtaining which group the VM belongs to. Thus, each VM’s *mm_struct* structure is registered to the *ksm_scan* of its group. Thus, each group has its own registered *mm_struct* list. By doing so, every page to be scanned should reference its GID first to choose its corresponding G-trees. Since each page has its private *rmap_item* structure, we can speed up the search in G-trees by bounding page’s GID into the address field of its *rmap_item* structure.

To save the cost of CPU and memory, we use a single thread as KSM (ksmd) to manage all groups. Since KSM is a linear scanner, to treat every group fairly every group has a weight value based on its sharing opportunities. At first, the weight is initialized by the ratio of the total memory of each group and total memory of mergeable. During runtime, the weight will be recounted by the shared pages of each group at the end of each round. The following formula is used to calculate the number of pages that should be scanned for each group, where N is the number of pages to be scanned per scan round, which is specified by the administrator.

$$\text{PerGroupScannedPages} = N \times \frac{\text{PerGroupSharedPages}}{\text{TotalSharedPages}} \quad (1)$$

Classifying the Advised Pages. We further discover that the sharing possibility between two pages of the same access permissions is more than those of not. For example, pages mapped to binary file will not be merged with pages of stack. Pages with most sharing are heap, shared library and page cache, which validates page access permission based classification is practicable. Thus, we further divide G-trees of each group into multiple classification trees (GC-trees) to improve KSM more effective.

To classify memory footprints of the VM into several classifications, we utilize VMI to obtain the page access permissions. During the page scan, PPC dynamically captures the access permissions for each candidate page, and pages with similar permissions are gathered into the same classification. In the current implementation, we defined five static page classifications as listed in Table 1. Therefore, each G-tree will be divided into five local GC-trees. That means each classification has a stable GC-tree and an unstable GC-tree. Since pages of

different classifications are cross-distribution, we need to obtain the access permissions in real time. For performance optimization, we provide a buffer in PPC for getting the access permission more rapidly. Each time access permissions of ten pages are obtained from the guest OS. In doing so, otiose operations are reduced, which will decrease the runtime to detect the same proportion shareable pages compared to traditional KSM. Thus, the scanner possesses more time to detect short-lived page sharing opportunities. In some case, new page sharing opportunities may be detected, which will be shown in Sect. 4.

Discussion. Except efficiency improvement, security is another advantage for grouping and classification. If the sender and receiver of a covert channel may be separated into different groups, thus the covert channels will be completely prevented without reducing the benefits of memory deduplication.

Since current KSM does not consider page access permission, the attacker's writable pages can be merged with pages of victim's application code to detect target vulnerable application [23, 25]. However, this situation cannot happen in PCA, since PCA has classified pages into different classification based on their access permissions. Besides, page classification provides a prerequisite for defending intra-group covert channels.

3.3 KSM Cache

The extreme countermeasure for covert channels is to forbid the pages of the sender and receiver to be merged, like our VM grouping. But if the sender and receiver are arranged in a same group, what should we do? The best way is to distinguish the pages employed by covert channels from others. However, it is challenging and may be impossible. We adopt another direction that pages used by covert channels are allowed to merge, but with some additional efforts to reduce the difference of write access time between deduplicated and KSM pages. For this purpose, we introduce KSM cache, which disturbs the receiver to decode the transmitted data by mitigating the write latency of COW. Thus, PCA can mitigate or even prevent covert channels between VMs in any cases.

In our design, each group has a dedicated KSM cache that is used for storing the deduplicated pages. Note, we only cache pages with writable access permission used to build covert channels. When two identical pages are detected, we do not free the duplicated page immediately. Instead we move it from the stable GC-tree to KSM cache, as shown in Fig. 4. Thus we can find a copy of the deduplicated pages as quickly as possible when a COW occurs, without requiring a new copy of the KSM page.

To implement KSM cache, we have chosen two kinds of data structures to store the cached pages with low overhead: a red-black tree (kcache tree) and a LRU (least recently used) linked list (kcache list). Duplicated pages are managed by the kcache tree whose node includes PFN (page frame number), hva (host virtual address), child nodes, *mm_struct* of deduplicated pages, and a list containing the VM identity etc. During tree search and insertion, there is no checksum and byte-for-byte content comparisons. So it is lightweight compared

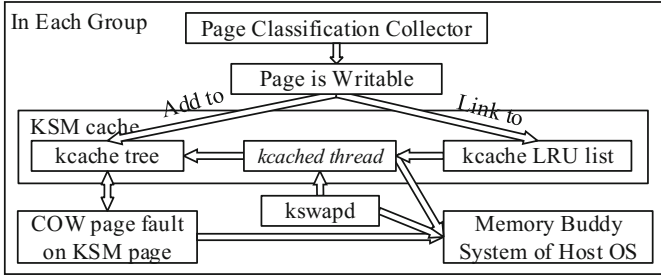


Fig. 4. The KSM cache architecture

to the KSM trees. The time complexity of the insert, delete and search operations is $O(\log n)$ in both average and worst-case. Besides, duplicated pages are maintained in a global kcache list for releasing them in a unified manner.

We have extended KSM to incorporate our KSM cache through a suite of hooks embedded in KSM. During KSM initialization, the KSM cache interfaces are mounted to the hooks. Like KSM, KSM cache is carried out as a kernel module that executes as a kernel thread, named *kcached*. Thus, since each group has its own data structures, *kcached* can manage the KSM cache group by group.

The aim of memory deduplication is to reduce the consumption of physical memory, so we must timely release the cached pages from the KSM cache. There is a tradeoff between the residence time of the duplicated page in KSM cache and the available memory freed by KSM. The longer the deduplicated pages reside in the KSM cache, the more security is ensured, and the less available memory is gained. To keep a better balance between available memory capacity and security, KSM cache is allowed to free its captured pages in three conditions. The first one is when a COW page fault occurs on a KSM page. In this case, we first find whether there is a match with the fault host virtual address and its VM identity in the KSM cache. If so, we directly map the matched page to this fault host virtual address. Thus, we avoid recreating a new copy. If not, the general COW process is followed. The second one is that we use the kcache list to periodically release a certain amount of outdated pages. Due to the natural characteristics of the LRU list, we will always process the oldest pages first. To better defend the covert channels, we add a random probability to the kcache list, in which outdated pages are freed randomly. We can therefore prevent the covert channels with justifiable overhead. Meanwhile keep the benefit of memory deduplication. The last one is when the system memory is insufficiency. We extend the memory deallocation to reclaim the pages of our KSM cache.

3.4 VMI-Based Memory Deduplication Scanner

Combing the above techniques, we implement a VMI-based memory deduplication scanner, using semantic information of VM’s memory footprints. Figure 5 shows how the pages are organized in PCA. VMs’ pages are first organized into

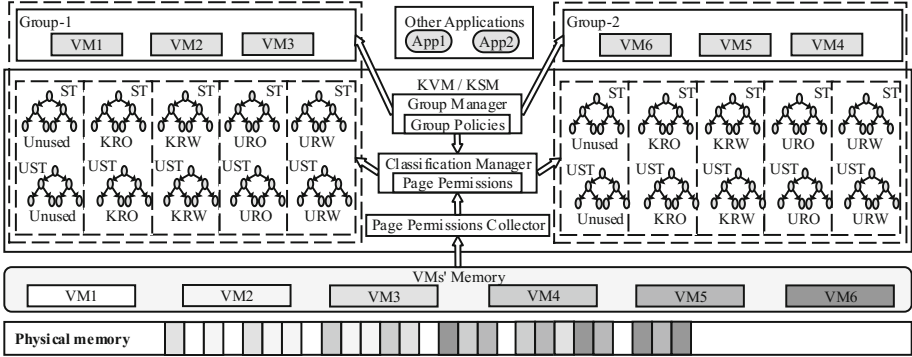


Fig. 5. Dividing VMs’ pages into different sets

different groups by G-trees. In each group, pages are classified by GC-trees. During grouping and classification, PCA needs to get the guest OS internal state. For simplicity, we take advantage of LibVMI [20] to achieve these. To reduce interaction, the VMI tool is divided into two parts: a LibVMI-based user application and a kernel module. They interact with each other through a shared memory mapped to a character device file (see Fig. 3).

To arrange VMs into proper group, we get their OS type, version and workloads during their startup. To classify the pages of each group, the scanner requires to know the access permission of pages in guest OS. To obtain these, we firstly translate the candidate page’s host virtual address into guest physical address (GPA) through the structure *kvm_memory_slot*. Then through VMI tool we can get the page structure array of guest OS, like *vmemmap* or *mem_map*. Thus, we can get the page structure in this array indexed by GPA. After that, we can learn whether this page is used via the *_mapcount* field. If the value is -1 , this page isn’t used. Otherwise, if the mapping field of the page structure is not empty, we can get the *anon_vma* or *address_space* structure. In this structure, we can get the access permissions. Otherwise, this page is used by kernel. According to its virtual field, we can know its access permissions. Note, except kernel code and module code, other pages are considered to be writable.

Figure 6 shows PCA’s flowchart. In each periodic scan, the KSM thread gets a candidate page, it firstly gets the GID from the GM. Then obtains the CID (classification ID) from CM. Thus, it can locate its local trees. Like KSM the candidate page will be searched in its stable GC-tree and then in its unstable GC-tree. The difference is that when a match is found, the deduplicated page is inserted into the kcache tree and kcache list. After finishing each scan round, all unstable GC-trees need to be rebuilt in the next scan round. KSM cache also needs to be freed periodically based on its kcache list.

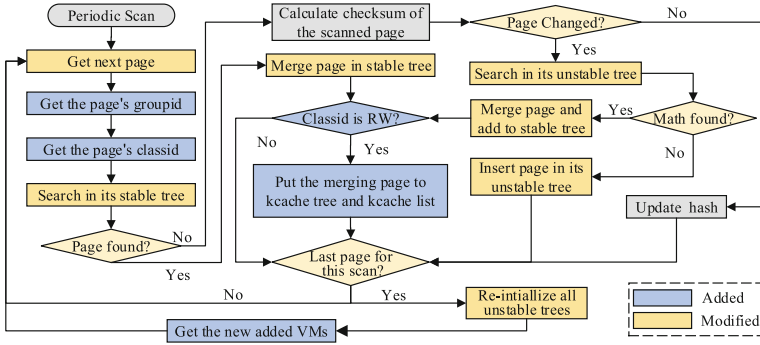


Fig. 6. The work-flow of PCA.

4 Evaluation

In this section, we firstly run the following benchmarks in VMs to show the effectiveness of PCA. Then we present its defense against covert channels and its overhead of preliminary evaluation. Our experiments are performed on a server with 4 2.6 GHz Intel Xeon E7 processors and 16 GB memory. Each processor has 8 physical cores. The server runs CentOS-6.5 virtualized by KVM with Linux-3.18.1. While VMs run CentOS-6.5 with Linux-2.6.38.

- Kernel Build: we compile the Linux kernel-3.18.1 in VMs. We begin this after the stable sharing opportunities are detected.
- Web Service: we run the Apache httpd server in VMs. We test the ab [38] benchmark with a local webpage.
- Bonnie++: we use bonnie++ [39] tool to do the hard disk benchmarking. The test file size is 2 times of the VM's memory.

4.1 Deduplication Effectiveness

We were particularly interested in seeing how does PCA work on merging pages compared to original KSM. To verify this, we have tested different workloads in PCA and KSM. In experiments, 4 VMs are booted, two in a group, the other two in different groups. To be able to measure the count of page comparisons and checksum calculations accurately, we modified some KSM functions to output the count per second. For simplicity, we don't show the inter-group test result, since the VMs in different groups will not be merged.

Figure 7 shows the page sharing opportunities. For Kernel Build and Apache, we can see that PCA is able to detect almost all page sharing opportunities, which is more than 97% of KSM. The results prove that fine-granularity is a good hint for page classification, and page access permission is a better guide for page classification. However, there is still room for improvement, because pages with same content but with different access permission will be separated

into different classifications. For bonnie++, PCA detects more page sharing opportunities than KSM. It might be because PCA can find many additional short-lived sharing opportunities that KSM is not able to detect. This further proves that fine-granularity classification can achieve higher accuracy, since the close contacts between page content and access permissions. We can also see that PCA can merge pages more quickly than KSM. Because PCA costs less comparison time in its classification trees, so that equal pages are identified earlier, thereby new sharing opportunities may be detected.

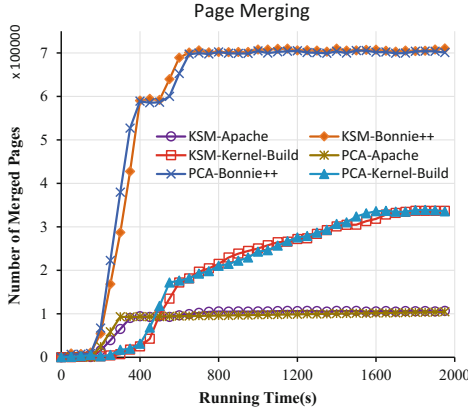


Fig. 7. Page sharing opportunities between 2 VMs with different workloads.

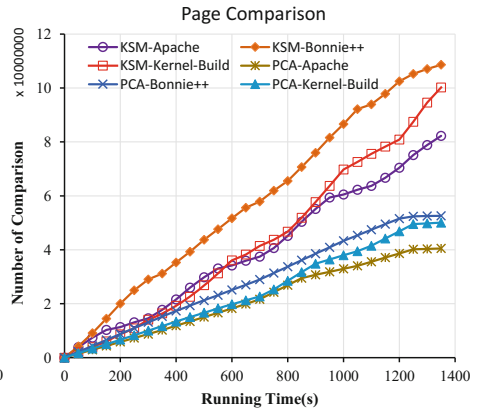


Fig. 8. The number of pages that need to be compared with different workloads.

Figures 8 and 9 respectively show the amount of page comparisons and checksum calculations. Figures show that KSM owns the largest number of page comparisons and checksum calculations due to its large global trees. While PCA has almost forty percent optimizations of KSM. Because in PCA multiple small classification trees contain less page nodes but have a much higher probability to have same content, which significantly reduces the otiose operations. Combining with Fig. 7, we can conclude that PCA has a available tradeoff between detecting page sharing opportunities and reducing otiose operations. The average reduction is about 40%. This results prove that based on page access permissions of guest OS PCA can accurately classify the pages.

4.2 Effectiveness of Covert Channel Defense

To evaluate the effectiveness of PCA against covert channels, we perform two types of experiment: sender and receiver in the same group and in different groups. We boot two virtual machines to deploy the sender and receiver process respectively. Each VM is configured with 1 VCPU and 512MB memory. They load a 404KB file (i.e. 101 4KB pages) into memory. To ensure each page is

unique, the file is generated randomly by /dev/random. To guarantee that all pages are merged, we set the sleeping time to 5 s. In each type, we test five times. In each test, the sender transfer different data to the receiver, and in receiver we record the write access time of the 101 pages to decode the delivered data.

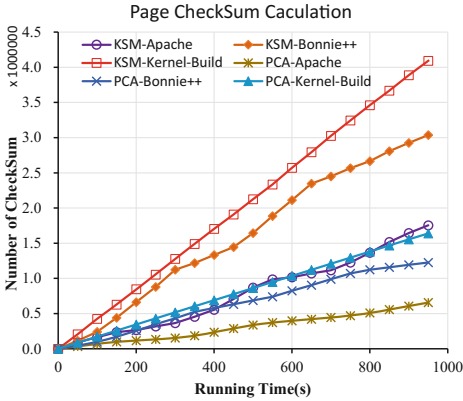


Fig. 9. The number of page checksums with different workloads.

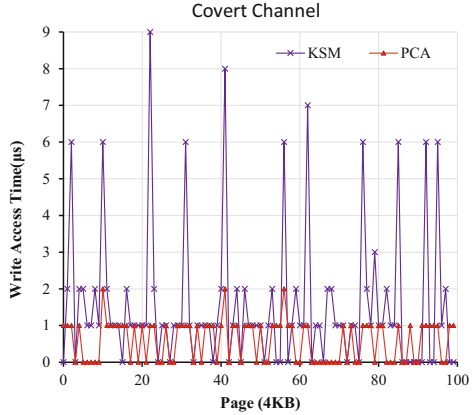


Fig. 10. Based on VM grouping and page classification to against covert channel.

Figure 10 shows the experimental results (VMs in the same group), which transmits a 101-bit data. To do this, the sender modifies the 3rd, 11th, 22th, 32th, 42th, 57th, 63th, 77th, 86th, 93th, 96th pages to encode a data. From Fig. 10, we can see that although different tests demonstrate different write access spikes, the write access time of sender-modified pages is always much less than the write access time of sender-unmodified pages in traditional KSM. While in PCA, there is no difference between sender-modified pages and sender-unmodified pages. This is because in PCA, once a page is merged, it will be added into the kcache tree immediately, and it will be hit in the kcache tree when the receiver decodes the data. Hence the write access time to the deduplicated pages has a negligible effect. Thus, the receiver cannot correctly recover the transmitted data.

4.3 Performance Overhead

The general trade-off of memory scanner is CPU utilization and memory consumption versus the security and efficiency. We have measured the CPU consumption of the scanner with some benchmarks by top measurements taken by per second. Figure 11 shows the average CPU utilization. We can see that PCA reduces the CPU overhead compared with original KSM. The highest impact is kernel build workload about 6%. Since kernel build workload has high memory usage during runtime, it exists vast otiose operations. However, PCA can effectively reduce these otiose operations. Thus the CPU overhead is reduced. While apache has low memory usage, so its overhead is reduced relatively less.

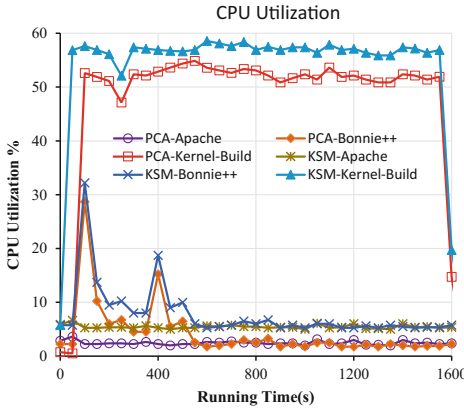


Fig. 11. The CPU utilization of KSM and PCA with different workloads.

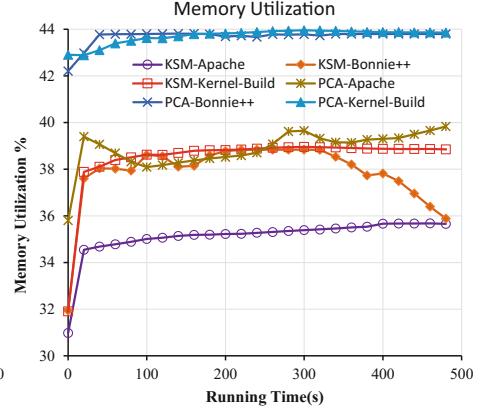


Fig. 12. The memory consumption of KSM and PCA with different workloads

Although the VMI used in our work may lead to a certain of performance loss. But the overhead is very little. Further, PCA provides a buffer to store access permissions, which reduces the number of VMI invocation. However, inserting duplicated pages into kcache tree is very cheap.

The only additional memory space is used by storing the page permission, the kcache tree, the kcache list, some structures and a few locks to sequential access shared data structures. From Fig. 12, we can see that PCA only consumes little memory during VMs execution. This overhead comes from KSM cache, because some key fields are recorded into the kcache tree nodes.

5 Related Work

Page Sharing. Limited main memory size has become one of the major bottlenecks in virtualization environment, and as an efficient approach to reduce server memory requirement, memory deduplication thus has attracted a large body of work on it. Disco [1] was the first system to implement page sharing on code pages with assistance from guest OS. CBPS requires no assistance from the guest OS, and was firstly implemented in VMware ESX server [2]. Then, CBPS was introduced in Xen [3] and KVM [4] to increase the memory density of VMs. But they can only merge anonymous pages, as the host regards the guests' memory as anonymous memory due to the semantic gap.

To acquire more sharing opportunities, Difference Engine [13] and Memory Buddies [14] proposed sub-page sharing, which not only explores the potential of same pages but also similar pages. Satori [15] employed sharing-aware virtual disks to find short-lived sharing opportunities. KSM++ [16] found pages in host cache are strong sharing candidates and preferential scan them can exploit short-lived sharing opportunities. Based on this, XLH [17] generates page hints in the host's virtual file system for merging them earlier. Singleton [18] combined the

host and guest double-caching into an exclusive cache. However, they didn't consider reducing needless overhead of KSM. While we argue that page sharing needs to consider classification for reducing otiose operations.

Empirical studies [7, 8] show CBPS can achieve memory savings up to 50% on I/O intensive workloads. But CBPS has higher runtime overhead by otiose comparison. To solve this, an adaptive policy [9] was proposed to obtain more sharing opportunities but with little CPU overhead. Sindelar et al. [10] proposed two hierarchical sharing models through sharing-aware algorithms without heavy CPU overhead. CMD [11] proposed a classification-based approach with a dedicated hardware. While IBM's AMD [12] generates a signature for each physical page to avert page comparison. PageForge [19] firstly proposed a hardware-based design for same-page merging that effectively reduces the CPU overhead. Except the performance improvement, PCA also concerns security.

Covert Channels. Recent research efforts [23–26] have mentioned the potential threat of covert channels based on memory deduplication. However, in their context, the covert channel is used primarily for leaking information. Xiao et al. [27] firstly constructed a rough covert channel to transmit information between two VMs. In an ideal situation, the bit rate of such covert channel can be around 1kpbs. However, its bit errors make it impractical in reality due to uncertain merging time. For this, Rong et al. [28] proposed a robust communication protocol for high-speed transmission and reliability. Xen's event channel can be used to conduct covert channels [29, 30], which has been demonstrated in Amazon EC2 [31]. Moreover, Gruss et al. [32] proposed the JavaScript based covert channel to collect private information in sandboxes. Our work aims to implement a defense scheme to these attacks.

Cloud providers can tackle covert channels through either preventative or detective approach, since they are much more resourceful. Amazon EC2 provides a dedicated instances service [33], in which different tenants' VMs do not share physical hardware. While the significant service charge reduces its attractiveness. Also, Wu et al. [31] advised the cloud provider to define a policy, which only allows two tenants to be shared in each server. But the tenant's neighbor is predetermined. These approaches may mitigate covert channels, but the memory utilization is low. In contrast, PCA has a low cost, but allows all tenants to share system memory.

Kim et al. [34] proposed a group-based memory deduplication scheme that aims to provide performance isolation on each single server. Deng et al. [35] also proposed a similar memory sharing mechanism, in which the global KSM thread is divided into per-group threads. Also, Ning et al. [36] proposed a covert channel defense mechanism based on VM grouping. SEMMA [37] provides a security architecture for performance isolation and security assurance. All of the above work only provides inter-group protection. Further, they did not consider otiose operations. Our work not only prevent covert channels in both inter-group and intra-group, but also reduce otiose operations to improve performance.

6 Conclusion

Memory deduplication is an important feature in modern hypervisors. However, it has otiose operations, and induces covert channels. In this paper, we put forward a highly efficient and secure memory deduplication approach called page correlation aggregation (PCA). PCA achieves two important objectives. First, it significantly reduces otiose operations. Meanwhile it speeds up the identification of page sharing and boosts the sharing opportunities. Second, it effectively mitigates or even prevents covert channels. We have implemented our design and evaluated it on KVM with different workloads. The experimental results show that PCA is effective, efficient and practical. In the future, we will plan to investigate the combination of PCA and balloon technique for efficiently managing the memory in the cloud. Also, we are interested in research logging mechanism with machine learning to detect the covert channels.

Acknowledgments. We would like to thank the anonymous reviewers for their constructive suggestions. This work was supported by the National Key Research and Development Program of China under grant No. 2016YFB0801002.

References

1. Bugnion, E., Devine, S., Govil, K., et al.: Disco: running commodity operating systems on scalable multiprocessors. *ACM Trans. Comput. Syst.* **15**(4), 412–447 (1997)
2. Waldspurger, C.A.: Memory resource management in VMware ESX server. *ACM SIGOPS Oper. Syst. Rev.* **36**(1), 181–194 (2002)
3. Kloster, J.F., Kristensen, J., Mejlholm, A., et al.: On the feasibility of memory sharing: content-based page sharing in the Xen virtual machine monitor (2006)
4. Arcangeli, A., Eidus, I., Wright, C.: Increasing memory density by using KSM. In: *Proceedings of the Linux Symposium*, pp. 19–28 (2009)
5. Srivastava, A., Giffin, J.: Tamper-resistant, application-aware blocking of malicious network connections. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) *RAID 2008*. LNCS, vol. 5230, pp. 39–58. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87403-4_3
6. Payne, B.D., Carbone, M., Sharif, M., Lee, W.: Lares: an architecture for secure active monitoring using virtualization. In: *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 233–247 (2008)
7. Chang, C.R., Wu, J.J., Liu, P.: An empirical study on memory sharing of virtual machines for server consolidation. In: *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications*, IEEE, pp. 244–249 (2011)
8. Barker, S.K., Wood, T., et al.: An empirical study of memory sharing in virtual machines. In: *Proceedings of the Annual Technical Conference*, pp. 273–284 (2012)
9. Rachamalla, S., Mishra, D., Kulkarni, P.: All page sharing is equal, but some sharing is more equal than others (2013). <http://www.cse.iitb.ac.in/internal/techreports/reports/TR-CSE-2013-49.pdf>
10. Sindelar, M., Sitaraman, R.K., Shenoy, P.: Sharing-aware algorithms for virtual machine colocation. In: *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 367–378 (2011)

11. Chen, L., Wei, Z., Cui, Z., et al.: CMD: classification-based memory deduplication through page access characteristics. In: Proceedings of the ACM International Conference on Virtual Execution Environments, vol. 49, no. 7, pp. 65–76 (2014)
12. Ceron, R., Folco, R., Leitao, B., et al.: Power systems memory deduplication. In: IBM Redbooks (2012). <http://www.redbooks.ibm.com/abstracts/redp4827.html>
13. Varghese, G., Voelker, G.M., Vahdat, A., et al.: Difference engine: harnessing memory redundancy in virtual machines. In: Proceedings of the USENIX Symposium on Operating Systems Design and Implementation, pp. 309–322 (2008)
14. Wood, T., Tarasuk-Levin, G., Shenoy, P., et al.: Memory buddies: exploiting page sharing for smart colocation in virtualized data centers. In: Proceedings of the International Conference on Virtual Execution Environments (2009)
15. Mios, G., Murray, D.G., Hand, S., Fetterman, M.A.: Satori: enlightened page sharing. In: Proceedings of the Annual Technical Conference, USENIX, pp. 1–14 (2009)
16. Miller, K., Franz, F., Groeninger, T., et al.: KSM++: using I/O-based hints to make memory-deduplication scanners more efficient. In: Proceedings of the ASPLOS Workshop on Runtime Environments, Systems, Layering and Virtualized Environments (2012)
17. Miller, K., Franz, F., Rittinghaus, M., Hillenbrand, M., Bellosa, F.: XLH: more effective memory deduplication scanners through cross-layer hints. In: Proceedings of the Annual Technical Conference, USENIX, pp. 279–290 (2013)
18. Sharma, P., Kulkarni, P.: Singleton: system-wide page deduplication in virtual environments. In: Proceedings of the International Symposium on High-Performance Parallel and Distributed Computing, ACM, pp. 15–26 (2012)
19. Skarlatos, D., Kim, N.S., Torrellas, J.: Pageforge: a near-memory content-aware page-merging architecture. In: Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture, ACM, pp. 302–314 (2017)
20. LibVMI tool. <http://libvmi.com/>
21. Irazoqui, G., Inci, M.S., Eisenbarth, T., Sunar, B.: Wait a minute! A fast, cross-VM attack on AES. In: Stavrou, A., Bos, H., Portokalidis, G. (eds.) RAID 2014. LNCS, vol. 8688, pp. 299–319. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11379-1_15
22. Irazoqui, G., Eisenbarth, T., Sunar, B.S.: \$ A: a shared cache attack that works across cores and defies VM sandboxing-and its application to AES. In: Proceedings of the Security and Privacy, pp. 591–604. IEEE (2015)
23. Ristenpart, T., Tromer, E., Shacham, H., et al.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 199–212. ACM (2009)
24. Suzaki, K., Iijima, K., Yagi, T., et al.: Software side channel attack on memory deduplication. In: Symposium on Operating Systems Principles, Poster session (2011)
25. Suzaki, K., Iijima, K., et al.: Memory deduplication as a threat to the guest OS. In: Proceedings of the Fourth European Workshop on System Security. ACM (2011)
26. Suzaki, K., Iijima, K., Yagi, T., et al.: Implementation of a memory disclosure attack on memory deduplication of virtual machines. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **96**(1), 215–224 (2013)
27. Xiao, J., Xu, Z., Huang, H., et al.: Security implications of memory deduplication in a virtualized environment. In: Proceedings of the Dependable Systems and Networks, pp. 1–12. IEEE (2013)

28. Rong, H., Wang, H., Liu, J., et al.: WindTalker: an efficient and robust protocol of cloud covert channel based on memory deduplication. In: Proceedings of the Big Data and Cloud Computing, pp. 68–75. IEEE (2015)
29. Shen, Q., Wan, M., Zhang, Z., Zhang, Z., Qing, S., Wu, Z.: A covert channel using event channel state on Xen hypervisor. In: Qing, S., Zhou, J., Liu, D. (eds.) ICICS 2013. LNCS, vol. 8233, pp. 125–134. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02726-5_10
30. Wu, J.Z., Ding, L., Wang, Y., et al.: Identification and evaluation of sharing memory covert timing channel in Xen virtual machines. In: Proceedings of the Cloud Computing, pp. 283–291. IEEE (2011)
31. Wu, Z., Xu, Z., Wang, H.: Whispers in the hyper-space: high-speed covert channel attacks in the cloud. In: Proceedings of USENIX Security Symposium, pp. 159–173 (2012)
32. Gruss, D., Bidner, D., Mangard, S.: Practical memory deduplication attacks in sandboxed Javascript. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 108–122. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_6
33. Amazon Web Services. Amazon EC2 dedicated instances. <http://aws.amazon.com/dedicated-instances/>
34. Kim, S., Kim, H., Lee, J.: Group-based memory deduplication for virtualized clouds. In: Alexander, M., et al. (eds.) Euro-Par 2011. LNCS, vol. 7156, pp. 387–397. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29740-3_44
35. Deng, Y., Hu, C., Wo, T., et al.: A memory deduplication approach based on group in virtualized environments. In: Proceedings of the Service Oriented System Engineering, pp. 367–372. IEEE (2013)
36. Ning, F., Zhu, M., You, R., et al.: Group-based memory deduplication against covert channel attacks in virtualized environments. In: Proceedings of the Trust-com/BigDataSE/I SPA, pp. 194–200. IEEE (2016)
37. Chen, X., Chen, W., Long, P., et al.: SEMMA: secure efficient memory management approach in virtual environment. In: Proceedings of the Advanced Cloud and Big Data, pp. 131–138. IEEE (2013)
38. Apache http server benchmarking tool. <http://httpd.apache.org/docs/2.2/programs/ab.html>
39. Bonnie++ file system benchmark. <https://www.coker.com.au/bonnie++/>



Role Recognition of Illegal Online Gambling Participants Using Monetary Transaction Data

Xiaohui Han^(✉), Lianhai Wang, Shujiang Xu, Dawei Zhao, and Guangqi Liu

Shandong Computer Science Center (National Supercomputer Center in Jinan),
Shandong Provincial Key Laboratory of Computer Networks, Qilu University of
Technology (Shandong Academy of Sciences), Jinan, China
{hanxh,wanglh,xushj,zhaodw,liuguangqi}@sdas.org

Abstract. Online gambling has become a substantial global industry during the past two decades. However, it is explicitly prohibited or restricted by most countries in the world due to social problems caused by it. This results in rapid expansion of the illegal online gambling (IOG) market where players profits are under little protection. To fight against IOG, this paper addresses the IOG participant-role recognition (PRR) problem by learning a supervised classifier with monetary transaction data. We propose two sets of features, i.e., transaction statistical features and network structural features, to effectively represent participants. Based on the feature representation, we adopt an ensemble learning strategy in the training phase of a PRR classifier to reduce the impact of unbalanced data. Results of experiments performed on real-world IOG case data demonstrate the feasibility and validity of the proposed approach. The proposed approach could help investigators in a law enforcement agency find the key members of an IOG organization quickly and destroy the ecosystem efficiently.

Keywords: Illegal online gambling · Role recognition · Online crime
Monetary transaction

1 Introduction

Online gambling, which was initiated in the mid-1990s, has exploded from a minor sideshow on the Internet into a substantial global industry over the past two decades. Due to factors such as convenience, accessibility, affordability, anonymity, and interactivity, online gambling could be potentially more tempting and addictive to consumers than traditional offline gambling. However, several recent studies suggest that it would exacerbate gambling problems in society, such as problem gambling, pathological gambling, and underage gambling [6, 8]. Consequently, a number of countries, including Mainland China, the United States, and Russia, explicitly prohibit most or all forms of online gambling. In the most remaining countries, online gambling is under strict regulations.

The prohibition and restriction of online gambling result in rapid expansion of the illegal online gambling (IOG) market where players' profits are under little protection. Several studies allege that IOG sites victimize participants rather than benefit them [1–4]. More seriously, it is reported that the IOG business is correlated with money laundering and other complex fraudulent and extortionist activities that, in turn, can fund yet other criminal activities [3]. Although states have adopted legal measures targeted at both consumers and operators to limit citizens' access to IOG sites. Yet the effectiveness of such measures may well be limited.

To fight against IOG, it is necessary to gain sufficient knowledge regarding how these businesses operate in an environment characterized by extreme uncertainty and high risk. This would require a systematic analysis of sophisticated organizational structures formed by IOG participants. However, to the best of our knowledge, fairly few researches have been carried out on automatic techniques for analyzing IOG ecosystems. This study addresses this absence by investigating the problem of automatically recognizing the roles that IOG participants played in their ecosystem. The proposed approach could help investigators in a law enforcement agency (LEA) find the key members of an IOG organization quickly and destroy the ecosystem efficiently. To be specific, this study makes the following contributions:

- We present a novel participant-role recognition (PRR) approach which learns a supervised classifier based on monetary transaction data to predict the roles of IOG participants.
- We propose two sets of features, i.e. transaction statistical features and network structural features, to effectively represent participants.
- We adopt an ensemble learning strategy in the training phase of the PRR classifier to reduce the impact of unbalanced training data.
- We evaluate the performance of the proposed approach using real-world IOG monetary transaction data. Experimental results demonstrate the feasibility and validity of the proposed approach.

The remainder of this paper is organized as follows: Sect. 2 reviews related work. Section 3 provides preliminaries related to this paper. Section 4 describes the details of the proposed approach. Section 5 gives the experimental results and analysis. Section 6 concludes the content of this paper.

2 Related Work

Prosperity on Internet gambling has drawn much attention from academics [11]. A number of studies have investigated possible precipitating factors for the expansion of online gambling. For example, Gainsbury et al. [6] claimed that technological innovations, including the availability of cheap, fast broadband connections in essentially any location, the emergence of mobile technology, and the use of trustable online payment systems, have played an important role in



Fig. 1. The pyramid structure of an IOG organization.

the growth of online gambling. They also found that the primary reasons people gave for preferring to gambling were: ease of access, convenience, comfort, greater privacy, and anonymity. Similar motivations for online gambling have been found in [14]. Several pieces of research studied demographic profiles of online gamblers and found that Internet gamblers were significantly more likely to be male gender, younger age, from higher socio-economic strata, employed full time, more technologically savvy, having more positive attitudes toward gambling, and better educated [6, 8, 9, 12, 13]. However, it was reported more women and young people are engaging in the activity [7].

Many recent studies reported online gambling is more harmful to gamblers compared to terrestrial gambling [6, 8, 9]. Wood and William [13] found online gamblers are under a higher probability of using drugs and alcohol than non-online gamblers. Due to the fragmented nature of governance, online gambling also presents a significant opportunity for crime and victimization. McMullan and Rege [10] investigated the types, techniques, and organizational dynamics of crime at portals of online gambling sites using document analysis based on data retrieved with the Google search engine.

In jurisdictions, such as the United States and Mainland China, for example, the prohibition of gambling has given rise to illegal online gambling business. Banks [3] summarized three principal forms of the illegal gambling business, i.e., accepting bets from a resident in a country where gambling is illegal, operating without appropriate licenses, and accepting bets from underage gamblers. There are numerous documented cases in which providers of illegal online gambling have been found to cheat customers on payouts, have apparently not paid winnings, have cheated players with unfair games, or have absconded with player deposits [1–4]. As claimed by McMullan and Rege [10], these cases identified by existing studies are likely to represent “the tip of the iceberg”, with many more crimes going unreported and unrecorded.

Based on our review of the related work, we have found that although IOG has been studied for years, most existing researches only focused on types of crime related to IOG. There continues to be a paucity of research on the systematic analysis of sophisticated organizational structures IOG ecosystems.

Especially, few automatic techniques have been proposed to analyze IOG ecosystems. In this study, we address this absence by presenting an approach to automatically recognize the roles that IOG participants played in their ecosystem.

3 Preliminaries

3.1 The IOG Ecosystem and Participant Roles

Figure 1 shows the typical structure of an IOG ecosystem, which is like a pyramid. A small number of **investors** provide funds for building online gambling platforms in countries/regions where online gambling is legal. Then these investors seek local organizers in a country/region, where online gambling is forbidden, to run the IOG business in that country/region. **Local organizers** employ gambling agents to attract gamblers to participate in IOG games. **Gambling agents** collect bets from gamblers and distribute IOG proceeds among participants. **Gamblers** can only participate in IOG games by transferring their bets to gambling agents. Participant roles in a higher level of the pyramid structure will keep interest at a definite ratio from the proceeds they collected from the direct lower level role. Based on the understanding of the IOG ecosystem, we aim to identify IOG participants into one of the four roles, i.e., investor, local organizer, gambling agent, and gambler.

3.2 Problem Statement

The participant-role recognition (PRR) problem can be considered as a supervised classification problem. We give a formal definition of PRR as follows.

Definition 1. *Given a set of K pre-defined roles $\mathcal{R} = \{r_1, r_2, \dots, r_K\}$ and a set of IOG participants $\mathcal{P} = \{p_1, p_2, \dots, p_M\}$, the task of participant-role recognition is to find a function $f(\cdot)$ to decide if a participant p_i 's role is r_j , i.e., $f : \mathcal{P} \times \mathcal{R} \mapsto \{0, 1\}$ such that for any pair $(p_i, r_j) \in \mathcal{P} \times \mathcal{R}$, we have*

$$f(p_i, r_j) = \begin{cases} 1, & \text{if } p_i \text{'s role is } r_j \\ 0, & \text{otherwise} \end{cases}. \quad (1)$$

For the i -th participant, we represent them as a n -dimensional feature vector $p_i = [p_{i1}, p_{i2}, \dots, p_{in}]^T$, where $p_i \in \mathbb{R}^n$. By this definition, solving the PRR problem involves extracting discriminative features as well as finding an appropriate classification scheme. We present our solution in the following section.

4 The Proposed Approach

Figure 2 provides an illustration of our PRR approach. We first compute transaction statistics and build a money flow network based on monetary transaction data obtained from an IOG ecosystem. Then we extract transaction statistical features and network structural features to form a vector representation for each participant sample. Based on the extracted features, we further learn a classifier to predict the role of new participant samples. This section describes each component of the approach in detail.

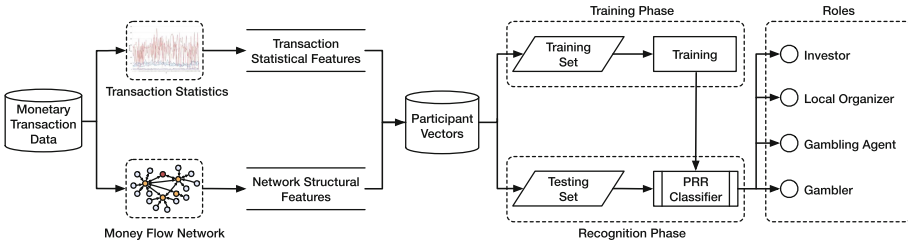


Fig. 2. Illustration of the proposed participant-role recognition approach.

4.1 Monetary Transaction Data

Monetary transaction data (MTD), which records the monetary transactions between IOG participants, could be obtained from bank accounts of IOG participants or derived from IOG servers. Each transaction record typically consists of information including the sender, the recipient, the amount of money transferred, and a time stamp. MTD contains very useful information for understating an IOG ecosystem. However, it is usually difficult to access MTD due to privacy reasons. The MTD used in this study was provided by a law enforcement agency. They obtained the data with warrants during the investigation of an IOG case.

4.2 Feature Extraction

We propose two sets of features to represent IOG participants in the learning process of a PRR classifier.

Transaction Statistical Features. We assume that participants in different roles have discriminative money transfer patterns which can be captured by statistics extracted from MTD. Here, we first introduce some notations used in the computation of transaction statistical features. Let M be the number of participants in an IOG ecosystem, and let N_D be the number of days covered by the MTD. We denote $\mathbf{A}^{(in)}$, $\mathbf{A}^{(out)}$, $\mathbf{F}^{(in)}$, $\mathbf{F}^{(out)}$, $\mathbf{C}^{(in)}$, and $\mathbf{C}^{(out)}$ as $M \times N_D$ dimensional matrices, where $\mathbf{A}_{ij}^{(in)}$ ($\mathbf{A}_{ij}^{(out)}$) is the amount of money participant p_i received (sent out) on the j -th day, $\mathbf{F}_{ij}^{(in)}$ ($\mathbf{F}_{ij}^{(out)}$) is the number of transfers p_i received (sent out) on the j -th day, and $\mathbf{C}_{ij}^{(in)}$ ($\mathbf{C}_{ij}^{(out)}$) is the number of incoming (outgoing) counterparties of p_i on the j -th day, respectively. To capture money transfer patterns of IOG participants, we compute the following features.

- **Means of Daily Income ($\hat{\mu}_i^{(ain)}$) and Expenditure ($\hat{\mu}_i^{(aout)}$)** measure the average amounts of money participant p_i receives from and sends to other participants daily, which are computed by the following equations:

$$\hat{\mu}_i^{(ain)} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{A}_{ij}^{(in)}, \tag{2}$$

$$\hat{\mu}_i^{(aout)} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{A}_{ij}^{(out)}. \quad (3)$$

- **Variances of Daily Income** ($\hat{\sigma}_i^{(ain)}$) **and Expenditure** ($\hat{\sigma}_i^{(aout)}$) measure how far the daily incomes and expenditures of participant p_i are spread out from the mean values, which are computed by the following equations:

$$\hat{\sigma}_i^{(ain)} = \frac{1}{N_D - 1} \sum_{j=1}^{N_D} [\mathbf{A}_{ij}^{(in)} - \hat{\mu}_i^{(ain)}]^2, \quad (4)$$

$$\hat{\sigma}_i^{(aout)} = \frac{1}{N_D - 1} \sum_{j=1}^{N_D} [\mathbf{A}_{ij}^{(out)} - \hat{\mu}_i^{(aout)}]^2. \quad (5)$$

- **Means of Daily Incoming** ($\hat{\mu}_i^{(fin)}$) **and Outgoing Transfer Numbers** ($\hat{\mu}_i^{(fout)}$) measure the average numbers of daily transfers participant p_i receives and sends out, which are computed by the following equations:

$$\hat{\mu}_i^{(fin)} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{F}_{ij}^{(in)}, \quad (6)$$

$$\hat{\mu}_i^{(fout)} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{F}_{ij}^{(out)}. \quad (7)$$

- **Variances of Daily Incoming** ($\hat{\sigma}_i^{(fin)}$) **and Outgoing Transfer Numbers** ($\hat{\sigma}_i^{(fout)}$) measure how far p_i 's daily numbers of incoming and outgoing transfers are spread out from their mean values, which are computed by the following equations:

$$\hat{\sigma}_i^{(fin)} = \frac{1}{N_D - 1} \sum_{j=1}^{N_D} [\mathbf{F}_{ij}^{(in)} - \hat{\mu}_i^{(fin)}]^2, \quad (8)$$

$$\hat{\sigma}_i^{(fout)} = \frac{1}{N_D - 1} \sum_{j=1}^{N_D} [\mathbf{F}_{ij}^{(out)} - \hat{\mu}_i^{(fout)}]^2. \quad (9)$$

- **Means of Daily Incoming** ($\hat{\mu}_i^{(cin)}$) **and Outgoing Counterparty Numbers** ($\hat{\mu}_i^{(cout)}$) measure p_i 's average numbers of daily incoming and outgoing counterparties, which are computed by the following equations:

$$\hat{\mu}_i^{(cin)} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{C}_{ij}^{(in)}, \quad (10)$$

$$\hat{\mu}_i^{(cout)} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{C}_{ij}^{(out)}. \quad (11)$$

– **Variiances of Daily Incoming ($\hat{\sigma}_i^{(cin)}$) and Outgoing Counterparty Numbers ($\hat{\sigma}_i^{(cout)}$)** measure how far p_i 's daily numbers of incoming and outgoing counterparties are spread out from the mean values, which are computed by the following equations:

$$\hat{\sigma}_i^{(cin)} = \frac{1}{N_D - 1} \sum_{j=1}^{N_D} [C_{ij}^{(in)} - \hat{\mu}_i^{(cin)}]^2, \tag{12}$$

$$\hat{\sigma}_i^{(cout)} = \frac{1}{N_D - 1} \sum_{j=1}^{N_D} [C_{ij}^{(out)} - \hat{\mu}_i^{(cout)}]^2. \tag{13}$$

Network Structural Features. All IOG participants are in pursuit of money. Gamblers want to win money by playing various kinds of games provided by an IOG platform. Investors, local organizers, and gambling agents aim to earn interest from gamblers. Therefore, money flow can be seen as the “blood” of an IOG ecosystem. We use a money flow network (MFN) to describe the “flow” of “blood”. Let f_{ij} be the total amount of money transferred from participant p_i to participant p_j , and let f_{ji} be the total amount of money transferred from p_j to p_i . We first define the money flow between p_i and p_j as follows.

Definition 2. *The money flow between two participants p_i and p_j is the absolute value of the difference between f_{ij} and f_{ji} , which is computed by:*

$$flow(p_i, p_j) = |f_{ij} - f_{ji}|. \tag{14}$$

The direction of the flow is from p_i to p_j if $f_{ij} > f_{ji}$, and is from p_j to p_i if $f_{ji} > f_{ij}$.

Based on the definition of money flow, we further give the definition of an MFN.

Definition 3. *A money flow network is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} is a finite set of vertices representing participants, and \mathcal{E} is a set of directed edges representing money flows between participants.*

In \mathcal{G} , there will be an edge $e_{ij} \in \mathcal{E}$ between two vertices v_i and v_j in \mathcal{V} if $flow(p_i, p_j) \neq 0$, and the weight and direction of the edge are the same with $flow(p_i, p_j)$ and the direction of the flow, respectively. An MFN can further be represented by an $M \times M$ dimensional adjacency matrix \mathbf{W} , where

$$\mathbf{W}_{ij} = \begin{cases} flow(p_i, p_j), & \text{if } e_{ij} \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

We assume that the importance of different roles in an MFN should be different. Thus, we compute the following centrality measures as network structural features to identify the importance of participants in an MFN.

- **In-degree.** This measure counts the number of incoming ties of a vertex. As edges in an MFN are weighted, values of incoming edge weights are summed up. The in-degree for vertex v_i is represented by the equation:

$$D_{in}(v_i) = \sum_{j=1}^M \mathbf{W}_{ji} \tag{16}$$

- **Out-degree.** This measure counts the number of out-going ties of a vertex. The weighted out-degree for vertex v_i is represented by the equation:

$$D_{out}(v_i) = \sum_{j=1}^M \mathbf{W}_{ij} \tag{17}$$

- **All-degree.** This measure counts the number of ties connecting a vertex to the others, regardless of their directionality. In the weighted version, the all-degree for vertex v_i is represented by the equation:

$$D_{all}(v_i) = \sum_{j=1}^M (\mathbf{W}_{ij} + \mathbf{W}_{ji}) \tag{18}$$

- **Betweenness.** Degrees are local measures, i.e., they do not take into account the whole network, but only the local neighborhood of a vertex. As a global measure, betweenness quantifies how frequently a vertex acts as a bridge along the shortest paths that connect every other couple of vertices. More formally, the betweenness centrality of node v_k is computed by the following equation:

$$C_b(v_k) = \frac{1}{(M-1)(M-2)} \sum_{i,j \neq k} \frac{N_{ij}^s(v_k)}{N_{ij}^s}, \tag{19}$$

where N_{ij}^s is the number of shortest paths linking the couple of vertices v_i and v_j , and $N_{ij}^s(v_k)$ is the number of that paths which contain v_k . $[(M-1)(M-2)]$ is the total number of pairs of vertices not including v_k .

- **Closeness.** This feature measures the inverse of the distance of a vertex from all the others in the network, considering the shortest paths that connect each couple of vertices. That is, it denotes how close a vertex is to others. Let $dist(v_i, v_j)$ be the number of edges in the shortest path linking vertices v_i and v_j , the closeness centrality of the vertex v_i is computed by the following equation:

$$C_c(v_i) = \frac{1}{\sum_{j=1}^M dist(v_i, v_j)}, \tag{20}$$

where $\sum_{j=1}^M dist(v_i, v_j)$ is the distance of vertex v_i from all the other vertices in the graph.

4.3 Participant Role Recognition

After we determine the features, we use them to represent IOG participants as vectors. Based on the vector representation, we train a classifier using labeled data for role recognition. One thing should be noted is that there may be an imbalance between the number of samples in each role category. For example, in an IOG ecosystem, the number of gamblers is much larger than that of investors. This imbalance could affect the performance of the trained classifier.

To reduce the negative impact of unbalanced training data, we adopt an ensemble learning strategy based on the AdaBoost algorithm [5]. We use Naive Bayesian as the base classifier. Each training tuple is assigned with a weight. A series of t classifiers are iteratively learned. In each learning iteration, the samples from the original training set are re-sampled to form a new training set. The samples with higher weights are selected with a higher probability. After a classifier H_i is learned, the samples misclassified by H_i are assigned higher weights. In the following learning iteration, the classifier H_{i+1} will pay more attention to the misclassified samples. In each round, we restrict the number of re-sampled samples in each role category to be the same (i.e., the size of the smallest category). The final class prediction is based on the weighted votes of the classifiers learned in each iteration. We named the overall ensemble classifier as EC4PRR (Ensemble Classifier for Participant Role Recognition).

5 Experiments and Analysis

In this section, we present a set of experiments to evaluate the performance of our PRR approach using real-world IOG monetary transaction data. We implemented all experiments in Java and MATLAB.

5.1 Data

As aforementioned, the MTD used in experiments were provided by an LEA in Shandong Province of China. The data contains two years of monetary transaction records between participants of an IOG platform. The LEA obtained this data during the investigation of the IOG case with warrants, and some of the key members of the IOG organization have been arrested. However, to protect the privacy, the MTD has been pre-processed by the LEA. Each participant was represented by an ID, and only a few fields of each transaction record were kept, including source ID, target ID, amount of transferred money, and time stamp. All the other information was removed. There are totally 4690 participants and 3.9 million transactions in the MTD. Each participant has been assigned a role manually by LEA investigators during the investigation. In Tables 1 and 2, we summarize some essential characteristics of the MTD. From Table 1, we can see the imbalance between the number of samples in each role category. In the training phase, we randomly select 85% of the samples as the training set to build a classifier and 15% as testing samples to validate the performance of the classifier.

Table 1. Participants distribution among roles.

	Investor	Local organizer	Gambling agency	Gamblers
# of participants	113	376	1930	2271

Table 2. Experimental data statistics.

Statistics	Incoming	Outgoing
Max amount of money a participant transfer daily	998,083 RMB	999,866 RMB
Min amount of money a participant transfer daily	1,000 RMB	1,000 RMB
Max # of daily transfers for a single participant	1136	1419
Min # of daily transfers for a single participant	1	1
Max # of daily counterparties for a single participant	836	655
Min # of daily counterparties for a single participant	1	1

5.2 Evaluation Measures

We adopt *precision*, *recall*, and F_1 score as performance measures. Let $\tilde{\mathcal{R}} = \{\tilde{R}_1, \tilde{R}_2, \dots, \tilde{R}_K\}$ refer to the role assignment output by a PRR classifier, and $\mathcal{R} = \{R_1, R_2, \dots, R_K\}$ be the ground-truth. Let TP_i be the number of participants assigned to \tilde{R}_i correctly according to R_i , FP_i refer to the number of participants assigned to \tilde{R}_i by mistake, and FN_i refer to the number of participants should be assigned to R_i but are assigned to other roles, the *precision*, *recall*, and F_1 for the whole experimental results can be briefed as follows:

$$precision = \frac{\sum_{i \in [1, K]} TP_i}{\sum_{i \in [1, K]} TP_i + \sum_{i \in [1, K]} FP_i} \quad (21)$$

$$recall = \frac{\sum_{i \in [1, K]} TP_i}{\sum_{i \in [1, K]} TP_i + \sum_{i \in [1, K]} FN_i} \quad (22)$$

$$F_1 = \frac{2 \times recall \times precision}{recall + precision} \quad (23)$$

5.3 Effectiveness of Features

To tested the effectiveness of the proposed features, we trained EC4PRR with different feature configurations. Three classifiers, named EC4PRR-TSF, EC4PRR-NSF, and EC4PRR-All, were learned using only transaction statistical features, only network structural features, and a combination of all features, respectively. For each classifier, we ran it using a 10-fold cross-validation.

The prediction results are shown in Fig. 3. From the results, we can see both EC4PRR-TSF and EC4PRR-NSF achieved relatively satisfactory performances. Precision scores of EC4PRR-TSF and EC4PRR-NSF achieved 0.76 and 0.71, respectively. The overall performances of the two classifiers are 0.69 and 0.65 respectively in terms of F_1 . This indicates that both transaction statistical

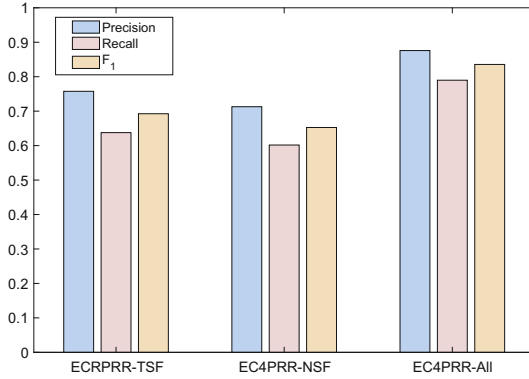


Fig. 3. Prediction results of EC4PRR classifiers learned with different feature configurations.

features and network structural features are discriminative for role recognition. EC4PRR-All, which was trained with all features, obtained the best performance in terms of all evaluation measures. The precision, recall, and F_1 of EC4PRR-All were 0.8759, 0.7898, and 0.8356, respectively. This reveals that a combination of the two sets of features can improve the performance of the EC4PRR classifier.

As the efficiency of transaction statistical features can be impacted by the time coverage of MTD, we also examined the sensitivity of EC4PRR-TSF and EC4PRR-All to the size of a dataset. We divided the two years of MTD into eight units with a time coverage of three months for each unit. We then derived subsets of the MTD by varying the time coverage from one to eight units and tested F_1 performances of the two classifiers using these eight subsets. Figure 4 shows the performance variations of EC4PRR-TSF and EC4PRR-All with data size enlarging. We can see that F_1 scores of both EC4PRR-TSF and EC4PRR-All become stable when the data size is larger than five units.

Table 3. Performance comparison of different classifiers.

	Naive Bayesian	SVM	Random forest	EC4PRR
Precision	0.5893	0.8286	0.8463	0.8759
Recall	0.5514	0.7832	0.7866	0.7898
F_1	0.5697	0.8053	0.8154	0.8356

5.4 Role Recognition Results

We compared the proposed EC4PRR with three common-used classifiers, which include Naive Bayesian, SVM, and Random Forest, to validate its predictive

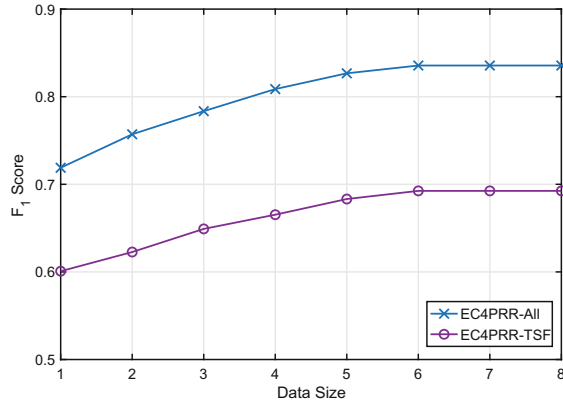


Fig. 4. Performance variations of EC4PRR-TSF and EC4PRR-All with the size of data enlarging.

performance. These classifiers have been implemented in WEKA¹ and LIBSVM². We also ran each of the classifiers using a 10-fold cross-validation.

Table 3 gives the results of this experiment. From Table 3, we can see that EC4PRR outperformed all the other three classifiers in terms of precision, recall, and F_1 . This demonstrates the effectiveness of EC4PRR for recognizing the roles of IOG participants. The performance of Naive Bayesian, which we used as the base classifier in our ensemble learning process, was the worst among all classifiers. The ensemble learning strategy made the final classifier much better. Note that the Random Forest is also a kind of classifier based on ensemble learning. However, our modification of the re-sampling strategy made EC4PRR more suitable for the unbalanced training data and achieve better performance.

6 Conclusion

In this study, we propose an automatic approach to address the IOG participant-role recognition problem. The proposed approach extracts two sets of features, i.e. transaction statistical features and network structural features, from monetary transaction data to effectively represent participants. A classifier named EC4PRR is trained based on the feature representation to predict the roles of participants. To reduce the impact of unbalanced training data, EC4PRR adopts an ensemble learning strategy in the training phase. Experiments were carried out on real-world IOG case data. The results indicate that both transaction statistical features and network structural features are discriminative for role recognition, and a combination of the two sets of features can improve the performance of EC4PRR. In comparison with other common-used

¹ <http://www.cs.waikato.ac.nz/ml/weka/>.

² <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

classifiers, EC4PRR achieved the best performance. The result also reveals that our modification of the ensemble learning process makes EC4PRR more suitable for unbalanced training data.

There are primarily two limitations of this study we should pay attention to in our future work. First, although we used real-world data in our experiments, the data was only extracted from a specific IOG case, more case data should be collected in the future. Second, we did not consider the situation that an IOG participant plays multiple roles. New techniques should be proposed to address this problem.

Acknowledgment. This work was supported in part by National Natural Science Foundation of China (61602281, 61702309), Natural Science Foundation of Shandong Province of China (ZR2015YL018, ZR2016YL011, and ZR2016YL014), Shandong Provincial Key Research and Development Program (2018CXGC0701, 2018GGX106005, 2017CXGC0701, and 2017CXGC0706).

References

1. Albanese, J.S.: Illegal gambling businesses & organized crime: an analysis of federal convictions. *Trends Organ. Crime* **21**(3), 262–277 (2018)
2. Banks, J.: Edging your bets: advantage play, gambling, crime and victimisation. *Crime Media Cult.* **9**(2), 171–187 (2013)
3. Banks, J.: Internet gambling, crime and the regulation of virtual environments. In: Banks, J. (ed.) *Gambling, Crime and Society*, pp. 183–223. Palgrave Macmillan UK, London (2017). https://doi.org/10.1057/978-1-137-57994-2_6
4. Blaszczynski, A.: Online gambling and crime: causes, controls and controversies. *Int. Gambl. Stud.* **15**(2), 340–341 (2016)
5. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *European Conference on Computational Learning Theory*, pp. 23–37 (1995)
6. Gainsbury, S., Wood, R., Russell, A., Hing, N., Blaszczynski, A.: A digital revolution: comparison of demographic profiles, attitudes and gambling behavior of internet and non-internet gamblers. *Comput. Hum. Behav.* **28**(4), 1388–1398 (2012)
7. Gainsbury, S.M., Russell, A., Hing, N., Wood, R., Blaszczynski, A.: The impact of internet gambling on gambling problems: a comparison of moderate-risk and problem internet and non-internet gamblers. *Psychol. Addict. Behav.* **27**(4), 1092–1101 (2013)
8. Gainsbury, S.M., Russell, A., Hing, N., Wood, R., Dan, L., Blaszczynski, A.: How the internet is changing gambling: findings from an Australian prevalence survey. *J. Gambl. Stud.* **31**(1), 1–15 (2013)
9. Gainsbury, S.M., Russell, A., Wood, R., Hing, N., Blaszczynski, A.: How risky is internet gambling? A comparison of subgroups of internet gamblers based on problem gambling status. *New Media Soc.* **17**(6), 861–879 (2015)
10. McMullan, J.L., Rege, A.: Online crime and internet gambling. *J. Gambl. Issues* **24**(5), 115–116 (2010)
11. Tong, S., Zhang, H., Shen, B., Zhong, H., Wang, Y., Jin, B.: Detecting gambling sites from post behaviors. In: *2016 IEEE 11th Conference on Industrial Electronics and Applications*, pp. 2495–2500 (2016)

12. Wardle, H., Moody, A., Griffiths, M., Orford, J., Volberg, R.: Defining the online gambler and patterns of behaviour integration: evidence from the british gambling prevalence survey 2010. *Int. Gambl. Stud.* **11**(3), 339–356 (2011)
13. Wood, R.T., Williams, R.J.: A comparative profile of the internet gambler: demographic characteristics, game-play patterns, and problem gambling status. *New Media Soc.* **13**(13), 1123–1141 (2011)
14. Wood, R.T., Williams, R.J., Lawton, P.K.: Why do internet gamblers prefer online versus land-based venues? Some preliminary findings and implications. *J. Gambl. Issues* **20**, 235–252 (2007)



Certifying Variant of RSA with Generalized Moduli

Yao Lu¹, Noboru Kunihiro¹, Rui Zhang^{2,4}(✉), Liqiang Peng^{2,3}(✉),
and Hui Ma²(✉)

¹ The University of Tokyo, Tokyo, Japan

² State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
Beijing 100093, China

{r-zhang, pengliqiang, mahui}@iie.ac.cn

³ Data Assurance and Communication Security Research Center,
Chinese Academy of Sciences, Beijing 100093, China

⁴ School of Cyber Security, University of Chinese Academy of Sciences,
Beijing 100049, China

Abstract. Let N be an arbitrary integer with unknown factorization. In Asiacrypt 2012, Kakvi et al. proposed an algorithm that, given prime $e \geq N^{\frac{1}{4}} + \epsilon$, certifies whether the RSA function $\text{RSA}_{N,e}(x) := x^e \bmod N$ defines a permutation over \mathbb{Z}_N^* or not. In this paper, we extend Kakvi et al.'s work by considering the case with generalized moduli $N = \prod_{i=1}^n p_i^{z_i}$. Surprisingly, when $\min\{z_1, \dots, z_n\} \geq 2$, we show that it can be efficiently decided whether the RSA function defines a permutation over \mathbb{Z}_N^* or not even for the prime $e < N^{\frac{1}{4}}$. Our result can be viewed as an extension of Kakvi et al.'s result.

Keywords: Coppersmith's method · Lattices · RSA
Public key cryptosystem · LLL algorithm

1 Introduction

RSA function is one of the most well known cryptographic primitives, it is defined as $\text{RSA}_{N,e} : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$, $x \rightarrow x^e \bmod N$, where N is a public modulus and e is an exponent integer. Moreover, it is believed that RSA function (with the appropriate choice of parameters) defines a family of trapdoor permutations, which has a number of applications in public-key cryptosystems.

In Crypto'92, Bellare and Yung [2, 3] introduced a new primitive called certified trapdoor permutations, compared with standard trapdoor permutation, it requires an additional efficient permutation checking procedure, roughly speaking, a trapdoor permutation is certified if one can verify from the public key that it is actually a permutation. Using certified trapdoor permutations as a building block, we can construct many useful cryptographic protocols: ZAPS and verifiable PRF [6]; Sequential aggregate signatures [1, 12]. More importantly, NIZK

protocols for any NP-statement can be built from certified trapdoor permutations [8].

Among all the known candidate trapdoor permutations (factoring-based), RSA trapdoor function is the most efficient certified trapdoor permutation currently known. It is well known that RSA trapdoor function defines a permutation over the domain \mathbb{Z}_N^* iff $\gcd(e, \phi(N)) = 1$ where $\phi(\cdot)$ is Euler's totient function i.e. the number of positive integers less than or equal to N that are coprime to N . So we only need to check whether $\gcd(e, \phi(N)) = 1$ to tell whether RSA trapdoor function defines a permutation or not.

Generally speaking, RSA function is not a certified trapdoor permutation. In [2,3], Bellare and Yung proposed a generalized approach that can transform every trapdoor permutation into a certified trapdoor permutation. Using their method, we can easily make RSA function to be certified, however, since Bellare-Yung transformation brings an additional computational overhead, which makes their method relatively inefficient. Besides, in order to keep the same data structure, we prefer to use directly method rather than artificial method. In [4,12], the authors showed that if prime $e > N$, the RSA function is a certified permutation (since e is a prime and $\phi(N) < N$, thus $\gcd(e, \phi(N)) = 1$), but in practice, using large exponent e ($e > N$) will bring heavy costs for modular exponentiation. Therefore, the question naturally arises: Is the RSA function certified for the case of $e < N$?

There exist several research results on the above problem. Suppose that N is a RSA modulus i.e. $N = pq$ where p and q are of the same bit-size. If prime $e \geq N^{1/4}$, we can efficiently decide whether e divides $\phi(N)$ or not by using Coppersmith's result [5]. On the other hand, if prime $e < N^{1/4}$, it is hard to decide whether e divides $\phi(N)$ or not, which is called Phi-Hiding Assumption, by Cachin, Micali and Stadler [4] in the context of efficient single database private information retrieval, which has found a lot of applications in cryptography.

Later, in Asiacrypt 2008, Schridde and Freisleben [15] showed that the Phi-Hiding Assumption does not hold for special composite integers of the form $N = pq^{2k}$ for $k > 0$. Such integers are often used in cryptography to speed up certain operations [16].

Suppose that N is an arbitrary integer with unknown factorization. Recently, in Asiacrypt 2012, Kakvi et al. [10] proposed an algorithm that, given prime $e \geq N^{1/4+\epsilon}$, efficiently decides whether e divides $\phi(N)$ or not. Kakvi et al. [10] gave an efficient certification procedure that works for any prime exponent $e > N^{1/4}$ (rather than $e > N$). However, until now, if prime $e < N^{1/4}$, we do not know that whether Phi-Hiding Assumption holds or not (we know the results for RSA moduli [4] and moduli of the form pq^{2k} with $k > 1$ [15], but we know nothing for arbitrary integer with unknown factorization).

1.1 Our Contributions

In this paper, given an arbitrary modulus N with unknown factorization and a prime $e < N^{1/4}$, we can extract more information than previously expected, which enable us to efficiently decide whether e divides $\phi(N)$ or not in some

circumstances, that means we can further improve [10]’s result in this circumstances.

In particular, using our algorithm, if prime e satisfies $N^{1/4r} < e < N^{\min\{1/4(r-1), 1\}}$ (r is a positive integer), we can check whether exists secret factor p s.t. $e|p-1$ and $N \equiv 0 \pmod{p^r}$. Note that when $r = 1$, that is exactly [10]’s result. Thus, our result can be viewed as a generalization of [10]’s result.

Although when $e < N^{1/4}$, we can not directly decide whether $\gcd(e, \phi(N)) = 1$ or not (since we can only check the factor $p: N \equiv 0 \pmod{p^r}$, r is related to the size of exponent e), we can identify the scenarios of $\gcd(e, \phi(N)) \neq 1$ if moduli of form $N \equiv 0 \pmod{p^r}$ are used and p hides e . In addition, let $N = \prod_{i=1}^n p_i^{z_i}$, for the case of $\min\{z_1, \dots, z_n\} \geq 2$, we can improve [10]’s result: for example, for $N = p^2q^3$, we can improve [10]’s result to 1/8. Therefore, using moduli of this form, we can further decrease the size of exponent e while publicly verifying the permutation. However, on the other hand, this indicates that cryptographic schemes using moduli of this form and relying on the Phi-Hiding Assumption must be handled with care.

Our technique is similar to Kakvi et al. [10], we also use Coppersmith’s method [5] to find prime divisors p of N in a specific range, and the key problem is to show that the number of invocations of Coppersmith’s algorithm in our certification algorithm is polynomial-time.

Like [10]’s algorithm, our algorithm also only works for prime e and how to extend to arbitrary integers e of unknown factorization is still an open problem.

2 Preliminary

2.1 Certified Trapdoor Permutation

Definition 1. A family of trapdoor permutations is a tripe of algorithms (G, E, D) such that:

- $G(\cdot)$ is a randomized algorithm that takes no input and generates a pair (pk, sk) , where pk is a public key and sk is a secret key;
- $E(\cdot)$ is a deterministic algorithm such that, for every fixed public key pk , the mapping $x \rightarrow E(pk, x)$ is a bijection;
- $D(\cdot)$ is a deterministic algorithm such that for every possible pair of keys (pk, sk) generated by $G(\cdot)$ and for every x we have

$$D(sk, E(pk, x)) = x$$

A family of permutations Π is said to be certified if the permutation can be verified in polynomial-time given pk . We follow the definition in [10, 12].

Definition 2. $\Pi = (G, E, D, C)$ is called a family of certified trapdoor permutations if (G, E, D) is a family of trapdoor permutations and $C(\cdot)$ is a deterministic polynomial-time algorithm such that, for an arbitrary pk (potentially not generated by $G(\cdot)$), returns 1 iff $(E(pk, \cdot))$ defines a permutation over domain Dom_{pk} .

2.2 Coppersmith’s Method

Let us introduce Coppersmith’s algorithm for finding small roots of modular polynomial equations. Our main algorithm uses Coppersmith’s algorithm as sub-routine.

Theorem 1 (Coppersmith [5], May [13]). *Let N be an integer of unknown factorization, which has a divisor $p \geq N^\beta$, $0 < \beta \leq 1$. Let $0 < \mu \leq \frac{1}{7}\beta$. Furthermore, let $f(x)$ be a univariate monic polynomial of degree δ . Then we can find all solutions x_0 for the equation:*

$$f(x_0) = 0 \pmod p \quad \text{with} \quad |x_0| \leq \frac{1}{2}N^{\frac{\beta^2}{\delta} - \mu}$$

This can be achieved in time $\mathcal{O}(\mu^{-7}\delta^5 \log^2 N)$. The number of solutions x_0 is bounded by $\mathcal{O}(\mu^{-1}\delta)$.

In the rest of our paper, one of our main algorithms is to find small roots of polynomial equation $f(x) = x + a = 0 \pmod p$, where p is unknown that satisfies $N = 0 \pmod{p^r}$. We can model this problem as the univariate polynomial with degree r :

$$f(x) = (x + a)^r \pmod{p^r}$$

A direct application of Theorem 1 can get the desired result. However, we notice that the form of polynomial $f(x) = (x + a)^r$ is kind of special, actually we can use a smarter way to solve this type of equation. In this paper, we exploit Lu et al’s. [11] algorithm because of its better performance and lower complexity.

Theorem 2 (Lu et al. [11]). *For every $0 < \mu < \beta$, let N be a sufficiently large composite integer (of unknown factorization) with a divisor p^r ($p \geq N^\beta$ and r is a positive integer: $r \geq 1$). Let $f(x) \in \mathbb{Z}[x]$ be a univariate monic linear polynomial. Then we can find all the solutions x_0 of the equation $f(x) = 0 \pmod p$ with $|x_0| \leq N^\gamma$ if*

$$\gamma < r\beta^2 - \mu$$

The time complexity is $\mathcal{O}(\mu^{-7} \log^2 N)$.

For completeness, we give the whole proof here.

Proof. Consider the following univariate linear polynomial:

$$f_1(x) = a_0 + a_1x \pmod p$$

where N is known to be a multiple of p^r for known r and unknown p . Here we assume that $a_1 = 1$, since otherwise we can multiply f_1 by $a_1^{-1} \pmod N$. Let $f(x) = a_1^{-1}f_1(x) \pmod N$.

We define a collection of polynomials as follows:

$$g_k(x) := f^k(x)N^{\max\{\lceil \frac{t-k}{r} \rceil, 0\}}$$

for $k = 0, \dots, m$ and integer parameters t and m with $t = \tau m$ ($0 \leq \tau < 1$), which will be optimized later. Note that for all k , $g_k(y) \equiv 0 \pmod{p^t}$.

Let $X := N^{r\beta^2 - \mu} (= N^\gamma)$ be the upper bound on the desired root y . We will show that this bound can be achieved for any chosen value of μ by ensuring that $m \geq m^* := \lceil \frac{\beta(2r+1-r\beta)}{\mu} \rceil - 1$.

We build a lattice \mathcal{L} of dimension $d = m + 1$ using the coefficient vectors of $g_k(xX)$ as basis vectors. We sort these polynomials according to the ascending order of g , i.e., $g_k < g_l$ if $k < l$.

From the triangular matrix of the lattice basis, we can compute the determinant as the product of the entries on the diagonal as $\det(\mathcal{L}) = X^s N^{s_N}$ where

$$\begin{aligned}
 s &= \sum_{k=0}^m k = \frac{m(m+1)}{2} \\
 s_N &= \sum_{k=0}^{t-1} \lceil \frac{t-k}{r} \rceil = \sum_{k=0}^{t-1} \left(\frac{t-k}{r} + c_k \right) \\
 &= \frac{\tau m(\tau m + 1)}{2r} + \sum_{k=0}^{t-1} c_k.
 \end{aligned}$$

Here we rewrite $\lceil \frac{t-k}{r} \rceil$ as $(\frac{t-k}{r} + c_k)$ where $c_k \in [0, 1)$. To obtain a polynomial with short coefficients that contains all small roots over integer, we apply *LLL*-basis reduction algorithm to the lattice \mathcal{L} . Lemma 1 gives us an upper bound on the norm of the shortest vector in the *LLL*-reduced basis.

Lemma 1 (LLL [7]). *Let \mathcal{L} be a lattice of dimension w . Within polynomial-time, *LLL*-algorithm outputs a set of reduced basis vectors v_i , $1 \leq i \leq w$ that satisfies*

$$\|v_1\| \leq \|v_2\| \leq \dots \leq \|v_i\| \leq 2^{\frac{w(w-1)}{4(w+1-i)}} \det(\mathcal{L})^{\frac{1}{w+1-i}}$$

If the bound is smaller than the bound given in Lemma 2 (below), we can obtain the desired polynomial.

Lemma 2 (Howgrave-Graham [9]). *Let $g(x_1, \dots, x_k) \in \mathbb{Z}[x_1, \dots, x_k]$ be an integer polynomial that consists of at most w monomials. Suppose that*

1. $g(y_1, \dots, y_k) \equiv 0 \pmod{p^m}$ for $|y_1| \leq X_1, \dots, |y_k| \leq X_k$ and
2. $\|g(x_1 X_1, \dots, x_k X_k)\| < \frac{p^m}{\sqrt{w}}$

Then $g(y_1, \dots, y_k) = 0$ holds over integers.

We require the following condition:

$$2^{\frac{d-1}{4}} \det(\mathcal{L})^{\frac{1}{d}} < \frac{N^{\beta\tau m}}{\sqrt{d}}$$

where $d = m + 1$. We plug in the values for $\det(\mathcal{L})$ and d , and obtain

$$2^{\frac{m(m+1)}{4}} (m+1)^{\frac{m+1}{2}} X^{\frac{m(m+1)}{2}} < N^{\beta\tau m(m+1) - \frac{\tau m(\tau m + 1)}{2r} - \sum_{k=0}^{t-1} c_k}$$

To obtain the asymptotic bound, we let m grow to infinity. Note that for sufficiently large N the powers of 2 and $m + 1$ are negligible. Thus, we only consider the exponent of N . Then we have

$$X < N^{2\beta\tau - \frac{\tau(\tau m + 1)}{r(m+1)} - \frac{2\sum_{k=0}^{t-1} c_k}{m(m+1)}}$$

Setting $\tau = r\beta$, and noting that $\sum_{k=0}^{t-1} c_k \leq t$, the exponent of N can be lower bounded by

$$r\beta^2 - \frac{\beta(1 - r\beta)}{m + 1} - \frac{2r\beta}{m + 1}$$

We appropriate the negative term $\frac{*}{m+1}$ by $\frac{*}{m}$ and obtain

$$r\beta^2 - \frac{\beta(2r + 1 - r\beta)}{m}$$

Enduring that $m \geq m^*$ will then guarantee that X satisfies the required bound for the chosen value of μ .

The running time of our method is dominated by *LLL*-algorithm, which is polynomial in the dimension of the lattice and in the maximal bit-size of the entries. We have a bound for the lattice d

$$d = m + 1 \geq \lceil \frac{\beta(2r + 1 - r\beta)}{\mu} \rceil$$

Since $r\beta < 1$, then we obtain $d = \mathcal{O}(\mu^{-1})$. The maximal bit-size of the entries is bounded by

$$\max\{\frac{t}{r} \log(N), dr\beta^2 \log(N)\} = \max\{\beta d \log(N), dr\beta^2 \log(N)\}$$

Since $r\beta < 1$ and $d = \mathcal{O}(\mu^{-1})$, the bit-size of the entries can be upper bounded by

$$\max\{\mathcal{O}(\beta\mu^{-1}) \log(N), \mathcal{O}(\beta\mu^{-1}) \log(N)\} = \mathcal{O}(\mu^{-1} \log(N))$$

Nguên and Stehlé [14] proposed a modified version of the *LLL*-algorithm called *L²*-algorithm. The *L²*-algorithm achieves the same approximation quality for a shortest vectors as the *LLL*-algorithm, but has an improved worst case running time analysis. Its running time is $\mathcal{O}(d^5(d + \log b_d) \log b_d)$, where $\log b_d$ is the maximal bit-size of an entry in lattice. Thus, we can obtain the running time of our algorithm

$$\mathcal{O}\left(\left(\frac{1}{\mu}\right)^5 \left(\frac{1}{\mu} + \frac{\log N}{\mu}\right) \frac{\log N}{\mu}\right)$$

Therefore, the running time of our algorithm is $\mathcal{O}(\mu^{-7} \log^2 N)$. Eventually, the vector output by *LLL*-algorithm gives a univariate polynomial $g(x)$ such that $g(y) = 0$, and one can find the root of $g(x)$ over the integers.

3 Our Main Result

In this section we give our main result.

Theorem 3. *Let N be an integer of unknown factorization and $e < N$ (suppose $\gamma = \log_N e$) be a prime integer and $\gcd(e, N) = 1$. First determine a positive integer r such that $\frac{1}{4r} < \gamma < \min\{\frac{1}{4(r-1)}, 1\}$. Let $\epsilon = \gamma - \frac{1}{4r}$, then we can decide whether exists secret factor p s.t. $e|p - 1$ and $N \equiv 0 \pmod{p^r}$ in time $\mathcal{O}(\epsilon^{-8} \log^2 N)$.*

Before we provide a proof for Theorem 3, we would like to interpret its implications. Notice that Theorem 3 yields in the special case $r = 1$ the bound $\frac{1}{4} < \gamma < 1$ that corresponds to [10]’s result. Thus, our result can be viewed as a generalization of [10]’s result.

On the other hand, we would like to give an example to clarify our result. If $\frac{1}{8} < \gamma < \frac{1}{4}$, we can check whether exists secret factor p s.t. $e|p - 1$ and $N \equiv 0 \pmod{p^2}$, which means that if some secret factor p hides e ($e|p - 1$) and p^2 divides modulus N ($N \equiv 0 \pmod{p^2}$), our proposed algorithm can recover such factor p . Although our result does not guarantee that $\gcd(e, \phi(N)) = 1$ (since we can only check the factor p : $N \equiv 0 \pmod{p^2}$), we have $\gcd(e, \phi(N)) \neq 1$ if our algorithm outputs a factor p . In addition, let $N = \prod_{i=1}^n p_i^{z_i}$, for the case of $\min\{z_1, \dots, z_n\} \geq 2$, we can identify whether $\gcd(e, \phi(N)) = 1$ or not, which improve [10]’s result to $\gamma > \frac{1}{8}$.

Proof. At first we get the value of integer r from the exponent e . Then we have

$$\phi(N) = \prod_{i=1}^n p_i^{z_i-1}(p_i - 1)$$

Let us focus on the case $e|(p_i - 1)$ and $N \equiv 0 \pmod{p_i^r}$ for some i . Denote that $p := p_i$, there exists an $x_0 \in \mathbb{Z}$ s.t.

$$ex_0 + 1 = p$$

Next our goal is to find x_0 , which is a small root of the polynomial equation $f(x) = ex + 1 \pmod{p}$ ($N \equiv 0 \pmod{p^r}$).

In order to run Coppersmith’s algorithm, we have to know the parameter β : the bitsize of unknown divisor. However, we do not know the exact value of β here. To overcome this problem, we give a lemma that can be used to check whether $e|p - 1$ for some p ($N \equiv \pmod{p^r}$) in a specific range. This following lemma can be regard as an extension of Lemma 5 of [10].

Lemma 3. *Let N be an integer of unknown factorization with divisor p^r : $p \geq N^\beta$ ($\beta \in (0, 1]$) and $r \geq 1$. Further, let $e = N^\gamma$ with $e|p - 1$. Then there is an algorithm that, given (N, e, β, μ) , outputs p in time $\mathcal{O}(\mu^{-7} \log^2 N)$ provided that*

$$p \leq N^{r\beta^2 + \gamma - \mu}$$

If this algorithm can not find a non-trivial factor of N , it outputs \perp .

Proof. We can easily get the result from Theorem 2. We have $e|p - 1$, then we get the polynomial $f(x) = ex + 1$ has the root x_0 modulo p . By multiplying e^{-1} modulo N , we can get a monic polynomial. Since $p \leq N^{r\beta^2 + \gamma - \mu}$, we have

$$x_0 = \frac{p - 1}{e} < \frac{N^{r\beta^2 + \gamma - \mu}}{N^\gamma} = N^{r\beta^2 - \mu}$$

Therefore, we can recover x_0 by Theorem 2 in time $\mathcal{O}(\mu^{-7} \log^2 N)$. For every candidate of x_0 , we check whether $\gcd(ex_0 + 1, N)$ gives us the divisor p . Since the number of the candidate is bounded by $\mathcal{O}(\mu^{-1}r)$, this can be done in $\mathcal{O}(\mu^{-1}r \log^2 N)$, which can be negligible compared to the running time of Copersmith’s algorithm.

Using Lemma 3, we can check whether $e|p - 1$ for some p ($N \equiv \text{mod } p^r$) in the range $[N^\beta, N^{r\beta^2 - \mu + \gamma}]$. However, it is not enough, our target range is $p \in [e, N^{\frac{1}{r}}]$, much larger than the search range of Lemma 3. Next we apply [10]’s idea to solve the above problem.

At first, we set $r\beta^2 - \mu + \gamma = \frac{1}{r}$, which implies $\beta = \frac{\sqrt{1 - r(\gamma - \mu)}}{r}$. Then we can check p in the interval $[N^{\frac{\sqrt{1 - r(\gamma - \mu)}}{r}}, N^{\frac{1}{r}}]$ using Lemma 3. If we can not find such p , we turn to the range $[e, N^{\frac{\sqrt{1 - r(\gamma - \mu)}}{r}}]$, then we use Lemma 3 again, and set $r\beta^2 - \mu + \gamma = \frac{\sqrt{1 - r(\gamma - \mu)}}{r}$, which defines a new lower bound β . We then repeat this process.

To summary, we cover the target range by a sequence of intervals $[N^{\beta_1}, N^{\beta_0}], \dots, [N^{\beta_n}, N^{\beta_{n-1}}]$ where the β_i are defined by the recurrence relation

$$\beta_{i+1} = \max\left\{\sqrt{\frac{\beta_i - (\gamma - \mu)}{r}}, \gamma\right\} \quad \text{with } \beta_0 = \frac{1}{r}.$$

Here we suppose that

$$\frac{1}{4r} < \gamma - \mu < \gamma < \min\left\{\frac{1}{4(r - 1)}, 1\right\}$$

We have to prove that the number of invocations of Lemma 3 is polynomial. At first, we show by induction that the sequence of the β_i is monotone decreasing.

It is obvious that $\beta_1 < \beta_0$ since $\gamma < \frac{1}{r} = \beta_0$ and $\sqrt{\frac{\beta_0 - (\gamma - \mu)}{r}} < \sqrt{\frac{\beta_0}{r}} = \beta_0$. We now show that if $\beta_i \leq \beta_{i-1}$ for all $i \leq m$, we have $\beta_{m+1} \leq \beta_m$.

Since $\beta_m \leq \beta_{m-1}$, we have

$$\frac{\beta_m - (\gamma - \mu)}{r} \leq \frac{\beta_{m-1} - (\gamma - \mu)}{r}$$

which implies

$$\sqrt{\frac{\beta_m - (\gamma - \mu)}{r}} \leq \sqrt{\frac{\beta_{m-1} - (\gamma - \mu)}{r}}$$

That means

$$\max\left\{\sqrt{\frac{\beta_m - (\gamma - \mu)}{r}}, \gamma\right\} \leq \max\left\{\sqrt{\frac{\beta_{m-1} - (\gamma - \mu)}{r}}, \gamma\right\}$$

Thus $\beta_{m+1} \leq \beta_m$.

Since the sequence of $\{\beta_i\}$ is monotone decreasing and bounded below by γ , it converges. Now we investigate the length of interval $[\beta_i, \beta_{i-1}]$. Define a function $\Delta(\beta_{i-1}) = \beta_{i-1} - \beta_i \geq 0$, which is the length of the i^{th} interval. We have

$$\Delta(\beta_{i-1}) = \beta_{i-1} - \beta_i = \beta_{i-1} - \sqrt{\frac{\beta_{i-1} - (\gamma - \mu)}{r}}$$

We calculate the first two derivations of $\Delta(\beta)$ as follows

$$\Delta'(\beta) = 1 - \frac{1}{2\sqrt{r}}(\beta - (\gamma - \mu))^{-\frac{1}{2}}$$

and

$$\Delta''(\beta) = \frac{1}{4\sqrt{r}}(\beta - (\gamma - \mu))^{-\frac{3}{2}} > 0$$

It is clear that $\Delta(\beta)$ achieves its minimum at the point $\beta^{(0)} = \frac{1}{4r} + \gamma - \mu$ (when $\Delta'(\beta) = 0$). And the length of interval $\Delta(\beta)$ is of size at least

$$\Delta(\beta^{(0)}) = \gamma - \mu - \frac{1}{4r}$$

Let

$$k := \left\lceil \frac{\frac{1}{r} - \gamma}{\gamma - \mu - \frac{1}{4r}} \right\rceil + 1$$

That means after the number of k steps, the sequence β_i stabilize at the point $\beta_k = \gamma$.

Therefore, if we run the algorithm of Lemma 3 at most k times, we can test the entire range $[e, N^{\frac{1}{r}}]$.

Before we give the running time of our algorithm, we would like to discuss the choice of the parameter μ . Since $\epsilon := \gamma - \frac{1}{4r}$, we have the condition $\frac{1}{4r} < \gamma - \mu$, thus $\mu < \gamma - \frac{1}{4r} = \epsilon$, we simply choose $\mu = \frac{1}{c}\epsilon$ where c is a positive integer. In practice, if we choose larger c , which means smaller value of μ , then the value of k is smaller, and the same time, Lemma 3 may take more running time; if c is smaller, by contrast, means larger value of k and less running time of Lemma 3. However, note that the running time of our algorithm is mainly decided by the running time of Lemma 3, thus we would like to choose $c = 2$ in practice.

At last we give the total running time of our algorithm, we have to run the algorithm of Lemma 3 at most k times, which can be bounded as

$$k := \left\lceil \frac{\frac{1}{r} - \gamma}{\gamma - \mu - \frac{1}{4r}} \right\rceil + 1 \leq \left\lceil \frac{\frac{1}{r}}{(c-1)\mu} \right\rceil + 1 = \mathcal{O}(\mu^{-1}) = \mathcal{O}(\epsilon^{-1})$$

Table 1. Example: concrete values for $r = 2$

$r = 2$	β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7
$\gamma = 0.187500$ $\mu = 0.031250$	0.500000	0.414578	0.359394	0.314396	0.281199	0.249949	0.216447	0.187500
$\gamma = 0.187500$ $\mu = 0.015625$	0.500000	0.405046	0.341446	0.291179	0.244238	0.190214	0.187500	
$\gamma = 0.200000$ $\mu = 0.020000$	0.500000	0.400000	0.331662	0.275374	0.218374	0.200000		

and each iteration takes time $\mathcal{O}(\mu^{-7} \log^2 N)$, finally we obtain the total running time of our algorithm

$$\mathcal{O}(\mu^{-8} \log^2 N) = \mathcal{O}(\epsilon^{-8} \log^2 N)$$

In Table 1, we give three examples to show the concrete values of $\{\beta_i\}$. Note that if μ is smaller, we can take fewer steps to search the whole target range.

4 Conclusion

In this paper, we extend Kakvi et al.’s work by considering the case with generalized moduli $N = \prod_{i=1}^n p_i^{z_i}$. Surprisingly, when $\min\{z_1, \dots, z_n\} \geq 2$, we show that it can be efficiently decided whether the RSA function defines a permutation over \mathbb{Z}_N^* or not even for the prime $e < N^{\frac{1}{4}}$. Our result can be viewed as an extension of Kakvi et al.’s result.

Acknowledgments. The authors thank anonymous reviewers for their valuable comments. This work was supported in part by National Natural Science Foundation of China (Nos. 61632020, 61472416, 61772520, 61702505, 61732021), Key Research Project of Zhejiang Province (No. 2017C01062), Fundamental theory and cutting edge technology Research Program of Institute of Information Engineering, CAS (No. Y7Z0321102, Y7Z0341103), National Cryptography Development Fund (MMJJ20170115), JST CREST (No. JPMJCR14D6), and JSPS KAKENHI (No.16H02780).

References

1. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_37
2. Bellare, M., Yung, M.: Certifying cryptographic tools: the case of trapdoor permutations. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 442–460. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_31
3. Bellare, M., Yung, M.: Certifying permutations: noninteractive zero-knowledge based on any trapdoor permutation. *J. Cryptol.* **9**(3), 149–166 (1996)

4. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_28
5. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptol.* **10**(4), 233–260 (1997)
6. Dwork, C., Naor, M.: Zaps and their applications. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, Redondo Beach, California, USA, 12–14 November 2000, pp. 283–293 (2000)
7. Giuliani, K.: Factoring polynomials with rational coefficients. *Math. Ann.* **216**(4), 515–534 (1982)
8. Goldreich, O.: Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: the state of the art. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation. LNCS, vol. 6650, pp. 406–421. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22670-0_28
9. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0024458>
10. Kakvi, S.A., Kiltz, E., May, A.: Certifying RSA. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 404–414. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_25
11. Lu, Y., Zhang, R., Peng, L., Lin, D.: Solving linear equations modulo unknown divisors: revisited. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 189–213. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_9
12. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_5
13. May, A.: Using LLL-reduction for solving RSA and factorization problems. In: Nguyen, P., Vallée, B. (eds.) The LLL Algorithm - Survey and Applications, pp. 315–348. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-02295-1_10
14. Nguên, P.Q., Stehlé, D.: Floating-point LLL revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_13
15. Schridde, C., Freisleben, B.: On the validity of the Φ -hiding assumption in cryptographic protocols. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 344–354. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_21
16. Takagi, T.: Fast RSA-type cryptosystem modulo p^kq . In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 318–326. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055738>

Full Paper Session XII: Formal Analysis and Cryptanalysis



Attack Trees in Isabelle

Florian Kammüller^{1,2}(✉)

¹ Middlesex University London, London, UK

f.kammueLLer@mdx.ac.uk

² Technische Universität Berlin, Berlin, Germany

Abstract. In this paper, we present a proof theory for attack trees. Attack trees are a well established and useful model for the construction of attacks on systems since they allow a stepwise exploration of high level attacks in application scenarios. Using the expressiveness of Higher Order Logic in Isabelle, we succeed in developing a generic theory of attack trees with a state-based semantics based on Kripke structures and CTL. The resulting framework allows mechanically supported logic analysis of the meta-theory of the proof calculus of attack trees and at the same time the developed proof theory enables application to case studies. A central correctness and completeness result proved in Isabelle establishes a connection between the notion of attack tree validity and CTL. The application is illustrated on the example of a healthcare IoT system and GDPR compliance verification.

1 Introduction

Attack trees are an intuitive and practical formal method to analyse and quantify attacks on security and privacy. They are very useful to identify the steps an attacker takes through a system when approaching the attack goal. In this paper, we provide a proof calculus to analyse concrete attacks using a notion of attack validity. We define a state based semantics with Kripke models and the temporal logic CTL in the proof assistant Isabelle [1] using its Higher Order Logic (HOL)¹. We prove the correctness and completeness (adequacy) of attack trees in Isabelle with respect to the model. This generic Kripke model enriched with CTL does not use an action based model contrary to the main stream. Instead, our model of attack trees leaves the choice of the actor and action model to the application. Nevertheless, using the genericity of Isabelle, proofs and concepts of attack trees carry over to the application model.

There are many approaches to provide a mathematical and formal semantics as well as constructing verification tools for attack trees but we pioneer the use of a Higher Order Logic (HOL) tool like Isabelle that allows proof of meta-theory – like adequacy of the semantics – and verification of applications – while being ensured that the formalism is correct.

Attack trees have been investigated on a theoretical level quite intensively; various extensions exist, e.g., to attack-defense trees and probabilistic or timed

¹ In the following, we refer to Isabelle/HOL simply as Isabelle.

attack trees. This paper uses preliminary work towards an Isabelle proof calculus for attack trees presented at a workshop [2] but accomplishes the theoretical foundation by defining a formal semantics and providing the proof of correctness and completeness and thereby establishing a feasible link for application verification. The novelty of this proof theoretic approach to attack tree verification is to take a logical approach from the very beginning by imposing the rigorous expressive Isabelle framework as the technical and semantical spine. This approach brings about a decisive advantage which is beneficial for a successful application of the attack tree formalism and consequently also characterizes our contribution: meta-theory and application verification are possible simultaneously. Since Higher Order Logic allows expressing concepts like attack trees within the logic, it enables reasoning *about* objects like attack trees, Kripke structures, or the temporal logic CTL in the logic (meta-theory) while at the same time *applying* these formalised concepts to applications like infrastructures with actors and policies (object-logics).

This paper presents the following contributions.

- We provide a proof calculus for attack trees that entails a notion of refinement of attack trees and a notion of valid attack trees.
- Validity of attack trees can be characterized by a recursive function in Isabelle which facilitates evaluation and permits code generation.
- The main theorems show the correctness and completeness of attack tree validity with respect to the state transition semantics based on Kripke structures and CTL. This meta-theorem not only provides a proof for the concepts but is part of the proof calculus for applications.
- The Isabelle attack tree formalisation is applied to the case study of formalising GDPR properties over infrastructures.

In this paper, we first introduce the underlying Kripke structures and CTL logic (Sect. 2). Next, we present attack trees and their notion of refinement (Sect. 3). The notion of validity is given by the proof calculus in Sect. 4 followed by the central theorem of correctness and completeness (adequacy) of attacks in Sect. 5 including a high level description of the proof. Section 6 shows how the framework is applied to analyse an IoT healthcare system and Sect. 7 extends by labelled data to enable GDPR compliance verification. We then discuss, consider related work, and draw conclusions (Sect. 8). All Isabelle sources are available online [3].

2 Kripke Structures and CTL in Isabelle

Isabelle is a generic Higher Order Logic (HOL) proof assistant. Its generic aspect allows the embedding of so-called object-logics as new theories on top of HOL. An Isabelle theory introduces new types, constants, and definitions. Similar to a programming language, keywords indicate these items: for example, the keyword `datatype` marks the beginning of a new type definition or `definition` introduces a new constant and its definition. In this paper, we will provide

more detailed explanation of the concrete syntax when and where it is used. Object-logics, when added to Isabelle using constant and type definitions, constitute a so-called *conservative extension*. This means that no inconsistency can be introduced; conceptually, new types are defined as subsets of existing types and properties are proved using a one-to-one relationship to the new type from properties of the existing type. New properties within a theory can be subsequently proved in interaction with the user over any model defined in terms of the new types, constants, and definitions. These properties are introduced with the keyword `lemma` or `theorem` depending on their significance. Following the statement of such a property, the Isabelle tool expects the user to provide step-by-step instructions how to prove the property using existing lemmas and theorems from underlying theories or previously proved properties. In principle, proof can be a tedious process and requires expert knowledge. However, there are sophisticated proof tactics available to support reasoning: simplification, first-order resolution, and special macros to support arithmetic amongst others. The use of HOL has the advantage that it enables expressing even the most complex application scenarios, conditions, and logical requirements and HOL simultaneously enables the analysis of the meta-theory.

In this work, we make additional use of the class concept of Isabelle that allows an abstract specification of a set of types and properties to be instantiated later. We use it to abstract from states and state transition in order to create a generic framework for Kripke structures, CTL, and attack trees. Using classes the framework can then be applied to arbitrary object-logics that have a notion of state and state transition by instantiation. Isabelle attack trees have been designed as a generic framework meaning that the formalised theories can be applied to various applications. Figure 1 illustrates how the Isabelle theories in our framework are embedded into each other.

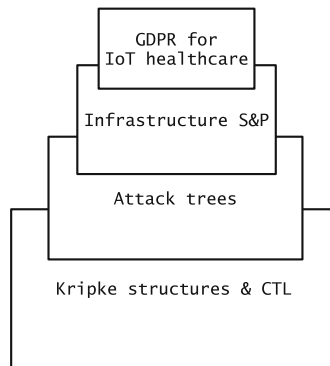


Fig. 1. Generic framework for attack trees embeds applications.

2.1 Kripke Structures and CTL

The expressiveness of Higher Order Logic (HOL) allows formalizing the notion of Kripke structures as sets of states and a generic transition relation over those in Isabelle. In addition, the branching time temporal logic CTL is embedded conservatively into HOL using Isabelle's fixpoint definitions for the CTL operators. We apply Kripke structures and CTL to model state based systems and analyse properties under dynamic state changes. Snapshots of systems are the states on which we define a state transition. Temporal logic is then employed to express and prove security and privacy properties over these system models.

In Isabelle, the system states and their transition relation are defined as a class called `state` containing an abstract constant `state_transition`. It introduces the syntactic infix notation $I \rightarrow I'$ to denote that system state I and I' are in this relation over an arbitrary (polymorphic) type σ .

```
class state =
fixes state_transition :: [ $\sigma$  :: type,  $\sigma$ ]  $\Rightarrow$  bool ("_  $\rightarrow$  _")
```

The above class definition introduces a new type class in Isabelle. This class `state` lies in the base class `type` which is encoded by the type judgment $::$ `type` following σ . All types in class `state` are characterized by having a constant called `state_transition`, with concrete infix syntax \rightarrow . The σ is a polymorphic type variable used here to represent an arbitrary type σ in the class `state` imposing that the state transition must be a predicate over two elements of σ , that is, a relation.

This type class `state` lifts Kripke structures and CTL to a general level allowing various instantiations to concrete state transition relations that will be provided using inductive definitions. The definition of such an inductive relation is given by a set of specific rules which are, however, part of an application like infrastructures (Sect. 6). Branching time temporal logic CTL is defined in general over Kripke structures with arbitrary state transitions and can later be applied to suitable theories, like infrastructures.

Based on the generic state transition \rightarrow of the type class `state`, the CTL-operators EX and AX express that property f holds in some or all next states, respectively.

```
AX  $f$   $\equiv$  { $s$ . { $f0$ .  $s \rightarrow f0$ }  $\subseteq f$ }
EX  $f$   $\equiv$  { $s$ .  $\exists f0 \in f$ .  $s \rightarrow f0$ }
```

The CTL formula $AG\ f$ means that on all paths branching from a state s the formula f is always true (G stands for 'globally'). It can be defined using the Tarski fixpoint theory by applying the greatest fixpoint operator.

```
AG  $f$   $\equiv$   $\text{gfp}(\lambda Z. f \cap AX\ Z)$ 
```

The function input to `gfp` is from a set of states Z to the set of states $f \cap AX\ Z$ transforms properties to properties – a so-called predicate transformer.

In a similar way, the other CTL operators are defined. The formal Isabelle definition of what it means that formula f holds in a Kripke structure M can be

stated as: the initial states of the Kripke structure `init M` need to be contained in the set of all states `states M` that imply `f`. This is stated in the definition of the operator `check` with infix syntax \vdash .

$$M \vdash f \equiv \text{init } M \subseteq \{s \in \text{states } M. s \in f\}$$

The left side of a definition fixes parameters, here, a Kripke structure `M` and a set of states `f`, which can be used on the right side of the \equiv to define its meaning. In this definition, we use the set theory operators for subset relation \subseteq , set membership \in , and set collection $\{x. P\ x\}$ denoting the set of all `x` with property `P` for any predicate `P`. These set notations are provided in the rich Isabelle theory database. In an application, the set of states of the Kripke structure will be defined as the set of states reachable by the infrastructure state transition from some initial state, say `example_scenario`.

$$\text{example_states} \equiv \{I. \text{example_scenario} \rightarrow^* I\}$$

The relation \rightarrow^* is the reflexive transitive closure $(_)^*$ – a generic operator supplied by the Isabelle theory library – applied to the relation \rightarrow . Again using here the generic theory library, automatically provides a realm of theorems about the reflexive transitive closure and powerful automated proof support for our application.

The `Kripke` constructor combines a state set, here the one of our dummy example, a set of initial states, here just the singleton set containing `example_scenario`, and a state transition relation, here \rightarrow , into a Kripke structure that we name here `example_Kripke`.

$$\text{example_Kripke} \equiv \text{Kripke } \text{example_states} \ \{\text{example_scenario}\} \ \rightarrow$$

In Isabelle, the concept of sets and predicates coincide². Thus a `property` is a predicate over states which is equal to a set of states. For example, we can then try to prove that there is a path (E) to a state in which the property eventually holds (in the Future) by starting the following proof in Isabelle.

$$\text{example_Kripke} \vdash \text{EF } \text{property}$$

Since `property` is a set of states, and the temporal operators are predicate transformers, that is, transform sets of states to sets of states, the resulting `EF property` is also a set of states – and hence again a property.

3 Attack Trees and Refinement

Attack Trees [4] are a graphical language for the analysis and quantification of attacks. If the root represents an attack, its children represent the sub-attacks. Leaf nodes are the basic attacks; other nodes of attack trees represent sub-attacks. Sub-attacks can be alternatives for reaching the goal (disjunctive node) or they must all be completed to reach the goal (conjunctive node). Figure 2 is

² In general, this is often referred to as *predicate transformer semantics*.

an example of an attack tree taken from a textbook [4] illustrating the attack of opening a safe. Nodes can be adorned with attributes, for example costs of attacks or probabilities which allows quantification of attacks (not used in the example).

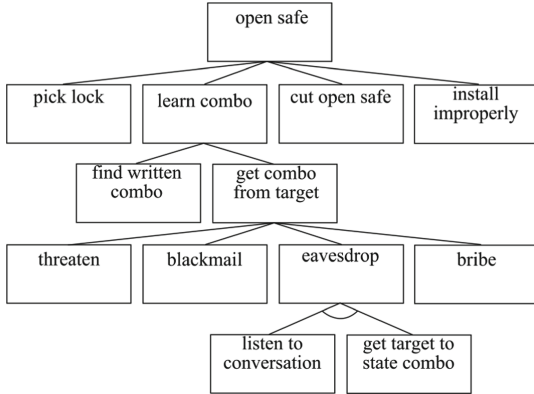


Fig. 2. Attack tree example illustrating disjunctive nodes for alternative attacks refining the attack “open safe”. Near the leaves there is also a conjunctive node “eavesdrop”.

3.1 Attack Tree Datatype in Isabelle

The following datatype definition `attree` defines attack trees. Isabelle allows recursive datatype definitions similar to the programming languages Haskell or ML. A datatype is given by a “|” separated sequence of possible cases each of which consists of a constructor name, the types of inputs to this constructor, and optionally a pretty printing syntax definition. The simplest case of an attack tree is a base attack. The principal idea is that base attacks are defined by a pair of state sets representing the initial states and the *attack property* – a set of states characterized by the fact that this property holds for them. Attacks can also be combined as the conjunction or disjunction of other attacks. The operator \oplus_{\vee} creates or-trees and \oplus_{\wedge} creates and-trees. And-attack trees $l\oplus_{\wedge}^s$ and or-attack trees $l\oplus_{\vee}^s$ consist of a list of sub-attacks – again attack trees.

```

datatype ( $\sigma :: \text{state}$ )attree =
  BaseAttack ( $\sigma \text{ set}$ )  $\times$  ( $\sigma \text{ set}$ ) (" $\mathcal{N}(\_)$ ")
| AndAttack ( $\sigma \text{ attree}$ )list ( $\sigma \text{ set}$ )  $\times$  ( $\sigma \text{ set}$ ) (" $\_ \oplus_{\wedge}^{(\_)}$ ")
| OrAttack ( $\sigma \text{ attree}$ )list ( $\sigma \text{ set}$ )  $\times$  ( $\sigma \text{ set}$ ) (" $\_ \oplus_{\vee}^{(\_)}$ ")
  
```

The attack goal is given by the pair of state sets on the right of the operator \mathcal{N} , \oplus_{\vee} or \oplus_{\wedge} , respectively. A corresponding projection operator is defined as the function `attack`.

```

primrec attack :: ( $\sigma :: \text{state}$ ) attree  $\Rightarrow$  ( $\sigma \text{ set}$ )  $\times$  ( $\sigma \text{ set}$ )
where
  attack (BaseAttack b) = b
| attack (AndAttack as s) = s
| attack (OrAttack as s) = s

```

Functions over datatypes can be given with `primrec` which enables defining an operator, here `attack`, by listing the possible cases and describing the semantics using simple equations and pattern matching on the left side.

3.2 Attack Tree Refinement

When we develop an attack tree, we proceed from an abstract attack, given by an attack goal, by breaking it down into a series of sub-attacks. This proceeding corresponds to a process of *refinement*. Therefore, as part of the attack tree calculus, we provide a notion of attack tree refinement. This can be done elegantly by defining an infix operator \sqsubseteq . The intuition of developing an attack tree from the root to the leaves is illustrated in Fig. 3. The example attack tree on the left side has a leaf that is expanded by the refinement into an and-attack with two steps. Formally, we define the semantics of the refinement operator by an inductive definition for the constant \sqsubseteq , that is, the smallest predicate closed under the set of specified rules.

```

inductive refines_to :: [ $(\sigma :: \text{state}) \text{ attree}, \sigma \text{ attree}$ ]  $\Rightarrow$  bool ("_  $\sqsubseteq$  _")
where
  refI: [ [ A = (1 @ [N(s1,s2)] @ 1'')  $\oplus_{\wedge}^{(s0,s3)}$ ; A' = 1'  $\oplus_{\wedge}^{(s1,s2)}$ ;
           A'' = 1 @ 1' @ 1''  $\oplus_{\wedge}^{(s0,s3)}$  ]  $\Longrightarrow$  A  $\sqsubseteq$  A''
| ref_or: [ [ as  $\neq$  [];  $\forall$  A'  $\in$  set(as). A  $\sqsubseteq$  A'  $\wedge$  attack A = s
             ]  $\Longrightarrow$  A  $\sqsubseteq$  as  $\oplus_{\vee}$  s
| ref_trans: [ [ A  $\sqsubseteq$  A'; A'  $\sqsubseteq$  A'' ]  $\Longrightarrow$  A  $\sqsubseteq$  A''
| ref_refl : A  $\sqsubseteq$  A

```

The rule `refI` captures the intuition expressed in Fig. 3: a sequence of leaves in an and-subtree can be refined by replacing a single leaf by a new subsequence (the `@` is the list append in Isabelle). Rule `ref_or` describes or-attack refinement. To refine a node into an or-attack, all sub-trees in the or-attack list need to refine the parent node. The remaining rules define \sqsubseteq as a pre-order on sub-trees of an attack tree: it is reflexive and transitive.

Refinement of attack trees defines the stepwise process of expanding abstract attacks into more elaborate attacks only syntactically. There is no guarantee that the refined attack is possible if the abstract one is, nor vice-versa. We need to provide a semantics for attacks in order to judge whether such syntactic refinements represent possible attacks. To this end, we now formalise the semantics of attack trees by a proof theory.

4 Proof Calculus

A valid attack, intuitively, is one which is fully refined into fine-grained attacks that are feasible in a model. The general model we provide is a Kripke structure,

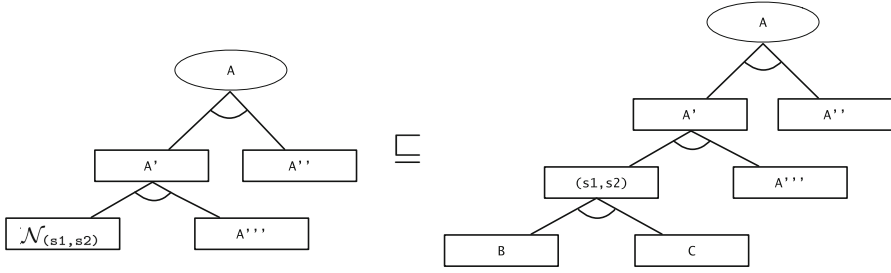


Fig. 3. Attack tree example illustrating refinement of an and-subtree.

i.e., a set of states and a generic state transition. Thus, feasible steps in the model are single steps of the state transition. We call them valid base attacks. The composition of sequences of valid base attacks into and-attacks yields again valid attacks if the base attacks line up with respect to the states in the state transition. If there are different valid attacks for the same attack goal starting from the same initial state set, these can be summarized in an or-attack.

```

fun is_attack_tree :: [( $\sigma$  :: state) attree]  $\Rightarrow$  bool ("|-")
where
  att_base:  $\vdash \mathcal{N}_s = \forall x \in \text{fst } s. \exists y \in \text{snd } s. x \rightarrow y$ 
| att_and:  $\vdash (\text{As} :: (\sigma :: \text{state attree list})) \oplus_{\wedge}^s =$ 
  case As of
    []  $\Rightarrow$  ( $\text{fst } s \subseteq \text{snd } s$ )
  | [a]  $\Rightarrow \vdash a \wedge \text{attack } a = s$ 
  | a # l  $\Rightarrow \vdash a \wedge \text{fst}(\text{attack } a) = \text{fst } s$ 
     $\wedge \vdash 1 \oplus_{\wedge}^{(\text{snd}(\text{attack } a), \text{snd}(s))}$ 
| att_or:  $\vdash (\text{As} :: (\sigma :: \text{state attree list})) \oplus_{\vee}^s =$ 
  case As of
    []  $\Rightarrow$  ( $\text{fst } s \subseteq \text{snd } s$ )
  | [a]  $\Rightarrow \vdash a \wedge \text{fst}(\text{attack } a) \supseteq \text{fst } s \wedge \text{snd}(\text{attack } a) \subseteq \text{snd } s$ 
  | a # l  $\Rightarrow \vdash a \wedge \text{fst}(\text{attack } a) \subseteq \text{fst } s \wedge \text{snd}(\text{attack } a) \subseteq \text{snd } s$ 
     $\wedge \vdash 1 \oplus_{\vee}^{(\text{fst } s - \text{fst}(\text{attack } a), \text{snd } s)}$ 

```

More precisely, the different cases of the validity predicate are distinguished by pattern matching over the attack tree structure.

- A base attack $\mathcal{N}_{(s_0, s_1)}$ is valid if from all states in the pre-state set s_0 we can get with a single step of the state transition relation to a state in the post-state set s_1 . Note, that it is sufficient for a post-state to exist for each pre-state. After all, we are aiming to validate attacks, that is, possible attack paths to some state that fulfills the attack property.
- An and-attack $\text{As} \oplus_{\wedge}^{(s_0, s_1)}$ is a valid attack if either of the following cases holds:
 - empty attack sequence As : in this case all pre-states in s_0 must already be attack states in s_1 , i.e., $s_0 \subseteq s_1$;
 - attack sequence As is singleton: in this case, the singleton element attack a in $[a]$, must be a valid attack and it must be an attack with pre-state s_0 and post-state s_1 ;

- otherwise, \mathbf{As} must be a list matching $\mathbf{a} \# \mathbf{l}$ for some attack \mathbf{a} and tail of attack list \mathbf{l} such that \mathbf{a} is a valid attack with pre-state identical to the overall pre-state $\mathbf{s0}$ and the goal of the tail \mathbf{l} is $\mathbf{s1}$ the goal of the overall attack. The pre-state of the attack represented by \mathbf{l} is $\mathbf{snd}(\mathbf{attack} \ \mathbf{a})$ since this is the post-state set of the first step \mathbf{a} .
- An or-attack $\mathbf{As} \oplus_{\bigvee}^{(s0,s1)}$ is a valid attack if either of the following cases holds:
 - the empty attack case is identical to the and-attack above: $\mathbf{s0} \subseteq \mathbf{s1}$;
 - attack sequence \mathbf{As} is singleton: in this case, the singleton element attack \mathbf{a} must be a valid attack and its pre-state must include the overall attack pre-state set $\mathbf{s0}$ (since \mathbf{a} is singleton in the or) while the post-state of \mathbf{a} needs to be included in the global attack goal $\mathbf{s1}$;
 - otherwise, \mathbf{As} must be a list $\mathbf{a} \# \mathbf{l}$ for an attack \mathbf{a} and a list \mathbf{l} of alternative attacks. The pre-states can be just a subset of $\mathbf{s0}$ (since there are other attacks in \mathbf{l} that can cover the rest) and the goal states $\mathbf{snd}(\mathbf{attack} \ \mathbf{a})$ need to lie all in the overall goal state set $\mathbf{s1}$. The other or-attacks in \mathbf{l} need to cover only the pre-states $\mathbf{fst} \ \mathbf{s} - \mathbf{fst}(\mathbf{attack} \ \mathbf{a})$ (where $-$ is set difference) and have the same goal $\mathbf{snd} \ \mathbf{s}$.

The proof calculus is thus completely described by one recursive function. This is a major improvement to the inductive definition provided in the preliminary workshop paper [2] that inspired this paper. Our notion of attack tree validity is more concise hence less prone to stating inconsistent definitions and still allows to infer properties important for proofs. The increase of consistency is because other important or useful algebraic properties can be derived from the recursive function definition. Note, that preliminary experiments on a proof calculus for attack trees in Isabelle [2] used an inductive definition that had a larger number of rules than the three cases we have in our recursive function definition `is_attack_tree`. The earlier inductive definition integrated a fair number of properties as inductive rules which are now proved from the three cases of `is_attack_tree`.

It might appear that Kripke semantics interprets conjunction as sequential (ordered) conjunction instead of parallel (unordered) conjunction. However, this is not the case: the ordering of events or actions is implicit in the states. Therefore, any kind of interleaving (or true parallelism) of state changing actions is possible. This is inserted as part of the application – for example in the `Infras` definition of the state transition in Sect. 6. There the order of actions between states depends on the pre-states and post-states only.

Given the proof calculus, the notion of validity of an attack tree can be used to identify valid refinements already at a more abstract level. The notion \sqsubseteq_V denotes that the refinement of the attack tree on the left side is to a valid attack tree on the right side.

$$A \sqsubseteq_V A' \equiv (A \sqsubseteq A' \wedge \vdash A')$$

Taking this one step further, we can say that an abstract attack tree is valid if there is a valid attack tree it refines to.

$$\vdash_V A \equiv (\exists A'. A \sqsubseteq_V A')$$

Thereby, we have achieved what we initially wanted: to state that an abstract attack tree A is actually a valid attack tree, we can conjecture $\vdash_V A$. This results in the proof obligation of finding a valid attack tree $\vdash A'$ such that $A \sqsubseteq A'$. For practical purposes, the following lemma implements this method.

lemma `ref_valI`: $A \sqsubseteq A' \implies \vdash A' \implies \vdash_V A$

We are going to use this method on the case study in Sect. 6.2 for the attack tree analysis.

5 Correctness and Completeness of Attack Trees

The novel contribution of this paper is to equip attack trees with a Kripke semantics. Thereby, a valid attack tree corresponds to an attack sequence. The following correctness theorem provides this: if A is a valid attack on property s starting from initial states described by I , then from all states in I there is a path to the set of states fulfilling s in the corresponding Kripke structure.

theorem `AT_EF`: $\vdash A :: (\sigma :: \text{state}) \text{ attree} \implies (I, s) = \text{attack } A$
 $\implies \text{Kripke } \{t . \exists i \in I. i \rightarrow^* t\} I \vdash \text{EF } s$

It is not only an academic exercise to prove this theorem. Since we use an embedding of attack trees into Isabelle, this kind of proof about the embedded notions of attack tree validity \vdash and CTL formulas like `EF` is possible. At the same time, the established relationship between these notions can be applied to case studies. Consequently, if we apply attack tree refinement to spell out an abstract attack tree for attack s into a valid attack sequence, we can apply theorem `AT_EF` and can immediately infer that `EF s` holds.

Theorem `AT_EF` also extends to validity of abstract attack trees. That is, if an “abstract” attack tree A can be refined to a valid attack tree, correctness in CTL given by `AT_EF` applies also to the abstract tree.

theorem `ATV_EF`: $\vdash_V A :: (\sigma :: \text{state}) \text{ attree} \implies (I, s) = \text{attack } A$
 $\implies \text{Kripke } \{t . \exists i \in I. i \rightarrow^* t\} I \vdash \text{EF } s$

The inverse direction of theorem `AT_EF` is a completeness theorem: if states described by predicate s can be reached from a finite nonempty set of initial states I in a Kripke structure, then there exists a valid attack tree for the attack (I, s) .

theorem `Completeness`: $I \neq \{\} \implies \text{finite } I \implies$
 $\text{Kripke } \{t . \exists i \in I. i \rightarrow^* t\} I \vdash \text{EF } s$
 $\implies \exists A :: (\sigma :: \text{state}) \text{ attree}. \vdash A \wedge (I, s) = \text{attack } A$

Correctness and Completeness are proved in Isabelle within the theory `AT.thy` [3]. The interactive proofs including auxiliary lemmas consist of nearly 1200 lines of proof commands. However, we have proved these theorems once for all. Owing to the modular organisation of our framework they are meta-theoretic theorems usable for any object logic that models an application.

6 Application to Infrastructures, Policies, and Actors

The Isabelle Infrastructure framework supports the representation of infrastructures as graphs with actors and policies attached to nodes. These infrastructures are the *states* of the Kripke structure.

The transition between states is triggered by non-parametrized actions `get`, `move`, `eval`, and `put` executed by actors. Actors are given by an abstract type `actor` and a function `Actor` that creates elements of that type from identities (of type `string`). Policies are given by pairs of predicates (conditions) and sets of (enabled) actions.

```
type_synonym policy = ((actor  $\Rightarrow$  bool)  $\times$  action set)
```

Actors are contained in an infrastructure graph.

```
datatype igrph = Lgrph (location  $\times$  location)set
                location  $\Rightarrow$  identity set
                actor  $\Rightarrow$  (string set  $\times$  string set)
                location  $\Rightarrow$  (string  $\times$  acond)
```

An `igrph` has just one constructor function `Lgrph`. It constructs an `igrph` from a set of location pairs representing the topology of the infrastructure as a graph of nodes and a list of actor identities associated to each node (location) in the graph. The third component of an `igrph` associates actors to a pair of string sets by a pair-valued function whose first range component is a set describing the credentials in the possession of an actor and the second component is a set defining the roles the actor can take on. More importantly in this context, the fourth component of an `igrph` assigns locations to a pair of a string that defines the state of the component and an element of type `acond`. This type `acond` is defined as a set of labelled data representing a condition on that data. Corresponding projection functions for each of these components of an `igrph` are provided; they are named `gra` for the actual set of pairs of locations, `agra` for the actor map, `cgra` for the credentials, and `lgra` for the state of a location and the data at that location.

Infrastructures are given by the following datatype that contains an infrastructure graph of type `igrph` and a policy given by a function that assigns local policies over a graph to all locations of the graph.

```
datatype infrastructure = Infrastructure igrph
                            igrph  $\Rightarrow$  location  $\Rightarrow$  policy set
```

There are projection functions `graphI` and `delta` when applied to an infrastructure return the graph and the policy, respectively. Policies specify the expected behaviour of actors of an infrastructure. They are defined by the `enables` predicate: an actor `h` is enabled to perform an action `a` in infrastructure `I`, at location `l` if there exists a pair `(p,e)` in the local policy of `l` (`delta I l` projects to the local policy) such that the action `a` is a member of the action set `e` and the policy predicate `p` holds for actor `h`.

`enables I l h a` $\equiv \exists (p,e) \in \text{delta I l. } a \in e \wedge p h$

We now flesh out the abstract state transition introduced in Sect. 2.1 by defining an inductive relation \rightarrow_n for state transition between infrastructures. This state transition relation is dependent on actions but also on enabledness and the current state of the infrastructure. For illustration purposes we consider the rule for `get_data` only (see the complete source code [3] for full details of other rules)³.

```
get_data : G = graphI I  $\implies$  a @G l  $\implies$ 
  enables I l' (Actor a) get  $\implies$ 
  I' = Infrastructure
    (Lgraph (gra G)(agra G)(cgra G)
      ((lgra G)(l := (fst (lgra G l),
        snd (lgra G l)  $\cup$  {new}))))))
    (delta I)
 $\implies$  I  $\rightarrow_n$  I'
```

The new state `I'` of the infrastructure can be reached from `I` if the actor `h` is in location `l`, action `get` is enabled for `h` at location `l`. Under those preconditions, data `new` can be added to the actor's current location `l` formalised here using the function update `:=` for the fourth component `lgra G` of the infrastructure's `igraph`.

6.1 Application Example from IoT Healthcare

The example of an IoT healthcare systems is from the CHIST-ERA project SUCCESS [5] on monitoring Alzheimer's patients. Figure 4 illustrates the system architecture where data collected by sensors in the home or via a smart phone helps monitoring bio markers of the patient. The data collection is in a cloud based server to enable hospitals (or scientific institutions) to access the data which is controlled via the smart phone. We show the encoding of the `igraph` for this system architecture in the Infrastructure model.

```
ex_graph  $\equiv$  Lgraph {(home, cloud), (sphone, cloud), (cloud,hospital)}
  (\lambda x. if x = home then {'Patient''}
    else (if x = hospital then {'Doctor''} else {}))
  ex_creds ex_locs
```

The identities `Patient` and `Doctor` represent patients and their doctors; double quotes `''s''` indicate strings in Isabelle/HOL. The global policy is 'only the patient and the doctor can access the data in the cloud':

```
fixes global_policy::[infrastructure, identity]  $\Rightarrow$  bool
defines global_policy I a  $\equiv$  a  $\notin$  gdpr_actors  $\longrightarrow$ 
   $\neg$ (enables I cloud (Actor a) get)
```

Local policies are represented as a function over an `igraph G` that additionally assigns each location of a scenario to its local policy given as a pair of

³ We deliberately omit here the DLM constraints for illustration purposes (see below).

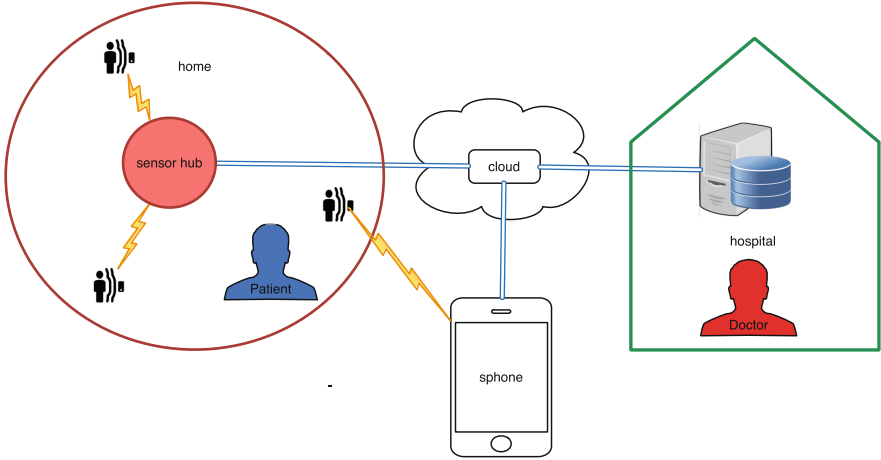


Fig. 4. IoT healthcare monitoring system for SUCCESS project

requirements to an actor (first element of the pair) in order to grant him actions in the location (second element of the pair). The predicate $@_G$ checks whether an actor is at a given location in the graph G .

```

local_policies G ≡
(λ x. if x = home then {(λ y. True, {put,get,move,eval})}
  else (if x = sphone then
    {(λ y. has G (y, 'PIN')}, {put,get,move,eval})}
    else (if x = cloud then {(λ y. True, {put,get,move,eval})}
      else (if x = hospital then
        {((λ y. (∃ n. (n @_G hospital) ∧ Actor n = y ∧
          has G (y, 'skey'))), {put,get,move,eval})} else {}))))
    
```

6.2 Using Attack Tree Calculus

Since we consider a predicate transformer semantics, we use sets of states to represent properties. For example, the attack property is given by the following set sgdpr .

$$\text{sgdpr} \equiv \{x. \neg (\text{global_policy } x \text{ 'Eve'})\}$$

The attack we are interested in is to see whether for the scenario

$$\text{gdpr_scenario} \equiv \text{Infrastructure ex_graph local_policies}$$

from the initial state $\text{Igdpr} \equiv \{\text{gdpr_scenario}\}$, the critical state sgdpr can be reached, i.e., is there a valid attack $(\text{Igdpr}, \text{sgdpr})$?

To set up this question as a proof goal, we can now use the meta-theory for valid abstract attack trees developed in Sect. 4 which allows setting out from this abstract attack.

$$\vdash_V [\mathcal{N}_{(\text{Igdpr}, \text{sgdpr})}] \oplus_{\wedge}^{(\text{Igdpr}, \text{sgdpr})}$$

We can then prove that there is a refinement to an and-attack where the set GDPR is an intermediate state where **Eve** accesses the cloud.

$$\begin{aligned} & [\mathcal{N}_{(\text{Igdpr}, \text{sgdpr})}] \oplus_{\wedge}^{(\text{Igdpr}, \text{sgdpr})} \\ \sqsubseteq & \\ & [\mathcal{N}_{(\text{Igdpr}, \text{GDPR})}, \mathcal{N}_{(\text{GDPR}, \text{sgdpr})}] \oplus_{\wedge}^{(\text{Igdpr}, \text{sgdpr})} \end{aligned}$$

We can then finish the proof by deriving that the second refined attack is a valid and-attack using the attack tree proof calculus.

$$\vdash [\mathcal{N}_{(\text{Igdpr}, \text{GDPR})}, \mathcal{N}_{(\text{GDPR}, \text{sgdpr})}] \oplus_{\wedge}^{(\text{Igdpr}, \text{sgdpr})}$$

For the Kripke structure

$$\text{gdpr_Kripke} \equiv \text{Kripke} \{I. \text{gdpr_scenario} \rightarrow^* I\} \text{Igdpr}$$

we can alternatively apply the Correctness theorem **AT_EF** to immediately derive from the previous result the following CTL statement.

$$\text{gdpr_Kripke} \vdash \text{EF sgdpr}$$

This application of the meta-theorem of Correctness of attack trees saves us proving the CTL formula tediously by exploring the state space. However, a more convincing case for using the Correctness and Completeness meta-theorems is given next when we consider how to improve the access control on data to guarantee GDPR level security and privacy.

7 Data Protection by Design for GDPR Compliance

7.1 General Data Protection Regulation (GDPR)

On 26th May 2018, the GDPR has become mandatory within the European Union and hence also for any supplier of IT products. Breaches of the regulation will be fined with penalties of 20 Million EUR. For this paper, we use the final proposal [6] as our source. Despite the relatively large size of the document of 209 pages, the technically relevant portion for us is only about 30 pages (pp. 81–111, Chaps. I to III, Sect. 3). In summary, Chap. III specifies that the controller must give the data subject *read access* (1) to any information, communications, and “meta-data” of the data, e.g., retention time and purpose. In addition, the system must enable *deletion of data* (2) and restriction of processing.

An invariant condition for data processing resulting from these Articles is that the system *functions* must *preserve* any of the access rights of personal data (3).

7.2 Security and Privacy by Labeling Data

The Decentralised Label Model (DLM) [7] introduced the idea to label data by owners and readers. We pick up this idea and formalize a new type to encode the owner and the set of readers.

```
type_synonym dlm = actor × actor set
```

Labelled data is then just given by the type $\text{dlm} \times \text{data}$ where **data** can be any data type. Additional meta-data, like retention time and purpose, can be encoded as part of this type **data**. We omit these detail here for conciseness of the exposition.

Using labeled data, we can now express the essence of Article 4 Paragraph (1): ‘personal data’ means any information relating to an identified or identifiable natural person (‘data subject’). Since we have a more constructive system view, we express this by defining the owner of a data item **d** of type **dlm** as the actor that is the first element in the pair that is the first of the pair **d**. Then, we use this function to express the predicate “owns”.

```
definition owner :: dlm × data ⇒ actor
where owner d ≡ fst(fst d)
```

```
definition owns :: [igraph, location, actor, dlm × data] ⇒ bool
where owns G l a d ≡ owner d = a
```

The introduction of a similar function for readers projecting the second element of a **dlm** label

```
definition readers :: dlm × data ⇒ actor set
where readers d ≡ snd (fst d)
```

enables specifying whether an actor may access a data item.

```
definition has_access :: [igraph, location, actor, dlm × data] ⇒ bool
where has_access G l a d ≡ owns G l a d ∨ a ∈ readers d
```

For our example of an IoT health care monitoring system, the data and its privacy access control definition is given by the parameter **ex_locs** specifying that the data 42, for example some bio marker’s value, is located in the cloud, is owned by the patient, and can be read by the doctor (‘free’ is the state of the cloud component).

```
ex_locs ≡ (λ x. if x = cloud
              then (‘free’, {(Actor ‘Patient’, {Actor ‘Doctor’}), 42})
              else (‘’, {}))
```

7.3 Privacy Preserving Functions

The labels of data must not be changed by processing: we have identified this finally as an invariant (3) resulting from the GDPR in Sect. 7. This invariant can be formalized in our Isabelle model by a type definition of functions on labeled data that preserve their labels.

```
typedef label_fun = {f :: dlm×data ⇒ dlm×data. ∀ x. fst x = fst (f x)}
```

We also define an additional function application operator \Downarrow on this new type. Then we can use this restricted function type to implicitly specify that only functions preserving labels may be applied in the definition of the system behaviour in the state transition rule for action `eval` (see [3]).

7.4 Policy Enforcement

We can now use the labeled data to encode the privacy constraints of the GDPR in the rules. For example, the `get_data` rule has now labelled data $((\text{Actor } a', \text{as}), n)$ and used the labeling in the precondition to guarantee that only entitled users can get data: Actor `a` has to be in the set of readers `as` to have this data item added to his location `l`.

```
get_data : G = graphI I ⇒ a @G l ⇒ enables I l' (Actor a) get ⇒
  ((Actor a', as), n) ∈ snd (lgra G l') ⇒ Actor a ∈ as ⇒
  I' = Infrastructure
    (Lgraph (gra G)(agra G)(cgra G)
      ((lgra G)(l := (fst (lgra G l)),
        snd (lgra G l) ∪ {((Actor a', as), new)})))
    (delta I)
  ⇒ I →n I'
```

Using the formal model of infrastructures, we can now prove privacy by design for GDPR compliance of the specified system. We can show how the properties relating to data ownership, processing and deletion can be formally captured using Kripke structures and CTL and the Infrastructure framework. As an example, consider the preservation of data ownership.

Processing Preserves Privacy. We can prove that processing preserves ownership as defined in the initial state for all paths globally (AG) within the Kripke structure and in all locations of the graph.

```
theorem GDPR_three: h ∈ gdpr_actors ⇒ l ∈ gdpr_locations ⇒
  owns (Igraph gdpr_scenario) l (Actor h) d ⇒
  gdpr_Kripke ⊢
  AG {x. ∀ l ∈ gdpr_locations. owns (Igraph x) l (Actor h) d}
```

Note, that it would not be possible to express this property in Modelcheckers (let alone prove it) since they only allow propositional logic within states. This generalisation is only possible since we use Higher Order Logic. The proof of this property is straightforward evaluation of the CTL rules.

Applying Correctness to Prove Absence of Attacks. The contraposition of the Correctness theorem grants that if $(\text{EF } f)$ does *not* hold in a Kripke structure, then there is *no* attack (I, f) for the initial states of the Kripke structure I . Since properties are expressed as sets, negation is expressed in the theorem by using the set complement $\neg P$ for the negation of property P .

$$\begin{aligned}
& h \in \text{gdpr_actors} \implies l \in \text{gdpr_locations} \implies \\
& \text{owns (Igraph gdpr_scenario) l (Actor h) d} \implies \\
& \text{attack A} = (\text{Igdpr}, \{-\{x. \forall l \in \text{gdpr_locations}. \\
& \quad \text{owns (Igraph x) l (Actor h) d}\}) \\
& \implies \neg(\vdash \text{A}::\text{infrastructure attree})
\end{aligned}$$

Proving the absence of attacks for attack trees is in general very difficult but becomes feasible owing to the meta-theorem Correctness and the possibility to interleave meta-theoretic reasoning with that in the object logic in Isabelle.

8 Conclusions

In this paper, we have presented a proof theory for attack trees in Isabelle’s Higher Order Logic (HOL). We have shown the incremental and generic structure of this framework, presented correctness and completeness results equating valid attacks to EF s formulas. The proof theory has been illustrated on an IoT healthcare infrastructure where the meta-theorem of completeness could be directly applied to infer the existence of an attack tree from CTL. The practical relevance has been demonstrated on GDPR compliance verification.

There are excellent foundations available based on graph theory [8]. They provide a very good understanding of the formalism, various extensions (like attack-defense trees [9]) and differentiations of the operators (like sequential conjunction (SAND) versus parallel conjunction [10]) and are amply documented in the literature. These theories for attack trees provide a thorough foundation for the formalism and its semantics. The main problem that adds complexity to the semantical models is the abstractness of the descriptions in the nodes. This leads to a variety of approaches to the semantics, e.g. propositional semantics, multiset semantics, and equational semantics for ADtrees [9]. The theoretical foundations allow comparison of different semantics, and provide a theoretical framework to develop evaluation algorithms for the quantification of attacks.

More practically oriented formalisations, e.g. [11], focus on an action based-approach where the attack goals are represented as labels of attack tree nodes which are actions that an attacker has to execute to arrive at the goal.

A notable exception that uses, like our approach, a state based semantics for attack trees is the recent work [12]. However, this work is aiming at assisted generation of attack trees from system models. The tool ATSyRA supports this process. The paper [12] focuses on describing a precise semantics of attack tree in terms of transition systems using “under-match”, “over-match”, and “match” to arrive at a notion of correctness. In comparison, we use additionally CTL logic to describe the correctness relation precisely. Also we use a fully formalised and proved Isabelle model.

Surprisingly, the use of an automated proof assistant, like Isabelle, has not been considered before despite its potential of providing a theory and analysis of attacks simultaneously. The essential attack tree mechanism of disjunction and conjunction in tree refinement is relatively simple. The complexity in the theories is caused by the attempt to incorporate semantics to the attack nodes

and relate the trees to actual scenarios. This is why we consider the formalisation of a foundation of attack trees in the interactive prover Isabelle since it supports logical modeling and definitions of datatypes very akin to algebraic specification but directly supported by semi-automated analysis and proof tools.

The workshop paper [2] has inspired the present work but is vastly superseded by it. The novelties are:

- Attack trees have a state based semantics formalised in the framework.
- Correctness and completeness are proved based on the formal semantics.
- The Isabelle framework is generic using type classes, that is, works for any state model. Infrastructures with actors and policies are an instantiation.
- The semantics, correctness and completeness theorems facilitate application verification.

References

1. Nipkow, T., Wenzel, M., Paulson, L.C. (eds.): LNCS, vol. 2283. Springer, Heidelberg (2002). <https://doi.org/10.1007/3-540-45949-9>
2. Kammüller, F.: A proof calculus for attack trees in Isabelle. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 3–18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_1
3. Kammüller, F.: Isabelle infrastructure framework with IoT healthcare s&p application (2018). <https://github.com/flokam/IsabelleAT>
4. Schneier, B.: Secrets and Lies: Digital Security in a Networked World. Wiley, Hoboken (2004)
5. CHIST-ERA, Success: Secure accessibility for the internet of things (2016). <http://www.chistera.eu/projects/success>
6. Union, E.: The EU general data protection regulation (GDPR). Accessed 20 Mar 2018, proposal for a Regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation) [first reading] - Analysis of the final compromise text with a view to agreement, Brussels, 15 December 2015. <http://www.eugdpr.org>
7. Myers, A.C., Liskov, B.: Complete, safe information flow with decentralized labels. In: IEEE Symposium on Security and Privacy. IEEE (1999)
8. Kordy, B., Piètre-Cambacédés, L., Schweitzer, P.: Dag-based attack and defense modeling: don't miss the forest for the attack trees. *Comput. Sci. Rev.* **13–14**, 1–38 (2014)
9. Kordy, B., Mauw, S., Radomirovic, S., Schweitzer, P.: Attack-defense trees. *J. Log. Comput.* **24**(1), 55–87 (2014)
10. Jhavar, R., Kordy, B., Mauw, S., Radomirović, S., Trujillo-Rasua, R.: Attack trees with sequential conjunction. In: Federrath, H., Gollmann, D. (eds.) SEC 2015. IAICT, vol. 455, pp. 339–353. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18467-8_23
11. Aslanyan, Z., Nielson, F., Parker, D.: Quantitative verification and synthesis of attack-defence scenarios. In: CSF 2016. IEEE (2016)
12. Audinot, M., Pinchinat, S., Kordy, B.: Is my attack tree correct? In: Foley, S.N., Gollmann, D., Sneekenes, E. (eds.) ESORICS 2017. LNCS, vol. 10492, pp. 83–102. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66402-6_7



Automated Verification of Noninterference Property

Fan Zhang^{1,2}, Cong Zhang¹, Mingdi Xu³, Xiaoli Liu⁴, Fangning Hu²,
and HanChieh Chao^{1,5,6}✉

¹ Mathematics and Computer Science School, Wuhan Polytechnic University,
Wuhan, China

hcchao@gmail.com

² School of Computer Science and Electrical Engineering, Jacobs University
Bremen, Bremen, Germany

³ Wuhan Digital and Engineering Institute, Wuhan, China

⁴ College of Information Science and Technology/College of Cyber Security,
Jinan University, Guangzhou 510632, China

⁵ Department of Electrical Engineering, National Dong Hwa University,
Hualien, Taiwan

⁶ Department of Computer Science and Information Engineering,
National Ilan University, Ilan, Taiwan

Abstract. Noninterference is an important information flow model that is widely applied in building secure information systems. Although the noninterference model itself has been thoroughly investigated, verifying the noninterference property in an efficient and automated manner remains an open problem. In this study, we explore the noninterference verification problem from the perspective of the state-equivalence relations between two automata running synchronously. Our results are as follows. (1) To the best of our knowledge, we are the first to propose a recursive form of the necessary and sufficient condition of noninterference. We also for the first time disclose the fact that Rushby's definition of noninterference model can also be formalized as a bi-simulation (over two automata) (2) We present an automated noninterference verification algorithm. The algorithm can finish the verification within $O(|S|^2 * |D|)$, where $|D|$ is the number of security domains and $|S|$ is the number of states. The time-complexity of our algorithm is the best among other existing studies.

Keywords: Intransitive noninterference · Information flow
Noninterference verification

1 Introduction

An information system is usually composed of trusted and untrusted components. Therefore, it is crucial to protect information from flowing from trusted components (with a high security level) to untrusted components (with a low security level) so as to maintain the confidentiality of the trusted components. Among the different technologies applied, “noninterference” is one of the most important approaches to achieving this requirement. Informally, noninterference addresses an information-flow

attack, in which an adversary obtains secrets that he is not authorized to occupy, by observing the running of an information system and making subtle deductions.

The first definition of noninterference was given by Goguen and Meseguer [1], who presented the formalization of an information flow and its causal relationships in the context of transitive information flow policies. However, it was subsequently noted that transitive policies are insufficient to characterize system architectures, and intransitive policies have often been required. For example, consider a security system with three processes, i.e., P_1 receives a plaintext from a keyboard, P_2 encrypts the information coming from P_1 , and P_3 sends the encrypted information to the network. In this case, the high-level domain (P_1) apparently cannot directly interfere with the low-level domain (P_3); otherwise, confidential information would be leaked. To address this issue, an intermediate downgrader (P_2) is needed. Information is permitted to first flow from the high-level domain to the downgrader domain, and then from the downgrader domain to the low-level domain ($P_1 \rightsquigarrow P_2 \wedge P_2 \rightsquigarrow P_3$), but not directly from the high-level domain to the low-level domain ($(P_1 \rightsquigarrow P_3)$), thereby motivating the use of intransitive policies.

Although Haigh and Young introduced a formal model for intransitive policies in [2], their formalization was found questionable [3]. Rushby [3] then presented the first perfect formalization by refining Haigh and Young's work. Note that it is essential to establish verification techniques to guarantee that a designed system indeed satisfies noninterference, and thus, Rushby introduced the "unwinding theorem" in [3]. "Unwinding technology" was the first effort at building a theoretical base for verification of the noninterference property, but it requires significant human ingenuity to define the unwinding relations between security domains. Engineers apparently prefer fully automated verification techniques, rather than techniques requiring highly specialized expertise. After Rushby, another breakthrough was the work of Hadj-Alouane et al. [4], who were the first to propose a necessary and sufficient condition-based noninterference verification algorithm. Nevertheless, the time complexity of their algorithm is the double exponential $O(2^{|S|*2^{|D|}})$, which is too high for practical use. In 2011, Eggert et al. investigated the time- and space-complexities of several noninterference definitions, including P-Security (Transitive-Policy Security), IP-Security (Intransitive-Policy Security), TA-Security, and TO-Security [5]. However, they merely analyzed the time- and space-complexities without presenting any detailed algorithms. More importantly, their definition of IP-Security was actually different from that of Rushby. Specifically, Eggert et al. focused on the occurrences of high-level actions and their order of occurrence [5], whereas Rushby focused on the system states and their equivalence relations on the security domains [3]. Therefore, although Eggert et al. claimed that IP-Security can be verified in polynomial time [5], it is still unknown of time- and space-complexity boundaries for Rushby's definition of intransitive noninterference.

Actually, to date, significantly few studies have been conducted on automated verification techniques for the intransitive noninterference property [5], and constructing a practical and automated algorithm for verifying the intransitive noninterference property remains an open problem. This paper attempts to address the above issues, and our contributions are as follows:

- (1) We propose a necessary and sufficient condition of intransitive noninterference based on the state-equivalence relation of two automata in a recursive manner.

To the best of our knowledge, we are the first to propose the recursive form of necessary and sufficient condition of intransitive noninterference.

(2) We present an automated noninterference verification algorithm.

The time complexity of our algorithm is the best among existing studies. Moreover, if a verification fails, our algorithm can precisely point out the exact reason why the system does not satisfy noninterference, which is helpful for security-bug repairs.

2 Problem Statement

For a clearer understanding, herein we use an example to state the problem at hand.

In a noninterference model, there are two key elements: **an automaton** modeling an information system M whose security is to be investigated, as well as **interference/noninterference relations** (also equivalently called **security policies**) indicating how information is allowed to flow between security domains within M [3].

With respect to the automaton, a noninterference model does not specify how an information system M should be modeled as an automaton (in fact, there are many standard methods, and researchers can choose whatever method they want), but simply assumes the automaton has been obtained. Without loss of generality, suppose Fig. 1 is the automaton that M is modeled as. In Fig. 1, s_0 (oval rectangle) is the initial state, s_1, s_2, \dots, s_{12} (single circles) are the intermediate states, and $s_{f0}, s_{f1}, \dots, s_{f4}$ (double circles) are the terminal states. The directed arrows between states indicate how the system transforms from one state into another, and the letters over the directed arrows, i.e., a, b, \dots, g , are the actions that trigger the state transformations.

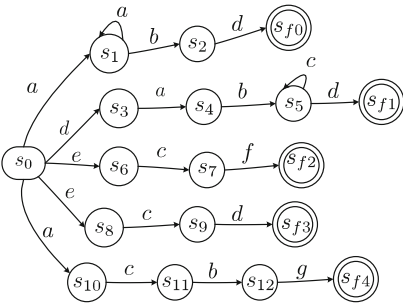
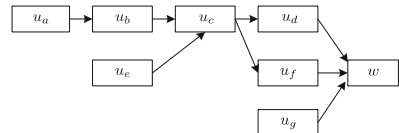


Fig. 1. The automaton of information system M



Here w is a special security domain, from which Low-level observers (attackers) can issue actions to observe runs to an information system and thereby deduce something. In this paper, we call w "the final security domain", because for any information "finally" flows into w , we must ensure that nothing confidential is leaked to (observed by) Low-level observers.

Fig. 2. Intransitive security policies

With respect to the interference/noninterference relations (i.e., security policies), they determine how information can flow between security domains within M . Without loss of generality, suppose the interference relations are $u_a \rightsquigarrow u_b \rightsquigarrow u_c \rightsquigarrow u_d \rightsquigarrow w$, $u_e \rightsquigarrow u_c \rightsquigarrow u_f \rightsquigarrow w$, and $u_g \rightsquigarrow w$. Any other two security domains are of noninterference with each other except for the above interference relations. The meaning of security policies is straightforward, for example, $u_e \rightsquigarrow u_c \rightsquigarrow u_f \rightsquigarrow w$ means that information can

flow starting from u_e , in turn passing through u_c and u_f , and finally ending at w . Figure 2 shows the security policies.

In a noninterference model, any action must belong to a security domain. Therefore, in Fig. 2 we use $u_{k(k=a,b,\dots,g)}$ to refer to the security domain that action k belongs to. Note that w in Fig. 2 is a special security domain, because from w Low-level observers (i.e., attackers) can issue actions to observe results of paired-runs of M (specifically, one is the actual run, the other is the ideal run under the protection of security policies. see definition 11), try to find differences between the paired-runs, and thereby deduce something that they are not authorized to know. Hence, in order to prevent Low-level observers from learning anything confidential from the information flowing to w , we must ensure that Low-level observers cannot observe any difference between paired-runs of M , no matter how they observe M (by choosing actions from w). This is the nature of noninterference, and the formal definition please refer to definition 10.

Now the challenge is, if given the automaton and security policies (for example, akin to Figs. 1 and 2) of an information system M , how can we determine whether M is secure (i.e., no confidential information is leaked to Low-level observers) in an automated and efficient way? Unfortunately, however, this is still an open problem thus far.

In the following, we will present our approach to address this issue.

3 Basic Definitions of Noninterference

Note that a transitive policy is simply a special case of an intransitive policy [3], and we hereafter refer to intransitive policy-based noninterference as simply “noninterference.”

Definition 1. A system (or machine) M consists of the following: (1) a set S of states, where $s_0 \in S$ is the sole initial state; (2) a set A of actions, where actions could be “inputs,” “commands,” or “instructions” to be performed by M ; (3) a set B of behaviors, where behaviors are sequences of actions concatenated by \circ ; (4) a set O of outputs, where O is composed of all storage locations that can be observed by users; (5) a set D of domains, where D comprises all security domains in M ; (6) a function $dom : A \rightarrow D$, which returns the domain that an action $a \in A$ is associated with, where each action a belongs only to a unique domain $dom(a)$; (7) a function $step : S \times A \rightarrow S$, which describes how M transforms from the pre-state $s_i \in S$, after performing an action a , to the post-state $s_{i+1} \in S$; (8) a function $behcon : S \times A \rightarrow O$, which returns outputs when using an action a to observe M in a state s ; (9) a function $exec : S \times B \rightarrow S$, which describes how M transforms from the pre-state s_i , after performing a behavior $\alpha = a_i \circ \dots \circ a_j \in B$, to the post-state s_j . If we use Λ to denote an empty behavior, $exec$ can be defined in a right recursive manner, i.e., $\begin{cases} exec(s, \Lambda) = s, \\ exec(s, a \circ \alpha) = exec(step(s, a), \alpha); \end{cases}$ (10) *direct* interference relation \rightsquigarrow and *direct* noninterference relation $\not\rightsquigarrow$. \rightsquigarrow and $\not\rightsquigarrow$ are two complement relations, where $p \rightsquigarrow q$ denotes information can *directly* flow from domain p to domain q , whereas $p \not\rightsquigarrow q$ denotes information cannot *directly* flow from p to q ; (11) *Indirect* interference relation \approx . In practice, we sometimes encounter the situation that information can not *directly*

flow from p to q , but *indirectly* flow from p to q . With respect to this situation, we say $p \overset{\approx}{\rightsquigarrow} q$. For example, consider a scenario where a confidential military plan is sent out over a network. The standard workflow is, firstly, the military plan is input through a keyboard (domain u_m), then the plaintext of the plan is encrypted by an encryption machine (domain u_e), and finally the encrypted plan is sent out over a network (domain u_n). The military plan must not be sent out over the network without being encrypted. Therefore, the security policies of the above scenario is $u_m \rightsquigarrow u_e \rightsquigarrow u_n \wedge u_m \not\rightsquigarrow u_n$. In this scenario, though information cannot *directly* flow from u_m to u_n ($u_m \not\rightsquigarrow u_n$), it can still *indirectly* flow to u_n via the intermediate downgrader u_e ($u_m \rightsquigarrow u_e \rightsquigarrow u_n$). Formally, for $\overset{\approx}{\rightsquigarrow}$ we have: $u \rightsquigarrow w$ **iff** $\exists v_1, v_2, \dots, v_{n(1 \leq n)}$. $u \rightsquigarrow v_1 \wedge \dots \wedge v_n \rightsquigarrow w \wedge u \not\rightsquigarrow w$.

In general, we use letters \dots, s, t, \dots to denote states, letters a, b, c, \dots to denote actions, and Greek letters α, β, \dots to denote behaviors.

To depict a system state, internal structures are needed. Definition 2 introduces the notations of *structured state*.

Definition 2. A machine M has a *structured state* if the following exist: (1) a set N of names that comprises all of the object names; (2) a set V of values; (3) a content function $contents : S \times N \rightarrow V$, which returns the value $v \in V$ of object $n \in N$ in the system state s ; (4) an observation function $observe : D \rightarrow \mathcal{P}(N)$ and an alteration function $alter : D \rightarrow \mathcal{P}(N)$, which return object names that can be observed and altered, respectively, where \mathcal{P} denotes a power set.

Definition 3. Reference Monitor Assumptions (RMAs). An access control policy is enforced accurately if the following three conditions are satisfied. Note that in this paper, we follow [3] to use \supset instead of \rightarrow to denote an implication.

- **Rule 1:** Two states s and t are said to satisfy the state-equivalence relation $\overset{u}{\rightsquigarrow}$ if and only if their object names are observed to have identical values. Formally,

$$s \overset{u}{\rightsquigarrow} t \text{ **iff** } \forall n \in observe(u). contents(s, n) = contents(t, n).$$

- **Rule 2:** When an action a transforms the system from a pre-state to a post-state, the new value of every changed object must depend solely on the values that can be observed by domain $dom(a)$. Formally,

$$s \overset{dom(a)}{\rightsquigarrow} t \wedge [contents(step(s, a), n) \neq contents(s, n) \vee contents(step(t, a), n) \neq contents(t, n)] \\ \supset contents(step(s, a), n) = contents(step(t, a), n)$$

- **Rule 3:** $dom(a)$ must be authorized the alter access before an action a can change the value of an object n . Formally,

$$contents(step(s, a), n) \neq contents(s, n) \supset n \in alter(dom(a)).$$

Definition 4. *output consistent* and *step consistent*.

$$\text{output consistent: } s \overset{dom(a)}{\rightsquigarrow} t \supset behcon(s, a) = behcon(t, a),$$

step consistent: $s \stackrel{u}{\sim} t \supset \text{step}(s, a) \stackrel{u}{\sim} \text{step}(t, a)$.

Definition 5. *Interference Source set* and function $IS : B \times D \rightarrow \mathcal{P}(D)$, where \mathcal{P} is the power set. IS can be formally defined in a right recursive manner, i.e., $IS(\Lambda, w) = \{w\}$, where Λ is the empty sequence, and

$$IS(a \circ \alpha, w) = \begin{cases} \{dom(a)\} \cup IS(\alpha, w) & \text{if } \exists v. v \in IS(\alpha, w) \wedge dom(a) \rightsquigarrow v \\ IS(\alpha, w) & \text{otherwise} \end{cases}.$$

Intuitively, IS returns all security domains that can directly or indirectly interfere with domain w , so we call the resulting set $IS(\alpha, w)$ “*Interference Source set*”.

Definition 6. Function $wexpected : B \times D \rightarrow B$. For any behavior $\alpha \in B$, function $wexpected$ maintains all actions whose domains can directly or indirectly interfere with domain w , and meanwhile deletes all otherwise actions, thus forming an expected behavior. Formally, we have $wexpected(\Lambda, w) = \Lambda$, and

$$wexpected(a \circ \alpha, w) = \begin{cases} a \circ wexpected(\alpha, w) & \text{if } dom(a) \in IS(a \circ \alpha, w) \\ \Lambda \circ wexpected(\alpha, w) & \text{otherwise} \end{cases}$$

Definition 7. *Domain set equivalence relation* $\stackrel{C}{\approx}$. $s \stackrel{C}{\approx} t$ iff $\forall u \in C. s \stackrel{u}{\sim} t$.

Definition 8. *Weakly step consistent*. Formally,

$$dom(a) \rightsquigarrow u \wedge s \stackrel{dom(a)}{\sim} t \wedge s \stackrel{u}{\sim} t \supset \text{step}(s, a) \stackrel{u}{\sim} \text{step}(t, a).$$

Intuitively, when an action a interferes with a domain u ($dom(a) \rightsquigarrow u$), those names visible to u change in a way that depends on those same names, plus those names visible to $dom(a)$. Since all of the aforementioned names have the identical values ($s \stackrel{u}{\sim} t \wedge s \stackrel{dom(a)}{\sim} t$), those names visible to u will be altered by a to the identical values as well, i.e., $\text{step}(s, a) \stackrel{u}{\sim} \text{step}(t, a)$ (By rule 1 of Definition 3).

Definition 9. *Locally respects* \rightsquigarrow . $dom(a) \rightsquigarrow u \supset s \stackrel{u}{\sim} \text{step}(s, a)$.

Intuitively, if the security domain that an action a belongs to is of noninterference with a domain u ($dom(a) \rightsquigarrow u$), then action a does not alter any of names visible to u , i.e., $s \stackrel{u}{\sim} \text{step}(s, a)$.

4 Automated Verification of Noninterference Property

Definition 10. [3] A system M satisfies noninterference if and only if

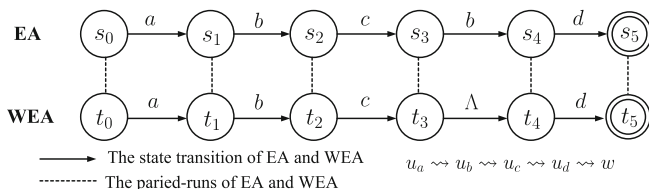
$$\forall \gamma \forall a. \text{behcon}(\text{exec}(s_0, \gamma), a) = \text{behcon}(\text{exec}(s_0, wexpected(\gamma, w)), a) \quad (1)$$

where $\gamma \in B$, $a \in A$, $w = dom(a)$.

Rushby regarded the whole Eq. (1) as a automaton, and proposed the “unwinding theorem” by induction on the length of γ . Different from Rushby, we regard the left and right sides of Eq. (1) as two automata, i.e., “EA” and “WEA” (see Definition 11), respectively. EA depicts the actual runs of M, and WEA depicts the ideal runs of M under the protection of security policies. Attackers observe the paired-runs of EA and WEA, and compare them (function *behcon* returns what attackers can observe) to find differences so as to deduce something that they are not allowed to know. If paired-runs of M are always indistinguishable, then attackers can learn nothing, and thus M is said to satisfy noninterference [3].

Note that Rushby pointed out that γ and a in Eq. (1) are read as universally quantified (Definition 2 of [3]), i.e., $\forall\gamma$ and $\forall a$. Here we need to simply explain the essence of “universally quantified”. Initially, we believe it is impossible in the real world to determine all possible attack actions issued by attackers. For example, taking the “Row Hammer” attack (shorted for “RH” hereafter) as an example, before the “RH” attack is discovered, what is the action a corresponding to “RH”? Specifically, since no one knows details of the “RH”, then no one knows how to alter the structured states of M (i.e., names, values, and so on, see Definition 2) so as to launch “RH”. In other words, the action a corresponding to “RH” cannot be constructed, which means a is in fact unknown to humans. More generally, any attack that has not yet been discovered by humans is not in the action set A , even if these attacks do exist. Therefore, for $\forall a \in A$, a is limited to attacks that have already been known to humans.

In the following, we present in detail how to automatically verify the noninterference property based on Eq. (1). For ease of understanding, we use Fig. 3, a simple automaton which has only one single behavior $\alpha = a \circ b \circ c \circ b \circ d$, to explain intuitions of definitions in this section, before presenting our technology. Note that in reality, an automaton modeled from an information system M is far more complicated than the example in Fig. 3. However, Fig. 3 is sufficient to demonstrate the nature of noninterference and explain intuitions of our definitions. Without loss of generality, suppose in Fig. 3 the security policy is $u_a \rightsquigarrow u_b \rightsquigarrow u_c \rightsquigarrow u_d \rightsquigarrow w$, and any other two



The nature of noninterference is that, EA and WEA (which corresponds to the left and right sides of equation (1), respectively) always maintain the state-equivalence relations on the security domains that in the future will (directly or indirectly) interfere with w .

Fig. 3. Example for the explanation of our technology

security domains are of noninterference with each other except for the aforementioned interference relations.

In the following, we give intuitions of our technology based on Fig. 3.

(1) Intuitions for EA and WEA.

In a noninterference model, an information system M is modeled as an automaton. This automaton corresponds to the left side of Eqs. (1), and describes the actual runs of M , hence we call it “EA (Equivalent Automaton, see definition 11)”.

For every behavior α in EA, we can correspondingly calculate its theoretically expected behavior $\beta = wexpected(\alpha, w)$, thus forming another automaton. This automaton corresponds to the right side of Eq. (1), and describes the ideal runs of M under the protection of security policies, hence we call it “WEA(Weakly Equivalent Automaton, see definition 11)”.

For example, in Fig. 3, EA has a single behavior $\alpha = a \circ b \circ c \circ b \circ d$. According to $u_a \rightsquigarrow u_b \rightsquigarrow u_c \rightsquigarrow u_d \rightsquigarrow w$, we have $\beta = wexpected(\alpha, w) = a \circ b \circ c \circ \Lambda \circ d$ and obtain WEA.

(2) Intuition for the nature of noninterference property.

Let us return to Eq. (1). Attackers perform an action a from w to observe the paired-runs of EA and WEA, trying to find differences so as to deduce confidential information. Consequently, as long as all names visible to w (i.e., visible to attackers) always have the same values in EA and WEA, then attackers can find nothing different. Furthermore, as we mentioned before, values of names visible to w change in a way that depends on those same names, plus names visible to domains that directly/indirectly interfere with w . Note that EA and WEA start operations from the same initial state $s_0 = t_0$, and thus names visible to w must be identical in the initial state. Therefore, **intuitively, as long as EA and WEA always guarantee values of names visible to domains that directly/indirectly interfere with w are identical (i.e., EA and WEA always maintain state-equivalence on domains that directly/indirectly interfere with w , see Rule 1 of Definition 3), then values of names visible to w are sure to be always identical as well. In other words, what observed from EA and WEA by attackers are always the same, and thus nothing can be learned. M is secure.** This is the intuition of Theorem 1, Detailed proof please refer to Lemmas 2, 3, and Theorem 1.

Let us go back to Fig. 3. We simulate the paired-runs of EA and WEA to illustrate the above intuition. The simulation consists of six steps.

(6.0) Initially, EA and WEA starts runs from the same initial state $s_0 = t_0$.

With respect to EA, calculate domains that in the future interference with w . EA has a single behavior $\alpha_0 = \alpha = a \circ b \circ c \circ b \circ d$. According to $u_a \rightsquigarrow u_b \rightsquigarrow u_c \rightsquigarrow u_d \rightsquigarrow w$ and the definition of IS , we have $IS(\alpha_0, w) = \{u_a, u_b, u_c, u_d\}$, which contains all the domains that in the future interfere with w . Since $s_0 = t_0$, then $s_0 \stackrel{IS(\alpha_0, w)}{\approx} t_0$.

(6.1) The EA performs an action a to transform its state from s_0 to $s_1 = step(s_0, a)$. Correspondingly, the WEA also performs the action a to transform its state from t_0 to $t_1 = step(t_0, a)$.

With respect to EA, the rest sub-behavior to be performed is $\alpha_1 = b \circ c \circ b \circ d$. Similarly, we can calculate $IS(\alpha_1, w) = \{u_b, u_c, u_d\}$, which contains all the domains that in the future interfere with w . Therefore, EA and WEA must maintain state equivalence on these domains, i.e., $s_1 \stackrel{IS(\alpha_1, w)}{\approx} t_1$, so as to protect the security of M .

(6.2) The EA performs an action b to transform its state from s_1 to $s_2 = \text{step}(s_1, b)$. Correspondingly, the WEA also performs the action b to transform its state from t_1 to $t_2 = \text{step}(t_1, b)$.

With respect to EA, the rest sub-behavior to perform is $\alpha_2 = c \circ b \circ d$. Similarly, we have $IS(\alpha_2, w) = \{u_c, u_d\}$. Therefore, EA and WEA must maintain state equivalence on $IS(\alpha_2, w)$, i.e., $s_2 \stackrel{IS(\alpha_2, w)}{\approx} t_2$, so as to protect the security of M.

(6.3) The EA performs an action c to transform its state from s_2 to $s_3 = \text{step}(s_2, c)$. Correspondingly, the WEA also performs the action c to transform its state from t_2 to $t_3 = \text{step}(t_2, c)$.

With respect to EA, the rest sub-behavior to perform is $\alpha_3 = b \circ d$. Similarly, we have $IS(\alpha_3, w) = \{u_d\}$. Therefore, EA and WEA must maintain state equivalence on $IS(\alpha_3, w)$, i.e., $s_3 \stackrel{IS(\alpha_3, w)}{\approx} t_3$, so as to protect the security of M.

(6.4) The EA performs an action b to transform its state from s_3 to $s_4 = \text{step}(s_3, b)$. Correspondingly, the WEA performs Λ to transform its state from t_3 to $t_4 = \text{step}(t_3, \Lambda) = t_3$.

With respect to EA, the rest sub-behavior to perform is $\alpha_4 = d$. Similarly, we have $IS(\alpha_4, w) = \{u_d\}$. Therefore, EA and WEA must maintain state equivalence on $IS(\alpha_4, w)$, i.e., $s_4 \stackrel{IS(\alpha_4, w)}{\approx} t_4$, so as to protect the security of M.

(6.5) The EA performs an action d to transform its state from s_4 to the final state $s_5 = \text{step}(s_4, d)$. Correspondingly, the WEA performs d to transform its state from t_4 to $t_5 = \text{step}(t_4, d)$.

With respect to EA, the rest sub-behavior to perform is $\alpha_5 = \Lambda$. According to definition of IS , we have $IS(\alpha_5, w) = \{w\}$. Therefore, EA and WEA must maintain state equivalence on $IS(\alpha_5, w)$, i.e., $s_5 \stackrel{IS(\alpha_5, w)}{\approx} t_5$, so as to protect the security of M.

The simulation of paired-runs of EA and WEA ends. From the above simulation we can see that without loss of generality, suppose EA reaches any state s_i , then WEA must correspondingly reach to a state t_i . We call (s_i, t_i) the “paired-states”. Then **an information system M is secure, if and only if for any paired-states (s_i, t_i) , (s_i, t_i) maintains state-equivalence on domains that in the future directly/indirectly interfere with w .** This is the essence of Theorem 1.

Definition 11. *EA, WEA, and buddy automata.* The left side of Eq. (1) can be abstracted as an automaton, which denotes how an information system M performs behavior γ and correspondingly transforms its states. We call this automaton the *Equivalent Automaton* (EA). Similarly, the right side of Eq. (1) can be abstracted as another automaton, which denotes how M, under the control of intransitive security policies, performs the expected behavior $w_{\text{expected}}(\gamma, \text{dom}(a))$ and correspondingly transforms its states. We call this automaton the *Weakly Equivalent Automaton* (WEA). EA and WEA are collectively called the *buddy automata*.

We stress that our approach is not a bi-simulation analysis between EA and WEA, and give the detailed discussion in Sect. 7.

Lemma 1. For any information system M , construct its EA and WEA. Then, for any behavior α in EA, and the corresponding expected behavior $\beta = wexpected(\alpha, w)$ in WEA, we have $IS(\alpha, w) = IS(\beta, w)$.

Proof. We can prove $IS(\alpha, w) \subseteq IS(\beta, w)$ and $IS(\alpha, w) \supseteq IS(\beta, w)$ based on the definition of IS and $wexpected$, and thus $IS(\alpha, w) = IS(\beta, w)$.

Lemma 1 shows that for any information system M , any behavior α in the EA (and correspondingly $\beta = wexpected(\alpha, w)$ in the WEA), α and β must have the identical interference source set $IS(\alpha, w) = IS(\beta, w)$. This is an important conclusion, because in the following we need to consider the state-equivalence relation between EA and WEA only on these identical set.

Definition 12. Execution Sub-sequence. For $\forall \gamma = a_0 \circ a_1 \circ \dots \circ a_n$, EA will in turn perform actions a_0, a_1, \dots, a_n . Each time after EA performs an action $a_{i(0 \leq i \leq n)}$, we call the remaining part an *execution sub-sequence*. In addition, we use $||\gamma||$ to denote the number of execution sub-sequences.

Taking the behavior $\forall \gamma = a_0 \circ a_1 \circ \dots \circ a_n$ in Definition 12 as an example, with the EA's performing of actions a_0, a_1, \dots, a_n , the execution sub-sequences of γ are, respectively, $\gamma_0 = \gamma = a_0 \circ \underline{a_1 \circ \dots \circ a_n} = a_0 \circ \underline{\gamma_1}$, $\gamma_1 = a_1 \circ \underline{a_2 \circ \dots \circ a_n} = a_1 \circ \underline{\gamma_2}$, $\dots, \gamma_{n-1} = a_{n-1} \circ \underline{a_n} = a_{n-1} \circ \underline{\gamma_n}$, $\gamma_n = a_n = a_n^{\circ} \underline{\Lambda} = a_n^{\circ} \underline{\gamma_{n+1}}$, and $\gamma_{n+1} = \Lambda$. Therefore, $||\gamma|| = n + 2$.

Lemma 2. Let M be a system with a structured state that satisfies the reference monitor assumption. For any behavior $\forall \gamma$ of M and any observation action $\forall a (dom(a) = w)$, without loss of generality, supposing $\forall \gamma = a_0 \circ a_1 \circ \dots \circ a_n$, we construct all of the execution sub-sequences of γ , i.e., $\gamma_0, \gamma_1, \dots, \gamma_n, \gamma_{n+1}$, where $\gamma_0 = \gamma, \gamma_{n+1} = \Lambda$ and $||\gamma|| = n + 2$. Let $N = ||\gamma|| - 2$, then we have: $\forall i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$ is a sufficient condition for M to satisfy noninterference.

In $\forall i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$, γ denotes the behavior sequence to be performed by EA, and $\gamma_{i(0 \leq i \leq N+1)}$ are the execution sub-sequences of γ . In addition, s_i and t_i denote the current state of EA and WEA, respectively. Because EA and WEA start from the same initial state, we have $s_0 = t_0$. Without loss of generality, let $\gamma_i = a_i \circ \gamma_{i+1(0 \leq i \leq N)}$, then s_{i+1} denotes the post-state $s_{i+1} = step(s_i, a_i)$ that EA will transform into after it performs a_i from the pre-state s_i ; t_{i+1} denotes the post-state that WEA will transform into after it synchronously performs a_i (in this case, $dom(a_i) \in IS(\gamma_i, w)$ and $t_{i+1} = step(t_i, a_i)$) or Λ (in this case, $dom(a_i) \notin IS(\gamma_i, w)$ and $t_{i+1} = t_i$) from the pre-state t_i .

Proof. Recursively invoking the condition of Lemma 2 $\forall i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$, we can prove that Eq. (1) holds.

(1) Initially, EA and WEA start from the same initial state $s_0 = t_0$. Thus, we have

$$s_0 \stackrel{IS(\gamma_0, w)}{\approx} t_0.$$

(2) Next, recursively invoking the condition in Lemma 2, we obtain

$$s_0 \stackrel{IS(\gamma_0, w)}{\approx} t_0 \supset s_1 \stackrel{IS(\gamma_1, w)}{\approx} t_1 \supset s_2 \stackrel{IS(\gamma_2, w)}{\approx} t_2 \supset \dots \supset s_{N+1} \stackrel{IS(\gamma_{N+1}, w)}{\approx} t_{N+1} \quad (2)$$

Note that $N = \|\gamma\| - 2 = n$, and $\gamma_{N+1} = \gamma_{n+1} = \Lambda$. From the definition of function IS , the following is provided:

$$IS(\gamma_{N+1}, w) = IS(\Lambda, w) = \{w\} \quad (3)$$

Substituting $N = n$ and $IS(\gamma_{N+1}, w) = \{w\}$ (Eq. (3)) into $s_{N+1} \stackrel{IS(\gamma_{N+1}, w)}{\approx} t_{N+1}$ (formula (2)), we have $s_{n+1} \stackrel{\{w\}}{\approx} t_{n+1}$, which is equivalent to

$$s_{n+1} \stackrel{w}{\sim} t_{n+1} \quad (4)$$

Note that system M satisfies the reference monitor assumption, and thus, by invoking “output consistent” and $s_{n+1} \stackrel{w}{\sim} t_{n+1}$ (formula (4)), we have the following

$$behcon(s_{n+1}, a) = behcon(t_{n+1}, a) \quad (dom(a) = w) \quad (5)$$

The definitions of EA and WEA give $s_{n+1} = exec(s_0, \gamma)$ and $t_{n+1} = exec(t_0, wexpected(\gamma, w))$, respectively. Substituting $s_0 = t_0$, s_{n+1} and t_{n+1} into Eq. (5), and note that the behavior sequence γ and the observation action $a(dom(a) = w)$ are arbitrary, we can immediately determine that Eq. (1) holds.

Lemma 3. Let M be a system with a structured state that satisfies the reference monitor assumption. For any behavior $\forall \gamma$ of M and any observation action $\forall a$ ($dom(a) = w$), without loss of generality, supposing $\forall \gamma = a_0 \circ a_1 \circ \dots \circ a_n$, we construct all of the execution sub-sequences of γ , i.e., $\gamma_0, \gamma_1, \dots, \gamma_n, \gamma_{n+1}$, where $\gamma_0 = \gamma$, $\gamma_{n+1} = \Lambda$ and $\|\gamma\| = n + 2$. Let $N = \|\gamma\| - 2$, then we have: $\forall i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$ is a necessary condition for M to satisfy noninterference.

The meanings of the notations in Lemma 3 are the same as those in Lemma 2.

Lemma 3 can be proved by a reduction to absurdity. The basic idea is as follows:

Assume $\exists i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \not\supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$, which can be equally written as $\exists i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \not\stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$. Based on the definition of $\stackrel{C}{\approx}$, we immediately have $\exists v_j \in IS(\gamma_{i+1}, w). s_{i+1} \not\stackrel{v_j}{\approx} t_{i+1}$. In other words, EA and WEA are non-equivalent on the security domain v_j .

Next, according to the definition of execution sub-sequence, EA will perform $\gamma_{i+1} = a_{i+1} \circ \gamma_{i+2}$. Note that we must be able to “design” a new behavior γ'_{i+1} based on γ_{i+1} , such that after EA performs γ'_{i+1} (and meanwhile WEA synchronously performs $wexpected(\gamma'_{i+1}, w)$), we have the following:

- (a) The non-equivalent relation (suppose on domain v_j , see formula (6)) between EA and WEA can never be corrected.

- (b) Worse, the non-equivalent relation between EA and WEA can be spread from v_j to other domains that can directly or indirectly interfere with w , based on the intransitive security policies.
- (c) Until finally EA and WEA reach terminal states that are non-equivalent on domain w , which means system M does not satisfy noninterference. This contradicts the premise of Lemma 3. Therefore, our assumption is wrong, and Lemma 3 holds.

Proof. Proof by contradiction.

Assume at some point that EA and WEA respectively transform their states to s_{i+1} and t_{i+1} , such that $\exists i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\not\approx} t_{i+1}$. This can be equally written as

$$\exists v_j \in IS(\gamma_{i+1}, w). s_{i+1} \stackrel{v_j}{\not\approx} t_{i+1} \tag{6}$$

According to the definition of execution sub-sequence, EA will now perform $\gamma_{i+1} = a_{i+1} \circ \gamma_{i+2}$. In the following, we will then prove that we must be able to “design” a new behavior γ'_{i+1} , such that after system M performs γ'_{i+1} , M does not satisfy noninterference, thus inducing the contradiction. Generally, the proof comprises three main steps as follows: note that $\gamma_{i+1} = a_{i+1} \circ \gamma_{i+2}$, then

STEP 1. Firstly, we can prove that once $s_{i+1} \stackrel{v_j}{\not\approx} t_{i+1}$ (formula (6)), we can design a new action a'_{i+1} to replace a_{i+1} , such that EA and WEA maintain non-equivalent relation on v_j after they both execute a'_{i+1} .

STEP 2. Furthermore, for every action in γ_{i+2} , we can also correspondingly design a new action, forming γ'_{i+2} . When system M performs the newly designed γ'_{i+2} , the non-equivalent relation between EA and WEA can be passed from v_j to other domains (in γ'_{i+2}) that can directly or indirectly interfere with w , according to the intransitive security policies.

STEP 3. Finally, when the system finishes executing γ'_{i+2} , EA and WEA will reach terminal states that are non-equivalent on w , and thus Eq. (1) does not hold. This means that there must be at least one behavior that makes system M NOT satisfy noninterference, which is contradictory to the precondition of Lemma 3. Therefore, our assumption (formula (6)) is wrong, and Lemma 3 holds.

Owing to a limited amount of space, the detailed proof is not presented here.

Theorem 1. Let M be a system with a structured state that satisfies the reference monitor assumption. For any behavior $\forall \gamma$ of M and any observation action $\forall a(dom(a) = w)$, without loss of generality, supposing $\forall \gamma = a_0 \circ a_1 \circ \dots \circ a_n$, we construct all of the execution sub-sequences of γ , i.e., $\gamma_0, \gamma_1, \dots, \gamma_n, \gamma_{n+1}$, where $\gamma_0 = \gamma, \gamma_{n+1} = \Lambda$ and $||\gamma|| = n + 2$. Let $N = ||\gamma|| - 2$, then we have: $\forall i_{(0 \leq i \leq N)}. s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$ is a necessary and sufficient condition for M to satisfy noninterference.

Algorithm 1 Verification Algorithm for Noninterference Property**Function Name:** Boolean NoninterferenceChecking(e, p, s_0, w, α, c)**Function Return:** TRUE/FALSE (system satisfies/does not satisfy noninterference)**Input Params:** e describes the EA of system M ; p indicates the intransitive security policies; s_0 is the initial state of M .**Output Params:** if TRUE is returned, $\alpha = c = \Lambda$; otherwise, α is an untrusted behavior, and c is a causal action in α that describes why α fails the verification.**Algorithm Procedure:****(1) Initialization.** Let $B = \emptyset$ be the behavior set, and then put all of the behaviors in EA into B ; let $\alpha = c = \Lambda$.**(2) Body.**/* There are two WHILE loops in the body of the algorithm, where WHILE1 is used to select behavior γ from the behavior set B , and WHILE2 performs the actual non-interference verification to γ (as the way illustrated in figure 3) */

/* If there are still behavior(s) in EA to be verified */

WHILE ($B = \emptyset$) { //WHILE1Select one behavior γ from B , and let $B = B - \{\gamma\}$. Let $\gamma_i = \gamma$ be the current behavior to be performed by EA, and s_i, t_i be the current state of EA and WEA, respectively.Initially, $s_i = t_i = s_0$, because EA and WEA always start from the same initial state s_0 each time a new behavior is verified. Let $\alpha = \gamma_i, c = \Lambda$./* If the recursive verification to γ_i is not finished */WHILE ($\gamma_i \neq \Lambda$) { //WHILE2Suppose $\gamma_i = a_i \circ \gamma_{i+1}$, calculate $\beta_i = \text{wexpected}(\gamma_i, w)$ to be performed by WEA;EA performs a_i , and transforms from its current state s_i into the next state $s_{i+1} = \text{step}(s_i, a_i)$;IF ($\beta_i = a_i \circ \beta_{i+1}$) { /* If WEA also performs a_i . This corresponds to the case that $\text{dom}(a_i)$ (directly/indirectly) interferes with w */WEA performs a_i , and transforms from its current state t_i into the next state $t_{i+1} = \text{step}(t_i, a_i)$;

} //End of IF

ELSE IF ($\beta_i = \Lambda \circ \beta_{i+1}$) { /* if WEA does not perform a_i . This corresponds to the case that $\text{dom}(a_i)$ does not (directly/indirectly) interfere with w at all*/WEA performs then empty action, and keeps its state unchanged as $t_{i+1} = t_i$;

} //End of ELSE

/* Ready to verify whether EA and WEA are state-equivalent on the interference security domain set of the execution sub-sequence γ_{i+1} */Calculate $IS(\gamma_{i+1}, w)$;IF ($s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$) { /* If EA and WEA are state-equivalent *//* Initialization for the next recursive verification (based on $\forall i_{(0 \leq i \leq N)} \cdot s_i \stackrel{IS(\gamma_i, w)}{\approx} t_i \supset s_{i+1} \stackrel{IS(\gamma_{i+1}, w)}{\approx} t_{i+1}$) */Let $\gamma_i = \gamma_{i+1}$;Let $s_i = s_{i+1}$;Let $t_i = t_{i+1}$;

CONTINUE; /* recursive verification */

```

} //End of IF
ELSE { /* If EA and WEA are not state-equivalent */
   $c = a_i$  ;
  RETURN FALSE; /* Behavior  $\alpha = \gamma_i$  of M does not satisfy noninterference and
the causal action is  $c$  */
} //End of ELSE
} // End of WHILE2
} //End of WHILE1
RETURN TRUE; /* If all behaviors in  $B$  satisfy noninterference, return true */

```

(3) Termination. If a value of TRUE is returned, then system M satisfies noninterference. Otherwise, M does not satisfy noninterference, and α is the behavior that makes M insure, and c is the causal action.

Proof. By invoking Lemmas 2 and 3, Theorem 1 immediately holds.

A running example please refer to the simulation of paired-runs of EA and WEA in Fig. 3.

5 Time Complexity

Theorem 2. Regardless of whether a deterministic or nondeterministic system, the time complexity of algorithm 1 is $O(|S|^2 * |D|)$.

Proof. Note that the nature of noninterference is as follows: The buddy automata always maintain state-equivalent on the interference source sets of the execution subsequences. Based on this observation, we provide a review of the time complexity analysis in the following:

For the time complexity, we first need to calculate the total amount of all possible state-pairs (s, t) of EA and WEA. Because EA has $|S|$ states, then WEA has $|T| \leq |S|$ states according to the definitions of *wexpected* and WEA. Thus, under the worst condition, the total amount of state-pairs (s, t) is $|S| \times |T| \leq |S|^2$. Second, we need to determine the state-equivalence relation $\overset{u}{\sim}$ of the $|S|^2$ state-pairs (s, t) on all of the $|D|$ domains, which requires a total of $|S|^2 * |D|$ comparisons. Therefore, the time complexity is $O(|S|^2 * |D|)$. This result is better than $O(|S|^3 * |A|^2 * |D|)$ (S&P'11, [5]) and $O(2^{|S| * 2^{|D|}})$ (IEEE TSMC'05, [4]).

6 Related Work

Table 1 shows the comparisons of our approach with related work.

To establish a formal model of noninterference, researchers have conducted in-depth studies using various security scenarios based on distinct modeling tools, and

Table 1. Comparisons with related work

Approach	Idea	Semantic model	Sound & complete	Algorithm or tool	Time complexity	Causal action
Rushby [3]	Unwinding theorem	SMBM	×	×	—	×
Hadj-Alouane [4]	Observability of discrete event system	SMBM	√	√	$O(2^{ S *2^{ D }})$	×
Eggert [5]	Reachability problem in directed graphs	SMBM	√	×	$O(S ^3 * A ^2 * D)$	×
Meyden [6]	Doubling constructions & model checking	SMBM	√	√	$O(S ^2 * A)$	×
Souza [7]	Language-theoretic operations	SMBM	√	×	State explosion	×
Focardi [8]	Behavior bi-simulation	PABM	×	√	State explosion	×
Ryan [9]	Model checking or theorem proof	PABM	×	×	State explosion	×
Ours	Recursive state-equivalence relation between buddy automata	SMBM	√	√	$O(S ^2 * D)$	√

^aIn Table 1, in the “Time-Complexity” column, “—” indicates “does not exist” or “not mentioned by authors”.

have proposed abundant noninterference models [8–15]. Among these models, the State Machine-Based Model (SMBM) and the Process Algebra-Based Model (PABM) are two representatives. Rushby was the first to establish a perfect SMBM of noninterference [3], and Focardi [8] and Ryan [9] were the first to systematically present a PABM of noninterference based on CCS (Calculus of Communicating System) and CSP (Communication Sequential Process), respectively. It should be noted that the semantics of SMBM and PABM are not consistent. In particular, SMBM focuses on the state transition of a system, as well as the state-equivalence relation after each state transition, whereas PABM focuses on the bi-simulation relation between the behavior sequences of a system and target security behavior specifications. Both SMBM and PABM are widely applied in practice [16–20].

Unfortunately, although extensive research has been conducted on the modeling of noninterference, there has been less work on the verification of noninterference [5]. To verify the noninterference property, Rushby [3] was the first to propose the “unwinding theorem”, which laid a solid foundation for the subsequent work of other researchers. However, unwinding theorem requires significant human ingenuity to define equivalence relations between domains, making it difficult to construct an automated verification algorithm. After Rushby, Hadj-Alouane et al. [4] proposed the first necessary and sufficient verification algorithm based on the observability of DES (Discrete Event System). The problem of [4] is that the time complexity is too high (double exponential) to be practical. In 2011, Eggert et al. [5] for the first time explored the time- and space-complexities of different noninterference definitions, including P-, IP-, TA-, and TO-Security. In [5], the authors reduced the verification of intransitive noninterference (which they call IP-Security) to the reachability problem in directed graphs, and claimed that the time complexity is $O(|S|^3 * |A|^2 * |D|)$. However, IP-Security is not consistent with Rushby’s original definition of noninterference but a variant, so the time complexity of Rushby’s definition of intransitive noninterference is still unknown. Moreover, they did not present a concrete verification algorithm. For any given deterministic system M , Meyden et al. [6] introduced the concept of doubling construction M^2 , and then reduced the noninterference property to the *safety* property based on M^2 , which finally enables the noninterference property to be checked by using standard model checking technology. The time complexity of their work seems comparable to ours; however, their work can handle no more than three security domains, H, L, D , whereas our approach can deal with any finite number of security domains. Souza et al. [7] boiled down the noninterference verification problem to the set containing problem of regular language. In [7], they first defined some language-theoretic operations on a regular language, and then established the connections between basic security predicates (BSPs) and language-theoretic operations. Thus, the verification of the noninterference property can be reduced to check whether $L_1 \subseteq L_2$, where L_1 is the regular language used to describe the system, and L_2 is the regular language for a given BSP. Nevertheless, their approach is trace-set based, rather than the stronger structure-based notion [3, 7]. Worse, [7] has to face state explosions under the worst conditions.

With respect to process algebra-based models, Focardi et al. implemented a compositional security checker called CoSeC [8, 21] on the basis of the Concurrency Workbench (CW), which can automatically check the observation equivalence relations between information flow security properties and CCS-based security specifications. Nonetheless, CoSec cannot avoid state explosions caused by the parallel composition operator. Ryan [9] proposed verifying the noninterference property by using model checking and theorem provers, but this requires highly specialized expertise, and hence it is difficult to build automated verification tools for common engineers. On the other hand, [9] cannot avoid state explosion problem, either.

7 Discussion

We stress that our work is completely different from bi-simulation based approaches. (1) Firstly, all existing bi-simulation based approaches adopt process algebra (i.e., CCS or CSP) as a tool for mathematical modeling [8, 9]. Note that process algebra studies behaviors, rather than states. Specifically, process algebra explores the bi-simulation relation between behaviors, rather than the equivalence relation between system states. Due to this observation, process algebra may not be suitable as a mathematical modeling tool for SMBM. In fact, no process algebra (and hence no bi-simulation) is used for SMBM at all [3–7]. (2) Secondly, the semantics of PABM and SMBM are different from each other (As we mentioned before, PABM focuses on behavior bi-simulation, whereas SMBM focuses on state equivalence), leading that bi-simulation approach used by PABM cannot tackle the SMBM issue of intransitive noninterference introduced by Rushby [9]. Furthermore, all verification algorithms and tools developed for PABM cannot be applied by SMBM, either. (3) Finally, a bi-simulation based approach only deals with two levels of actions, i.e., High and Low (though there may be multilevel of actions, these actions are grouped into two clusters [8]). Instead, however, our work needs to handle any number of security levels (i.e., security domains). Therefore, traditional bi-simulation approaches cannot be used by our work at all.

8 Conclusion

In this paper, we for the first time established the recursive necessary and sufficient condition of noninterference (Theorem 1), and based on which proposed a noninterference property verification algorithm. A theoretical analysis showed that the time-complexity of our algorithm is the best among existing studies.

Our work can be a step toward bridging the gap between the theoretical analysis and the practical application of the noninterference model. In the future, we will demonstrate how to put algorithm 1 into practice, for example, by using algorithm 1 to formally verify software security, to mine software vulnerability, and to construct a real-time software trust measurement framework.

Acknowledgements. We thank the anonymous reviewers for their insightful feedback. This work is sponsored in part by the National Natural Science Foundation of China (Grant No. 61502438, 61502362), Hubei Provincial Science and Technology Major Project (Grant No. 2018ABA099).

References

1. Goguen, J., Meseguer, J.: Security policies and security models. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 11–20. IEEE, Oakland (1982)
2. Haigh, J., Young, W.: Extending the noninterference version of MLS for SAT. IEEE Trans. Softw. Eng. **SE-13**(2), 141–150 (1987)

3. Rushby, J.: Noninterference, transitivity, and channel control security policies. SRI International. Technical report CSL-92-02, December 1992. <http://www.csl.sri.com/papers/csl-92-2/>
4. Hadj-Alouane, N., Lafrance, S., Lin, F., Mullins, J., Yeddes, M.: On the verification of intransitive noninterference in multilevel security. *IEEE Trans. on Syst. Man Cybern. Part B* **35**(5), 948–958 (2005)
5. Eggert, S., Meyden, R.V.D., Schnoor, H., Wilke, T.: The complexity of intransitive noninterference. In: *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 196–211. IEEE, Oakland (2011)
6. Meyden, R.V.D., Zhang, C.: Algorithmic verification of noninterference properties. *Electr. Notes Theor. Comput. Sci.* **168**(10), 61–75 (2007)
7. D’Souza, D., Raghavendra, K.R., Sprick, B.: An automata based approach for verifying information flow properties. *Electr. Notes Theor. Comput. Sci.* **135**(1), 39–58 (2005)
8. Focardi, R., Gorrieri, R.: Classification of security properties (part I: information flow). In: Focardi, R., Gorrieri, R. (eds.) *Foundations of Security Analysis and Design 2000*. LNCS, vol. 2171, pp. 331–396. Springer, Heidelberg (2000)
9. Ryan, P.Y.A.: Mathematical models of computer security. In: Focardi, R., Gorrieri, R. (eds.) *FOSAD 2000*. LNCS, vol. 2171, pp. 1–62. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45608-2_1
10. Wittbold, J.T., Johnson, D.M.: Information flow in nondeterministic systems. In: *IEEE Symposium on Security and Privacy*, pp. 144–161. IEEE, Oakland (1990)
11. McCullough, D.: Noninterference and the composability of security properties. In: *IEEE Symposium on Security and Privacy*, pp. 177–186. IEEE, Oakland (1988)
12. Mantel, H.: A uniform framework for the formal specification and verification of information flow security. Ph.D. dissertation, Universitat des Saarlandes (2003)
13. Roscoe, A.W., Goldsmith, M.H.: What is intransitive noninterference? In: *IEEE Computer Security Foundations Workshop*, pp. 228–238. IEEE, Mordano (1999)
14. Oheimb, D.: Information flow control revisited: noninfluence = noninterference + nonleakage. In: Samarati, P., Ryan, P., Gollmann, D., Molva, R. (eds.) *ESORICS 2004*. LNCS, vol. 3193, pp. 225–243. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30108-0_14
15. Engelhardt, K., Meyden, R.V.D., Zhang, C.: Intransitive noninterference in nondeterministic systems. In: *Proceedings of ACM Conference on Computer and Communications Security*, pp. 869–880. ACM, Raleigh (2012)
16. Murray, T., et al.: seL4: from general purpose to a proof of information flow enforcement. In: *IEEE Symposium on Security and Privacy*, pp. 415–429. IEEE, Oakland (2013)
17. Dam, M., Guanciale, R., Schwarz, O.: Formal verification of information flow security for a simple arm-based separation kernel. In: *Proceedings of ACM Conference on Computer and Communications Security*, pp. 223–234. ACM, Berlin (2013)
18. Ko, C., Redmond, T.: Noninterference and intrusion detection. In: *IEEE Symp. on Security and Privacy*, pp:177. IEEE, Oakland (2002)
19. Murray, T., Matichuk, D., Brassil, M., Gammie, P., Klein, G.: Noninterference for operating system kernels. In: Hawblitzel, C., Miller, D. (eds.) *CPP 2012*. LNCS, vol. 7679, pp. 126–142. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35308-6_12
20. Krohn, M., Tromer, E.: Noninterference for a practical DIFC-based operating system. In: *IEEE Symposium on Security and Privacy*, pp. 61–76. IEEE, Oakland (2009)
21. Focardi, R., Gorrieri, R.: The compositional security checker: a tool for the verification of information flow security properties. *IEEE Trans. Softw. Eng.* **23**(9), 550–572 (1997)



Automatical Method for Searching Integrals of ARX Block Cipher with Division Property Using Three Subsets

Ya Han^{1,2(✉)}, Yongqiang Li^{1,2}, and Mingsheng Wang^{1,2}

¹ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

{hanya,wangmingsheng}@iie.ac.cn, yongq.lee@gmail.com

Abstract. Bit-based division property was first proposed to find integral for SIMON32 by Todo *et al.* at FSE 2016. Xiang *et al.* improved the work with aid of Mixed Integer Linear Programming(MILP) method and applied the method to block ciphers with wider block size. Later on, Sun *et al.* applied division property to ARX block ciphers. Todo *et al.* proposed a more precise division property using three subsets method to describe integral propagation at FSE 2016, which can not be applied to wide state ARX block ciphers. In this paper, we extend bit-based division property using three subsets and propose an automatic method for finding integral distinguishers for ARX block ciphers with SAT/SMT solvers. Firstly, we study bit-based division property using three subsets through three basic operations (Copy, AND, XOR). Then, we model division property using three subsets through Addition Modulo function. Finally, by constructing and solving division property using three subsets propagation system, we find integral distinguishers for round reduced ARX block cipher. As a result, we propose 15 round integral distinguishers for SIMON32 automatically and verify the secure margins Todo *et al.* proposed for SIMON48, 64, 96, 128. Also, we can find one more 6 round integral distinguishers for SPECK32, which can not be found with conventional division property without using three subsets. It is interesting that no more integral distinguishers are found for SPECK48,64,96,128. Moreover, we apply to SIMECK, HIGHT, LEA, TEA and XTEA *et al.* Unfortunately, we find no more new results than conventional division property can do.

Keywords: Conventional division property · Three subsets · SIMON SPECK · SAT/SMT

1 Introduction

Division property was first proposed by Todo *et al.* [1] at Eurocrypt 2015 to find integral distinguishers of block cipher. Todo studied the division property

rules through basic components and proposed searching algorithm. With the technique, Todo found a 6-round integral distinguisher for MISTY1 and provided the first theoretical integral attack for full round MISTY1 at Crypto 2015 [2]. Also 5-round gap exist between the proved and experimental one [3] for SIMON32. Later on, Todo *et al.* [4] proposed bit-based division property at FSE 2016.

Xiang *et al.* [5] studied the propagation rules of division property through components and applied MILP method to search integral distinguisher of 6 block ciphers. Sun *et al.* [6,7] applied division property to ARX block ciphers. By modeling the propagation through Addition Modulo, Rotation and XOR of bit-based division property in bit-level, Sun *et al.* proposed wonderful results for ARX block ciphers.

At FSE 2016, Todo *et al.* proposed a precise method to describe the integral namely bit-based division property using three subsets. With this method, Todo *et al.* found 14-round integral distinguisher for SIMON32, which matched the experimental one. Unfortunately, it can not be applied to find exact integrals but “provable security” boundaries for SIMON family block ciphers with block size larger than 32-bits. Moreover, there is still no automatical method for finding integrals of ARX block ciphers with division property using three subsets.

Our Contribution. In this paper, we propose a generic method with SAT/SMT solver to search the integral distinguishers based on division property using three subsets for ARX block ciphers. Our contributions are summarized as follows.

1. We study the propagation rules of conventional division property through basic components (Copy, Xor, Rotation) of ARX block cipher carefully and propose corresponding rules of division property using three subsets in bit-vector mode.
2. Combining with division property using three subsets through basic components, we model the propagation rules of division property using three subsets through Addition Modulo 2^n .
3. With SAT/SMT solver, we apply division property using three subsets to search integral distinguisher for some ARX block ciphers. For SIMON family block ciphers, we propose 15,16,18,22,26-round integral distinguishers automatically and verify the results proposed by Todo *et al.* [4]. For SPECK32, we find one more 6-round integral distinguisher, which can not be found with conventional division property.

Moreover, we confirm the integral distinguisher for some other ARX block cipher like SIMECK [8], HIGHT [9], LEA [10], TEA [11] and XTEA found by conventional division property without using three subsets. Some comparison between our results and exist results can be found in Table 1.

Organization. The remainder of this paper is organized as follows. In Sect. 2, we present important notations throughout the paper. Section 3 covers construction of division property using Three Subsets through basic primitives of ARX block ciphers and searching algorithm. In Sect. 4, we apply the method to some ARX block ciphers. Finally, we conclude the paper in Sect. 5.

Table 1. Results for some ARX block ciphers.

Cipher	Round	Data	Number	Reference	Method
SPECK32	6	31	1	[7]	M1
	6	31	2	This paper	M2
SIMON32	14	31	-	[5]	M1
	15	31	-	[3]	M3
	15	31	-	[4]	M4
	15	31	32	This paper	M2
SIMON48	16	47	-	[5]	M1
	<17	47	-	[4]	M5
	16	47	48	This paper	M2
SIMON64	18	63	-	[5]	M1
	<20	63	-	[4]	M5
	18	63	64	This paper	M2
SIMON96	22	95	-	[5]	M1
	<25	95	-	[4]	M5
	22	95	96	This paper	M2
SIMON128	26	127	-	[5]	M1
	<29	95	-	[4]	M5
	26	127	128	this paper	M2

M1: Division Property.
 M2: SAT/SMT Based Division Property using Three Subsets.
 M3: Experimental Search.
 M4: Based Division Property using Three Subsets.
 M5: Based Division Property using Three Subsets from Lazy Propagation

2 Preliminaries

2.1 Notations

Let \mathbb{F}_2^n denote n -bit length vector over \mathbb{F}_2 . \oplus, \wedge and \neg denote n -bit bitwise Xor, And and Negation respectively. Let $\mathbf{0}$ denote the vector with n consecutive zero-bits, $\mathbf{1}$ denote the vector with n consecutive one-bits. For any $a \in \mathbb{F}_2^n$, $a_{[i]}$ denote the i -th bit of a , $a_{[n-1]}$ is the least significant bit(LSB). Let e_i denote n -bit length unit vector, where the i -th bit equals 1. $wt(a)$ denote the hamming weight of a , which is calculated as $\sum_{i=0}^{n-1} a_{[i]}$. For any $\mathbf{a} = (a_0, a_1, \dots, a_{m-1}) \in \mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \dots \times \mathbb{F}_2^{n_{m-1}}$, the vectorial hamming weight of \mathbf{a} is defined as $Wt(\mathbf{a}) = (wt(a_0), wt(a_1), \dots, wt(a_{m-1})) \in \mathbb{Z}^m$. For any $\mathbf{k} = (k_0, k_1, \dots, k_{m-1}) \in \mathbb{Z}^m$ and $\mathbf{k}' = (k'_0, k'_1, \dots, k'_{m-1}) \in \mathbb{Z}^m$, we define $\mathbf{k} \succeq \mathbf{k}'$ if $k_i \geq k'_i$ for all $i \in \{0, 1, \dots, m-1\}$. Otherwise, $\mathbf{k} \not\succeq \mathbf{k}'$.

2.2 Division Property

Division property is a new method to find integral characteristics. In this section, we define the integral division propagation rules.

Bit Product Function. Let $\pi_u : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ be a bit product function for any $u \in \mathbb{F}_2^n$. Let $x \in \mathbb{F}_2^n$ be the input, and $\pi_u(x)$ is the AND of $x[i]$ satisfying $u[i] = 1$, i.e., it is defined as

$$\pi_u(x) := \prod_{i=0}^{n-1} x[i]^{u[i]}.$$

Let $\pi_u : (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \dots \times \mathbb{F}_2^{n_{m-1}}) \rightarrow \mathbb{F}_2$ be a bit product function for any $\mathbf{u} \in (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \dots \times \mathbb{F}_2^{n_{m-1}})$. Let $\mathbf{x} \in (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \dots \times \mathbb{F}_2^{n_{m-1}})$ be the input, and $\pi_u(\mathbf{x})$ is defined as

$$\pi_u(\mathbf{x}) := \prod_{i=0}^{m-1} \pi_{u_i}(x_i).$$

Definition 1 (Division Property [1]). Let \mathbb{X} be a multiset whose elements in $(\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \dots \times \mathbb{F}_2^{n_{m-1}})$. If \mathbb{X} has the division property $\mathcal{D}_{\mathbb{K}}^{n_0, n_1, \dots, n_{m-1}}$, it fulfills the following conditions:

$$\bigoplus_{x \in \mathbb{X}} \pi_u(x) = \begin{cases} \text{unknown} & \text{if there is } \mathbf{k} \in \mathbb{K} \text{ s.t. } \text{Wt}(\mathbf{u}) \succeq \mathbf{k}, \\ 0 & \text{otherwise.} \end{cases}$$

Todo proved the propagation rules for division property through basic components of block ciphers and these rules were summarized in [2].

2.3 Conventional Bit-Based Division Property

Conventional bit-based division property [4] is a special case of division property, whose division property propagate in bit level. Xiang *et al.* [5] proposed the propagation rules of bit-based division property through Copy, And and Xor operations by linear conditions. Sun *et al.* generalized the model of Copy and Xor in [6].

Theorem 1 (Generalized Copy [6]). Denote $(a) \rightarrow (b_0, b_1, \dots, b_{m-1})$ a division propagation through Copy operation, the following conditions describe the propagation rule.

$$\begin{cases} a - b_0 - b_1 - \dots - b_{m-1} = 0 \\ a, b_0, b_1, \dots, b_{m-1} \text{ are binaries.} \end{cases}$$

Theorem 2. (Generalized Xor [6]). Denote $(a_0, a_1, \dots, a_{m-1}) \rightarrow (b)$ a division propagation through Xor operation, the following conditions describe the propagation rule.

$$\begin{cases} a_0 + a_1 + \dots + a_{m-1} - b = 0 \\ a_0, a_1, \dots, a_{m-1}, b \text{ are binaries.} \end{cases}$$

Theorem 3. (And [5]). Denote $(a_0, a_1) \rightarrow (b)$ a division propagation through And operation, the following conditions describe the propagation rule.

$$\begin{cases} b - a_0 \geq 0 \\ b - a_1 \geq 0 \\ b - a_0 - a_1 \leq 0 \\ a_0, a_1, b \text{ are binaries.} \end{cases}$$

2.4 Bit-Based Division Property Using Three Subsets

Conventional division property uses \mathbb{K} set to represent the subset of u fulfilling $\bigoplus_{x \in \mathbb{X}} \pi_u(x)$ is unknown. The bit-based division property using three subsets use another \mathbb{L} set to represent the subset of u such that $\bigoplus_{x \in \mathbb{X}} \pi_u(x) = 1$.

Definition 2. (Division Property using Three Subsets [1]). Let \mathbb{X} be a multiset whose elements in $(\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \dots \times \mathbb{F}_2^{n_{m-1}})$. If \mathbb{X} has the division property using three subsets $\mathcal{D}_{\mathbb{K}, \mathbb{L}}^{n_0, n_1, \dots, n_{m-1}}$, it fulfills the following conditions:

$$\bigoplus_{x \in \mathbb{X}} \pi_u(x) = \begin{cases} \text{unknown} & \text{if there is } \mathbf{k} \in \mathbb{K} \text{ s.t. } \text{Wt}(\mathbf{u}) \succeq \mathbf{k}, \\ 1 & \text{else if there is } \mathbf{l} \in \mathbb{L} \text{ s.t. } \text{Wt}(\mathbf{u}) = \mathbf{l}, \\ 0 & \text{otherwise.} \end{cases}$$

Assuming that \mathbb{X} has division property using three subsets $\mathbb{D}_{\mathbb{K}, \mathbb{L}}^m$. The propagation rules for \mathbb{K} -set of division property using three subsets is the same as the conventional \mathbb{K} -set do. Moreover, the propagation rule for \mathbb{L} -set through Xor function is the same as the conventional \mathbb{K} -set of division property do.

Proposition 1. Denote $(a) \xrightarrow{\text{Copy}} (b_0, b_1, \dots, b_{m-1})$ one division propagation through Copy operation, the following conditions describe the propagation rule for \mathbb{L} -set of bit-based division property using three subsets through Copy operation.

$$\begin{cases} a - b_0 - \dots - b_{m-1} + ab_0 + \dots + ab_{m-1} + \bar{a}\bar{b}_0 \dots \bar{b}_{m-1} = 1 \\ a, b_0, b_1, \dots, b_{m-1} \text{ are binaries.} \end{cases}$$

Proposition 2. Denote $(a_0, a_1) \xrightarrow{\text{And}} (b)$ a division propagation through And operation, the following conditions describe the propagation rule for \mathbb{L} -set of bit-based division property using three subsets through And operation.

$$\begin{cases} b + ba_0 + ba_1 + \bar{a}_0\bar{a}_1 = 1 \\ b, a_0, a_1 \text{ are binaries.} \end{cases}$$

2.5 Bit-Vector Based Division Property Using Three Subsets

We propose an equivalent form of division property using three subsets in bit-vector level. In this section, we introduce the propagation rules of bit-vector based division property using three subsets through basic components and propose the proof for division property using three subsets through Copy function in Appendix A.

Proposition 3. Denote $(x) \rightarrow (y_0, y_1, \dots, y(m - 1))$ a division propagation through Copy operation using three subsets, \mathbb{K} -set and \mathbb{L} -set propagation rule satisfy Theorem 1 and Proposition 1 respectively.

$$DP3_{\mathbb{K}}(x \rightarrow y_0, y_1) = \neg x \wedge \neg y_0 \wedge \neg y_1 \oplus x \wedge (y_0 \oplus y_1) = \mathbf{1}.$$

$$DP3_{\mathbb{K}}(x \rightarrow y_0, y_1, y_2) = \neg y_1 \wedge \neg y_2 \wedge (y_0 \oplus \neg x) \oplus \neg y_0 \wedge x \wedge (y_1 \oplus y_2) = \mathbf{1}.$$

We define \mathbb{L} -set through general Copy operation as

$$DP3_{\mathbb{L}}(x \rightarrow y_0, y_1, \dots, y(m - 1)) = x \oplus \neg y_0 \wedge \neg y_1 \wedge \dots \wedge \neg y(m - 1) = \mathbf{1}.$$

Proposition 4. Denote $(x_0, x_1, \dots, x(m - 1)) \rightarrow (y)$ a division propagation through Xor operation using three subsets, \mathbb{K} -set and \mathbb{L} -set propagation rule satisfy Theorem 2 and satisfy the same propagation rule.

$$DP3(x_0, x_1 \rightarrow y) = \neg y \wedge \neg x_0 \wedge \neg x_1 \oplus y \wedge (x_0 \oplus x_1) = \mathbf{1}.$$

$$DP3(x_0, x_1, x_2 \rightarrow y) = \neg x_1 \wedge \neg x_2 \wedge (x_0 \oplus \neg y) \oplus \neg x_0 \wedge y \wedge (x_1 \oplus x_2) = \mathbf{1}.$$

Proposition 5. Denote $(x_0, x_1) \rightarrow (y)$ a division propagation through And operation using three subsets, \mathbb{K} -set and \mathbb{L} -set propagation rule satisfy Theorem 3 and Proposition 2 respectively.

$$DP3_{\mathbb{K}}(x_0, x_1 \rightarrow y) = y \oplus \neg x_0 \wedge \neg x_1 = \mathbf{1}.$$

$$DP3_{\mathbb{L}}(x_0, x_1 \rightarrow y) = y \oplus x_0 \oplus x_1 \oplus x_0 \wedge x_1 \oplus y \wedge x_0 \oplus y \wedge x_1 = \mathbf{0}.$$

In [4], Todo *et al.* proposed the dependencies between \mathbb{K} -set and \mathbb{L} -set of division property using three subsets when propagating through “Xor Round Key” function.

Theorem 4 (Dependencies between \mathbb{K} -set and \mathbb{L} -set). Let \mathbb{X} and \mathbb{Y} be the input and output multiset of operation $y = x \oplus rk$, where rk is round key. Let $\mathcal{D}_{\mathbb{K}, \mathbb{L}}^n$ and $\mathcal{D}_{\mathbb{K}', \mathbb{L}'}^n$ be the division property using three subsets of \mathbb{X} and \mathbb{Y} , respectively. For any vector $l = (l_0, l_1, \dots, l_{n-1}) \in \mathbb{L}$ and all $l_i = 0$, where $i \in [0, n - 1]$. \mathbb{K}' is computed as

$$\mathbb{K}' \leftarrow (l_0, \dots, l_i \vee 1, \dots, l_{n-1}).$$

3 Modeling Division Property Using Three Subsets Through Addition Modulo 2^n

Addition Module 2^n is the only nonlinear component of ARX block ciphers. Assuming that $\mathbf{x} = (x_{[0]}, x_{[1]}, \dots, x_{[n-1]})$, $\mathbf{y} = (y_{[0]}, y_{[1]}, \dots, y_{[n-1]})$, $\mathbf{c} = (c_{[0]}, c_{[1]}, \dots, c_{[n-1]})$ and $\mathbf{z} = (z_{[0]}, z_{[1]}, \dots, z_{[n-1]})$ are n -bit length vectors in \mathbb{F}_2^n . For ARX block ciphers like SPECK, LEA etc., the operation has the form $z = x \boxplus y$, where the input parameters x, y are mid-states of block cipher and z is the output state associated with x and y . For HIGHT etc., the operation has the form $z = x \boxplus rk$, where x is the mid-state of block cipher, rk is a constant (round key) value and z is the output state associated with x and c .

3.1 General Addition Modulo 2^n

Assuming that $\mathbf{x} = (x_{[0]}, x_{[1]}, \dots, x_{[n-1]})$, $\mathbf{y} = (y_{[0]}, y_{[1]}, \dots, y_{[n-1]})$, $\mathbf{z} = (z_{[0]}, z_{[1]}, \dots, z_{[n-1]})$ are n -bit length vectors in \mathbb{F}_2^n and $\mathbf{z} = \mathbf{x} \boxplus \mathbf{y}$. Addition Modulo 2^n can be expressed bit-by-bit as

$$\left\{ \begin{array}{l} z_{[n-1]} = x_{[n-1]} \oplus y_{[n-1]} \\ z_{[n-2]} = x_{[n-2]} \oplus y_{[n-2]} \oplus c_{[n-2]}, c_{[n-2]} = x_{[n-1]}y_{[n-1]} \\ z_{[n-3]} = x_{[n-3]} \oplus y_{[n-3]} \oplus c_{[n-3]}, \\ \quad c_{[n-3]} = x_{[n-2]}y_{[n-2]} \oplus (x_{[n-2]} \oplus y_{[n-2]})c_{[n-2]} \\ \quad \vdots \\ z_{[0]} = x_{[0]} \oplus y_{[0]} \oplus c_{[0]}, c_{[0]} = x_{[0]}y_{[0]} \oplus (x_{[0]} \oplus y_{[0]})c_{[0]}. \end{array} \right.$$

$$c_{[i]} = x_{[i]} + y_{[i]} + c_{[i+1]} \text{ and } c_{[n-1]} = 0.$$

\mathbb{K} -set Propagation Rule. Some variables are introduced to describe the propagation rule of \mathbb{K} -set for bit-vector based division property using three subsets through general Addition Modulo 2^n . The division propagation of \mathbb{K} -set through general Addition Modulo 2^n is shown in Appendix B.1. The propagation system of \mathbb{K} -set for bit-vector based division property using three subsets through Addition Modulo 2^n denoted $\mathcal{S}_{DP3_{\mathbb{K}}}(kx, ky \xrightarrow{\boxplus} kz)$ can be described as

$$\left\{ \begin{array}{l} DP3_{\mathbb{K}}(kx1, ky1, kc1 \xrightarrow{Xor} kz) = 1 \\ DP3_{\mathbb{K}}(kx \xrightarrow{Copy} kx1, kx2, kx3) = 1 \\ DP3_{\mathbb{K}}(ky \xrightarrow{Copy} ky1, ky2, ky3) = 1 \\ DP3_{\mathbb{K}}(kc2 \xrightarrow{Copy} kc1, kc2) = 1 \\ DP3_{\mathbb{K}}(kx2, ky2 \xrightarrow{And} ku) = 1 \\ DP3_{\mathbb{K}}(kx3, ky3 \xrightarrow{Xor} kv) = 1 \\ DP3_{\mathbb{K}}(kv, kc3 \xrightarrow{And} kw) = 1 \\ DP3_{\mathbb{K}}(ku \ll 1, kw \ll 1 \xrightarrow{Xor} kc2) = 1 \end{array} \right.$$

\mathbb{L} -set Propagation Rule. Some variables are introduced to describe the propagation rule of \mathbb{L} -set for bit-vector based division property using three subsets through general Addition Modulo 2^n . Similarly, the propagation system of \mathbb{L} -set for bit-vector based division property using three subsets through Addition Modulo 2^n denoted $\mathcal{S}_{DP3_{\mathbb{L}}}(lx, ly \xrightarrow{\boxplus} lz)$ can be described as

$$\left\{ \begin{array}{l} DP3_{\mathbb{L}}(lx1, ly1, lc1 \xrightarrow{Xor} lz) = 1 \\ DP3_{\mathbb{L}}(lx \xrightarrow{Copy} lx1, lx2, lx3) = 1 \\ DP3_{\mathbb{L}}(ly \xrightarrow{Copy} ly1, ly2, ly3) = 1 \\ DP3_{\mathbb{L}}(lc2 \xrightarrow{Copy} lc1, lc2) = 1 \\ DP3_{\mathbb{L}}(lx2, ly2 \xrightarrow{And} lu) = 0 \\ DP3_{\mathbb{L}}(lx3, ly3 \xrightarrow{Xor} lv) = 1 \\ DP3_{\mathbb{L}}(lv, lc3 \xrightarrow{And} lw) = 0 \\ DP3_{\mathbb{L}}(lu \ll 1, lw \ll 1 \xrightarrow{Xor} lc2) = 1 \end{array} \right.$$

3.2 Addition Modulo 2^n with Constant

Assuming that $\mathbf{x} = (x_{[0]}, x_{[1]}, \dots, x_{[n-1]})$, $\mathbf{rk} = (rk_{[0]}, rk_{[1]}, \dots, rk_{[n-1]})$, $\mathbf{z} = (z_{[0]}, z_{[1]}, \dots, z_{[n-1]})$ are n -bit length vectors in \mathbb{F}_2^n and $\mathbf{z} = \mathbf{x} \boxplus \mathbf{rk}$. Addition Modulo 2^n with constant can be expressed bit-by-bit as

$$\left\{ \begin{array}{l} z_{[n-1]} = x_{[n-1]} \oplus rk_{[n-1]} \\ z_{[n-2]} = x_{[n-2]} \oplus rk_{[n-2]} \oplus c_{[n-2]}, c_{[n-2]} = x_{[n-1]}rk_{[n-1]} \\ z_{[n-3]} = x_{[n-3]} \oplus rk_{[n-3]} \oplus c_{[n-3]}, \\ \quad c_{[n-3]} = x_{[n-2]}rk_{[n-2]} \oplus (x_{[n-2]} \oplus rk_{[n-2]})c_{[n-2]} \\ \quad \vdots \\ z_{[0]} = x_{[0]} \oplus rk_{[0]} \oplus c_{[0]}, c_{[0]} = x_{[0]}rk_{[0]} \oplus (x_{[0]} \oplus rk_{[0]})c_{[0]}. \end{array} \right.$$

\mathbb{K} -set Propagation Rule. Some variables are introduced to describe the propagation rule of \mathbb{K} -set for bit-vector based division property using three subsets through Addition Modulo 2^n with constant. The division propagation of \mathbb{K} -set through Addition Modulo 2^n with constant is shown in Appendix B.2. The propagation system of \mathbb{K} -set for bit-vector based division property through Addition Modulo 2^n with constant denoted $\mathcal{S}_{DP3_{\mathbb{K}}}(kx, rk \xrightarrow{\boxplus} kz)$ can be described as

$$\left\{ \begin{array}{l} DP3_{\mathbb{K}}(kx1, kc1 \xrightarrow{Xor} kz) = 1 \\ DP3_{\mathbb{K}}(kx \xrightarrow{Copy} kx1, kx2, kx3) = 1 \\ DP3_{\mathbb{K}}(kc2 \xrightarrow{Copy} kc1, kc3) = 1 \\ DP3_{\mathbb{K}}(kx3, kc3 \xrightarrow{And} ku) = 1 \\ DP3_{\mathbb{K}}(kx2 \ll 1, ku \ll 1 \xrightarrow{Xor} kc2) = 1 \end{array} \right.$$

\mathbb{L} -set Propagation Rule. Some variables are introduced to describe the propagation rule of \mathbb{L} -set for bit-vector based division property using three subsets through Addition Modulo 2^n with constant. Similarly, the propagation system of \mathbb{L} -set for bit-vector based division property using three subsets through Addition Modulo 2^n with constant denoted $\mathcal{S}_{DP3_{\mathbb{L}}}(lx, rk \stackrel{\boxplus}{\rightarrow} lz)$ can be described as

$$\begin{cases} DP3_{\mathbb{L}}(lx1, lc1 \xrightarrow{Xor} lz) & = 1 \\ DP3_{\mathbb{L}}(lx \xrightarrow{Copy} lx1, lx2, lx3) & = 1 \\ DP3_{\mathbb{L}}(lc2 \xrightarrow{Copy} lc1, lc3) & = 1 \\ DP3_{\mathbb{L}}(lx3, lc3 \xrightarrow{And} lu) & = 0 \\ DP3_{\mathbb{L}}(lx2 \ll 1, lu \ll 1 \xrightarrow{Xor} lc2) & = 1 \end{cases}$$

3.3 Searching Algorithm

We construct r -round reduced propagation system of bit-vector based division property using three subsets. With the syntax rules of SAT/SMT solver, we turn r -round propagation system into problem, which the solver can recognize. Given one constant input division property using three subsets denoted $\mathcal{D}_{k,l}^n$ where k, l are non-zero bit-vector in \mathbb{F}_2^n . The SAT/SMT solver can calculate the \mathbb{K} -set of division property after r -rounds propagation denoted \mathbb{K}_r . If \mathbb{K}_r contains all the n unit vectors $e_i \in \mathbb{K}_r$, for all $i \in \{0, 1, \dots, n - 1\}$. As for any bit-vector $u \in \mathbb{F}_2^n$, it always holds that $u \succeq e_i$. There exists no bit-vector u can fulfill the equation $\bigoplus_{x \in \mathbb{X}} \pi_u(x) = 0$, so no balanced sum bits exist. Otherwise, if some unit vector e_i for some $i \in \{0, 1, \dots, n - 1\}$ is not in \mathbb{K}_r , it holds that $e_i \not\preceq k$ for all $k \in \mathbb{K}$. Therefore, $\bigoplus_{x \in \mathbb{K}} \pi_{e_i}(x) = 0$ and the i -th bit is zero-sum bit. As for division property using three subsets $\mathcal{D}_{\mathbb{K}_r, \mathbb{L}_r}^n$, we check whether all the n unit vectors are in \mathbb{K}_r . If so, there is no r -round integral distinguisher with input division property using three subsets $\mathcal{D}_{k,l}^n$. Otherwise, some r -round integral distinguishers exist with input division property using three subsets. By exhausting all possible division property using three subsets, we can judge whether some r -round integral distinguishers exist.

4 Apply to ARX Block Ciphers

Our experiment platform is established with Python 3.4.3 and STP [12] (a SMT solver) on a virtual machine with Intel(R) Core(TM) CPU i5-4210M(2.60GHz, 1GB RAM, Ubuntu 14.04.1).

4.1 Apply to SPECK Family Block Ciphers

Brief Description. The SPECK- $2n$ encryption maps make use of bitwise Xor, Addition Modulo 2^n , left and right circular shift (S^j, S^{-j}) by j bits on n -bit word. As the round key is Xored with internal state, it has no effect on the propagation of division property. Division propagation through SPECK round function is shown as Fig. 1. With shift amounts $\alpha = 7$ and $\beta = 2$ if $n = 16$, $\alpha = 8$ and $\beta = 3$ otherwise.

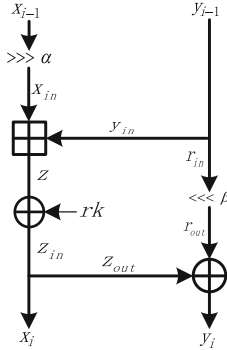


Fig. 1. Division propagation through SPECK round function

Modeling Round Function. $(x_{i-1}, y_{i-1}), (x_i, y_i)$ are the input and output of i -th round respectively. (x_{in}, y_{in}) are the input of Addition Modulo 2^n and z is the output of Addition Modulo 2^n . (y_{i-1}, z_{in}) propagate through Copy function to $(y_{in}, r_{in}), (z_{out}, x_i)$ respectively. (r_{out}, z_{out}) are Xored to y_i . (x_{i-1}, r_{in}) propagate through cyclic shift function to (x_{in}, r_{out}) . z propagate through Xored round key function to z_{in} and every possible value for lz affect kz_{in} . With the propagation rules of bit-vector based division property using three subsets, we model \mathbb{K} -set propagation system of bit-vector based division property using three subsets as

$$\left\{ \begin{array}{l} \mathcal{S}_{DP3_{\mathbb{K}}}(kx_{in}, ky_{in} \xrightarrow{\boxplus} kz) \\ (kx_{i-1} \ggg \alpha) \oplus kx_{in} = \mathbf{0} \\ (kr_{in} \lll \beta) \oplus kr_{out} = \mathbf{0} \\ DP3_{\mathbb{K}}(kz_{in} \xrightarrow{Copy} kz_{out}, kx_i) = \mathbf{1} \\ DP3_{\mathbb{K}}(ky_{i-1} \xrightarrow{Copy} ky_{in}, kr_{in}) = \mathbf{1} \\ DP3_{\mathbb{K}}(kr_{out}, kz_{out} \xrightarrow{Xor} ky_i) = \mathbf{1}. \end{array} \right.$$

Similarly, we model the \mathbb{L} -set propagation system of bit-vector based division property using three subsets as

$$\left\{ \begin{array}{l} \mathcal{S}_{DP3_{\mathbb{L}}}(lx_{in}, ly_{in} \xrightarrow{\boxplus} lz) \\ (lx_{i-1} \ggg \alpha) \oplus lx_{in} = \mathbf{0} \\ (lr_{in} \lll \beta) \oplus lr_{out} = \mathbf{0} \\ DP3_{\mathbb{K}}(lz \xrightarrow{Copy} lz_{out}, lx_i) = \mathbf{1} \\ DP3_{\mathbb{K}}(ly_{i-1} \xrightarrow{Copy} ly_{in}, lr_{in}) = \mathbf{1} \\ DP3_{\mathbb{K}}(lr_{out}, lz_{out} \xrightarrow{Xor} ly_i) = \mathbf{1}. \end{array} \right.$$

Construct the dependencies between variables lz and kz_{in} . As Theorem 4 shown, for any possible value $l = (l_0, l_1, \dots, l_{n-1}) \in lz$, we set value $(l_0, \dots, l_i \vee 1, l_{n-1})$ to kz_{in} for all $l_i = 0$, where $i \in [0, n - 1]$.

Table 2. Results for SPECK family block ciphers.

Block cipher	Round	Data	$\#\{\text{Balanced bits}\}$	Number
SPECK32	6	31	1	2
SPECK48	6	47	1	4
SPECK64	6	63	1	4
SPECK96	6	95	1	4
SPECK128	6	127	1	4

Results for SPECK. We find one more 6-round integral distinguishers for SPECK32 than the conventional division property can find. It is interesting that no more integral distinguishers are found for other variants. The results are summarized in Table 2. In Appendix C.1, we propose the detailed integral distinguishers for SPECK family block ciphers.

4.2 Apply to SIMON Family Block Ciphers

Brief Description. SIMON is a family of lightweight block ciphers proposed by NSA in 2013 [13]. It is a Feistel like cipher with block sizes 32, 48, 64, 96, 128 bits. The variant operating on $2n$ -bit state, where n is the word size and $n \in \{16, 24, 32, 48, 64\}$, is referred to as SIMON $2n$, which have an efficient implementation in hardware with rotational constants (1, 8, 2). The subkeys are derived from a master key by key scheduling. For detailed description and refer the reader to [13]. Division propagation through SIMON round function is shown as Fig. 2.

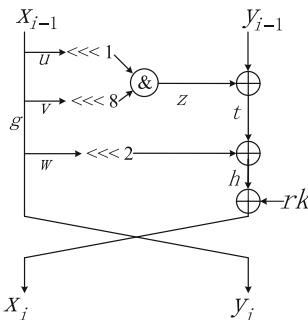


Fig. 2. Division Propagation through SIMON Round Function

Modeling Round Function. $(x_{i-1}, y_{i-1}), (x_i, y_i)$ are the input and output of i -th round respectively. x_{i-1} propagate through Copy function to (u, v, w, y_i) . $(u \lll 1, v \lll 8)$ are compressed by And operation to z . $(y_{i-1}, z), (w \lll 2, t)$ are Xored to t, h respectively. h propagate through Xored round key function to x_i

and every possible value for lh affect kx_i . With the propagation rules of bit-vector based division property using three subsets through ARX round components, we model the \mathbb{K} -set propagation system of bit-vector based division property using three subsets through SIMON- $2n$ round function as

$$\begin{cases} DP3_{\mathbb{K}}(kx_i \xrightarrow{Copy} ku, kv, kg) = \mathbf{1} \\ DP3_{\mathbb{K}}(kg \xrightarrow{Copy} kw, ky_i) = \mathbf{1} \\ DP3_{\mathbb{K}}(ku \lll 1, kv \lll 8 \xrightarrow{And} kz) = \mathbf{1} \\ DP3_{\mathbb{K}}(ky_{i-1}, kz \xrightarrow{Xor} kt) = \mathbf{1} \\ DP3_{\mathbb{K}}(kw \lll 2, kt \xrightarrow{Xor} kh) = \mathbf{1}. \end{cases}$$

Similarly, we model the \mathbb{L} -set propagation system of bit-vector based division property using three subsets through SIMON- $2n$ round function as

$$\begin{cases} DP3_{\mathbb{L}}(lx_i \xrightarrow{Copy} lu, lv, lg) = \mathbf{1} \\ DP3_{\mathbb{L}}(lg \xrightarrow{Copy} lw, ly_i) = \mathbf{1} \\ DP3_{\mathbb{L}}(lu \lll 1, lv \lll 8 \xrightarrow{And} lz) = \mathbf{0} \\ DP3_{\mathbb{L}}(ly_{i-1}, lz \xrightarrow{Xor} lt) = \mathbf{1} \\ DP3_{\mathbb{L}}(lw \lll 2, lt \xrightarrow{Xor} lx_i) = \mathbf{1}. \end{cases}$$

Construct the dependencies between variables lh and kx_i . As Theorem 4 shown, for any possible value $l = (l_0, l_1, \dots, l_{n-1}) \in lh$, we set value $(l_0, \dots, l_i \vee 1, l_{n-1})$ to kx_i for all $l_i = 0$, where $i \in [0, n - 1]$.

Results for SIMON. For SIMON- $2n$ family block cipher with block size 32, 48, 64, 96 and 128, we propose 15-, 16-, 18-, 22- and 26-round integral distinguishers respectively. For SIMON32, we propose 15-round integral distinguishers with division property using three subsets automatically, which erase the one round gap found by conventional division property without using three subsets. For SIMON48/64/96/128, we verify the margin Todo proposed in [4] with the strategy of “Lazy propagation”. The results are summarized in Table 3. In Appendix C.2, we propose some detailed integral distinguishers for SIMON family block ciphers.

5 Conclusion and Future Work

In this paper, we propose a SAT/SMT based method for searching integral distinguishers of ARX block ciphers with bit-vector based division property using three subsets. We apply the method to search some ARX block ciphers including SPECK, SIMON, SIMECK, HIGHT, LEA, TEA and XTEA *et al.* For SPECK family block ciphers, we present 6-, 6-, 6-, 6- and 6-round integral distinguishers. Interestingly that, we find one more integral distinguisher for SPECK32, which can not be found with conventional division property. Unfortunately, we

Table 3. Results for SIMON family block ciphers.

Block cipher	Round	Data	$\#\{\text{Balanced bits}\}$	Number
SIMON32	15	31	32	3
SIMON48	16	47	48	24
SIMON64	18	63	64	22
SIMON96	22	95	96	5
SIMON128	26	127	128	3

can not find longer integral distinguisher with bit-vector based division property using three subsets. For SIMON32, we find 15-round integral distinguishers with bit-vector based division property using three subsets automatically. For other variants of SIMON family block ciphers, we verify the secure margin Todo proposed. Moreover, we apply our method to SIMECK, HIGHT, LEA, TEA and XTEA *et al.* Unfortunately, we find no new results. In the future, we will apply bit-vector based division property using three subsets technique to S-box based block ciphers.

Acknowledgements. Thanks to the reviewers for their valuable comments. This work is supported by the National Science Foundation of China (No. 61772516, 61772517), Youth Innovation Promotion Association CAS.

A Proof of Propagation Rules of Bit-Vector Based Division Property Through Copy Function

Proof of Propositions 3. For 2-Copy operation $(x) \rightarrow (y_0, y_1)$, there are three possible conditions $(0) \rightarrow (0, 0)$, $(1) \rightarrow (0, 1)$ and $(1) \rightarrow (1, 0)$ for any division propagation $(x_{[i]}) \rightarrow (y_0_{[i]}, y_1_{[i]})$. It is possible to describe these conditions with boolean function $\neg x_{[i]} \wedge \neg y_0_{[i]} \wedge \neg y_1_{[i]} \oplus x_{[i]} \wedge (y_0_{[i]} \oplus y_1_{[i]}) = 1$. As any two division propagations are independent with each other, we can describe the propagation rule of \mathbb{K} -set for bit-vector based division property using three subsets through 2-Copy operation with function $\neg x \wedge \neg y_0 \wedge \neg y_1 \oplus x \wedge (y_0 \oplus y_1) = \mathbf{1}$. The propagation rule of \mathbb{K} -set and \mathbb{L} -set for division property using three subsets through 3-Copy operation satisfy the same process of proof.

B Division Propagation of \mathbb{K} -set through Addition Modulo 2^n

B.1 General Addition Modulo 2^n

We list the division propagation of \mathbb{K} -set through general Addition Modulo 2^n in Table 4.

Table 4. Division propagation of \mathbb{K} -set through general addition modulo 2^n .

Bit	Division propagation	Bit	Division propagation
$kz_{[n-1]}$	$\underbrace{kx_{[n-1]}}_{kx1_{[n-1]}} \oplus \underbrace{ky_{[n-1]}}_{ky1_{[n-1]}} \oplus \underbrace{0}_{kc1_{[n-1]}}$	$kc_{[n-1]}$	$\underbrace{0}_{kx2_{[0]}} \oplus \underbrace{0}_{ky2_{[0]}} \oplus \underbrace{0}_{kx3_{[0]}} \oplus \underbrace{0}_{ky3_{[0]}} \oplus \underbrace{0}_{kc3_{[0]}}$
$kz_{[n-2]}$	$\underbrace{kx_{[n-2]}}_{kx1_{[n-2]}} \oplus \underbrace{ky_{[n-2]}}_{ky1_{[n-2]}} \oplus \underbrace{kc_{[n-2]}}_{kc1_{[n-2]}}$	$kc_{[n-2]}$	$\underbrace{kx_{[n-1]}}_{kx2_{[n-1]}} \oplus \underbrace{ky_{[n-1]}}_{ky2_{[n-1]}} \oplus \underbrace{0}_{kx3_{[n-1]}} \oplus \underbrace{0}_{ky3_{[n-1]}} \oplus \underbrace{0}_{kc3_{[n-1]}}$
$kz_{[n-3]}$	$\underbrace{kx_{[n-3]}}_{kx1_{[n-3]}} \oplus \underbrace{ky_{[n-3]}}_{ky1_{[n-3]}} \oplus \underbrace{kc_{[n-3]}}_{kc1_{[n-3]}}$	$kc_{[n-3]}$	$\underbrace{kx_{[n-2]}}_{kx2_{[n-2]}} \oplus \underbrace{ky_{[n-2]}}_{ky2_{[n-2]}} \oplus \underbrace{kx_{[n-2]}}_{kx3_{[n-2]}} \oplus \underbrace{ky_{[n-2]}}_{ky3_{[n-2]}} \oplus \underbrace{kc_{[n-2]}}_{kc3_{[n-2]}}$
\vdots	\vdots	\vdots	\vdots
$kz_{[0]}$	$\underbrace{kx_{[0]}}_{kx1_{[0]}} \oplus \underbrace{ky_{[0]}}_{ky1_{[0]}} \oplus \underbrace{kc_{[0]}}_{kc1_{[0]}}$	$kc_{[0]}$	$\underbrace{kx_{[1]}}_{kx2_{[1]}} \oplus \underbrace{ky_{[1]}}_{ky2_{[1]}} \oplus \underbrace{kx_{[1]}}_{kx3_{[1]}} \oplus \underbrace{ky_{[1]}}_{ky3_{[1]}} \oplus \underbrace{kc_{[1]}}_{kc3_{[1]}}$

B.2 Addition Modulo 2^n with Constant

We list the division propagation of \mathbb{K} -set through Addition Modulo 2^n with constant in Table 5.

C Integral Distinguishers for ARX Block Ciphers

In this section, we propose the detailed integral distinguishers for ARX block ciphers. ‘ \mathcal{A}^n ’ represents n connected active bits, ‘ \mathcal{C}^n ’ represents n connected constant bits. ‘ \mathcal{U}^n ’ represents n connected unknown bits, ‘ \mathcal{B}^n ’ represents n connected balanced bits satisfying zero-sum property.

C.1 Integral Distinguishers for SPECK Family Block Ciphers

For SPECK32, we find 2 integral distinguisher. For SPECK48/64/96/128, we find 4, 4, 4 and 4 integral distinguishers respectively. In Table 6, we present the detailed integral distinguishers for SPECK family block ciphers.

C.2 Integral Distinguishers for SIMON Family Block Ciphers

For SIMON32, we find 15-round integral distinguisher, which can not be found with conventional division property. In Table 7, we present the detailed integral distinguishers for SIMON family block ciphers.

Table 5. Division propagation of \mathbb{K} -set through addition modulo 2^n with constant.

Bit	Division propagation	Bit	Division propagation
$kz_{[n-1]}$	$\underbrace{kx_{[n-1]}}_{kx1_{[n-1]}} \oplus \underbrace{0}_{kc1_{[n-1]}}$	$\underbrace{kC_{[n-1]}}_{kc2_{[n-1]}}$	$\underbrace{0}_{kx2_{[0]}} \oplus \overbrace{\underbrace{0}_{kx3_{[0]}} \underbrace{0}_{kc3_{[0]}}}^{ku_{[0]}}$
$kz_{[n-2]}$	$\underbrace{kx_{[n-2]}}_{kx1_{[n-2]}} \oplus \underbrace{kC_{[n-2]}}_{kc1_{[n-2]}}$	$\underbrace{kC_{[n-2]}}_{kc2_{[n-2]}}$	$\underbrace{kx_{[n-1]}}_{kx2_{[n-1]}} \oplus \overbrace{\underbrace{0}_{kx3_{[n-1]}} \underbrace{0}_{kc3_{[n-1]}}}^{ku_{[n-1]}}$
$kz_{[n-3]}$	$\underbrace{kx_{[n-3]}}_{kx1_{[n-3]}} \oplus \underbrace{kC_{[n-3]}}_{kc1_{[n-3]}}$	$\underbrace{kC_{[n-3]}}_{kc2_{[n-3]}}$	$\underbrace{kx_{[n-2]}}_{kx2_{[n-2]}} \oplus \overbrace{\underbrace{kx_{[n-2]}}_{kx3_{[n-2]}} \underbrace{kC_{[n-2]}}_{kc3_{[n-2]}}}^{ku_{[n-2]}}$
\vdots	\vdots	\vdots	\vdots
$kz_{[0]}$	$\underbrace{kx_{[0]}}_{kx1_{[0]}} \oplus \underbrace{kC_{[0]}}_{kc1_{[0]}}$	$\underbrace{kC_{[0]}}_{kc2_{[0]}}$	$\underbrace{kx_{[1]}}_{kx2_{[1]}} \oplus \overbrace{\underbrace{kx_{[1]}}_{kx3_{[1]}} \underbrace{kC_{[1]}}_{kc3_{[1]}}}^{ku_{[1]}}$

Table 6. Integral distinguishers for SPECK family block ciphers.

Block cipher	Integral distinguisher	Balanced
SPECK32	$(A^4 A^4 A^4 A^4 A^4 A^4 A^4 A^4) \xrightarrow{GR} (U^4 U^4 U^4 U^3 B U^4 U^4 U^4 U^4)$ $(A^4 A^4 A^4 A^4 A^4 A^4 A^2 C A A^4) \xrightarrow{GR} (U^4 U^4 U^4 U^3 B U^4 U^4 U^4 U^4)$	1
SPECK48	$(A^6 A^6 A^6 A^6 A^6 A^6 A^3 C A^2 A^6) \xrightarrow{GR} (U^6 U^6 U^6 U^5 B U^6 U^6 U^6 U^6)$ $(A^6 A^6 A^6 A^6 A^6 A^6 A^4 C A A^6) \xrightarrow{GR} (U^6 U^6 U^6 U^5 B U^6 U^6 U^6 U^6)$ $(A^6 A^6 A^6 A^6 A^6 A^6 A^5 C A^6) \xrightarrow{GR} (U^6 U^6 U^6 U^5 B U^6 U^6 U^6 U^6)$ $(A^6 A^6 A^6 A^6 A^6 A^6 A^6 C A^5) \xrightarrow{GR} (U^6 U^6 U^6 U^5 B U^6 U^6 U^6 U^6)$	1
SPECK64	$(A^8 A^8 A^8 A^8 A^8 A^8 A^7 C A^8) \xrightarrow{GR} (U^8 U^8 U^8 U^7 B U^8 U^8 U^8 U^8)$ $(A^8 A^8 A^8 A^8 A^8 A^8 A^8 C A^7) \xrightarrow{GR} (U^8 U^8 U^8 U^7 B U^8 U^8 U^8 U^8)$ $(A^8 A^8 A^8 A^8 A^8 A^8 A^8 A C A^6) \xrightarrow{GR} (U^8 U^8 U^8 U^7 B U^8 U^8 U^8 U^8)$ $(A^8 A^8 A^8 A^8 A^8 A^8 A^8 A^2 C A^5) \xrightarrow{GR} (U^8 U^8 U^8 U^7 B U^8 U^8 U^8 U^8)$	1
SPECK96	$(A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^3 C A^8) \xrightarrow{GR} (U^{12} U^{12} U^{12} U^{11} B U^{12} U^{12} U^{12} U^{12})$ $(A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^4 C A^7) \xrightarrow{GR} (U^{12} U^{12} U^{12} U^{11} B U^{12} U^{12} U^{12} U^{12})$ $(A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^5 C A^6) \xrightarrow{GR} (U^{12} U^{12} U^{12} U^{11} B U^{12} U^{12} U^{12} U^{12})$ $(A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^{12} A^6 C A^5) \xrightarrow{GR} (U^{12} U^{12} U^{12} U^{11} B U^{12} U^{12} U^{12} U^{12})$	1
SPECK128	$(A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^7 C A^8) \xrightarrow{GR} (U^{16} U^{16} U^{16} U^{15} B U^{16} U^{16} U^{16} U^{16})$ $(A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^8 C A^7) \xrightarrow{GR} (U^{16} U^{16} U^{16} U^{15} B U^{16} U^{16} U^{16} U^{16})$ $(A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^9 C A^6) \xrightarrow{GR} (U^{16} U^{16} U^{16} U^{15} B U^{16} U^{16} U^{16} U^{16})$ $(A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^{16} A^{10} C A^5) \xrightarrow{GR} (U^{16} U^{16} U^{16} U^{15} B U^{16} U^{16} U^{16} U^{16})$	1

Table 7. Integral distinguishers for SIMON family block ciphers.

Block cipher	Integral distinguisher	Balanced bits
SIMON32	$(CA^3A^4A^4A^4A^4A^4A^4A^4) \xrightarrow{14R} (U^4U^4U^4U^4UBU^2U^4BU^3U^3B)$	3
SIMON48	$(CA^5A^6A^6A^6A^6A^6A^6A^6) \xrightarrow{15R} (U^6U^6U^6U^6B^6B^6B^6B^6)$	24
SIMON64	$(CA^8A^8A^8A^8A^8A^8A^8A^8) \xrightarrow{17R} (U^8U^8U^8U^8B^3B^3U^5BU^5B^2B^8)$	22
SIMON96	$(CA^{11}A^{12}A^{12}A^{12}A^{12}A^{12}A^{12}A^{12}) \xrightarrow{21R} (BUBU^4BU^4U^{12}U^{12}U^5BU^4BU)$	5
SIMON128	$(CA^{15}A^{16}A^{16}A^{16}A^{16}A^{16}A^{16}A^{16}) \xrightarrow{25R} (BUBU^{13}U^{16}U^{16}U^{14}BU)$	3

References

1. 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
2. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_20
3. Wang, Q., Liu, Z., Varici, 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
4. 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
5. 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
6. Sun, L., Wang, W., Liu, R., Wang, M.: Milp-aided bit-based division property for ARX-based block cipher. IACR Cryptology ePrint Arch. **2016**, 1101 (2016)
7. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5
8. 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
9. Hong, D., et al.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_4
10. Hong, D., Lee, J.-K., Kim, D.-C., Kwon, D., Ryu, K.H., Lee, D.-G.: LEA: a 128-bit block cipher for fast encryption on common processors. In: Kim, Y., Lee, H., Perrig, A. (eds.) WISA 2013. LNCS, vol. 8267, pp. 3–27. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05149-9_1
11. Wheeler, D.J., Needham, R.M.: Tea, a tiny encryption algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_29

12. Ganesh, V., Hansen, T., Dill, D.L., Cadar, C.: Stp constraint solver (2014). <https://github.com/stp/stp>
13. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers, IACR CryptologyPrint Archive, vol. 2013, p. 404 (2013)



Improved Automatic Search Algorithm for Differential and Linear Cryptanalysis on SIMECK and the Applications

Mingjiang Huang^{1,2(✉)}, Liming Wang¹, and Yan Zhang^{1,2}

¹ State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{huangmingjiang,wangliming,zhangyan}@iie.ac.cn

² University of Chinese Academy of Sciences, Beijing, China

Abstract. In CHES'15, Yang et al. proposed a family of lightweight block cipher SIMECK which combines the good designs of SIMON and SPECK. In this paper, we analysis the properties of the round function of SIMECK, and eliminate the repeated use of rotational independence judgment condition in Liu's algorithm that proposed in FSE'17, constructing the partial difference distribution table with limited Hamming weight of input difference to improve the search results. We get new differentials of 14/21/27 rounds for SIMECK32/48/64 which can provide higher probability than previous results, and find a new 28 rounds differential for SIMECK64. We also get new 13/21/27 rounds linear hulls with higher square correlation for SIMECK32/48/64, and we find new 14/22/28 rounds linear hulls for SIMECK32/48/64, which are the best linear hulls of SIMECK as far as we know. With the application of the new distinguishers and combination with the dynamic key-guessing techniques, we mount key recovery attacks on SIMECK variants, which can reduce the computational complexity and/or data complexity.

Keywords: SIMECK · Differential · Linear hull · Cryptanalysis
Block cipher

1 Introduction

With the development of the Internet of Things, the security issues in the IoT application systems are getting more and more attention. The cryptographic primitives are the basic components of a security application. The research of lightweight cryptographic algorithms aims at protecting the application security for these terminal devices with limited resources. In recent years, various lightweight block ciphers have been proposed, such as: PRINCE [10], PRESENT [9], TWINE [21], SIMON [5], SPECK [5], SIMECK [23], RECTANGLE [25], GIFT [4], etc.

In 2017, authors of SIMON cited the latest researches of cryptanalysis and gave the explanations on the security of SIMON, but still did not give their own

results of SIMON's security analysis [6]. At SAC'17, AlTawy et al. proposed a set of permutation algorithm sLiSCP that based on SIMECK for lightweight sponge cryptographic primitive, used in the sponge framework to construct authenticated encryption, stream cipher, MAC and hash function [2].

At present, there are three mainstream methods to search for the differential/linear distinguishers of SIMON/SIMECK, which are the mixed integer linear programming (MILP) based technique adopted by Sun et al. [20], the SAT/SMT solver method for satisfaction problem solving adopted by Kölbl et al. [12, 13], and the Matsui's branch and bound automated search algorithm adopted by Abed et al. [1], Biryukov et al. [8], and Liu et al. [14]. In the MILP method of Sun et al., the non-independence of the input differences are not considered yet. At CRYPTO'15, Kölbl et al. derived the differential propagation relationship for SIMON-like round function, but it takes much time for the SAT/SMT solver to find the optimal differential trails in the large block size variants of SIMON. At FSE'14, Abed et al. firstly searched for the possible output difference corresponding to the input difference in the SIMON-like round function, and took into account the dependence of the input differences on the bitwise AND operation, but they did not find the optimal differential trails. Biryukov et al. introduced the concept of pDDT, and applied the Matsui's approach to the ARX cipher by using the threshold search method, but by the limitation of the heuristic search methods they used, the optimal differential trails are may not obtained. At FSE'17, Liu et al. separately considered the independence and dependence of the inputs of the bitwise AND operation, and they introduced the concept of small block size DDT of bitwise AND operation with independent inputs, constructing the possible output difference space corresponding to an fixed input difference with a large block size. But in the search algorithm, they reused the independence condition, which will lead more computational efforts. Differential and linear analysis for SIMECK and SIMON are similar. Based on the explicit formula for the differential and linear propagation probability of SIMON-like round function, the optimal differential and linear trails to achieve the security bounds of each variants of SIMON and SIMECK were searched out in [14, 15]. For SIMECK32/48/64, the optimal differential trails cover 13/19/25 rounds, and the optimal linear trails also cover 13/19/25 rounds, respectively. In the previous works, the differentials and linear hulls obtained are based on the optimal trails that do not exceed the security boundary, but the differentials and linear hulls based on lower potential trails are not considered. For SIMECK, the probability of the 14/21/27 rounds differentials and the linear square correlation of the 13/21/27 rounds linear hulls are not tight enough.

For the key recovery attacks, 19/26/33 rounds on SIMECK32/48/64 were attacked by differential cryptanalysis in [13], and 22/28/35 rounds were attacked by dynamic key-guessing technique in [17]. However, the differentials that used to attack SIMECK in [13] and [17] are both 13/20/26 rounds, respectively. Intuitively, with the application of differentials with longer rounds and higher probability, the key recovery attack on SIMECK may need less complexity.

In [18], 13/20/26 rounds linear hulls were used to mount the 23/30/37 rounds key recovery attack on SIMECK32/48/64. However, the squared correlation of the linear hulls they used in the attack are derived from the previous differentials, which are estimated values and not tight enough yet. There are also some other analysis results on SIMECK worth attention under different cryptanalysis model, such as the linear cryptanalysis [3], zero correlation linear cryptanalysis [24], and distinguishing attack [19].

Our Contributions. In this paper, we further investigate the differential and linear propagation properties of the non-linear bitwise AND operation of the round function of SIMECK. We propose improved efficient algorithms to find the possible output differences of nonzero probabilities with a fixed input difference, eliminating the repeated use of the independence judgment condition in Liu's algorithm, and constructing a partial difference distribution table with Hamming weight less than a set threshold. We get 14/21/27 rounds differentials for SIMECK with higher probability than the previous works, and find a new 28 rounds differential for SIMECK64. Simultaneously, we get 13/21/27 rounds linear hulls with higher square correlation for SIMECK32/48/64 respectively. And the 14/22/28 rounds linear hulls for SIMECK32/48/64 we find are the best linear hulls so far. We improve the key recovery attack on SIMECK and reduce the computational complexity and/or data complexity. The 29-round differential attack on SIMECK48 is the longest so far.

Outline. This paper is organized as follows. In Sect. 2, we give the notations used in this paper. In Sect. 3, we give improved efficient algorithms to search for the possible nonzero probability output differences with fixed input difference. And we construct all the possible valid input mask space by using the base vectors of input mask introduced in [15]. In Sect. 4, we apply the obtained differentials to key recovery attacks on all variants of SIMECK for reducing the computational complexity and/or data complexity, as listed in Table 5. Conclusions are given in Sect. 5.

2 Preliminaries

2.1 Notations

The main notations used in this paper are shown in Table 1.

Let $P(\alpha, \beta)$ be the probability of a given input difference α propagate to a given output difference β , which is defined as

$$P(\alpha, \beta) = 2^{-n} \cdot \#\{x : f(x) \oplus f(x \oplus \alpha) = \beta\}.$$

Let $f(x) : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a vectorial boolean function on n bits with the input mask α and output mask β , we denote by $g(\alpha, \beta) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\alpha \cdot x \oplus \beta \cdot f(x)}$, and

Table 1. The notations used in the following paper.

Notation	Description of the notation
\oplus	bitwise XOR
\wedge	bitwise AND
\vee	bitwise OR
$x \lll i$	rotate x to the left by i bits
(x_i, x_{i-1})	the 2n-bit input state of round i , for $1 \leq i \leq r$
x^j	the j th bit of x , $0 \leq j \leq n - 1$
Δx	the difference of $x \oplus x'$
rk_{i-1}	the n bits subkey of round i , for $1 \leq i \leq r$

$g(\alpha, \beta) = \varepsilon \cdot 2^n$ for all $x \in \mathbb{F}_2^n$. Hence, the square correlation of α propagate to β , we denote by

$$C^2(\alpha \rightarrow \beta) = \varepsilon^2 = \left(\frac{g(\alpha, \beta)}{2^n} \right)^2.$$

Under the Markov’s assumption, the probability of a differential (or linear) trail is the product of the probability of each round. Let α be the input difference, and β is the given output difference after r rounds, then the probability of the r -round differential is the sum of all r -round differential trails with the same input and output difference. Similarly, the square correlation of the r -round linear hull is the sum of all r -round linear trails with the same input and output mask.

2.2 Description of SIMECK

The SIMECK family has 3 variants: SIMECK32/64 (32 rounds), SIMECK48/96 (36 rounds) and SIMECK64/128 (44 rounds). They share the same rotational constant set $(a, b, c) = (0, 5, 1)$, with which SIMECK32/48/64 to achieve full diffusion needs 8/9/11 rounds respectively as investigated in [13]. In this paper, we denote the input states of round i by (x_{i+1}, x_i) . The state transformation function of SIMECK can be presented as $x_{i+1} = (x_i \lll a) \wedge (x_i \lll b) \oplus (x_i \lll c) \oplus x_{i-1} \oplus rk_{i-1}$. Let $x_{i+1} = f(x_i) \oplus x_{i-1} \oplus rk_{i-1}$, we generally call $f(x)$ the round function of SIMECK. The differential and linear propagation in the round function of SIMECK is shown in Fig. 1.

3 Automatic Search Algorithm for Differentials and Linear Hulls of SIMECK

3.1 The Properties of SIMECK Round Function

The bitwise AND operation is the only non-linear component in the SIMECK round function, we first study its differential and linear propagation properties.

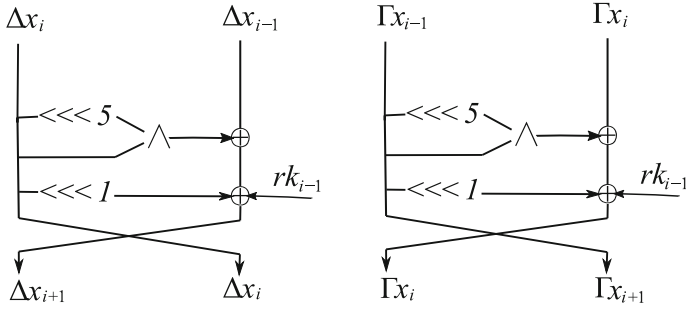


Fig. 1. The differential and linear propagation of the round function of SIMECK.

Table 2. The probability propagation relationship of bitwise AND operation.

α	β	γ	$P(\gamma = 0)$	$P(\gamma = 1)$
0	0	0	1	0
0	1	x	1/2	1/2
1	0	y	1/2	1/2
1	1	$x \oplus y \oplus 1$	1/2	1/2

Property 1. Let $x, x', y, y', \alpha, \beta, \gamma \in \{0, 1\}$, $f(x, y) = x \wedge y$, and $x = x' \oplus \alpha$, $y = y' \oplus \beta$, $\gamma = f(x, y) \oplus f(x', y')$, so α, β, γ have the probability propagation relationship in Table 2. Hence, when $P\{(\alpha, \beta) \rightarrow \gamma\} \neq 0$ is satisfied, if and only if $\bar{\alpha} \wedge \bar{\beta} \wedge \gamma = 0$ is satisfied [1].

According to the definition of differential probability, let α, β, γ be the n bits XOR difference, and $x, x', y, y' \in \mathbb{F}_2^n$, $f(x, y) = x \wedge y$, and $x = x' \oplus \alpha$, $y = y' \oplus \beta$, $\gamma = f(x, y) \oplus f(x', y')$, the probability of two n bits inputs lead to one n bits output is the product of the n probabilities of bitwise operation. As $P\{(\alpha, \beta) \rightarrow \gamma\} = 2^{-2n} \cdot \#\{(x, y) | f(x, y) \oplus f(x \oplus \alpha, y \oplus \beta) = \gamma\}$, which implies the following lemma.

Lemma 1. Let α, β, γ be the n -bit XOR difference, and $x, x', y, y' \in \mathbb{F}_2^n$, $f(x, y) = x \wedge y$, and $x = x' \oplus \alpha$, $y = y' \oplus \beta$, $\gamma = f(x, y) \oplus f(x', y')$, then

$$P\{(\alpha, \beta) \rightarrow \gamma\} = \begin{cases} 2^{-wt(\bar{\alpha} \wedge \bar{\beta})}, & \text{if } \bar{\alpha} \wedge \bar{\beta} \wedge \gamma = \mathbf{0}; \\ 0, & \text{else.} \end{cases}$$

Where $wt(\alpha)$ denotes the Hamming weight of the vector $\alpha \in \mathbb{F}_2^n$, the $\mathbf{0}$ represents an all-zero vector of n bits, and correspondingly $\mathbf{1}$ represents an all-one vector of n bits. When one of the inputs of the bitwise AND is rotated r bits to the left(or right), i.e. $f(x, y) = x \wedge (y \lll r)$, it is easy to get

$$P\{(\alpha, \beta) \rightarrow \gamma\} = \begin{cases} 2^{-wt(\bar{\alpha} \wedge \overline{\beta \lll r})}, & \text{if } \bar{\alpha} \wedge \overline{\beta \lll r} \wedge \gamma = \mathbf{0}; \\ 0, & \text{else.} \end{cases}$$

If the two input values of bitwise AND are mutual rotational dependent of each other, i.e. let $x, x', \alpha, \beta \in \mathbb{F}_2^n, x = x' \oplus \alpha, f(x) = x \wedge (x \lll r),$ and $\beta = f(x) \oplus f(x \oplus \alpha),$ then the differential probability is

$$P\{\alpha \rightarrow \beta\} = 2^{-n} \cdot \#\{x|f(x) \oplus f(x \oplus \alpha) = \beta\},$$

then

$$P\{\alpha \rightarrow \beta\} = 2^{-n} \cdot \#\{x|x \wedge (\alpha \lll r) \oplus \alpha \wedge (x \lll r) \oplus \alpha \wedge (\alpha \lll r) \oplus \beta = \mathbf{0}\},$$

and let

$$L_{\alpha,\beta}(x) = \{x \wedge (\alpha \lll r) \oplus \alpha \wedge (x \lll r) \oplus \alpha \wedge (\alpha \lll r) \oplus \beta = \mathbf{0}\}$$

be a set of equations with variable $x,$ where $\alpha, \beta \in \mathbb{F}_2^n$ as parameters. By solving the number of solutions of $x,$ considering these 2^n equations with the relationship between $(\alpha, \beta),$ Kölbl et al. derived the explicit formula of the differential probability propagation relationship between input and output difference [12]. Therefore, there is an equivalent relationship as shown in Theorem 1. Similarly, the linear square correlation can be defined in Theorem 2 [12].

Theorem 1. *Let α, β be fixed n -bit XOR differences, $x \in \mathbb{F}_2^n, x' = x \oplus \alpha,$ and $f(x) = x \wedge (x \lll r), \beta = f(x) \oplus f(x'),$ define variables $u = (\alpha \lll r) \vee \alpha,$ and $v = \alpha \wedge (\alpha \lll r) \wedge (\alpha \lll 2r),$ so the differential propagation probability of the bitwise AND with two inputs mutual rotational dependent is*

$$P\{\alpha \rightarrow \beta\} = \begin{cases} 2^{-n+1}, & \text{if } \alpha = \mathbf{1}, \text{ and } wt(\beta) \equiv 0 \pmod 2; \\ 2^{-wt(v \oplus u)}, & \text{if } \alpha \neq \mathbf{1}, \text{ and } \beta \wedge \bar{v} = \mathbf{0}, \text{ and } (\beta \oplus (\beta \lll r)) \wedge u = \mathbf{0}; \\ 0, & \text{else.} \end{cases}$$

Theorem 2. *Let α, β be an input and an output mask, $f(x) = x \wedge (x \lll r),$ where $x \in \mathbb{F}_2^n,$ and the set U_β is defined by $U_\beta = \{x|(\beta \wedge (x \lll r)) \oplus ((\beta \wedge x) \ggg r) = \mathbf{0}\},$ the dimension of U_β is $d = \dim U_\beta,$ and U_β^\perp is the orthogonal complement space of $U_\beta.$ Hence, the squared correlation calculation of $f(x)$ that α propagate to β can be denoted by*

$$C^2(\alpha, \beta) = \begin{cases} 2^{-n+2}, & \text{if } \beta = \mathbf{1}, \text{ and } \alpha \in U_\beta^\perp; \\ 2^{-n+d}, & \text{if } \beta \neq \mathbf{1}, \text{ and } \alpha \in U_\beta^\perp; \\ 0, & \text{else.} \end{cases}$$

Corollary 1. *Let α, β be fixed n -bit XOR differences, $x, x' \in \mathbb{F}_2^n, x' = x \oplus \alpha,$ $f(x) = x \wedge (x \lll r),$ and $\beta = f(x) \oplus f(x'),$ so the probability $P(\alpha \rightarrow \beta)$ is only related to the value of the input difference $\alpha,$ and the differential probability corresponding to each valid output difference β is equivalent. And the number of valid output differences is equal to $1/P\{\alpha \rightarrow \beta\},$ where $P\{\alpha \rightarrow \beta\} \neq 0.$*

Corollary 2. *Let $\{\Delta x_1, \Delta x_0\}$ be the $2n$ -bit input XOR difference, $\{\Delta x_{i+1}, \Delta x_i\}$ is the $2n$ -bit output XOR difference of i round differential of SIMECK $2n,$ and the probability is $P.$ There exist other $n - 1$ differentials with similar probability, the input difference and output difference can be constructed as $\{\Delta x_1 \lll j, \Delta x_0 \lll j\}$ and $\{\Delta x_{i+1} \lll j, \Delta x_i \lll j\}$ respectively, for $j \in [1, n - 1].$*

3.2 Improved the Differentials of SIMECK Variants

When the inputs of the bitwise AND are independent of each other, i.e. let $f(x, y) = x \wedge y$, the DDT can be constructed directly according to Lemma 1, as shown in Algorithm 1. If the inputs of the bitwise AND are mutually rotational dependent, i.e. let $f(x) = x \wedge (x \lll r)$, the difference distribution table of bitwise AND can be deduced by Theorem 1. Algorithm 2 constructs the nonzero probability difference distribution table of the bitwise AND with inputs rotational dependent. When the block size n is not too large, generally when $n \leq 16$, the difference distribution table (DDT) generated by it can be stored feasibly. The storage size of the DDT of the bitwise AND operation with input rotational dependent can be reduced from 2^{2n} to $2^n \cdot \mathcal{O}$, when $n=16$, $\mathcal{O} \approx 2^{8.62}$.

Algorithm 1. Given two m -bit input difference α, β , constructing the nonzero probability difference distribution table $DDT_m(\alpha, \beta)$.

```

1: for each  $\alpha, \beta = 0$  to  $2^m - 1$  do
2:   cnt = 0;
3:   for  $\gamma = 0$  to  $2^m - 1$  do
4:     if  $\gamma \wedge (\bar{\alpha} \wedge \bar{\beta}) = \mathbf{0}$  then
5:       beta[ $\alpha || \beta$ ][cnt + +] =  $\gamma$ ;
6:     end if
7:   end for
8:   p[ $\alpha || \beta$ ] = 1/cnt;
9: end for

```

When the inputs of the bitwise AND are mutually rotational dependent and the block length is larger than 16 bits, it will be difficult to store the large DDT produced by Algorithm 2 directly. We are inspired by Liu et al., in order to reduce the storage complexity, consider the independent and dependent conditions of the inputs separately. Firstly, assuming that the inputs are independent of each other, referring to Algorithm 1, a large block of blocksize n can be split into several small blocks with m bits length each, and the possible output differences corresponding to the fixed input difference are reconstructed by the output of small blocksize DDT whose inputs are mutually independent. For example, the blocksize of a cipher is $2n = 64$, let $m = 8$ in Algorithm 1, the input of round function with $n = 32$ bits length can be split into four 8-bit blocks, then for each block just lookup the DDT produced by Algorithm 1, and then recombine the 4 sets of 8-bit possible output difference which construct the 32 bits length possible output differences. Secondly, verifying whether the recombined output differences are valid possible output differences with nonzero probability or not through the judgment condition in Theorem 1, and considering the constraints of output difference and input inter dependence. Here, for filtering out the possible output differences of nonzero probability, we eliminate the repeated use of input independent condition (i.e. $\beta \wedge (\bar{\alpha} \vee \alpha \lll r) = \mathbf{0}$) in Liu’s algorithm. Thirdly, considering the increasement in the Hamming weight of the input difference α ,

Algorithm 2. Computing the difference distribution table $DDT_n(\alpha)$ when the inputs are mutually dependent.

```

1:  $p[2^n] = \{0\}$ ,  $\beta\_value[2^n][ ] = \{0\}$ ,  $\beta\_num[2^n] = \{0\}$ ;
2: for  $\alpha = 0$  to  $2^n - 2$  do
3:    $u = \alpha \vee \alpha \lll 5$ ;
4:    $v = \alpha \wedge \bar{\alpha} \lll 5 \wedge \alpha \lll 10$ ;
5:   for  $\beta = 0$  to  $2^n - 1$  do
6:     if  $(\bar{u} \wedge \beta) \vee (v \wedge (\beta \oplus \beta \lll 5)) = 0$  then
7:        $\beta\_value[\alpha][\beta\_num[\alpha]] = \beta$ ;
8:        $\beta\_num[\alpha] ++$ ;
9:     end if
10:  end for
11:   $p[\alpha] = 1/\beta\_num[\alpha]$ ;
12: end for
13: let  $\alpha = 2^n - 1$ ;
14: for  $\beta = 0$  to  $2^n - 1$  do
15:  if  $wt(\beta) \equiv 0 \pmod{2}$  then
16:     $\beta\_value[2^n - 1][\beta\_num[2^n - 1]] = \beta$ ;
17:     $\beta\_num[2^n - 1] ++$ ;
18:  end if
19: end for
20:  $p[2^n - 1] = 1/\beta\_num[2^n - 1]$ ;
21: return  $p[ ]$ ,  $\beta\_value[ ]$ ;

```

the probability of the corresponding output difference decrease, shown in [7, 14]. When the Hamming weight of the input difference increases, the upper bound of the probability of the round function will also decrease, which lead to the Matsui's pruning condition in the search procedure will not be satisfied mostly. The output differences and probabilities corresponding the low Hamming weight input difference are used frequently. Inspired by the concept of partial difference distribution table introduced by Biryukov et al. [8], we can just precompute and store the difference distribution table corresponding to the input difference with low Hamming weight, where the Hamming weight is less than a certain threshold. When the input difference Hamming weight is smaller than the set threshold, just look up the precomputed table, otherwise, calculating the possible output difference from $POD_n(\alpha)$ in Algorithm 3.

To search for the differentials of SIMECK, we firstly employ the Matsui's branch-bound search approach similar to [14] for finding the optimal differential characteristics(trails). Try to search for longer rounds of differential trails that exceed security bound with limiting the Hamming weight of the input difference according to the differential probability upper bound of the input Hamming weight [7, 14]. Afterward, we fix the input difference α and output difference β to find enough amount of trails, and statistic the probability of all trails. In order to effectively search for trails as much as possible within a feasible time, we limit the search range of probability weights ($-\log_2(p)$) from wt_{\min} to wt_{\max} . And wt_{\min} is the probability weight of the differential characteristic, wt_{\max} is the probability

Algorithm 3. Given a n -bit input difference α , computing the set $POD_n(\alpha)$ of possible output differences β with nonzero probability.

```

Input:  $\alpha \in \mathbb{F}_2^n, n = mt;$ 
1:  $p = 0, \beta\_value[] = \{0\}, \beta\_num = 0;$ 
2: if  $\alpha \neq 2^n - 1$  then
3:   let  $X = \alpha, Y = \alpha \lll 5;$ 
4:   divide  $X = \{x_{t-1}, \dots, x_0\}, Y = \{y_{t-1}, \dots, y_0\}, x_i, y_i \in \mathbb{F}_2^m;$ 
5:   for  $i = 0$  to  $t - 1$  do
6:     lookup  $DDT_m(x_i, y_i);$  //Computed by Algorithm 1.
7:      $\beta^i[cnt^i] = beta[x_i, y_i][cnt^i] \in \{0, 1\}^m$ 
8:   end for
9:   for each  $\beta := \{\beta^{t-1}[cnt^{t-1}] || \dots || \beta^0[cnt^0]\}$  do
10:    if  $(\beta \oplus \beta \lll 5) \wedge (\alpha \wedge \alpha \lll 5 \wedge \alpha \lll 10) = 0$  then
11:       $\beta\_value[\beta\_num++] = \beta;$ 
12:    end if
13:  end for
14: else
15:   for  $\beta = 0$  to  $2^n - 1$  do
16:    if  $wt(\beta) \equiv 0 \pmod 2$  then
17:       $\beta\_value[\beta\_num++] = \beta;$ 
18:    end if
19:  end for
20: end if
21:  $p = 1/\beta\_num;$ 
22: return  $p, \beta\_value[];$ 

```

weight of the minimal probability that be limited. In the first two rounds of the search process, using the left and right part of the input difference as the inputs of round function. During the branch search process of differential trails of 3 to $r - 1$ rounds, searching the possible output differences corresponding to the input difference by lookup table generated by Algorithm 4, when the Hamming weight of input difference is less than H . Otherwise utilizing the Algorithm 3 to generate the possible output differences. In the process of the middle rounds, we use Matsui’s branch pruning condition $wt(p_1) + wt(p_1) + \dots + wt(p_i) + wt(p_{r-i}) \leq wt_{max}$ as the stop condition, when the probability of searched truncated path is larger than the set value, remove it in advance. Even so, there are still some trails in the front $r - 1$ round maybe satisfying the brach pruning condition, but when multiplied by the differential probability of the last round, the probability weight will larger than the limited wt_{max} , while they are also propagate to the same output difference of the differential. In our statistical approach, these part trails with lower probability are also counted. Probability weight marked by * in Table 3 means there are some trails with lower probability than the set probability weight wt_{max} be counted. The differential probability is calculated

$$by P = \sum_{w=wt_{min}}^{wt_{max}} \#trails[w] \times 2^{-w}.$$

Algorithm 4. Pre-calculating and store the partial difference distribution table $PDDT_{n,H}(\alpha)$ with Hamming weight less than H .

```

1: for  $wt(\alpha) = 0$  to  $H$  do
2:    $POD_n(\alpha)$ ;
3:    $\beta[\alpha][ ] \leftarrow \beta\_value[ ]$ ;
4:    $p[\alpha] = p$ ;
5: end for
    
```

Table 3. The Differentials of SIMECK variants.

Cipher	Rd	Δ_{in}	Δ_{out}	wt_{min}	wt_{max}	Prob	#Trails	Ref
SIMECK32	13	0,2	2,0	38	52	$2^{-28.91}$	1846518	[18]
	13	8000,4011	4000,0	36	49	$2^{-27.28}$	/	[13]
	14	1,800A	4,8002	/	/	$2^{-31.64}$	/	[14]
	14	2,15	8,5	36	57	$2^{-31.63}$	8065284	This paper
	14	0,2	4,2	40	53	$2^{-30.90}$	1678405	This paper
SIMCEK48	21	20000,470000	50000,20000	/	100	$2^{-45.65}$	/	[13]
	21	1,800002	800002,1	/	/	$2^{-45.28}$	/	[14]
	21	2,5	5,2	52	73	$2^{-45.18}$	34899905	This paper
SIMECK64	26	0,4400000	8800000,400000	/	121	$2^{-60.02}$	/	[13]
	27	0,10	5,2	/	/	$2^{-61.49}$	/	[14]
	27	0,11	5,2	70	89*	$2^{-60.75}$	32649265	This paper
	28	0,11	A8,5	74	93*	$2^{-63.91}$	617703755	This paper

For a differential of i rounds, take the output difference of the i_{th} round as the input difference of $(i + 1)_{th}$ round, according to Algorithm 3, one more round can be extended by check the number of output difference. For every valid possible output difference of $(i + 1)_{th}$ round, probability of the new $i + 1$ round differential can be calculated by Corollary 1. The obtained differentials for SIMECK is shown in Table 3, in which the number of possible output difference corresponding to the output difference of the 13-round differential $(0x0, 0x2 \rightarrow 0x2, 0x0)$ used in [17] of SIMECK32 is 4. Hence, there exists 14 rounds differential of probability at least $\frac{1}{4} \times 2^{-28.91}$, the input difference of it is $(0x0, 0x2)$ and the output difference is one of $\{(0x4, 0x2), (0x6, 0x2), (0x44, 0x2), (0x46, 0x2)\}$, and the searched experimental result of the probability is $2^{-30.90}$. In addition, the result of 27/28 round differential of SIMECK64 that also confirmed the guesswork of Corollary 1. Additionally, we also searched out some differentials that follow Corollary 2, such as the 14-round differential $(2, 15) \rightarrow (8, 5)$, which is the rotational pair of $(1, 800A) \rightarrow (4, 8002)$ that rotated 1-bit to the left for each half state¹. More differentials can be constructed from Table 3 by the Corollary 2, for example, the 14-round differential of SIMECK32 implies $((0, 2) \rightarrow (4, 2)) \Rightarrow ((0 \lll j, 2 \lll j) \rightarrow (4 \lll j, 2 \lll j))$, $j \in [1, 15]$ with the similar probability that larger than $2^{-30.90}$. And the derivation process for SIMECK48/64 is similar. The obtained 14/21/27 rounds differential of SIMECK32/48/64 are the best so far, meanwhile, the 28-round differential of SIMECK64 is the longest so far.

¹ The reason why the experimental result of the probability is higher than that of [14] is because of more trails are counted.

Algorithm 5. Using the non-zero base vectors of α to construct the space of all valid possible α .

Input: Nonzero vectors of $y[i]$ construct the bases of U_{β}^{\perp} , $i \in [0, n - 1]$.

```

1:  $\alpha_{num} = 0, \alpha[ ] = \{0\}, z[ ] = \{0\}, j = 0;$ 
2: for  $i = 0$  to  $n - 1$  do
3:   if  $y[i] \neq 0$  then
4:      $z[j] = y[i];$ 
5:      $j + +;$  //Record the number of non-zero base vectors.
6:   end if
7: end for
8:  $\alpha_{num} = 2^j;$ 
9: for  $k = 0$  to  $\alpha_{num} - 1$  do //Construct each valid  $\alpha$ .
10:  for  $t = 0$  to  $j - 1$  do
11:    if  $(k \wedge (1 \ll t)) \neq 0$  then
12:       $\alpha[k] = \alpha[k] \oplus z[t];$ 
13:    end if
14:  end for
15: end for
16: return  $\alpha_{num}, \alpha[ ];$ 

```

3.3 Improved the Linear Hulls of SIMECK Variants

In [15], an automatic search algorithm to search for the optimal linear trails of SIMON and SIMECK are proposed. They gave an algorithm to find the base of all possible input masks $\alpha \in U_{\beta}^{\perp}$ in Theorem 2. By using the base vectors of α , we can construct all the possible valid input masks by Algorithm 5. Then, we use the method in [15] and search for longer round linear trails with the square correlation exceed the security boundary. Take the input and output mask of the trails as that of the linear hull, and we limit the search scope by the lower bound of the square correlation ($-\log_2 C^2(\alpha, \beta)$). Also, setting the branch pruning condition $wt(p_1) + wt(p_1) + \dots + wt(p_i) + wt(p_{r-i}) \leq wt_{\max}$ as the stop condition. By using the longer round linear trails with square correlation exceed the security boundary, the new longer round linear hulls are found in Table 4. The 14/22/28 rounds linear hulls of SIMECK32/48/64 are the longest linear hulls so far².

4 Key Recovery Attack on Round Reduced SIMECK

In 2014, Wang et al. proposed dynamic key-guessing techniques in differential attack on SIMON [22], and then these techniques had also been used to achieve linear hull attack on SIMON at FSE'16 [11]. The differential and linear hull attack on SIMECK with dynamic key-guessing techniques are the most efficient method until now [17, 18]. Hence, by applying the new differentials obtained, we

² All experiments code are runned on a PC with Intel[®] Core[™] i7-2600 CPU@3.40GHz \times 8.

Table 4. The Linear hulls of SIMECK variants.

Cipher	Rd	Γ_{in}	Γ_{out}	wt_{min}	wt_{max}	Potential	#Trails	Ref
SIMECK32	13	2,0	0,2	40	/	$2^{-30.91}$	1846518	[18]
	13	11,0	2,15	/	/	$2^{-29.43}$	/	[15]
	13	22,0	4,2A	32	38*	$2^{-28.11}$	876	This paper
	14	800A,1	8008,4004	36	42*	$2^{-31.90}$	1456	This paper
SIMECK48	20	280000,100000	200000,100000	/	/	$2^{-45.66}$	/	[18]
	21	800002,1	1,800002	/	/	$2^{-46.30}$	/	[15]
	21	880002,1	1,800002	52	64*	$2^{-44.48}$	352999	This paper
	22	800000,1	200000,500001	56	68*	$2^{-47.68}$	1326121	This paper
SIMECK64	26	440000,0	400000,200000	/	/	$2^{-62.09}$	/	[18]
	27	11,0	8,14	/	/	$2^{-61.14}$	/	[15]
	27	22,0	10,28	70	86*	$2^{-59.79}$	27489363	This paper
	28	80000001,5	0,40000004	74	88	$2^{-63.67}$	9103911	This paper

try to get some new results for the key recovery attacks on SIMECK variants. And for the attack on SIMECK with the new linear hulls obtained, better results with less complexity may also occur combining with dynamic key-guessing techniques in linear hulls attack. Even so, we leave it as subsequent researchs because the attack process is too cumbersome, it is recommended to refer to [11] for the process details of using linear hulls for key recovery.

4.1 Dynamic Key-Guessing in Differential Attack

In [17,22], the dynamic key guessing technique was introduced to the differential attack on SIMON and SIMECK. By observing the round function of SIMECK, the bit-transformation relationship of it is denoted by follows. Let $x_i = \{x_i^{n-1}x_i^{n-2} \dots x_i^j \dots x_i^0\}$, $x_i^j \in \{0, 1\}$, there have,

$$x_{i+1}^j = x_i^j \wedge x_i^{j-5} \oplus x_i^{j-1} \oplus x_{i-1}^j \oplus rk_{i-1}^j.$$

The bits relationship of the differential propagation of the SIMECK round function is expressed as follows:

$$\Delta x_{i+1}^j = \Delta x_i^j \wedge x_i^{j-5} \oplus x_i^j \wedge \Delta x_i^{j-5} \oplus \Delta x_i^j \wedge \Delta x_i^{j-5} \oplus \Delta x_i^{j-2} \oplus \Delta x_{i-1}^j$$

and the plaintext bits x_i^j and x_i^{j-5} involved secret subkey bits for $i \geq 2$.

$$x_i^j = x_{i-1}^j \wedge x_{i-1}^{j-5} \oplus x_{i-1}^{j-1} \oplus x_{i-2}^j \oplus rk_{i-1}^j$$

$$x_i^{j-5} = x_{i-1}^{j-5} \wedge x_{i-1}^{j-10} \oplus x_{i-1}^{j-6} \oplus x_{i-2}^{j-5} \oplus rk_{i-1}^{j-5}$$

As the chosen plaintexts are known, considering Δx_i^j and Δx_i^{j-5} as parameters under different values, and discussing the number of subkey bits satisfy the equations. When some subkey bits are determined, which can reduce the number of the remaining subkey bits that need to be exhaustive searched.

If $(\Delta x_i^j, \Delta x_i^{j-5}) = (0, 0)$, $\Delta x_{i+1}^j = b$, $b \in \{0, 1\}$, when $\Delta x_i^{j-2} \oplus \Delta x_{i-1}^j \neq b$, the differential propagation relationship of round function is not satisfied, this condition can be used to filter out the wrong plaintext pairs in the first and last round. When $\Delta x_i^{j-2} \oplus \Delta x_{i-1}^j = b$, the bit differential propagation relationship can not be used for determining subkey bits, so $(rk_{i-1}^j, rk_{i-1}^{j-5}) \in \mathbb{F}_2^2$ with 4 solutions. And if $(\Delta x_i^j, \Delta x_i^{j-5}) = (0, 1), (1, 0)$ or $(1, 1)$, then $(rk_{i-1}^j, rk_{i-1}^{j-5})$ have only 2 solutions. After appending r_0 rounds forward and r_1 rounds backward, generate the extended path with sufficient bit conditions, from which the related subkey bits for each sufficient bit condition can be solved by whether to satisfy the equations.

4.2 Applying the Differentials to Key Recovery Attack on SIMECK Variants

In [17, 22], to extend the differential path according to the rules that the output differences of AND operation is 0, if and only if its input differences are (0,0), otherwise set the output difference of AND operation to * as uncertain bit. However, the rotational dependence of the input differences of AND operation is not considered in these previous works. To extend the differential path with adding r_0 round forward and r_1 round backward, we use the nonzero probability outputs produce by Algorithm 3. In the data collection phase, there are Q_{r_0} possible plaintext differences in the set $\{\Delta_P\}$, which will lead to at least 1 input difference Δ_{in} of the r -round truncated differential that be appended r_0 round forward. There are Q_{r_1} possible ciphertext output differences in the set $\{\Delta_C\}$, which can be deduced from 1 output difference Δ_{out} of the r -round truncated differential that be appended r_1 round backward. The number of possible plaintext difference Q_{r_0} is less than the plaintext pairs in each data structure constructed in [22]. Selecting one plaintext X , then the plaintexts set $\{X \oplus \Delta_P\}$ obtained by Q_{r_0} XOR additions which can yield Q_{r_0} plaintext pairs which lead to at least one input difference Δ_{in} . Choosing 2^t arbitrary plaintext X , for example $X \in \{0, 1\}^t$, then we can get $2^t \times Q_{r_0}$ plaintext pairs that lead to 2^t pairs with intermediate state in $(r_1)_{th}$ round with Δ_{in} . Since the set of ciphertext differences resulting from the output difference of chosen differential, there exists conditions that some bit positions of the ciphertext difference are fixed, which can be used to filter out the invalid plaintext difference in $\{\Delta_P\}$, so does the invalid plaintexts, and then check the remaining ciphertext belong to $\{\Delta_C\}$ or not, which can reduce the storage complexity.

For SIMECK32/64, similarly to the attack in [17], we use the 14-round differential $\Delta_{in} : 0x0000\ 0002 \rightarrow \Delta_{out} : 0x0004\ 0002$ with the probability of $2^{-30.90}$ in Table 3, and extended 4 rounds forward and 4 rounds backward of it to mount the key recovery attack against the 22 rounds of SIMECK32/64. The extended differential path with sufficient bit conditions listed in Table 6.

In the data collection phase, we choose 2^{32} plaintexts, with $2^{32} \times Q_{r_0} \approx 2^{32+21.3} = 2^{53.3}$ times XOR addition, $2^{53.3}$ plaintext pairs can be get, which will lead to $2^{32} \times p = 2^{32-30.90} \approx 2.14$ right pairs occurred in average. For the chosen

Table 5. Comparison of Differential Cryptanalysis on SIMECK.

Cipher	Total Rounds	Diff. Rounds	Diff. Prob	Attacked Rounds	Time	Data	Ref
SIMECK32	32	13	$2^{-27.28}$	19	2^{40}	2^{31}	[13]
		13	$2^{-29.64}$	22	$2^{57.9}$	2^{32}	[17]
		14	$2^{-30.90}$	22	$2^{54.3} A^a + 2^{56} E^b$	2^{32}	This paper
SIMECK48	36	20	$2^{-43.65}$	26	2^{62}	2^{47}	[13]
		20	$2^{-43.65}$	28	$2^{68.3}$	2^{46}	[17]
		21	$2^{-45.18}$	29	$2^{77.04} A + 2^{83.14} E$	2^{47}	This paper
SIMECK64	44	26	$2^{-60.02}$	33	2^{115}	2^{63}	[13]
		26	$2^{-60.02}$	35	$2^{116.3}$	2^{63}	[17]
		27	$2^{-60.75}$	35	$2^{90.6} A + 2^{105.5} E$	2^{62}	This paper

^a ‘A’ represents a XOR addition operation.
^b ‘E’ represents an encryption of attacked rounds

Table 6. Extended Differential Path of 22-round SIMECK32 with sufficient conditions.

Rd	Input difference of each round represented in bits	Q_{r_0}
0	0,0,0,*,*,0,0,*,*,*,0,1,*,*,*,*,0,0,*,*,*,0,*,*,*,*,*,*,*,*,*	2589180
1	0,0,0,0,*,0,0,0,*,*,0,0,1,*,*,0,0,0,0,*,*,0,0,*,*,*,0,1,*,*,*,*	5638
2	0,0,0,0,0,0,0,0,0,0,*,0,0,0,1,*,0,0,0,0,0,*,0,0,0,*,*,0,0,1,*,*,0	68
3	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,*,*,0,0,1,*,*,0	4
4	0,1,0	1
4-18	$\Delta in : 0x0000\ 0002 \rightarrow \Delta out : 0x0004\ 0002$	Q_{r_1}
18	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0	1
19	0,0,0,0,0,0,0,0,*,0,0,0,1,*,*,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0	4
20	0,0,0,*,*,0,0,*,*,*,0,1,*,*,*,0,0,0,0,0,0,0,0,*,*,0,0,0,1,*,*,0	176
21	0,0,*,*,*,0,*,*,*,*,1,*,*,*,0,0,0,0,*,0,0,0,*,*,*,0,1,*,*,*,0	34336
22	0,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,0,0,*,*,*,0,*,*,*,*,1,*,*,*,*,0	30906788

plaintext pairs, after $2^{32} \times (Q_{r_0} + 1) \approx 2^{53.3}$ times $(r_0 + r + r_1)$ rounds encryption, using the fixed 6 bits of the ciphertext difference as conditions to filter out the wrong pairs with $2^{47.3}$ pairs left in approximately. Then use the $\{\Delta_C\}$ as the filter oracle, $2^{47.3} / Q_{r_1} \approx 2^{47.3-24.85} = 2^{22.45}$ pairs remained and should be stored in table T.

Considering the recovery of 4 consecutive rounds of subkeys, by partial decrypt the last four round, there are totally 35 bits of $rk_{21}^{10,15}$, $rk_{20}^{0,5,9-11,14,15}$, $rk_{19}^{10,15}$, $rk_{18}^{0,4-5,8-11,13-15}$ involved in the 18 bit conditions in the 18 to 22 rounds. Create a list of counters for the 35 subkey bits, and solve the 18 bit condition equations with each pairs remained in T. If the candidate subkey bits matches the equations then increment the counter. The counter associated with the candidate subkey bits has the highest count value.

For the data complexity, there requires 2^{32} plaintexts, and 3 tables of $\{\Delta_P\}$, $\{\Delta_C\}$ and T with $2^{21.3} + 2^{24.85} + 2^{22.45} \approx 2^{26.3}$ storage size. The computa-

Table 7. Extended Differential Path of 29-round SIMECK48 with sufficient conditions.

Rd	Input difference of each round represented in bits	Q_{r_0}
0	00000000*0***** 0000000*****	$\approx 2^{29.41}$
1	00000000000*0*0***1*** 00000000*0*****	950080
2	000000000000000*0*01*0* 00000000000*0*0***1***	1156
3	000000000000000000000101 000000000000000*0*01*0*	16
4	000000000000000000000010 00000000000000000000101	1
4-25	$\Delta_{in} : 0x000002\ 000005 \rightarrow \Delta_{out} : 0x000005\ 000002$	Q_{r_1}
25	0000000000000000000000101 000000000000000000000010	1
26	000000000000000*0*01*0* 000000000000000000000101	16
27	00000000000*0*0***1*** 000000000000000*0*01*0*	1156
28	00000000*0***** 00000000000*0*0***1***	950080
29	0000000***** 00000000*0*****	$\approx 2^{29.41}$

tional effort contains the XOR addition, encryptions, filtering phase, subkey bits guessing, and the brute-force search phase. The computational time complexity $C_t = (2^{32} \times 2^{21.3})A + (2^{32} \times (2^{21.3} + 1))E + ((2^{53.3})A + (2^{47.3})A) + (2 \cdot 2^{22.45} \cdot 2^{35} \cdot \frac{4}{22})E + 2^{64-35}E \approx 2^{54.3}A + 2^{56}E$. Here, ‘A’ represents a XOR addition operation and ‘E’ represents the encryption operation of attacked rounds. For the calculation of the success probability, we refer to the theory in [22], the success probability equals to $1 - Poisscdf(s, \lambda_r)$, where $s = \lfloor \lambda_r \rfloor$ is the number of hits that no more than right pairs λ_r , and $Poisscdf(s, \lambda_r)$ is the probability density function of Poisson distribution. Hence, we can get the success probability is 73% for the attack on SIMECK32.

For SIMECK48/96, we use a 21-round differential $\Delta_{in} : 0x0002\ 0005 \rightarrow \Delta_{out} : 0x0005\ 0002$ with the probability of $2^{-45.18}$ in Table 3, and add 4 rounds forward and 4 rounds backward of it to mount the key recovery attack against the 29 rounds of SIMECK48/96. The extended differential path with sufficient bit conditions listed in Table 7.

We choose 2^{47} plaintexts, with $2^{47} \times Q_{r_0} \approx 2^{47+29.41} = 2^{76.41}$ times XOR addition, $2^{76.41}$ plaintext pairs can be get, which will lead to $2^{47-45.18} \approx 3.53$ right pairs occurred in average. After $2^{47} \times (Q_{r_0} + 1) \approx 2^{76.41}$ times encryption for all pairs, we use the fixed 16 difference bits in the last round to filter out most part of wrong pairs with $2^{60.41}$ pairs left. Then use the $\{\Delta_C\}$ as the filter oracle, $2^{60.41-29.41} = 2^{31}$ pairs remained and should be stored in table T. Create counters for the candidate 54 subkey bits $rk_{27}^{9,11-23}, rk_{26}^{4,6,8-23}, rk_{25}^{1,3-23}$ involved in the last 4 rounds, and solve the 32 bit condition equations with each pairs remained in T. For the data complexity, there requires 2^{47} chosen plaintexts, and $2^{29.41} + 2^{29.41} + 2^{31} \approx 2^{31.5}$ storage size for $\{\Delta_P\}, \{\Delta_C\}$ and table T is needed. And for the computational effort, there needs $C_t = (2^{76.41})A + (2^{76.41})E + ((2 \cdot$

Table 8. Extended Differential Path of 35-round SIMECK64 with sufficient conditions.

Rd	Input difference of each round represented in bits
0	0000000000000000*00***0***1*** 00000000000000***0*****
1	0000000000000000*000**00***01** 0000000000000000*00***0***1***
2	0000000000000000000000*000**001* 0000000000000000*00***0***01**
3	0000000000000000000000000000010001 00000000000000000000*00***001*
4	000000000000000000000000000000000 00000000000000000000000010001
4-31	$\Delta_{in} : 0x00000000\ 00000011 \rightarrow \Delta_{out} : 0x00000005\ 00000002$
31	0000000000000000000000000000000101 0000000000000000000000000000010
32	000000000000000000000000*0*01*0* 000000000000000000000000000000101
33	00000000000000000000*0*0***1*** 0000000000000000000000*0*01*0*
34	0000000000000000*0***** 000000000000000000*0*0***1***
35	0000000000000000***** 0000000000000000*0*****
Q_{r_0}	$\approx 2^{28.6}, 363076, 1156, 16, 1$ for Rd.= 0,1,2,3,4,respectively
Q_{r_1}	$1, 16, 1156, 950080, \approx 2^{29.41}$ for Rd.= 31,32,33,34,35,respectively

$2^{76.41})A + (2 \cdot 2^{60.41})A) + (2 \cdot 2^{31} \cdot 2^{54} \cdot \frac{4}{29})E + 2^{96-54}E \approx 2^{77.04}A + 2^{83.14}E$, and the success probability is 78%.

For SIMECK64/128, we use the 27-round differential $\Delta_{in}:0x00000000\ 00000011 \rightarrow \Delta_{out}:0x00000005\ 00000002$ with the probability of $2^{-60.75}$ in Table 3, by appending 4 rounds on the top and 4 rounds at the bottom, and extend it to mount the key recovery attack against the 35 rounds of SIMECK64/128. The extended differential path of the 35 rounds SIMECK64/128 is listed in Table 8. Choosing 2^{62} plaintexts, we can get $2^{90.6}$ plaintext pairs with $2^{90.6}$ XOR additions, and $2^{62-60.75} \approx 2.13$ right pairs occurred in average. For all chosen plaintext pairs, there need $2^{90.6}$ encryptions of 35 rounds which lead to $2^{58.6}$ pairs left, and then use the $\{\Delta_C\}$ as the filter oracle, $2^{29.19}$ pairs stored in table T. There are 78 bits of $rk_{34}^{9,11-31}, rk_{33}^{4,6,8-31}, rk_{32}^{1,3-31}$ can be guessed. There requires 2^{62} chosen plaintexts for data complexity, and storage complexity is $2^{28.6} + 2^{29.41} + 2^{29.19} \approx 2^{30.7}$ for $\{\Delta_P\}, \{\Delta_C\}$ and table T. For the computational complexity, there needs $C_t = (2^{90.6})A + (2^{90.6})E + ((2^{58.6})A + (2^{29.19})A) + (2 \cdot 2^{29.19} \cdot 2^{78} \cdot \frac{4}{35})E + 2^{128-78}E \approx 2^{90.6}A + 2^{105.5}E$, and the success probability is 73%.

5 Conclusions

In this paper, we analyzed the differential and linear propagation properties of the bitwise AND operation with inputs mutually independent or dependent, and improved efficient algorithms by constructing the partial difference distribution table for bitwise AND operation. We searched out new differentials and linear

hulls for SIMECK, which are the best results as far as we know. We applied our differentials to the key recovery attack on SIMECK and less complexity required while compared to previous differential attack.

Acknowledgements. The authors are very grateful to the anonymous reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0801900).

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
2. AlTawy, R., Rohit, R., He, M., Mandal, K., Yang, G., Gong, G.: sLiSCP: Simeck-based permutations for lightweight sponge cryptographic primitives. In: Selected Areas in Cryptography - SAC 2017–24th International Conference, Ottawa, ON, Canada, 16–18 August 2017, Revised Selected Papers, pp. 129–150 (2017)
3. Bagheri, N.: Linear cryptanalysis of reduced-round SIMECK variants. In: Biryukov, A., Goyal, V. (eds.) INDOCRYPT 2015. LNCS, vol. 9462, pp. 140–152. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26617-6_8
4. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: a small present. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_16
5. 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)
6. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: Notes on the design and analysis of SIMON and SPECK. IACR Cryptology ePrint Archive 2017/560 (2017)
7. Beierle, C.: Pen and paper arguments for SIMON and SIMON-like designs. In: Zikas, V., De Prisco, R. (eds.) SCN 2016. LNCS, vol. 9841, pp. 431–446. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44618-9_23
8. 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
9. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
10. Borghoff, J., et al.: Prince - a low-latency block cipher for pervasive computing applications (full version). In: Proceedings of Advances in Cryptology - ASIACRYPT 2012–18th International Conference on the Theory and Application of Cryptology and Information Security (2012)
11. Chen, H., Wang, X.: Improved linear hull attack on round-reduced SIMON with dynamic key-guessing techniques. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 428–449. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_22

12. 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
13. Kölbl, S., Roy, A.: A brief comparison of SIMON and SIMECK. In: Bogdanov, A. (ed.) LightSec 2016. LNCS, vol. 10098, pp. 69–88. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55714-4_6
14. Liu, Z., Li, Y., Wang, M.: Optimal differential trails in simon-like ciphers. IACR Trans. Symmetric Cryptol. **2017**(1), 358–379 (2017)
15. Liu, Z., Li, Y., Wang, M.: The security of simon-like ciphers against linear cryptanalysis. IACR Cryptology ePrint Archive 2017/576 (2017)
16. 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>
17. Qiao, K., Hu, L., Sun, S.: Differential analysis on simeck and SIMON with dynamic key-guessing techniques. In: Camp, O., Furnell, S., Mori, P. (eds.) ICISSP 2016. CCIS, vol. 691, pp. 64–85. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54433-5_5
18. Qin, L., Chen, H., Wang, X.: Linear hull attack on round-reduced simeck with dynamic key-guessing techniques. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 2016. LNCS, vol. 9723, pp. 409–424. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40367-0_26
19. Ryabko, B., Soskov, A.: The distinguishing attack on speck, simon, simeck, HIGHT and LEA. Cryptology ePrint Archive, Report 2018/047 (2018)
20. 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)
21. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: *TWINE*: a lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_22
22. Wang, N., Wang, X., Jia, K., Zhao, J.: Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2014/448 (2014)
23. 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
24. Zhang, K., Guan, J., Hu, B., Lin, D.: Security evaluation on simeck against zero-correlation linear cryptanalysis. IET Inf. Secur. **12**(1), 87–93 (2018)
25. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. SCIENCE CHINA Inf. Sci. **58**(12), 1–15 (2015)

Short Paper Session I: Attack Detection



MalHunter: Performing a Timely Detection on Malicious Domains via a Single DNS Query

Chengwei Peng^{1,3}, Xiaochun Yun^{1,2}(✉), Yongzheng Zhang², and Shuhao Li²

¹ Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{pengchengwei,yunxiaochun}@iie.ac.cn

² Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhangyongzheng,lishuhao}@iie.ac.cn

³ University of Chinese Academy of Sciences, Beijing, China

Abstract. Domain names have been abused for illicit online activities for decades. A wealth of effort has been devoted to detect malicious domains in the past. However, these works primarily identify suspicious DNS behaviors (*e.g.*, lookup patterns, resolution graphs) to distinguish legitimate domains from malicious ones. Whereas, these behaviors can only be observed *after* malicious activity is already underway, thus are often too late to prevent miscreants from reaping benefits of the attacks, delaying detection. In this paper, we propose **MalHunter**, a timely detection technique that determines a domain's reputation via only a *single* DNS query. We base it on the insight that miscreants need to host malicious domains on IPs that they control, which makes different malicious domains are commonly hosted on the same IPs and creates intrinsic associations. To capture these inherent associations, we employ a deep neural network architecture based method, thus making it possible for detecting malicious domains via only a *single* DNS query. We evaluate **MalHunter** using real-world DNS traffic collected from three large ISP networks in China over two months. Compared to previous approaches, our method significantly reduces the time delay of detection from days or weeks to approximate ten microseconds while maintaining as high detection accuracy.

Keywords: Domain reputation · Timely detection
Single DNS query · Neural network · Malicious domain

1 Introduction

The Domain Name System (DNS) is a hierarchical decentralized naming system and provides critical services for mapping domain names to IP addresses. Unfortunately, it has been abused by miscreants for various illicit activities constantly. For instance, botnets exploit algorithmically generated domains to circumvent the take-down efforts of authorities [10], and scammers set up phishing websites on domains resembling well known legitimate ones [13].

To mitigate these threats, tremendous efforts [3, 6, 8, 9, 12] have been devoted in the last decades to establish domain reputation and blacklisting systems. These works derive domain reputation based on the DNS behaviors (*e.g.*, lookup patterns [3], resolution graphs [6, 8, 9, 12]). However, these information is obtained *after* the domain is *used* for a period time, delaying detection. For instance, Exposure [3] analyzes the time-based features based on a domain’s 20 times queries. However, our study shows that it on average costs 17.45 days to accumulate 20 times queries for malicious domains in a ISP network. On the other hand, we find that many domains only occur once in our two-month-long dataset. Therefore, most suspicious DNS patterns identified in previous studies do not work well for these disposable domains. Hao *et al.* [5] proposed PREDATOR that only uses the time-of-registration features so as to detect malicious domains before they are actually used. However, it cannot detect the domains that change malicious afterward.

In this paper, we propose **MalHunter**, a lightweight and timely technique to detect malicious domains through only a *single* DNS query. Our insight is that, though miscreants frequently change the domain names in different attack campaigns to avoid detection, the set of infected users and the IPs hosted are relatively stable. Therefore, over a period of time, the newly malicious domains will be visited by previously infected users and hosted on the same IPs that miscreants control. Driven by this insight, we employ a deep neural network architecture to mine the latent associations. Specifically, we first conduct an embedding learning to represent every user, domain, IP into a vector. Thus, we can convert a DNS query, $q = (user, domain, IP)$, to a tensor as the combination of the embedded vectors of the three items. Next, we design a multi-layer neural network to enhance the original features. At last, we use a standard classifier to decide if the domain is malicious using the enhanced features.

We evaluate **MalHunter** using real-world DNS traffic collected from three ISP networks in China over two months. Compared to previous methods [3, 12], the noteworthy advantage is that **MalHunter** significantly reduces the detection delay from days or weeks to several microseconds. On average, it costs about 10.41 *us* to determine its reputation after a domain is appeared in the monitoring ISP network on a machine with a NVIDIA GeForce GTX 1080 GPU of 12GB RAM. Therefore, **MalHunter** can approximately deal with 100,000 DNS queries per second, making it possible for online detection. On the other hand, **MalHunter** achieves as high detection accuracy with a 92.49% F-Measure (*i.e.*, 98.34% precision and 87.32% recall) as most previous methods.

In summary, our paper makes the following contributions:

- We present **MalHunter**, a lightweight technique to detect malicious domains timely. It determines a domain’s reputation via a *single* DNS query, which provides early detection (*e.g.*, first usage) of potentially malicious domains.
- We identify two types of inherent associations: (i) malicious domains tend to hosted on a stable set of IPs that miscreants control; and (ii) infected users tend to visit newly malicious domains.

- We employ a deep neural network architecture to capture the latent associations, making it possible to detect malicious domains via a *single* DNS query.
- We perform a comprehensive evaluation of **MalHunter** using two months real-world DNS traffic collected from three large ISP networks in China. The results demonstrate our technique can subversively reduce the detection delay from days or weeks to several microseconds, thus, providing a timely defense against the misuse of DNS domains.

We organize our paper as follows. Section 2 is related work. Section 3 elaborates the technical details of the proposed approach. Experiment setup and results analysis are reported in Sect. 4. We discuss a few issues of our approach in Sect. 5 and conclude the paper in Sect. 6.

2 Related Work

A wealth of research has been conducted on detecting malicious domains. Notos [2] is a pioneer work for establishing a dynamic reputation system to detect malicious domains. Bilge *et al.* proposed Exposure [3] with original time-based features, which requires less training time and data. Khalil *et al.* [6] argue that many local features used in detecting malicious domains, such as domain name and temporal patterns tend to be relatively brittle and allow attackers to take advantages of these features to evade detection. Peng *et al.* [9] proposed a malicious domain detection method focusing on domains that are not resolved to IP addresses directly, but only appear in DNS CNAME records. Rahbarinia *et al.* proposed Segugio [12] for efficiently tracking the occurrence of new malware-control domain names in very large ISP networks. Unfortunately, these works analyze DNS patterns that only can be obtained after the domain is used for a period of time to identify malicious domains, delaying detection.

Our method determines a domain’s reputation via only a single DNS query so that it can identify malicious domains at the same time they are used, performing timely detection. On the other hand, our technique utilizes the inherent associations to detect malicious domains, thus can identify the malicious domains even if they are registered normally.

3 Proposed Approach

3.1 Overview

MalHunter is designed as a timely detection to identify malicious domains via a *single* DNS query. Figure 1 shows a sample DNS query in our dataset. The *user* field represents the host who queries the domain. Notice that it has been anonymized and represented as a unique ID for the privacy concern. The *rdata* field is the first resolution in the answer section of the response packet. Note that, we only preserve the first resolution due to the efficiency consideration.

```
## A single DNS Query
{"user": "10342", "dns server": "8.8.8.8", "domain": "www.google.com",
 "rrtype": "A", "rdata": "216.58.216.4", "timestamp": "1483580120"}
```

Fig. 1. A sample DNS query

Unlike other DNS reputation systems that observe the ways a domain is used in practice, it is more challengeable to determine a domain’s reputation with only a *single* DNS query due to the limited information. Our insight is that, though miscreants frequently change the domain names in attack campaigns to avoid detection, these domains must be hosted on the IPs that they control or have access to. On the other hand, previously infected users tend to visit newly malicious domains afterward. Therefore, when detecting malicious domains via a *single* query, it is more effective to consider *who* queries the domain and *what IPs* the domain is hosted on, instead of the linguistic features of a domain alone. To capture these intrinsic associations, we employ a deep neural network architecture.

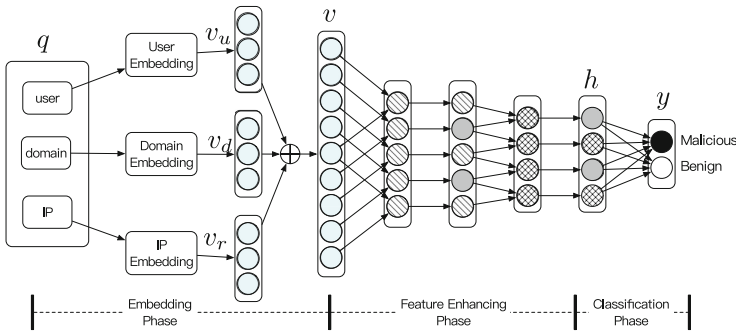


Fig. 2. Overview of MalHunter

Figure 2 depicts the architecture of MalHunter. It is composed of three phases: (1) *embedding* phase; (2) *feature enhancing* phase; and (3) *classification* phase. The *embedding* phase is responsible for representing every single DNS query, q , into a vector, v . Then, the *feature enhancing* phase aims at obtaining a more essential features v' from the vector representation v through a multilayer perceptron. Finally, the *classification* phase outputs the reputation score based on the enhanced feature v' . Notice that all of the three phases are optimized jointly using stochastic gradient descent [7].

3.2 Embedding Phase

Feature learning based on the embedding architecture (e.g., word2vec [4]) have been proven a big success in natural language processing. The basic idea is to

embed every word into a vector automatically based on its context. For our problem, we adapt the embedding operation to transform every input triple $q = (user, domain, IP)$ into a vector representation.

Specifically, we treat each user, domain and IP occurring in the DNS traffic as a word. Then, we use the *User Embedding* operations to embed the *user* item to vector v_u , use the *Domain Embedding* operations to embed the *domain* item to vector v_d and the *IP Embedding* operations to embed the *IP* item to vector v_I . The three different views of embedding provide us a more comprehensive recognition of the query. This phase involves three parameters: (i) L_u , the dimension of v_u ; (ii) L_d , the dimension of v_d ; and (iii) L_I , the dimension of v_I . Finally, after obtaining the embedded vectors of the user, domain and IP items, we concatenate the three vectors as the representation of the query, $v = v_u \oplus v_d \oplus v_I$, where \oplus is the concatenation operator.

3.3 Feature Enhancing Phase

The multilayer perceptron (MLP) is a class of feed-forward artificial neural network. The representation v learned from the *Embedding* phase represents the original representation of the query. Therefore, we employ MLP to mine more effective features. Denote $n = L_u + L_d + L_I$ to be the length of v .

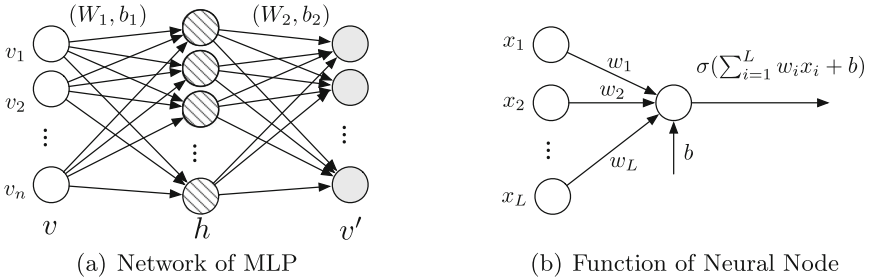


Fig. 3. Feature enhancing

Figure 3(a) shows the network of the MLP employed in our method. It consists of three layers: (i) the input layer, v ; (ii) the hidden layer, h and (iii) the output layer, v' . The three layers are connected by standard full connected operations. Figure 3(b) shows the detail function on each neural node, where w_i is the weight of node x_i , b is the bias and $\sigma(\cdot)$ is the non-linear activate function. A neural node receives the input from all the neural nodes of its forward layer. We choose $\sigma(x) = \tanh(x)$, due to $\tanh(x)$ is an effective and widely used non-linear function.

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1}$$

In summary, the transforming processes are: (i) $h = \tanh(W_1^T * v + b_1)$ and (ii) $v' = \tanh(W_2^T * h + b_2)$. Parameters (W_1, b_1, W_2, b_2) are first randomly

initialized and then adjusted gradually by optimizing the defined loss function, equation (3). There are two parameters, the number of neurons in hidden layer, n_1 , and in output layer, n_2 , in this phase. We discuss the performances of using different values of n_1 and n_2 in experiments.

3.4 Classification Phase

We adopt a standard full connected network with softmax normalization function on the enhanced feature v' to calculate the final reputation score as follow:

$$\tilde{y} = \text{softmax}(W^T * v' + b) \quad (2)$$

where W is the weight and b is the bias. $\tilde{y} = [y_1, y_2]$ is a two-dimensional vector. y_1 denotes the probability of the domain is malicious and y_2 denotes the probability of the domain is benign. The *softmax* is a normalization function to ensure the sum of \tilde{y} to 1.

3.5 Model Optimizing

We use cross entropy loss to measure the training process:

$$L(y, \tilde{y}) = -\frac{1}{N} \sum_{i=1}^N [y^{(i)} \log \tilde{y}^{(i)} + (1 - y^{(i)}) \log(1 - \tilde{y}^{(i)})] \quad (3)$$

where N is the number of training samples, y is the vector of true classes (*i.e.*, 0 for benign, 1 for malicious) and \tilde{y} is the vector of predicted classes. We use stochastic gradient descent (SGD) [7] algorithm to minimize Eq. (3). In each step, the SGD algorithm samples a mini-batch of data (We empirically set batch size equal to 64) and then updates the model parameters. Besides, we train the model with many epochs (100 epochs in our experiments).

4 Experiment

In this section, we perform a comprehensive evaluation of **MalHunter**. First, we introduce the datasets used in the experiments. Then, we evaluate the effectiveness of **MalHunter**. At last, we assess the time delay of **MalHunter** and compare it with two previous works.

4.1 Dataset

1. **Malicious Domains:** We collected malicious domains from Malware Domains List¹, Phishtank² and Openphish³ everyday from Jan. 03, 2017

¹ <http://www.malwaredomains.com>.

² <http://www.phishtank.com>.

³ <https://openphish.com>.

Table 1. Data description. Each row in the table represents the ISP network of data source, the number of total, the known malicious and benign samples in the ISP

ISP	First month (Dec 5, 2016–Jan 5, 2017)		Second month (Jan 6, 2017–Feb 5, 2017)	
	Malicious	Benign	Malicious	Benign
ISP_1	157,329	157,329	70,073	70,073
ISP_2	108,945	108,945	62,292	62,292
ISP_3	121,651	121,651	59,269	59,269

to Oct. 14, 2017. In addition, we use the Zeus Block List⁴ and the list of domains that are used by Conficker [11]. These malicious domains represent a wide variety of malicious activity, including botnet C&C servers, drive-by download sites, phishing pages and so on.

- Benign Domains:** We collected legitimate domains according to Alexa⁵. We chose domains that are consistently ranked among the top 1 million from Jan. 16, 2015 to Mar. 5, 2017 (513 days). In addition, we manual filter out domains that allow for the “free registration” of subdomains, such as popular blog-publishing services or dynamic DNS domains (*e.g.*, wordpress.com and dyndns.com), as their subdomains are often abused by attackers. Finally, this produced a list of 270,778 popular domains.
- Real-world DNS Traffic:** CNCERT/CC [1] provided us two-month-long (Dec 5, 2016 - Feb 5, 2017) passive DNS traffic, about 530 millions DNS queries, collected from three large ISP networks⁶ in China. We refer to these ISP networks simply as ISP_1 , ISP_2 and ISP_3 . Notice that this paper is part of an IRB-approved study; appropriate steps have been taken by our data provider to minimize privacy risks for the network users. Table 1 presents the number of labeled datasets in the two months using the collected malicious and benign domains.

4.2 Effectiveness Analysis

We conducted our experiments using the data described in Table 1. We train the model with the data in the first month and test its performance on the data in the second month.

Our approach involves five parameters: three (*i.e.*, L_u , L_d and L_I) in the embedding phase and two (*i.e.*, n_1 and n_2) in feature enhancing phase. We conducted the experiments using various values for each parameter to demonstrate the effectiveness. Specifically, we try $n_1 \in \{96, 128, 160, 256\}$, $n_2 \in \{32, 48, 64, 80\}$, $L_u \in \{50, 100, 200\}$, $L_d \in \{50, 100, 200\}$ and $L_I \in \{50, 100, 200\}$

⁴ <https://zeustracker.abuse.ch/blocklist.php>.

⁵ <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>.

⁶ The three ISP networks are located in the provinces of Anhui, Guangdong and Shanghai respectively.

Table 2. Detail experimental results on the three ISP datasets using different values of L_u , L_d and L_I . Precision (Prec.) = $\frac{N_{TP}}{N_{TP}+N_{FN}}$, Recall (Rec.) = $\frac{N_{TP}}{N_{TP}+N_{FP}}$ and F-Measure (F-Mea.) = $2 * \frac{Prec.*Rec.}{Prec.+Rec.}$. We set $n_1 = 128$ and $n_2 = 64$.

Parameter			ISP ₁			ISP ₂			ISP ₃		
L_u	L_d	L_I	Prec. %	Rec. %	F-Mea. %	Prec. %	Rec. %	F-Mea. %	Prec. %	Rec. %	F-Mea. %
50	50	50	97.47	87.30	92.11	99.07	88.34	93.40	98.70	86.00	91.91
50	50	100	97.69	85.87	91.40	99.14	89.88	94.28	98.44	87.03	92.39
50	50	200	98.81	86.81	92.42	97.31	91.29	94.20	99.06	88.01	93.21
50	100	50	98.64	89.06	93.60	98.64	89.30	93.74	98.75	88.94	93.59
50	100	100	98.34	88.70	93.27	98.70	91.35	94.88	99.44	87.81	93.26
50	100	200	98.10	90.53	94.16	98.73	91.36	94.90	98.76	89.13	93.70
50	200	50	98.45	90.36	94.23	99.39	89.63	94.26	97.70	88.97	93.13
50	200	100	98.80	89.95	94.17	98.90	92.34	95.51	99.30	89.35	94.06
50	200	200	98.47	91.43	94.82	99.30	91.22	95.09	99.16	89.62	94.15
100	50	50	97.16	87.35	92.00	97.95	90.07	93.85	98.01	85.76	91.48
100	50	100	97.16	88.67	92.72	98.16	90.22	94.02	98.31	88.65	93.23
100	50	200	97.84	88.18	92.76	98.30	91.03	94.53	98.60	88.54	93.30
100	100	50	97.51	88.82	92.96	98.64	90.86	94.59	98.90	89.34	93.88
100	100	100	98.30	89.75	93.83	98.77	91.23	94.85	98.56	90.66	94.45
100	100	200	98.59	89.49	93.82	99.08	91.11	94.93	98.44	89.76	93.90
100	200	50	98.39	90.00	94.01	98.70	91.35	94.88	98.85	89.06	93.70
100	200	100	97.96	90.40	94.03	99.13	89.94	94.31	99.02	88.55	93.49
100	200	200	97.98	91.53	94.64	98.51	91.97	95.13	99.12	89.33	93.97
200	50	50	96.70	86.26	91.19	96.67	89.73	93.07	97.60	89.69	93.48
200	50	100	96.98	87.77	92.14	98.25	89.82	93.84	97.55	89.79	93.51
200	50	200	96.95	87.81	92.15	98.33	91.22	94.64	97.65	91.38	94.42
200	100	50	96.71	90.27	93.38	98.09	90.10	93.93	98.46	86.75	92.24
200	100	100	97.30	90.67	93.87	97.07	91.57	94.24	98.34	88.11	92.94
200	100	200	97.08	91.12	94.01	98.74	91.05	94.74	98.55	89.14	93.61
200	200	50	97.50	90.07	93.64	98.57	91.42	94.86	98.57	89.33	93.72
200	200	100	98.14	90.23	94.02	97.43	92.50	94.90	99.21	89.01	93.84
200	200	200	98.40	90.29	94.17	98.73	92.87	95.71	98.35	90.34	94.17

to run the experiments. The results show that different values of these parameters lead to almost the same performance. Due to the space limitation, we only present the results of using different values of L_u , L_d and L_I in Table 2.

We observe that **MalHunter** can achieve promising accuracy on the three ISP datasets under all the combinations of the three parameters. On average, it achieves a 92.49% F-Measure with a 98.34% precision and a 87.32% recall, suggesting that **MalHunter** can effectively detect malicious domains even using only a *single* DNS query.

4.3 Time Delay Comparison

The time delay of detection is the duration between the domain is seen and the system finally outputs its detection result (*i.e.*, malicious or benign). It can be divided into two parts: (i) data delay, the time to collect the features evident after the domain is appeared; and (ii) calculation delay, the time to calculate the reputation after obtaining the features.

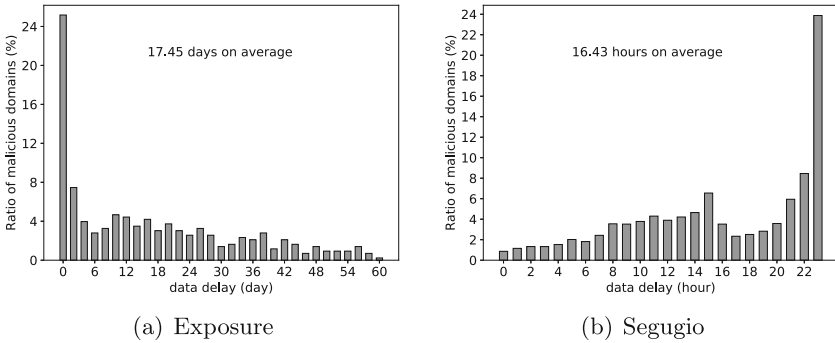


Fig. 4. The distributions of data delay of Exposure [3] and Segugio [12]

MalHunter only relies on a domain’s single DNS query to determine its reputation. Hence, when a domain is appeared in DNS system, we can determine its reputation immediately. Thus, the data delay is none of our approach. We compare the data delay with two existing reputation systems:

- (i) Exposure [3]: It identifies malicious domains according to their 20 times query behaviors. Therefore, it filters all domain names that requested less than 20 times.
- (ii) Segugio [12]: It uses one day’s DNS traffic to construct a *machine-domain* graph for extracting machine-based features. Besides, it discards all domain names that are queried by *only one* machine.

Figure 4 shows the time of data delay of these two works. On average, Exposure needs 17.45 days to accumulate 20 times requests and Segugio is about 16.43 hours late to detect the appeared malicious domains. For instance, if a domain is appeared at 9:00 am, Segugio can only detect it at 24:00. Compared to these two reputation systems, **MalHunter** saves the data delay time and detect malicious domains days or weeks earlier.

We further assess the calculation delay of our approach. We implemented **MalHunter** in PyTorch and conducted all experiments on a machine that has an 8-core Inter(R) Core(TM) i7-6700K CPU @ 4.00GHz with 32 GB RAM and a NVIDIA GeForce GTX 1080 GPU with 12 GB RAM. Table 3 presents the average elapsed time of **MalHunter**. Note that it takes about 11.92s to train per epoch on nearly 300 K samples, thus we need about 20 min (100 epochs) to

Table 3. Summary of elapsed time of **MalHunter**, “us” denotes microsecond (10^{-6} second)

ISP	<i>Train</i> (second/epoch)	<i>Test</i> (us/sample)
<i>ISP</i> ₁	14.36	10.33
<i>ISP</i> ₂	11.70	10.46
<i>ISP</i> ₃	9.69	10.44

train the model well. When predicting new samples, it costs about 10.41 *us* per sample. Therefore, **MalHunter** can approximately deal with 100,000 DNS queries per second, making it possible for deploying online.

5 Discussion and Future Work

In this section, we discuss some possible limitations and future work of our method.

First, **MalHunter** is a data-driven method, it relies on DNS traffic to train the model parameters. When deploying **MalHunter** in a new ISP network, the users served are different. Therefore, it needs to retrain the model with the DNS traffic collected in that ISP network. Once, **MalHunter** is well trained, it can detect malicious domains timely.

Second, **MalHunter** identify a new domain’s reputation based on its associations with previous users and IPs. Therefore, if a brand-new DNS query occur (*i.e.*, the user, domain and IPs are all never seen before), **MalHunter** cannot determine its reputation. However, we observe that the set of users and IPs in a given ISP network will gradually converge. Our study shows after collecting one-month DNS traffic, 86.82% users, 70.05% domains and 69.63% IPs in the second month are seen in the first month, therefore, the probability of a brand-new DNS query occurring is as low as $(1-86.82\%) * (1-70.05\%) * (1-69.63\%) = 1.20\%$. On the other hand, **MalHunter** is now a supervised method, for the future, we can employ an unsupervised machine learning algorithm to pre-train the DNS traffic to further reduce the probability of a brand-news query.

6 Conclusion

Determining the reputation of DNS domains provides significant benefits in defending against many Internet attacks. However, existing methods calculate a domain’s reputation based on the evidences *after* the domain is used, delaying detection. In this paper, we present **MalHunter**, a lightweight and timely detection technique to identify malicious domains via a *single* DNS query. The experimental results using real-word DNS traffic show that **MalHunter** can significantly reduce the time delay compared to existing detection systems while maintaining as high detection accuracy.

Acknowledgments. The research leading to these results has received funding from the National Key Research and Development Program of China (No. 2016YFB0801502) and National Natural Science Foundation of China (No. U1736218). The corresponding author is Xiaochun Yun.

References

1. CNCERT/CC. <https://www.cert.org.cn> (2018)
2. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for DNS. In: USENIX security symposium, pp. 273–290 (2010)
3. Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: Exposure: a passive DNS analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **16**(4), 14 (2014)
4. Goldberg, Y., Levy, O.: word2vec explained: deriving Mikolov et al’s negative-sampling word-embedding method. arXiv preprint [arXiv:1402.3722](https://arxiv.org/abs/1402.3722) (2014)
5. Hao, S., Kantchelian, A., Miller, B., Paxson, V., Feamster, N.: Predator: proactive recognition and elimination of domain abuse at time-of-registration. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1568–1579. ACM (2016)
6. Khalil, I., Yu, T., Guan, B.: Discovering malicious domains through passive DNS data graph analysis. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 663–674. ACM (2016)
7. Kiefer, J., Wolfowitz, J.: Stochastic estimation of the maximum of a regression function. *Ann. Math. Stat.* **23**, 462–466 (1952)
8. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 1–18. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_1
9. Peng, C., Yun, X., Zhang, Y., Li, S., Xiao, J.: Discovering malicious domains through alias-canonical graph. In: Trustcom/BigDataSE/ICSS, 2017 IEEE, pp. 225–232. IEEE (2017)
10. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A comprehensive measurement study of domain generating malware. In: USENIX Security Symposium, pp. 263–278 (2016)
11. Porras, P.A., Saïdi, H., Yegneswaran, V.: A foray into Conficker’s logic and rendezvous points. In: LEET (2009)
12. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: efficient behavior-based tracking of malware-control domains in large ISP networks. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 403–414. IEEE (2015)
13. Szurdi, J., Kocso, B., Cseh, G., Spring, J., Felegyhazi, M., Kanich, C.: The long “taile” of typosquatting domain names. In: USENIX Security Symposium, pp. 191–206 (2014)



Detecting Intrusion in the Traffic Signals of an Intelligent Traffic System

Abdullahi Chowdhury^{1(✉)}, Gour Karmakar^{1(✉)},
Joarder Kamruzzaman^{1(✉)}, and Tapash Saha^{2(✉)}

¹ Federation University Australia, Ballarat, Australia
{Abdullahi.Chowdhury, Gour.Karmakar,
Joarder.Kamruzzaman}@federation.edu.au

² VicRoads, Kew, Australia
Tapash.Saha@roads.vic.gov.au

Abstract. Traffic systems and signals are used to improve traffic flow, reduce congestion, increase travel time consistency and ensure safety of road users. Malicious interruption or manipulation of traffic signals may cause disastrous instants including huge delays, financial loss and loss of lives. Intrusion into traffic signals by hackers can create such interruption whose consequences will only increase with the introduction of driverless vehicles. Recently, many traffic signals across the world are reported to have intruded, highlighting the importance of accurate detection. To reduce the impact of an intrusion, in this paper, we introduce an intrusion detection technique using the flow rate and phase time of a traffic signal as evidential information to detect the presence of an intrusion. The information received from flow rate and phase time are fused with the Dempster Shaffer (DS) theory. Historical data are used to create the probability mass functions for both flow rate and phase time. We also developed a simulation model using a traffic simulator, namely SUMO for many types of real traffic situations including intrusion. The performance of the proposed Intrusion Detection System (IDS) is appraised with normal traffic condition and induced intrusions. Simulated results show our proposed system can successfully detect intruded traffic signals from normal signals with significantly high accuracy (above 91%).

Keywords: Traffic signal · Intrusion detection · Intelligent Traffic System

1 Introduction

With ever increasing population in urban areas, traffic congestion management became one of the major issues in the big cities. Intelligent Traffic Systems (ITS) uses adaptive traffic control system to improve the way of managing traffic on road. ITS aims to introduce different innovative traffic management services to make road safe for the commuters, use the existing transport network efficiently and make the Traffic Management System (TMS) more coordinated by providing real-time and better dissemination of traffic information. Innovative technologies surrounding ITS are on the rise, valuing the Global ITS market at US\$ 21,481.4 M in 2017, and projected to reach US\$ 70,798.4 M by 2027, indicating a compound annual growth rate of 12.7% [1]. In ITS,

different wireless communication methods (e.g., Dedicated Short Range Communication (DSRC) or cellular network) are being used to make the communication between vehicles and the road infrastructure. These wireless communication methods are vulnerable to different cyber-attacks. There are mainly four components in ITS: (a) On Board Unit (OBU), (b) Road Side Unit (RSU), (c) Vehicle Detector (VD), and (d) Signal Controller (SC). These components use wireless technologies in some point to communicate with each other. In last few years research on cyber-security of transportation systems was mainly focused on inter-vehicle communications. A number of security breaches have already been reported. An Argentinian security expert intruded into New York City's wireless vehicle detection system and showed that control of the devices (e.g., RSU, OBU, VD, and SC) can be compromised. It also showed that attackers can also send malicious or corrupted data [2–6].

Existing and future ITSs have huge risk of cyber-attacks. Attacks on automated vehicles and ITSs will have huge economic impact and even risk of loss of human life. These presents a pressing need to develop an IDS for ITS. This has been compounded by the fact that traffic control systems will be need to interact and manage many robotic systems (e.g., driverless cars) in the future. Vehicle-to-vehicle and infrastructure communication is based on wireless communication and ad-hoc in nature. Since wireless technologies are vulnerable to many attacks, consequently, future ITS will be very susceptible to various types of attacks including cyber-attacks. For this reason, to minimise the impact of the attack and study the cybersecurity of traffic infrastructure, the National Cooperative Highway Research Program (NCHRP) proposed few new projects to introduce some steps to mitigate the impacts cyber-attacks on TMS [7, 8].

Though reported attacks on ITS currently are limited to attacks on computers in the traffic controller, safety cameras installed in the RSU, and processing units installed in the signals of the intersection, undoubtedly such attacks will be on the rise in future [9]. Up to our knowledge, there is no Instruction Detection System (IDS) available in the current literature to detect attacks on traffic signal units and ITS in general. In this paper, for first time we have proposed an IDS for ITS. Our proposed system can detect the attacks in the road sensors, traffic signals and local traffic controller. We have theoretically modelled our proposed system using the DS decision theory considering the evidential observations of vehicle flow rate at intersections and the phase time of traffic signal changes and their historical data recorded by transportation authorities. For the verification and validation of our IDS, we developed a simulation based on the traffic simulator called SUMO [10] using many real scenarios and the data collected by the Victorian Transportation Authority, Australia called VicRoads. Simulated results show our proposed system can successfully detect overall 91.92% of original (non-intruded) and 91% of intruded traffic signals. Therefore, our proposed IDS can successfully detect most attacks on the traffic signals with a very number of false alarms.

2 Intelligent Traffic Systems

ITS uses different types of detectors to detect number of vehicles, speed of the vehicle, and type of the vehicle. There are mainly two types of detectors named strategic detectors and tactical detectors. Strategic detectors are used to gather vehicle data to

effectively use the signal phase time and tactical detectors gathers vehicle data to assist ITS make decision to set different state of the phase of any intersection, set the variable speed limit and cycle time of a traffic signal. Several driver assistance sensor systems installed in modern smart vehicles can communicate with the RSU to receive and provide information to increase road safety and smart traffic management. Communication technologies like the IEEE 802.11p standard is available in ITSs for V2V, V2I, I2I I2X and V2X communication [11, 12]. Different types of Adaptive traffic control systems (ATCSs) e.g., Sydney Coordinated Adaptive Traffic System (SCATS), OPAC, RHODES, ACS Lite and InSync [11, 13] have been developed to reduce travel time and congestion. In our study we have selected SCATS, as this one of the best ATCSs and used in approximately 37,000 intersections in 27 countries. A SCATS-compatible Traffic Signal Controller (TSC) collects vehicle information from many different methods. These methods are (i) Triangulation method, (ii) Vehicle re-identification, (iii) GPS based methods and (iv) Smartphone based rich monitoring. SCATS has different operation modes. They are (i) Master Link, (ii) Flexi Link, (iii) Isolated, (iv) Hurry Call and (iv) Manual Operation. Depending on the traffic condition and demand, SCATS can operate on any of these operation modes [13].

2.1 Attacks in ITS

In ITS, different signal controller units are used to communicate between the signals in adjacent intersections, maintain the phase time, cycle time, and the operational status provided by the central traffic controller. Road side sensor systems and different driver assistance sensor systems installed in the modern vehicles can communicate with the RSU to receive and provide information to increase road safety and smart traffic management. Communication standards like the IEEE 802.11p is available in ITSs for V2V, V2I, I2I I2X and V2X communication. Different types of Adaptive traffic control systems (ATCSs) e.g., Sydney Coordinated Adaptive Traffic System (SCATS), OPAC, RHODES, ACS Lite and InSync [11] use real time traffic data to optimize the cycle time of the traffic signals to reduce travel time and congestion [11, 12]. These systems are at risk of going under different types of cyber-attack based on their communication methods.

Researchers at the University of Michigan [13] shown that traffic control systems uses unencrypted wireless signals and default username and password to manage the traffic signals and traffic controller that controls the lights and walk signs. These can make the traffic signal and traffic controller system easy target for the hackers. Denial of Service attack Distributed Denial of Service attack on Vehicular Ad-hoc Network (VANET), ITS and TMS is possible by jamming communication channel, network overloading, and packets dropping [14, 15]. Fake data insertion, ransomware attack, rail network disruption and malware attacks in different ITS units were reported in different cities in the world [2, 17, 18]. These attacks cause huge traffic jam, financial loss, disruption to transport ticketing system and cyber threats to transport authority. Attacker can also use Sybil attack to provide misleading information to nearby vehicles or the road side traffic infrastructure [16]. GPS hacking can effect navigation of driverless cars, drones and automated emergency vehicles to send the vehicle to different location or provide false data to traffic controller. Data spoofing attack on even

single vehicle can increase the total delay by as high as 68.1% which completely reverses the benefit of using the Intelligent Traffic Signal System (I-SIG) system (26.6% decrease) by U.S. Department of transportation and cause the mobility to be even 23.4% worse than that without using the I-SIG system [17, 22].

3 Proposed Intrusion Detection Method

3.1 Overview of the Proposed IDS

Our proposed model mainly monitors the status of current traffic signal, which is statistically determined considering the flow rate and the density of vehicles, and the signal phase time of that traffic signal.

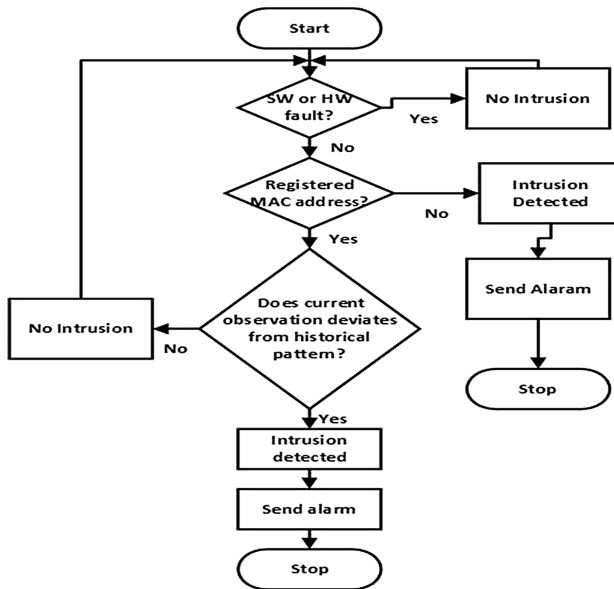


Fig. 1. An overview of the proposed intrusion detection system. SW = Software and HW = Hardware

To assess the whether the traffic signal is behaving as normal or unusual, the current status is compared and contrasted with the relevant status of that traffic signal derived from the corresponding historical data recorded by its TMS. The basic operating principle of the proposed model is shown in Fig. 1. To reduce the false alarms in detection, firstly, the proposed model checks whether there is any software or hardware malfunction. If the deployed mechanism signals no software or hardware malfunction, the model then further verifies whether the MAC address is registered in the system. If the MAC address is not registered, it sends a message to the controller system that the data is coming from an unauthorized sensor. Otherwise, our system checks whether the

current observation data sufficiently deviates from the corresponding historical observation pattern. If it sufficiently deviates, it confirms that there is an intrusion in that traffic signal. However, there may be special events (e.g., sports, festival) occurring seasonally and/or periodically throughout the year, which may affect the signal. To reduce the impact of those special events, the relevant historical data for the similar time that were affected by those events are chosen in our proposed model. Our proposed model consists of mainly two parts – (i) monitoring the status of the traffic signal and (ii) evaluating the traffic signal to detect intrusion. Those components are described below.

3.2 Monitoring the Status of the Traffic Signal

The traffic signal is monitored in two phases. In the first phase, we use the MAC address of the sensors. People (e.g., hackers) can use external devices equipped with sensors to connect with the TMS network through wireless communication infrastructure to exploit the system vulnerability and alter traffic data. Therefore, in the first stage, we can verify whether the MAC address of a particular sensor belongs to the list of registered MAC addresses. This verification process ensures to detect where any unregistered sensor is attempting to send signal data raising suspicion. In the second phase, our proposed approach determines whether a registered sensor have been compromised. For this, we can exploit historical traffic pattern probability mass function that has not been manipulated by intruder at a particular time within a particular time window (e.g., from 08:00 am to 09:00 am on Monday).

In this project, since we aim to use historical data to monitor the status of a current traffic signal at a particular time, we collected all required and relevant data from Vic Road's traffic data [19]. How the probability mass function of these data for a particular time window can be approximated, is detailed later.

For detecting the intrusion in this phase, we need to calculate continuous observed values of some signal attributes. For this project, we have chosen the observed value of the flow rate and phase time of a signal. This is because flow rate and phase time can be used to obtain additional green time for creating traffic signal disruptions or having illegal benefits. Other attributes (e.g., vehicle type, velocity, pedestrian count) are not significant as they are not so effective like flow rate and phase time to make major change in signal timing. Here, we need to use an inferencing method to assess the status (e.g., Normal or Abnormal) of a traffic signal. There are many methods available in the literature for inferencing such as Bayesian theory, rule based inferencing system and Dempster Shafer (DS) decision theory. We have chosen the DS decision theory because it is based on generalized Bayesian theory and provides distributing support for different propositions using temporal data. For our proposed model, the frame of discernment is defined as,

$$H = \{N, A\} \quad (1)$$

where, N and A represent the proposition of the current observation being Normal and Abnormal, respectively.

Since the flow rate and phase time are measured by respective individual sensor data, the belief function contributing over a particular preposition needs to be statistically measured for each sensor.

Let $R_{il\tau}$ be the id of a sensor placed in Lane l of intersection i having sensor type τ , where, $\tau = 1$ and $\tau = 2$ represent flow rate and phase time, respectively. Since we observe events such as flow rate and phase time of a particular intersection over time (e.g., the observed event in this case $E_\tau(t)$ at time t at a particular day), we observe the data in a time window (e.g., 08:00 am–09:00 am) of a day considering working and non-working days, we need to use the probability mass function of the historical data corresponding to that time window of that day to find out the probability of a particular observation being normal. Therefore, the belief over proposition N using the DS theory can be defined as,

$$\text{Belief}_{R_{il\tau}(N)} = \sum_{E_\tau(t) \subseteq N} m_{R_{il\tau}}(E_\tau(t)) \quad (2)$$

where, $R_{il\tau}$, N , and $E_\tau(t)$ are as defined before and $m_{R_{il\tau}}$ is the probability mass function of sensor $R_{il\tau}$. Note, Eq. (2) represents the lower bound of a confidence interval for estimating the status of proposition being N .

The upper bound of the confidence interval can be defined as,

$$\text{Plausibility}_{R_{il\tau}(N)} = 1 - \sum_{E_\tau(t) \cap N = \emptyset} m_{R_{il\tau}}(E_\tau(t)) \quad (3)$$

We have two types of evidences being continuously observed over time such as $E_F(t) =$ flow rate and $E_P(t) =$ phase time at time t . So for proposition N , the observation of both evidences can be fused based on the DS theory in the following way,

$$(m_{R_{iF}} \oplus m_{R_{iP}})(N) = \frac{\sum_{E_F(t) \cap E_P(t) = N} m_{R_{iF}}(E_F(t)) m_{R_{iP}}(E_P(t))}{1 - \sum_{E_F(t) \cap E_P(t) = \emptyset} m_{R_{iF}}(E_F(t)) m_{R_{iP}}(E_P(t))} \quad (4)$$

Either or both of the sensors can be compromised by the hackers. As mentioned before, to determine whether they have been attacked, individually or both, we can compare and contrast their observed values with their corresponding and authentic (e.g., not attacked or forged) historical values. This accentuates the development of probability mass function $m_{R_{il\tau}}$ used in the DS theory based fusion approach defined in (4). The development of $m_{R_{il\tau}}$ using the VicRoads's historical traffic signal data uploaded Victoria's state government website [19] is described in the following section.

3.3 Development of the Probability Mass Function

Special event usually happens in a particular period of a year. The occurrence of a special event can increase the likelihood of having unusual historical data than normal data during that specific timeframe. This implies that we need to use historical traffic signal data with and without the occurrence of events for both evidences. We calculated

histograms and their corresponding best fit normal distribution curves of the historical data for both flow rate and phase time as below.

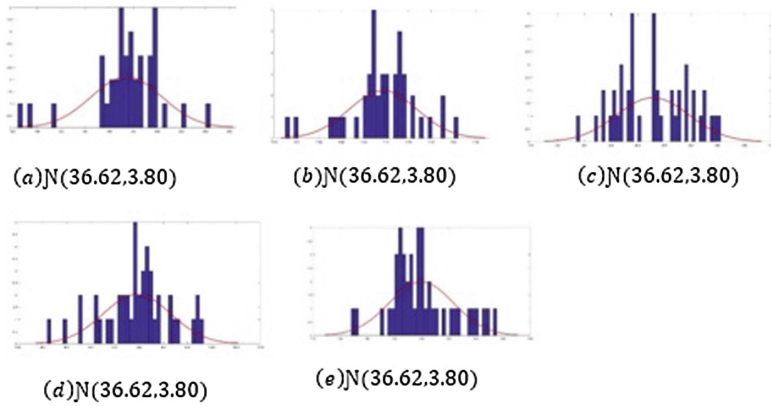


Fig. 2. Histogram and the corresponding fitting normal curves of different flow rates of five different intersections (a) Intersection 1, (b) Intersection 2, (c) Intersection 3, (d) Intersection 4, (e) Intersection 5.

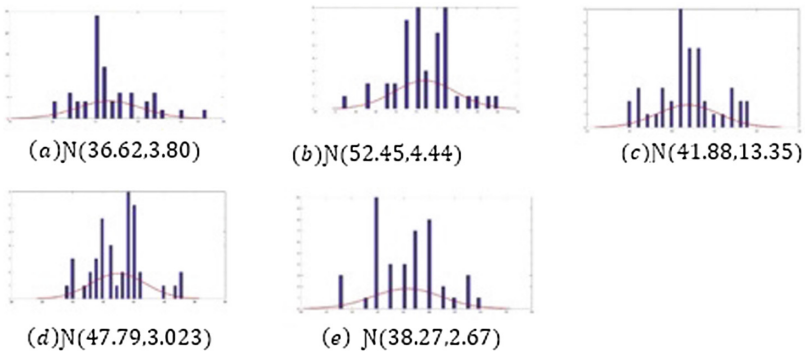


Fig. 3. Histogram and the corresponding fitting normal curves of different phase time of five different intersections (a) Intersection 1, (b) Intersection 2, (c) Intersection 3, (d) Intersection 4, (e) Intersection 5.

Without any event occurring, the histogram of the flow rate per hour and phase time on working Mondays from 08:00 am–09:00 am in 2017 and their corresponding best fit normal distribution curves of five different intersections are shown in Figs. 2(a)–(e) and 3(a)–(e), respectively. All of the figures for both flow rate and the phase time show that all probability mass functions are approximately normally distributed. This is vindicated by their corresponding well fitted normal probability mass functions for all curves as shown in the labels $N(\mu, \sigma)$ where μ and σ represent the mean and standard

deviation of the respective best fit normal curve. Using the probability mass functions developed from the historical data, we need to calculate the probability of observed evidences (flow rate and phase time) which is described below.

3.4 Calculating Probabilities of the Observed Evidences

In this section, we need to calculate the probabilities of the observed evidences for both flow rate and phase time, respectively. Since, as explained before, the probability mass functions are normally distributed, the probability of an evidence can be calculated as,

$$m_{R_{it\tau}(x)} = \begin{cases} 1 - \int_0^z \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx & z \geq 0 \\ 1 - \int_z^0 \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx & \textit{Otherwise} \end{cases} \tag{5}$$

where, $z = (x-\mu)/\sigma$, $x = E_\tau(t)$ for $\tau = F \textit{ or } P$, and μ and σ are the mean and standard deviation of the probability mass function, respectively. Once the probability of a particular evidence is calculated, we use this to evaluate the status of the traffic signal which is described in the next section.

3.5 Evaluating the Traffic Signal to Detect Intrusion

To detect an intrusion in the traffic signal, we need to evaluate the status of the traffic signals. This status is used in our model to determine the normal behavior of a traffic signal based on the historical data. If we know the value of current events (e.g., $E_F(t)$ and $E_P(t)$), we can calculate their probabilities as $m_{R_{iF}(E_F(t))}$ and $m_{R_{iP}(E_P(t))}$ using (5) and the μ and σ of their corresponding probability mass functions. Then next, the probability of $(m_{R_{iF}} \oplus m_{R_{iP}})$ being N. If $(m_{R_{iF}} \oplus m_{R_{iP}}) \geq \emptyset$, the traffic signal is assumed to be normal, otherwise, it is intruded. The sensitivity and accuracy of our proposed depend on the value of \emptyset . However, in average case, \emptyset can be considered 0.5.

4 Performance Evaluation

4.1 Simulation Environment

We instigated our model on the Simulation of Urban Mobility (SUMO) and simulated using real road map on SUMO to weigh the intrusion detection performance of our model. The following parameters were considered while setting up the simulation environment.

Map: We used the road map of Melbourne CBD and VicRoads’s real traffic data available in [19]. We have used five different intersections in Melbourne CBD.

Density and flow rate: We selected the peak-time density and flow rate in our simulation using the traffic data from 08:00 am to 09:00 am Monday at five busy corners of Melbourne CBD. The density and flow rate of an intersection at time t were calculated in the simulation using a popular microscopic traffic model presented in [21].

Vehicle type and traffic distribution: We considered mixed vehicle types where 65% vehicles were passenger vehicles, 20% delivery vans, 15% bus (both public and free shuttle services), 5% tram and some random pedestrians.

Car following model: We used the Krauss car following model in our simulation. Krauss car following model considers that car follow gradual deceleration while braking [22].

Normal and intrusion scenarios: For normal condition, the flow rate and phase time of an intersection for a particular scenario were derived for the simulation model (SUMO). In the simulation model, traffic distributions were initiated with the respective and non-compromised historical information of VicRoads online data [19]. For simulating intrusions to the traffic signals, flow rate of an intersection for a particular scenario was changed by intuitive induced phase time and vice versa. Either the flow rate or phase time was induced in such a way so that it remains within 68% to 95% confidence intervals in some scenarios and outside of 95% confidence intervals of the relevant historical data for the other scenarios.

4.2 Performance Metrics

We evaluated our intrusion detection results for all scenarios using the standard performance metrics widely used in event detection, such as specificity, sensitivity, F-score and overall accuracy.

4.3 Simulation Results and Analysis

We have tested 40, 41, 39, 42 and 44 scenarios for Intersection 1 to 5, respectively. As a representative sample, Table 2 shows the probabilities of signals being normal for 4 scenarios having various flow rates and phase time for each intersection. Since the initial and instantaneous traffic distributions were induced in the simulation from the corresponding historical data, our proposed method is able to successfully determine most of normal cases correctly with the exception of a few cases. Where in the simulation SUMO created normal traffic condition that deviates largely from historical value, our model detected those scenarios as false negative. This happened because if the traffic condition deviates highly from historical data, it can make disruption in normal traffic management.

If the induced flow rate/phase time is taken within 68% to 95% confidence intervals, for the short time (e.g., 1 or 2 cycles) attacks, our proposed IDS fails to detect intrusion. This is because since the induced flow rate/phase time is within 68% to 95% confidence intervals, the probability of a signal being normal becomes high (e.g., 0.525), which is greater than the threshold $\emptyset = 0.5$ used in making decision whether a signal is normal. This is not a major issue as short time intrusion cannot create considerable disruption in the traffic signals. However, in this case, if the intrusions prevail over a long time (e.g., longer than 2 cycles), our proposed method can successfully detect them. The reason is that long time intrusions can create disruption among all adjacent intersections, which has been clearly reflected in our simulation. This eventually reduces the probability value of signal being normal considerably. Our proposed

method can successful detect all intrusions if the induced flow rate/phase time is taken outside of 95% confidence interval. Because of the higher value, it can create considerable disruption among all closed by intersections, yielding the low value of the probability of a signal being normal (e.g., 0.024). Our proposed system produced 18, 18, 17, 19 and 19 true positives (TPs), 19, 20, 19, 20 and 20 true negatives (TNs), 2, 1, 1, 1 and 3 false positives (FPs) and 1, 2, 2, 2 and 2 false negatives (FNs) for Intersections 1 to 5, respectively (Table 1).

Table 1. Probabilities of signal being normal

I	Scenario#											
	1 (Normal)			2 (Abnormal)			3 (Normal)			4 (Abnormal)		
	FR	PT	Prob	FR	PT	Prob	FR	PT	Prob	FR	PT	Prob
1	770	45	0.53	1850	45	.007	784	36.5	0.69	1320	36.5	0.20
2	1107	53	0.85	1850	53	0.31	1180	34.5	0.42	1720	34.5	0.19
3	770	45	0.65	1850	45	0.32	784	36.5	0.88	1320	36.5	0.01
4	944	48.5	0.74	1843	48.5	0.13	930	52	0.68	1786	62	0.23
5	803	45	0.89	1456	67	0.02	870	52	0.65	920	50	0.35

I = Intersection, FR = Flow rate, PT = Phase time, Prob = Probability of signal being normal, abnormal.

For Intersection 5, the number of FPs (3) is slightly higher than that of the other intersections because for this intersection, the historical data used to generate traffic in SUMO for the specific time period (08:00 am to 09:00 am, Monday) during which an event occurred. This created more deviations compared with other intersections for the flow rate and phase time obtained from SUMO from their corresponding historical data distributions for the same events occurred in that time period throughout a year. Here, TN refers to a normal condition is detected as normal, FN represents a normal condition is detected as intrusion, TP means an intrusion is detected as intrusion and FP indicates an intrusion detected as normal. The specificity, sensitivity, F-score and accuracy of our proposed IDS for all intersections are shown in Table 3. These TP, TN, FP and FN are used to calculated sensitivity (true positive rate) and specificity (true negative rate). Our simulation result shows that we have high accuracy (91.74%) of detecting hacking even though some of the attacking data were for very small period of time.

Table 2. Performance metrics

Intersection	Specificity	Sensitivity	F-score	Accuracy
1	90.48	94.74	92.30	92.5
2	95.23	90	92.30	92.68
3	95	89.47	91.89	92.30
4	95.23	90.48	92.68	92.85
5	86.96	90.48	88.37	88.63
Overall	92.58	91.03	91.45	91.74

5 Conclusion

In this paper, we have introduced a model to detect an intrusion in ITSs for the first time. Our proposed IDS can detect any anomaly of traffic flow or signal phase time that can make considerable disruption in traffic system. Our model is based on the estimation of probability mass functions of traffic flow and phase time from the historical data collected from an ITS and fusion of those variable using DS theory. To test the efficacy of the system, we developed a simulation model considering the real traffic flow rate, density and signal phase time using the real map of Melbourne CBD and the historical data provided by VicRoads. The simulation model is built on SUMO, a known road traffic simulator, and we created various traffic signal scenarios including induced intrusions by making either flow rate or phase time or both intentionally shorter or longer than their designed permissible durations. We assessed the performance of the IDS using the standard performance metrics such as specificity, sensitivity, F-Score and accuracy. Our proposed system can achieve detection accuracy of 91.03% and 92.58% for intruded and normal traffic conditions, respectively. Currently our system misses intrusion when the intrusion duration is very short, i.e., 1 or 2 cycles time. The reason being, such short interruption does not have any noticeable impact on the traffic system and hence on the collected traffic data to show sufficient deviation from normal signal. Future ITSs will have driverless vehicles, smart road infrastructure and various sensors wirelessly connected to TMS, which will attract researchers to work on detecting the vulnerability of future ITSs.

References

1. Pojani, D., Stead, D.: Sustainable urban transport in the developing world: beyond megacities. *Sustainability* **7**(6), 7784–7805 (2015)
2. Feng, Y., et al.: Vulnerability of traffic control system under cyber-attacks using falsified data. In: *Transportation Research Board 2018 Annual Meeting (TRB)* (2018)
3. Yousef, K.M., Al-Karaki, M.N., Shatnawi, A.M.: Intelligent traffic light flow control system using wireless sensors networks. *J. Inf. Sci. Eng.* **26**(3), 753–768 (2010)
4. Khekare, G.S., Sakhare, A.V.: Intelligent traffic system for VANET: a survey. *Int. J. Adv. Comput. Res.* **2**(4), 99–102 (2012)
5. Sundar, R., Hebbar, S., Golla, V.: Implementing intelligent traffic control system for congestion control, ambulance clearance, and stolen vehicle detection. *IEEE Sens. J.* **15**(2), 1109–1113 (2015)
6. Al-Sakran, H.O.: Intelligent traffic information system based on integration of Internet of Things and Agent technology. *Int. J. Adv. Comput. Sci. Appl. (IJACSA)* **6**(2), 37–43 (2015)
7. Coogan, M.A., et al.: Intercity Passenger Rail in the Context of Dynamic Travel Markets. *Transportation Research Board* (2016)
8. Mahindra, R., et al.: A practical traffic management system for integrated LTE-WiFi networks. In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. ACM (2014)
9. Lakshmi, C.J., Kalpana, S.: Intelligent traffic signaling system. In: *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE (2017)

10. Chowdhury, A. Priority based and secured traffic management system for emergency vehicle using IoT. In: International Conference on Engineering & MIS (ICEMIS). IEEE (2016)
11. Nilsson, G., et al.: Entropy-like Lyapunov functions for the stability analysis of adaptive traffic signal controls. In: 2015 IEEE 54th Annual Conference on Decision and Control (CDC). IEEE (2015)
12. Services, N.T.R.a.M.: The benchmark in urban traffic control (2018). http://www.scats.com.au/files/an_introduction_to_scats_6.pdf. Accessed 6 July 2018
13. Passeri, P.: May 2016 Cyber Attacks Statistics (2016). <http://www.hackmageddon.com/category/security/cyber-attacks-statistics/>. Accessed 10 June 2018
14. Kaspersky Lab: Kaspersky DDoS Intelligence Report for Q1 2016 (2016). <https://securelist.com/analysis/quarterly-malware-reports/74550/kaspersky-ddos-intelligence-report-for-q1-2016/>. Accessed 10 June 2018
15. OWASP: Category: OWASP Top Ten Project (2016). https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013. Accessed 10 June 2018
16. Chawla, A., Patial, R.K., Kumar, D.: Comparative analysis of Sybil attack detection techniques in VANETs. *Indian J. Sci. Technol.* **9**(47) (2016)
17. Chowdhury, Abdullahi: Recent cyber security attacks and their mitigation approaches – an overview. In: Batten, Lynn, Li, Gang (eds.) ATIS 2016. CCIS, vol. 651, pp. 54–65. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-2741-3_5
18. Cerrudo, C.: An emerging us (and world) threat: cities wide open to cyber attacks. *Securing Smart Cities* (2015)
19. Directory, V.G.D.: Traffic signal strategic monitor detector data (2017). https://www.data.vic.gov.au/data/dataset/traffic_signal_strategic_monitor_detector_data. Accessed 8 June 2018
20. Ajayi, E.: The Impact of Cyber Crimes on Global Trade and Commerce. Available at SSRN (2016)
21. Poole, A., Kotsialos, A.: Second order macroscopic traffic flow model validation using automatic differentiation with resilient backpropagation and particle swarm optimisation algorithms. *Transp. Res. Part C: Emerg. Technol.* **71**, 356–381 (2016)
22. Song, J., et al.: Research on car-following model based on SUMO. In: 2014 IEEE 7th International Conference on Advanced Infocomm Technology (ICAIT), IEEE (2014)



A Linguistic Approach Towards Intrusion Detection in Actual Proxy Logs

Mamoru Mimura^(✉) and Hidema Tanaka

National Defense Academy, 1-10-20 Hashirimizu, Yokosuka, Kanagawa, Japan
mim@nda.ac.jp

Abstract. Modern malware imitates benign http traffic to evade detection. To detect unseen malicious traffic, a linguistic-based detection method for proxy logs has been proposed. This method uses Paragraph Vector to extract features automatically. To generate discriminative feature representation, a balanced corpus is required. In actual proxy logs, benign traffic is dominant, and occupies malicious feature representation. Therefore, the previous method does not perform accuracy in practical environment.

This paper demonstrates that the previous method is not effective in actual proxy logs because of the imbalance. To mitigate the imbalance, our method extracts important words from proxy logs based on the TFIDF (Term Frequency Inverse Document Frequency) scores. The experimental results show our method can detect unseen malicious traffic in actual proxy logs. The best F-measure achieves 0.94 in the timeline analysis.

Keywords: Drive by download · Neural network · Paragraph Vector
Doc2vec · TFIDF

1 Introduction

Modern malware imitates benign http traffic to evade detection. To detect unseen malicious traffic, a linguistic-based detection method for proxy logs has been proposed [1]. This method uses Paragraph Vector to extract features automatically. In the previous work, this method performed good accuracy in the balanced datasets. To generate discriminative feature representation with Paragraph Vector, a balanced corpus is required. If a corpus contains too much of words and too little of others, an imbalance occurs. Because, the large corpus occupies most of the whole corpus. Such a corpus cannot generate discriminative feature representation. This method expects that the corpus is extracted from actual benign proxy logs and malicious pcap files. Actual proxy logs contain a huge amount of words. To represent a huge amount of words, a corpus requires long-term and much traffic. In contrast, malicious pcap files contain only limited words. Thus, benign traffic is dominant, and occupies malicious feature representation. This is due to the base-rate fallacy phenomenon [2]. Therefore, this method might not perform accuracy in practical environment.

This paper demonstrates the effectiveness of the imbalance with actual proxy logs. To mitigate the imbalance, this paper proposes how to generate a balanced corpus from actual proxy logs. Our method extracts important words from proxy logs based on the TFIDF (Term Frequency Inverse Document Frequency) scores. TFIDF is a numerical statistic that is intended to reflect word importance. To the best of our knowledge, the sole example using TFIDF in the field of network security is extracting features of malware from host logs [3]. Our method can detect unseen malicious traffic from network devices which monitor a wide range.

The main contributions of this paper are as follows: (1) Demonstrated that dominant benign traffic occupied malicious feature representation. (2) Proposed a method to mitigate the imbalance with TFIDF scores. (3) Verified that our method could detect unseen malicious traffic in practical environment.

The rest of the paper is organized as follows. Next section describes Natural Language Processing (NLP) techniques. Section 3 introduces the linguistic-based detection method and indicates the imbalance. Section 4 proposes the method to mitigate the imbalance. Section 5 demonstrates that dominant benign traffic occupies malicious feature representation, and shows the effectiveness of our method. Section 6 evaluates the result and Sect. 7 discusses related work.

2 Natural Language Processing (NLP) Technique

2.1 Paragraph Vector (Doc2vec)

Paragraph Vector (Doc2vec) [4] is an extension of Word2vec, and constructs embedding from entire documents. Word2vec is a shallow two-layer neural network that is trained to reconstruct linguistic contexts of words. Word2vec has two models to produce a distributed representation of words. Continuous-Bag-of-Words (CBoW) model predicts the current word from a window of surrounding context words. Skip-gram model uses the current word to predict the surrounding window of context words with the order reversed. These models enable to calculate the semantic similarity between two words and infer similar words semantically. The same idea has been extended to entire documents. In the same manner, Doc2vec has two models to produce Paragraph Vector a distributed representation of entire documents. Distributed-Memory (DM) is the extension of CBoW, and the only change is adding a document ID as a window of surrounding context words. Distributed-Bag-of-Words (DBoW) is the extension of skip-gram, and the current word was replaced by the current document ID. Doc2vec enables to calculate semantic similarity between two documents and infer similar documents semantically.

2.2 TFIDF (Term Frequency Inverse Document Frequency)

TFIDF is a numerical statistic that is intended to reflect word importance to a document in a collection or corpus, and one of the most popular term-weighting schemes. A TFIDF score increases proportionally to the number of times a word

appears in the document, and is often offset by the frequency of the word in the corpus. TFIDF is the product of two statistics, term frequency and inverse document frequency. Term frequency is the number of times that a term occurs in a document. Inverse document frequency is a measure of how much information the term provides. This means whether the term is common or rare across all documents.

TFIDF of term i in document j in a corpus of D documents is calculated as follows.

$$TFIDF_{i,j} = frequency_{i,j} \times \log_2 \frac{D}{document_frequency_i}$$

A high weight in TFIDF is reached by a high term frequency in the given document and a low document frequency of the term in the whole collection of documents. Therefore, the weights tend to exclude common terms. The ratio inside the log function is always greater than or equal to 1. Hence, the TFIDF score is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the TFIDF score closer to 0.

Our method utilizes TFIDF scores to extract important unique words from proxy logs.

3 Linguistic-Based Detection Method

3.1 Extracting Words

Based on a linguistic approach, a malicious traffic detection method for proxy logs has been proposed [1]. The key idea of this method is treating proxy logs as a natural language. Proxy logs contain date and time at which transaction completed, request line from the client (includes the method, the URL and the user agent), HTTP status code returned to the client and size of the object returned to the client.

This method divides a single log line into HTTP status code, request line from the client, size of the object returned to the client and user agent by a single space. Then, the request line is divided into method, URL and protocol version by a single space. Furthermore, this method divides the URL into words by the delimiters which are “dot” (.), “slash” (/), “question mark” (?), “equal” (=) and “and” (&). This method treats each strings separated by a single space or the delimiters as a word. A single paragraph consists of the words extracted from 10 log lines. Thus, this method derives paragraphs from proxy logs.

3.2 Previous Method

The previous method [1] uses two types of machine learning techniques. One is Doc2vec an unsupervised learning model for feature extraction. This method constructs a Doc2vec model from the paragraphs and trains the model. The paragraphs are derived from benign and malicious proxy logs. Note that these

logs have to be known as benign or malicious. Then, the paragraphs are converted into feature vectors. In this process, the feature representation is automatically extracted by the model. That is why this method utilizes the neural network model. The other is a supervised learning model to classify the feature vectors. This method trains the classifiers with the feature vectors and labels.

Subsequently, this method derives paragraphs from unknown proxy logs. The paragraphs are converted into feature vectors with the trained Doc2vec model. Finally, the trained classifiers predict the label from the feature vectors.

3.3 Imbalance

This method performed good accuracy in the balanced datasets, which were generated from benign and malicious pcap files at the same ratio [1]. This method requires affordable benign and malicious corpuses continually. Because new websites are being created constantly as with malware. The benign corpus is extracted from proxy logs in each organization. The malicious corpus is extracted from malicious pcap files which are downloaded from the websites such as Malware-Traffic-Analysis.net¹. In reality, proxy logs in a large organization contain a huge amount of words. In contrast, these malicious pcap files contain only limited words. If we generate a dataset from both words at the same ratio, the generated corpus cannot represent benign feature adequately. Because the corpus contains only benign words as much as words in malicious corpus. If we merely generate a dataset from both whole words, an imbalance occurs. In this case, benign words occupy most of the corpus. Such a corpus cannot generate discriminative feature representation. Thus, benign traffic is dominant, and occupies malicious feature representation. Therefore, this method might not perform accuracy in practical environment. A further section describes the detail. To perform accuracy in practical environment, this method needs a balanced corpus.

4 Proposed Method

4.1 Notion

To mitigate the imbalance, this paper pursues how to generate a balanced corpus from actual traffic. In general, benign proxy logs contain a huge amount of words. In contrast, malicious pcap files contain only limited words. To generate a balanced corpus, both numbers of unique words have to be adjusted at the same ratio. To adjust the number of unique words, our method utilizes TFIDF scores. TFIDF is one of the most popular term-weighting schemes, and reflects word importance to a document in a corpus. Our method extracts important words based on the TFIDF scores, and removes the ephemeral words. This provides two benefits: generating a balanced corpus and improving classification accuracy. Our method extracts important words from benign proxy logs, and reduces the number of unique words. Then, we can obtain a balanced corpus which contains

¹ <http://www.malware-traffic-analysis.net/>.

both appropriate unique words. Furthermore, our method attempts to extract important words from malicious pcap files. This might improve classification accuracy slightly.

4.2 Overview

This paper proposes a practical linguistic-based detection method which includes how to generate a balanced corpus. Figure 1 shows an overview of the proposed method.

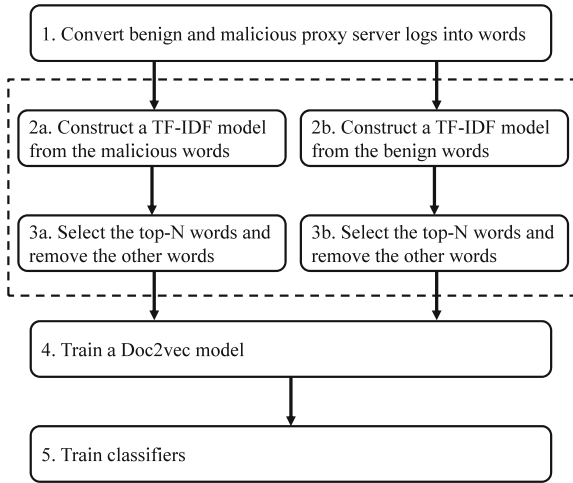


Fig. 1. An overview of the proposed method.

Updated processes are surrounded by a broken line. First, our method converts malicious and benign proxy logs or pcap files into words (1). Pcap files are converted into pseudo proxy logs. Both logs are separated by a single space or the delimiters. A single paragraph consists of the words extracted from 10 log lines. The number of log lines was determined by an empirical approach. Next, our method constructs TFIDF models from the malicious and benign words (2ab), and extracts top-N important words (3ab). Both models have to be constructed respectively. Hence, the extracted top-N unique words are different. Because these unique words have to equally represent both traffic. Note that both models use the same N. To construct a balanced corpus, both numbers of unique words have to be the same number. That is why our method extracts top-N important words, does not extract words above a threshold. The default value for N is the unique word number of the malicious words. N is a parameter value to adjust the number of unique words. Our method extracts important words based on the TFIDF score, and removes ephemeral words. Thus, our method mingles both words, and obtains a balanced corpus. Then, our method trains a

Doc2vec model with the balanced corpus, and converts both words into feature vectors with the trained model (4). Finally, our method trains classifiers with the feature vectors and labels (5). The classifiers are Support Vector Machine (SVM), Random Forests (RF) and Multi-Layer Perceptron (MLP).

These are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training data, each labeled as belonging to one or the other of two categories, these training algorithms build a model that assigns new examples to one category or the other.

Subsequently, our method derives paragraphs from unknown proxy logs in the same way. This method extracts the top-N important words from the paragraphs, which were extracted in the training process. The modified paragraphs are converted into feature vectors with the trained Doc2vec model. Then, the trained classifiers predicts the label from the feature vectors. The predicted label is either malicious or benign.

4.3 Implementation

We implemented our method with gensim-1.2.1, scikit-learn-0.19.1 and chainer-2.0.12. Gensim is a Python library to realize unsupervised semantic modelling from plain text, and provides a Doc2vec model. Scikit-learn is a machine-learning library for Python that provides tools for data mining with a focus on machine learning, which contain SVM and RF. Chainer is a flexible Python framework for neural networks which contain MLP. We use the same models and set the same parameters for comparison with the previous method [1].

5 Experiment

5.1 Dataset

In this experiment, we use captured pcap files from Exploit Kit (EK) between 2014 and 2017 as malicious traffic. EK is a software kit designed to run on web servers, with the purpose of identifying vulnerabilities in client computers. EK seeks and exploits vulnerabilities to upload and execute malicious code on the client computer. We chose some EKs which communicate via a standard protocol and attempt to imitate benign http communication, and downloaded the packet traces from the website MALWARE-TRAFFIC-ANALYSIS.NET. Table 1 shows the detail.

Table 1. The detail of the MTA dataset.

Period	Size	Number	Description
2014	238M	258	Angler, Fiesta, FlashPack, Magnitude, Neutrino, Nuclear, RIG
2015	186M	161	Angler, Fiesta, Magnitude, Neutrino, Nuclear, RIG
2016	373M	406	Angler, Magnitude, Neutrino, Nuclear, RIG
2017	109M	69	RIG

This dataset (MTA) contains the pcap files extracted from traffic of the latest EKs. Our method aims to detect malicious traffic from proxy logs. Thus, we converted these pcap files into pseudo proxy logs. We extracted http traffic from these pcap files and concatenated the requests and responses.

This paper uses actual proxy logs between 2016 and 2017 as benign traffic. These actual proxy logs were collected at a campus network, which belongs to a class B network. This network consists of more than 5,000 computers. Due to our computer resource constraints, we extracted 1G bytes of logs in 2016 and 2017 respectively. We assume that these proxy logs represent benign traffic.

We mingle the malicious logs and benign logs into datasets. We split the datasets into training data and test data for timeline analysis. This paper uses three metrics: Precision (P), Recall (R) and F-measure (F).

5.2 Demonstrating the Imbalance

To compare the previous method [1] with the proposed method, we conducted timeline analysis. In the timeline analysis, we used 2014’s MTA and 2016’s proxy logs as the training data, and 2015’s MTA and 2017’s proxy logs as the test data. Table 2 shows the numbers of unique words and vectors. The previous method does not extract important words. The number of original unique words in benign traffic is 608,805. The previous method constructed a Doc2vec model from the original words. On the other hand, the proposed method extracted 8,200 important words from benign traffic, and constructed a Doc2vec model from the reduced words. The numbers of vectors by both methods are the same, and the numbers of the malicious vectors and benign vectors are quite different. This means the experimental environment is more fair and practical.

Table 2. The numbers of unique words and vectors.

Method	Class	Dataset		
		Train		Test
		Word (N)	Vector	
Previous method	Malicious	8,200	363	233
	Benign	608,805	328,154	331,283
Proposed method	Malicious	8,200	363	233
	Benign	8,200	328,154	331,283

Next, Table 3 shows performance of the timeline analysis. In spite of reducing words, the performance of the benign traffic is almost perfect. We focus on the malicious traffic. The proposed method maintains the performance to a degree in the timeline analysis. The best F-measure has reached 0.90. With the previous method, the decline is too large to detect malicious traffic. The previous method shows quite poor performance. This is because dominant benign traffic occupies malicious feature representation. Recall that the number of unique

words in benign traffic was too larger than the number in malicious traffic. Thus, the previous method does not perform accuracy in practical environment. The proposed method can detect unseen malicious traffic in actual proxy logs.

Table 3. Performance of the timeline analysis.

Method	Classifier	Benign			Malicious		
		P	R	F	P	R	F
Previous method	SVM	1.00	1.00	1.00	0.02	0.61	0.03
	RF	1.00	1.00	1.00	0.00	0.00	0.00
	MLP	1.00	0.99	1.00	0.04	0.57	0.08
Proposed method	SVM	1.00	1.00	1.00	0.78	0.88	0.83
	RF	1.00	1.00	1.00	1.00	0.52	0.69
	MLP	1.00	1.00	1.00	0.95	0.86	0.90

5.3 Measuring Long-Term Performance

To measure long-term performance of the proposed method, we conducted timeline analysis with the all datasets. In this experiment, we used from 2014's to 2016's MTA as the training data and 2015's to 2017's MTA as the test data respectively. In the entire experiment, we used 2016's proxy logs as the training data and 2017's proxy logs as the test data. Moreover, we optimized the numbers of both unique words (N). Table 4 shows performance of the long-term analysis.

For the same reason, we focus on the performance of the malicious traffic. In the case that we used 2014's for training data and the others for test data, the best F-measure has reached 0.94. Overall, performance gradually decreases

Table 4. Performance of the long-term analysis.

Classifier	Training data	Test data	Malicious			Training data	Test data	Malicious		
			P	R	F			P	R	F
SVM	2014	2015	0.98	0.91	0.94	2015	2016	0.90	0.82	0.86
RF			0.98	0.55	0.71			0.94	0.34	0.50
MLP			0.99	0.88	0.93			0.92	0.77	0.84
SVM	2014	2016	0.88	0.71	0.78	2015	2017	0.69	0.66	0.67
RF			1.00	0.34	0.50			0.65	0.22	0.33
MLP			1.00	0.73	0.84			0.65	0.60	0.63
SVM	2014	2017	0.36	0.67	0.47	2016	2017	0.49	0.74	0.59
RF			1.00	0.31	0.47			0.96	0.38	0.54
MLP			0.84	0.64	0.73			1.00	0.71	0.83

every year. Three years later, an F-measure maintains 0.73. In the case that we used 2015's for training data and the following for test data, the best F-measure has reached 0.86. Next year, an F-measure maintains 0.67. In the case that we used 2016's for training data and 2017's for test data, the best F-measure has reached 0.83.

6 Discussion

As a result, our method using proxy logs could detect the latest EKs almost precisely in practical environment. In the practical environment, the previous method [1] showed quite poor performance. This is because dominant benign traffic occupies malicious feature representation. Our method could extract important words from benign traffic, and construct a balanced corpus. That is the reason that our method performed good accuracy in practical environment. Furthermore, our method optimized the numbers of both unique words, extracting important words from both benign traffic and malicious traffic. This adjustment improved classification accuracy slightly. Therefore, extracting important words with TFIDF scores from proxy logs is efficient, and can mitigate the imbalance.

Our method requires several tens of minutes to construct a Doc2vec model and train a classifier in practical environment. In this experiment, our method took roughly from 30 min to an hour. The greater the pcap files or proxy logs to construct the models, the longer the required time. However, our method can construct the models in advance. Therefore, our method performs within a practical time. Our method can classify unknown logs with these pre-trained models in a few seconds. Thus, our method can analyze network traffic or proxy logs in real time.

In this paper, we used malicious pcap files and benign proxy logs. We can obtain these files from the websites such as MTA, and own network system. Though these files might contain privacy sensitive information such as personal information, email addresses or client's IP addresses. To detect unseen malicious traffic, our method requires only pre-trained models. These models do not include any payload and logs. Furthermore, our method does not require client's IP addresses and even distinguishing the client's sources. This means our method accepts most vantage points to monitor traffic.

7 Related Work

Invernizzi et al. [5] built a network graph with IP addresses, domain names, FQDNs, URLs, paths and file names. Their method focuses on the correlation among nodes to detect malware distribution. In this method, the whole parameters are obtained from proxy logs. This method, however, has to cover many ranges of IP addresses, and performs in large-scale networks such as ISPs. In addition, this method requires the downloaded file types. Nelms et al. [6] proposed a trace back system which could go back to the source from the URL

transfer graph. This method uses hop counts, domain age and common features of the domain names to detect malicious URLs. Bartos et al. [7] categorized proxy logs into flows, and extracted various features from the flows. They proposed how to learn the feature vectors to classify malicious URLs. This method can decide the optimum feature vectors automatically. However, this method demands devising basic features for learning. Mimura et al. [8] categorized proxy logs by FQDNs to extract feature vectors, and proposed a RAT (Remote Access Trojan or Remote Administration Tool) detection method using machine learning techniques. This method uses the characteristic that RATs continues to access the same path regularly. This method, however, performs for only C&C traffic. Shibahara et al. [9] focus on a sequence of URLs which include malicious artifacts of malicious redirections, and built a detection system which uses Convolutional Neural Networks. This method uses a honey client to collect URL sequences and their labels, and performs for DbD attacks. Our method uses only proxy logs and does not require any additional information and does not require devising feature vectors at all. In addition, our method performs at any scale and can detect not only DbD attacks but also C&C traffic.

8 Conclusion

In this paper, we focused on the imbalance that benign traffic was dominant and reduced malicious feature representation. This paper demonstrated that the previous method [1] was not effective in actual proxy logs because of the imbalance. To mitigate the imbalance, this paper proposed how to generate a balanced corpus from actual proxy logs. Our method extracts important words from proxy logs based on the TFIDF scores. We conducted timeline analysis with actual proxy logs. The experimental results showed our method could detect unseen malicious traffic in the practical environment. The best F-measure achieved 0.94 in the timeline analysis. Our method does not rely on attack techniques and does not demand devising feature vectors. Furthermore, our method is adaptable, fast and has a few constraints in practical use.

In this paper, we used the datasets which were generated by mixing malicious pcap files and benign logs. Applying our method to other proxy logs is a future work.

Acknowledgment. This work was supported by JSPS KAKENHI Grant Number 17K06455.

References

1. Mimura, M., Tanaka, H.: Heavy log reader: learning the context of cyber attacks automatically with paragraph vector. In: Shyamasundar, R.K., Singh, V., Vaidya, J. (eds.) *ICISS 2017. LNCS*, vol. 10717, pp. 146–163. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72598-7_9
2. Axelsson, S.: The base-rate fallacy and the difficulty of intrusion detection. *ACM Tran. Inf. Syst. Secur.* **3**(3), 186–205 (2000)

3. Chen, Q., Bridges, R.A.: Automated behavioral analysis of malware: a case study of WannaCry Ransomware. In: Proceedings of 2017 16th IEEE International Conference on Machine Learning and Applications, pp. 454–460 (2017)
4. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of 31st International Conference on Machine Learning, pp. 1188–1196 (2014)
5. Invernizzi, L., et al.: Nazca: detecting malware distribution in large-scale networks. In: Proceedings of Network and Distributed System Security Symposium (2014)
6. Nelms, T., Perdisci, R., Antonakakis, M., Ahamad, M.: WebWitness: investigating, categorizing, and mitigating malware download paths. In: Proceedings of 24th USENIX Security Symposium, pp. 1025–1040 (2015)
7. Bartos, K., Sofka, M.: Optimized invariant representation of network traffic for detecting unseen malware variants. In: Proceedings of 25th USENIX Security Symposium, pp. 806–822 (2016)
8. Mimura, M., Otsubo, Y., Tanaka, H., Tanaka, H.: A practical experiment of the HTTP-based RAT detection method in proxy server logs. In: Proceedings of 12th Asia Joint Conference on Information Security, pp. 31–37 (2017)
9. Shibahara, T., et al.: Malicious URL sequence detection using event de-noising convolutional neural network. In: Proceedings of IEEE ICC 2017 Communication and Information Systems Security Symposium, pp. 1–7 (2017)

Short Paper Session II: Security Management



Simau: A Dynamic Privilege Management Mechanism for Host in Cloud Datacenters

Lin Wang^{1,2}, Min Zhu^{1,2}, Qing Li^{1,2}(✉), and Bibo Tu^{1,2}

¹ Institute of Information Engineering, Chinese Academy of Sciences,
Beijing 100093, China

{wanglin1993,zhumin,liqing,tubibo}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences,
Beijing 100049, China

Abstract. Nowadays, a majority of cyber-attacks are associated with the insider threats owing to improper privileges management. Though a number of access control mechanisms have been carried out, the insider threats are continuously increasing. In cloud, however, the physical machines of datacenters are still exposed to danger. Without the trusted hosts as the foundation, any further measurements for virtual machines are in vain. In this paper, we introduce Simau: a mechanism that constrains the privileges of root on each host in the cloud. It deploys a decision engine in user-space to support the variable security policies. The scope of Simau covers both kernel-space and user-space. Under Simau, once a system administrator logs into a host, he has only the least privileges to finish his missions and all his requests for privileged operations are determined by Simau. The experiments at last show good performance of our mechanism.

Keywords: Insider threats · Root privileges · Privilege management
Host · Cloud · Security

1 Introduction

With the popularity of cloud computing, the organizations who have transferred their local services to cloud datacenter are increasing. There is no doubt that the security of cloud datacenter draws a considerable attention. Cloud Security Alliance (CSA) published *Cloud Computing Top Threats in 2016*, pointing out that the breaches caused by improper access control are ranked the second [14]. Usually, the tragedy is caused by a worker who gains the privileges more than what he should have. Additionally, the security of hosts is the central premise of any security-related topic in a cloud datacenter. No protection for the hosts, no security for virtual machines. In conclusion, it is significant to apply appropriate privilege management to physical servers in cloud.

There are several circumstances when the cloud datacenters are posed to danger. Firstly, multifarious work of employees causes privileges abuse inadvertently. Moreover, once attracted by interests, the system administrator who

grants the total control rights over the system may copy confidential archives out or remove the database, causing immeasurable damage. Furthermore, the maintenance workers are often delegated with `root` privileges when asked to troubleshoot or upgrade. Finally, there are attacks targeted for the `root` privileges like social engineering against administrator's password.

Existing work has separated privileges in hosts through the combination of Linux Security Module (LSM) [1] and SELinux [11]. LSM is an integrated structure mediating access to the kernel's internal objects through hooks. SELinux is a successful application for LSM that implements a kind of Mandatory Access Control (MAC) in the kernel. However, to protect hosts from insider threats, we need a dynamic privilege management mechanism which assigns the privileges on-demand. The decision logic of SELinux is fully implemented in the kernel. As the result, though SELinux is equipped with several kinds of strategies, they should be perceived as immutable without code recompilation. Furthermore, the hooks of LSM is fixed, which is not enough flexible. Apart from some important objects under the protection of LSM, there are some prominent executions which are independent of any objects, or it is hard to find a clear range of entities attached to them. To manage this kind of operations, we have to define custom hooks. For example, "To install or uninstall software", it seems that the objects are the files related to the software while, in fact, it is not easy to identify all files and directories associated with the software so that it is impossible to bind pertinent data items to the procedure. In that case, we can insert a hook to the installer to block the unauthorized operation, while LSM is unable to support the custom hooks.

This paper introduces *Simau* which actualizes dynamic privileges management for the hosts in cloud datacenters. *Simau* precludes the unauthorized process from performing privileged operations by inserting hooks that spread over the whole system. Certainly, the hooks of *Simau* in a host are not something of a novelty. Nevertheless, upon this foundation, *Simau* provides autonomy—it supports user-defined hooks. It not only supports the reuse of LSM hooks to protect kernel objects but also has the ability to implant tailor-made hooks to satisfy different demands, as well as in user-space. *Simau* isolates decision-making from enforcement point by deploying a decision engine in user-space so that the security policies can vary as required and lead to the alteration of execution result in hooks timely. These policies can be operated by remote controller or local administrators and are carried into effects immediately.

Our contributions in this paper are:

1. We propose a dynamic privilege management mechanism for hosts in cloud datacenters that takes effect on both kernel-space and user-space.
2. We deploy a decision point in user-space that dominates the decision timely according to varied policies.
3. We devise a method of inserting user-defined hooks to kernel on-the-fly without such great pains by livepatch.

The rest of the paper is arranged as follows: in the next section, we state the background, including an overview of *Simau* and threat model. Then, we

introduce the design principles of Simau in Sect. 3. In Sect. 4 we depict the technical details of our prototype system. Experiment in Sect. 5 reveals the performance of our mechanism. The related work and conclusion are in Sects. 6 and 7 respectively.

2 Threat Model

In a typical cloud datacenter environment, the physical servers admit remote users and occasionally local maintenance. Simau that is distributed in each physical machine receives and enforces commands from controller. The controller is deployed in a central machine, keeping connection with each server host.

Trust Computing Base (TCB) is used for Simau to boot all components into a trusted initial state. We assume that Simau runs in a healthy environment with TCB, including secure hardware and operating system, as well as the built-in security mechanisms. The components of Simau are well protected by some process protection measurements [9, 10]. For the controller, we assume that the administrator of the controller is not allowed to operate on hosts locally and he will not be in collusion with one who may have the local access to hosts.

Given such a premise, our threat model is a single bad behaved employee who may perform some important operations on the hosts in a cloud datacenter, such as a service manager who has the right to start or stop a couple of crucial services, a system administrator who gains the `root` privileges on a certain host, or storekeeper and cleaner who has the opportunity to access physical machines. The employee may wield his power in wrong time, abuse his right or take the advantages of his job, as a result, cause great loss to an organization.

As a typical instance, a third-party maintenance worker is asked to upgrade software for an organization. In a traditional way, he may gain the administrator password and wield `root` privileges because he has to operate on some important directories. By this way, the worker obtains the additional rights that far more than he needs. He may plant a malware into the kernel, or steal the information asset through portable devices. If Simau, the worker will be assigned the privilege of installer only. He can neither insert kernel modules nor load the devices because Simau will refuse his requests.

3 Design

In this section, we introduce the design principles of Simau. Figure 1 shows the main components of Simau and the interactions among them. There are four main parts as the dash denotes: Policy Administration Point (PAP), Policy, Policy Decision Point (PDP) and Policy Enforcement Point (PEP). We will give the anatomy of each of them.

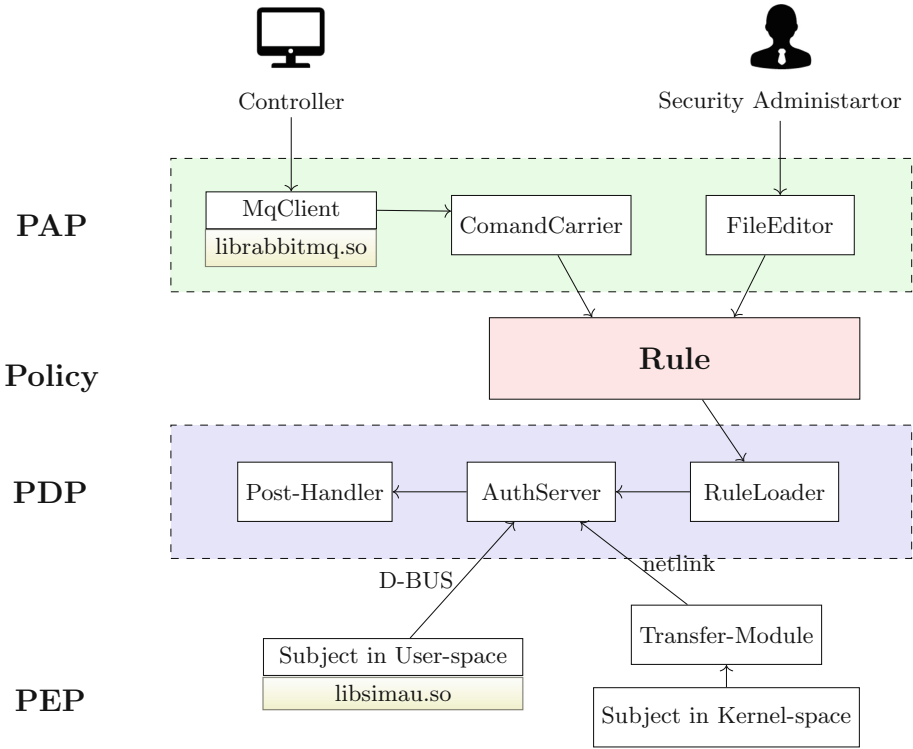


Fig. 1. Simau Architecture

3.1 Policy Administration Point and Policy

PAP is the interface for remote controller and local administrator to edit Rule files. MqClient which is an implementation of Message Queue is responsible for connection with controller. It receives messages from controller, parses them and passes the instructions to ComandCarrier to enforce the instructions on Rule files. The instructions are usually related to add a new rule, modify an existed one, or delete some rules. FileEditor in PAP refers to the common editor like *vim*. Security Administrators can carry local maintenance work out via editing the Rule files with the editors directly.

Policy is the security strategy we apply to our system. It exists in the form of rules set regulating how Simau performs.

3.2 Policy Decision Point

PDP is a central component where the decision logic is fully actualized. It has interactions with both Rule files and PEPs. Post-Handler, AuthServer, and RuleLoader are the main parts. RuleLoader loads the rules from Rule files whenever a modification of files is detected. AuthServer makes a decision on every

request according to the rules. The form of request is similar to the rule that can be regarded as the binding of a set of requests plus the effect upon them. For example, “Reading a file that is created on 2017-9-1” is a typical request and “Reading the files that are created after 2017-8-26 is not allowed” is a rule. The primary jobs of decision-making are searching and matching. In this case, the request is matched to the rule and a negative value will be returned.

Post-handler offers an obligation mechanism for Simau. That is, some extra processes are supported after modification detected on Rule files to enforce the new policies. RuleLoader acts as the monitor to stare at the Rule files here. AuthServer becomes conscious immediately whenever new rules are loaded and it invokes a proper post-handler to perform some extra duties if any. For example, a rule reading “Log-in is allowed from 8 am to 5 pm.” has an impact on the users who try to visit the system, rather than the online users, which is apparently irrational. The right way is that we kick out the illegal online users when the time is up so that we put all users under the control of Simau. In this case, when AuthServer is aware of the appending or alteration of rules related to log-in, it invokes the corresponding post-handler immediately to check whether the online users are in their valid time. This obligation mechanism can function as a “ruler” to make sure that every entity obeys our policies.

3.3 Policy Enforcement Point

PEP acts as the gateway to a privileged operation. It exists in a process in the form of a Simau-hook. *subject* is a process under the control of Simau with a served Simau-hook. Our hook separates the subject into two parts: one is the meaningful portion that we place control on, another is the unprivileged one. The meaningful portion often refers to functions or steps that directly have impacts on the outcome of procedures. If these functions were stepped over, the original procedure would fail. Simau-hook makes the “meaningful portion” be ignored if the subject is unauthorized no matter what other privileges it has been delegated by other mechanisms. Because implementation details of kernel-space and user-space are different, the subjects are segregated into user-space subjects and kernel-space subjects.

The Transfer-Module is designed for subjects in kernel-space and it works as a coordinator between AuthServer and subjects. For the Simau-hook of subjects in kernel-space, since it has to be inserted into kernel, it is inevitable to modify the code. The purpose of Transfer-Module is to share part of the responsibility and decrease the workload of adjustment. As is known to all, the communication with user-space in kernel is not an easy task. Without Transfer-Module, the new code will make the original segment long and convoluted. After alleviating the burden, Simau-hooks in kernel are only responsible for two necessary functions: communication with Transfer-Module which in kernel and collection some information for request constructing. It is obvious that to communicate with Transfer-Module which is written as a Loadable Kernel Module(LKM) is simpler than that with AuthServer.

3.4 Workflow

A subject will be trapped in the place where the Simau-hook serves. Under the impact of Simau-hook, the Simau authorization is carried out. If the subject is authorized, it would be allowed to continue, otherwise, it would go to fail. Note that Simau can coexist with other mechanisms like ACL or DAC. The combined policy of them is negative-override which means if any of them gains a negative value, the subject would go to fail.

The arrows in Fig. 1 indicate the workflow during an authentication. When the process arrives at a PEP, it generates an access request including the action identity and other collected elements that help match the right rule. For PEPs in kernel, the request is passed to Transfer-Module by a direct function call. Once the Transfer-Module receives the access request, it reconstructs it in a formal way and sends it to AuthServer through netlink. For PEPs in user-space, the request is forwarded to AuthServer by D-Bus [16]. AuthServer searches the matching rules for the request sequentially and the matching rule's answer will be returned. If there is no outcome for the request after scanning all the rules, a failed value is returned. As soon as the final decision is obtained, AuthServer returns the result to Transfer-Module or to PEPs in user-space directly. Finally, PEP gets the reply and enforces it.

Another flow that has an association with AuthServer is the interaction with a post-handler. Once Rule files are modified through PAP, either alteration or appending, AuthServer will be informed by RuleLoader immediately and post-handler is activated to perform its duty if provided.

4 Implement

According to our design principle, we realize a prototype of the system. In this section, we will give the technical details of main components. The last part is a demonstration of our prototype system.

4.1 Communication

The method we use in communication with a remote controller is message queue. The message-oriented middleware protocol is Advanced Message Queuing Protocol (AMQP) [17]. MqClient realizes both message queue consumer and publisher based *librabbitmq.so*. *librabbitmq.so* is an open-source C-language AMQP client library.

The interplay between AuthServer and Subjects is of paramount importance during an authentication. To provide service for subjects in both user-space and kernel-space, we apply two Inter-Process Communication (IPC) methods—D-Bus and netlink. D-Bus is for subjects in user-space. AuthServer will expose its authorization API on D-Bus. The subjects can lunch a call to AuthServer and get a reply from it through the bus. Netlink, as well as the Netlink socket family, is a Linux kernel interface used for IPC between user-space and kernel-space. As depicted above, a Transfer-Module is designed for sharing the load

of Simau-hooks in subjects of kernel-space. When it is initialized, it registers a special protocol via which Transfer-Module and AuthServer can communicate with each other.

4.2 Hooks

PEP is inserted in a process in the form of a Simau-hook. If a program in user-space is going to use Simau, it has to invoke Simau authentication functions explicitly in code. One approach is to add the appropriate Simau-related functions in source code and recompile the program. The other effective way relies on the extensibility of the program. If the program has interfaces for custom binary in some crucial points, such as the point where *Linux-PAM(Pluggable Authentication Modules for Linux)* [18] is deployed, we can realize the Simau-related functions in them too. Otherwise, it is impossible to use Simau. Despite this, for user-space, to insert PEP is not a tough task because coding in user-space is free and easy compared with that in kernel-space. Hence the kernel-space is our main concern.

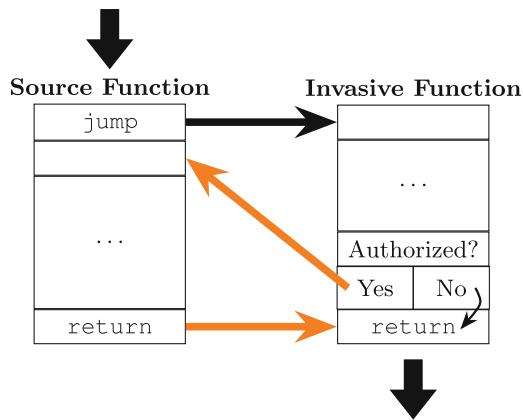


Fig. 2. Patch function structure in kernel-space

To plant PEP into kernel code escalates complexity of coding. To recompile or rebuild kernels unsuitable when reconstruction is not allowed. After all, for some operating systems, the source code is not available and it is also unwise to take great pains to install a software. As it is known to all, many patch methods have been actualized in kernel. For example, *Livepatch*, as the name indicates, is a small piece of code “sewn” on kernel to cover the original one. Figure 2 is the principle diagram. To be vivid, combining theory with livepatch, we call the original kernel code *Source Function (SF)*, and the covering code *Invasive Function (IF)*. The black arrows show the mechanism for live-patch itself. The patch plants a `jump` instruction as a tamp at the very beginning of

every function, namely SF in our picture, the destination of which is the first instruction of IF. When the process runs into the SF, the `jump` instruction is executed immediately after setting the runtime environments. Then, the next construction to be executed is redirected to the IF. The runtime environments, like stack or heap, however, are that of SF, because this `jump` can be perceived as the conditional branch in a sequential piece of code of `if`. When the process runs into `if`, there are two paths to jump to and has nothing to do with the runtime environments. So that, when the process runs into `return` in the IF, the address that invoked the SF is returned as if the IF were the invoked function.

The orange arrows are the flow of our mechanism. The code of PEP is written in IF. When the process is redirected to the IF under the force of patch, the PEP will perform its duty, to collect some elements etc. The communication method we used in kernel-space is the calling of the external function and synchronization primitive. Then, after receiving a message from Transfer-Module, the PEP will enforce the result commands. If it is authorized, the process will go to the next instruction of `jump` in the SF and the SF is processed as the way it is. Note that, the SF discarding the first instruction is taken as a complete function to be invoked here. That is, a function call happens and the new runtime environment for the SF without `jump` is set. Hence SF will be back to IF under the action of `return` in the last. Otherwise, if it not authorized, the SF is skipped completely and the process goes to the next instruction right after authorization, as the black arrow from “No”. Finally, the IF is returned with an error code indicating the failure of this function. The operation will be redirected to error treatment program. In this way, we deploy the PEP into kernel without such much cost.

5 Experiment

In this section, we measure the load that brings about by the Simau-hook. The experiments are carried in a test-server (Linux 4.4.0-87-generic, dual-core 64-bit Intel Core i5-3470 at 3.20 GHz, with 6MB/core cache and 2 GB memory).

We examine the running time of an authorized process with Simau-hook in both user-space and kernel-space. Log-in is for the test in user-space. Since Simau-hook has been inserted into PAM authentication management procedure, the running time of PAM authentication is examined. We test the spending time of PAM authentication management with Simau-check and record the average value which is shown in Table 1. As shown in the second column, PAM authentication management with Simau-check spends $6.095e-3$ s on average. Likewise, we record the time spent without Simau-check in the third column.

LKM is for the test in kernel-space. We examine the spending time of command `insmod` and `rmmod` respectively. Similarly, we have calculated the average time out. *Time delta* in Table 1 reveals that the cost of Simau-check is negligible because all of them are no more than $1e-3$ s which a human being can hardly feel about.

Table 1. Performance results of Simau

Action	with Simau-check(s)	without Simau-check(s)	Time delta(s)
Log in	6.095e-3	5.892e-3	2.03e-4
Insert module	1.924e-4	1.540e-4	3.84e-5
Remove module	1.870e-4	1.553e-4	3.17e-5

6 Related Work

The popular solutions to defend against insider threats are Access Control Mechanism [2, 7, 8, 20] and the various variants of them [3]. However, all of them pay attention to user-space with complicated policies. SELinux [11] are mechanisms in kernel-space based on LSM [1] but it is not flexible enough because it takes great pains to change security policies. CAP [4] and LandLock [6] implements an explicit function to alter policy but it is limited to a single process. AppArmor [21], known as a simple version of SELinux, aims to constrain a process with limited resources. In contrast, Simau provides global control for each host instead of a single process.

Container [5], Jails [12] and Zone [13] are intended for providing a isolated area to confine privileges. There is no doubt that they are not suitable for administration because the user has a limited perspective of the whole system.

TOMOYO Linux [22] and other security OS [23] proposed a new kind of OS or OS module to manage the privileges of user and process. Simau, on the other hand, does not require to recompile the kernel or replace OS.

7 Conclusion

We propose a dynamic privilege management mechanism for hosts in cloud datacenters. Our mechanism supports user-defined hooks to block processes and takes effects in both kernel-space and user-space. We deploy decision engine in user-space to actualize the variable security policies that can be altered on-demand. We further make a study on some important operations and try to regulate them under Simau. The experiment proves the ability of our mechanism. Finally, since it offers a programmable access control for hosts and realizes the separation of the control panel and action, Simau has great compatibility with the software-defined techniques.

Software-defined infrastructure (SDI), for example, brings many new approaches for managing, monitoring, etc. within clouds [15]. It breaks the restrains of the hardware-centric infrastructure, establishing a flexible and scalable fundamental structure. Simau can be one of the feasible technique that provides the enable support for SDI. We expect that Simau could play his significant role in the future.

Acknowledgement. We would like to thank the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Key Research and Development Plan of China under grant No. 2016YFB0801002.

References

1. Wright, C., Cowan, C., Morris, J., Smalley, S., Kroah-Hartman, G.: Linux security module framework. In: Ottawa Linux Symposium, vol. 8032, pp. 6–16 (2002)
2. David, F., Richard, K.: Role-based access controls. In: Proceedings of 15th NIST-NCSC National Computer Security Conference, vol. 563 NIST-NCSC, Baltimore (1992)
3. Rajkumar, P.V., Sandhu, R.: POSTER: security enhanced administrative role based access control models, pp. 1802–1804 (2016)
4. Hallyn, S.E., Morgan, A.G.: Linux capabilities: making them work. In: Linux Symposium, vol. 8 (2008)
5. Shalev, N., Keidar, I., Weinsberg, Y., Moatti, Y., Ben-Yehuda, E.: WatchIT: who watches your IT Guy?. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 515–530: ACM (2017)
6. Landlock: programmatic access control. <https://landlock.io/>
7. Lindqvist, H.: Mandatory access control. Master’s thesis in Computing Science, Umea University, Department of Computing Science, SE-901, vol. 87 (2006)
8. Li, N.: Discretionary access control. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security, pp. 353–356. Springer, Boston (2011). <https://doi.org/10.1007/978-1-4419-5906-5>
9. Costan, V., Devadas, S.: Intel SGX explained. IACR Cryptology ePrint Archive, vol. 2016, p. 86 (2016)
10. Azab, A.M., et al.: Hypervision across worlds: real-time kernel protection from the arm trustzone secure world. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 90–102. ACM (2014)
11. McCarty, B.: SELinux: NSA’s Open Source Security Enhanced Linux. O’Reilly, Newton (2005)
12. Kamp, P.-H., Watson, R.N.: Jails: confining the omnipotent root. In: Proceedings of 2nd International SANE Conference (2000)
13. Price, D., Tucker, A.: Solaris zones: operating system support for consolidating commercial workloads. In: LISA, vol. 4, pp. 241–254 (2004)
14. Cloud Security Alliance: Cloud Computing Top Threats in 2016, Cloud Security Alliance, Top Threats Working Group, February 2016
15. Gu, G., Hu, H., Keller, E., Lin, Z., Porter, D.E.: Building a security OS with software defined infrastructure. In: Proceedings of the 8th Asia-Pacific Workshop on Systems, p. 4. ACM (2017)
16. D-Bus. <https://www.freedesktop.org/wiki/Software/dbus/>
17. Vinoski, S.: Advanced message queuing protocol. IEEE Internet Comput. **10**(6), 87–89 (2006)
18. Samar, V.: Unified login with pluggable authentication modules (PAM). In: Proceedings of the 3rd ACM Conference on Computer and Communications Security, pp. 1–10. ACM (1996)
19. Linux Programmer’s Manual CAPABILITIES(7). <http://man7.org/linux/man-pages/man7/capabilities.7.html>

20. Ferraiolo, D., Chandramouli, R., Kuhn, R., Hu, V.: Extensible access control markup language (XACML) and next generation access control (NGAC), pp. 13–24 (2016)
21. AppArmor wiki. <https://wiki.ubuntu.com/AppArmor>
22. TOMOYO: A Security Module for System Analysis and Protection (2018). <http://tomoyo.osdn.jp/>
23. Santos, N., Rodrigues, R., Ford, B.: Enhancing the OS against security threats in system administration. In: Narasimhan, P., Triantafillou, P. (eds.) Middleware 2012. LNCS, vol. 7662, pp. 415–435. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35170-9_21



USB Packets Filtering Policies and an Associated Low-Cost Simulation Framework

Xiaoshu Ji¹(✉), Gervan Le Guernic², Nora Cuppens-Boualahia¹,
and Frédéric Cuppens¹

¹ IMT Atlantique, Rennes, France

{xiaoshu.ji,nora.cuppens,frederic.cuppens}@imt-atlantique.fr

² DGA Maîtrise de l'Information, Rennes, France

gervan.le-guernic@intradef.gouv.fr

Abstract. In recent years, USB has become the most popular standard for connecting hosts and peripherals due to its plug-and-play and fast speed features. However, with the emergence of attacks such as badUSB, USB security issues become increasingly prominent. In reaction, different USB protection mechanisms have been proposed, including USB communication filtering. Nevertheless, it is worth noting that currently there is no formalised universal USB filtering strategy, and the vast majority of those filters are implemented at the OS level, leaving a part of the OS and the firmware of USB host controllers unprotected. This paper proposes flexible, formalised, universal USB filtering policies and explores the differences between filtering USB communications at the OS level and directly at the USB packet transmission level. Moreover, we address a simulation framework that can be used in the early stages of research and development to conceive and evaluate USB packet filtering policies.

Keywords: USB · Security · Filtering policy

1 Introduction

The Universal Serial Bus (USB) is an industry standard proposed in the mid-1990s that provides ubiquitous plug-and-play connectivity for computer peripherals. Although it makes the use of computer peripherals more convenient, it brings many security issues. USB-related attacks have been transformed from the traditional use of USB storages as carriers for the spread of Trojan viruses to exploiting the vulnerability of the USB protocol, which lacks authentication mechanisms. Traditional USB antivirus software is not effective against such attack type.

Considering that the information declared by the USB device is not reliable and adding an authentication mechanism requires to modify the USB communication process, USB communication filtering is an excellent approach to limit the behaviours of USB devices to access the host. However, there is currently

no flexible, formalised, and extensible USB filtering policy, which makes current USB filtering lacks versatility. Additionally, the vast majority of those filters target USB Request Block (URB) packets and are implemented at the operating system (OS) level [9], leaving a part of the OS and the firmware of USB host controllers [2] unprotected.

In this paper, we not only explore the differences between filtering USB communication at the OS level (URB) and filtering directly at the USB packet transmission level, but also formalise the universal USB devices filtering policies and propose a simulation framework to evaluate them for the early research stage. The remainder of this paper is organized as follows: Sect. 2 presents the USB security issues and the motivation for proposing universal, formalized USB filtering policies and simulating the filtering on USB packet; Sect. 3 proposes the definition and the formalization of USB packet filtering policies; Sect. 4 shows the architecture of the USB packet filtering simulation framework, explains the filtering process and analyses the result of experimentation; and finally Sect. 5 concludes the paper and outlines the future work.

2 Background and Motivation

2.1 USB Security Issues

USB has been developed and improved for more than 20 years since its introduction. In the beginning, attackers used USB devices as carriers of attacks. However, in recent years, more and more attackers have focused their attacks on the USB protocol itself which lacks an effective authentication mechanism. There exists no uniform unalterable identifier for a USB device and the host passively believes the information declared by USB devices. An attacker can exploit this vulnerability by using some specialised hardware, such as Rubber Ducky, or by reprogramming the USB firmware [7] to deceive the trust of the host and gain control over the host. Since the attack occurs on the firmware of the USB device where we cannot detect, it becomes one of the USB-related attack types which are the most difficult to defend currently.

Besides, the earlier versions of USB protocol specification, such as USB specification version 2.0, use the broadcast mode to send USB packet from the host to USB devices. Under normal circumstances, when there is a broadcast packet from the host, only the destination USB device will respond, and other USB devices will discard the packet. However, this communication feature is exploited by attackers to conduct eavesdropping attacks.

2.2 Motivation of Proposing Universal Formalized Flexible USB Filtering Policies

Although the kind of USB attacks has continued to increase, fortunately more defence mechanisms against USB attacks have been proposed in recent years, such as USB communication filtering [4, 5, 10].

USB communication filtering is a USB defence mechanism that implements USB access control through the verification of the USB device information and restriction of the functions or behaviours of the USB device. At present, there is no research proposition for a universal formalised extended USB communication filtering policy which can promote the development of USB communication filtering research.

Considering the USB communication filtering policies from the top-view can involve less implementing technology making the policies more versatile. Besides, formalising filtering policies not only makes them easier to understand and express but also more conducive to improve their interactivity and reusability. This can save a lot of time and labour costs for practical applications. What's more, with the development of USB communication technology, USB has gradually replaced the traditional COM and PS/2 communication methods between personal computers and peripherals. Different kinds of USB devices have different functions and behaviours, which causes the varied of functional limitations and behaviour filterings. In order to adapt to this diversity, USB communication filtering policies must be flexible.

2.3 Motivation of Filtering USB Packets

USB communication filtering can be applied to different layers of the USB communication system. Figure 1 illustrates USB architecture. A Universal Serial Bus (USB) client driver cannot communicate with its device directly. Instead, the client driver creates requests and submits them to the USB driver stack for processing. Within each I/O request, the client driver provides a variable-length data structure called a USB Request Block (URB) [6], and sends it to the USB host controller driver via the USB core. Finally, a URB packet will be processed to USB packets which is the smallest logical transfer units of USB communication and be sent to the USB device. When a USB device sends information to a host, the process is the opposite of the above. When packets arrive at the host, the USB host controller will only extract the payloads of the packets to generate a URB packet according to the request information type.

Although both USB packet filtering and URB filtering can filter the USB communication, there are still many differences between them. These differences are mainly reflected in three aspects: the difficulty of implementation, the universality and the scope of protection. The implementation of USB packet filtering can be independent of the host and USB devices. URB filtering requires different implementations depending on the type and version of the system, which makes it less universal than USB packet filtering. Although implementing USB packet filtering requires some specific hardware, it can filter USB packets sent by USB devices before they reach the host. URB filtering filters URBs generated by the USB host controller after processing USB packets. It may let the original harmful information entering the OS system and cause a potential threat. That is the reason for applying our filtering policies on USB packet.

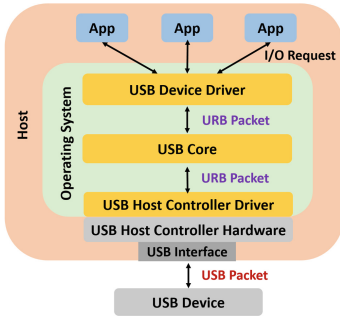


Fig. 1. USB architecture

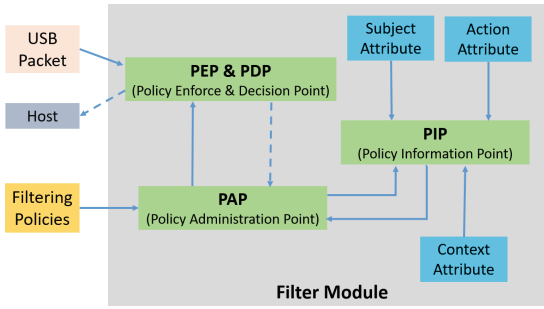


Fig. 2. Filtering module

3 Filtering Policy Definition

3.1 Filtering Requirement

The approach proposed in this paper uses USB packet filtering to achieve a more flexible lock mechanism for USB communication. Authenticating USB devices (based on manufacturer ID and product ID) and verifying their declared functionalities is the first step. It cannot ultimately defend against USB attacks, such as badUSB. Further filtering of the functionalities of the USB device is necessary.

According to the USB protocol [11], USB communication can be divided into two steps: enumeration process and functional communication. In the enumeration process, the host requests the basic information of the USB device and assigns the address to it. When the enumeration process is completed, the functions of the USB device can be used, which is also called functional communication. The context covers both of them as shown in Fig. 2. According to the actual situations, the following four kinds of USB packet filtering policies are proposed:

- **Type 1: Filtering policies solely based on time.** Such policies allow or disallow USB communications by the current time. If the access condition disallows USB communications, every USB packets are simply dropped without any processing. For example, such policies can be used to disallow USB connections outside of office hours.
- **Type 2: Filtering policies based on time and device class.** Such policies allow or disallow the use of specific USB device types by the current time and device class. In addition to the current time, the packets exchanged during the enumeration phase must be parsed in order to decide if the connection of this specific device class is allowed or not. Once a decision has been taken for a device, later USB packets from or to this device can be dropped without any further processing.
- **Type 3: Filtering policies based on device authentication.** Such policies authenticate devices by different information provided during the enumeration phase (such as product ID and vendor ID). After the enumeration phase, packets coming from unauthenticated devices are dropped. Now

more and more enterprises are aware of the importance of USB security. They require employees to use unified USB devices which are purchased by the enterprise. This policy can be used to achieve a similar effect. Furthermore, automatic mechanisms enforcing such policies are more reliable than the employee's consciousness.

- **Type 4: Filtering policies based on device functionalities.** Such policies map USB packets to device functionalities and drop every packet related to a functionality that is not authorised. As explained earlier, a USB device can have multiple interfaces, which means it can have multiple functions. However, sometimes we do not need all of them. In this case, the functions can be selected by the policy, and reduce potential risks.

3.2 Filtering Policy Formulation

Those policies are formalised below in order to clarify them and make them easier to understand. Each USB device has many properties, such as vendor ID, product ID, device class, and so on. The same type of USB device may have different usage restrictions according to different properties. Whereas Attribute-Based Access Control (ABAC) [3], based on attributes which can be about anything and anyone, takes more broad parameters into account. It is the most suitable model thanks to its flexibility and relatively fine-grained feature.

ABAC can define permissions based on any security-relevant characteristics, known as attributes [12]. For our filtering policies, we are concerned with three types of attributes:

- **Subject Attribute.** A subject is an entity that can apply an action to an object. Each subject has some attributes that define the identity and characteristics of the subject. In our case, a subject is a USB device and its associated attributes can be the information which is stored in the descriptors, such as Vendor ID, Product ID, USB class code and so on.
- **Action Attribute.** In this paper, action refers to a function of the USB device. One of the action attributes should be the endpoint which can identify one of the functions of the USB device as well as the transfer type and direction. Because the endpoint information is in the firmware of USB device, it is hard to get it before the enumeration process. We will use a dynamic table to express this endpoint attribute and complete the dynamic table during the enumeration process.
- **Context Attribute.** Context attributes refer to the operational, technical, and even situation when access occurs. For USB packet filtering policies, we only consider time as context. The context attributes can be a date, week or time.

Filtering policy rules can be formalized by ABAC model as follows:

1. S , A and C are respectively the sets of subjects, actions and contexts.
2. S_{att_i} ($1 \leq i \leq I$), A_{att_j} ($1 \leq j \leq J$) and C_{att_k} ($1 \leq k \leq K$) are respectively the pre-defined subjects attributes, actions attributes and contexts

attributes where I , J and K are respectively the number of pre-defined subjects attributes, actions attributes and contexts attributes.

3. Function notation is used to assign the value for individual attributes. Different types of attribute values, atomic elements in USB packet filtering policies can be divided into two types: categorical element and numerical element. For categorical element, the operator “=” is used for the string type attribute value. For numerical element, the operators “=”, “<”, “>”, “≤”, “≥” are used for the integer, real, date/time data types attribute values. The combination operators such as “∧”, “∨” and “−”, can be used when an attribute has many values. For example, vender ID = “Logitech” and $(9 : 00 \leq Time \leq 12 : 00) \vee (14 : 00 \leq Time \leq 18 : 00)$.
4. $ATTR(s)$, $ATTR(a)$, $ATTR(c)$ are respectively the set of attributes which belong to the subject (s), action (a), context (c) in one filter policy. If all the pre-defined attributes of subjects expressed as $\{S_{att_1}, S_{att_2}, \dots, S_{att_I}\}$, then $ATTR(s) \subseteq \{S_{att_1}, S_{att_2}, \dots, S_{att_I}\}$. $ATTR(a)$ and $ATTR(c)$ have the same relationship as $ATTR(s)$.
5. In the filtering policy form, a policy that decides on whether permitting a USB device s can access the host with an action a in a particular context c , is a function of s , a , and c 's attributes:
 Policy: $is_permitted(s, a, c) \leftarrow f(ATTR(s), ATTR(a), ATTR(c))$
 If the result of the function which evaluates all the attributes of s , a , and c is true, then the USB device is permitted to access the host with the action a ; otherwise the access will be denied.

4 Simulation Framework

4.1 Architecture

As far as we know, no software can directly capture USB packets without the help of specific hardware. In addition, the known USB packet capture hardware does not support modifying the content of USB packets. Therefore, how to simulate different USB communication cases to test the proposed packet filtering policies in the early stages of the research is an important issue. For this, a low-cost USB communication simulation framework is implemented.

Ellisys USB Explorer [1] is a USB protocol analyser hardware which can monitor and capture real USB communication packets. We use the structure of its export format in the simulation framework. A simple USB communication amounts to thousands of USB packets. Using Ellisys USB Explorer can get real USB packets more efficiently and export them in files. We can build different test cases easier by modifying the packet contents in the files. According to the user requirements, filtering policies are created by using the EXtensible Access Control Markup Language (XACML) [8] standard, which is a standard of declarative fine-grained, ABAC policy language.

Figure 3 shows the architecture of the simulation framework. The parser is in charge of analysing the USB packets file and classifying the packets according to their source (from host or device). The multi-threaded features of the Java

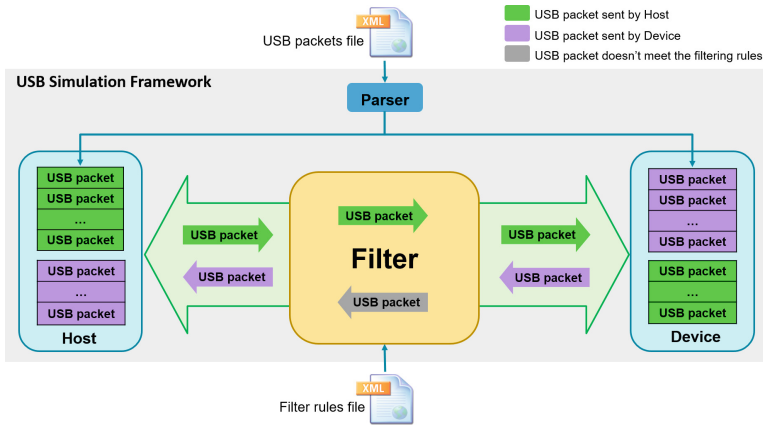


Fig. 3. Simulation framework architecture

language is used to simulate the communication between the host and USB devices. The filter module is responsible for parsing and enforcing the filtering rules. The module can be improved or updated according to the filtering policies.

4.2 Filtering Process

Figure 2 illustrates the most important components of the filtering module. Policy Administration Point (PAP) is responsible for parsing the file storing the filtering policies. According to the attributes in the policies, the PAP sends a request to the Policy Information Point (PIP) to get the real attribute value which will be used during the filtering process. For instance, the vendor id attribute value in the policy is “Kingston” which is the brand name known by the user. However, in the real USB communication, the USB device only sends the code (0x0951) of the brand instead of the name. We can get this information from the USB Implementers Forum (USB-IF) website. After getting the values from the PIP, the policy will be sent to the Policy Enforcement and Decision Point (PEP&PDP) to be enforced for filtering the USB packets. It is noteworthy that in general we do not know the internal structure of the USB device before the filtering. In other words, when setting up a filtering policy, we only know which functionality of the USB device should be limited, but we do not know what kind of value should be filtered. This kind of information can be collected from the enumeration process which occurs before starting the functional communication stage during the USB communication. This is the most significant difference from the traditional packet filtering firewall. This is a dynamic acquisition process which is shown by the imaginary line from the PEP&PDP to PAP.

The upstream filtering process is shown in Fig. 4. When an upstream (from a USB device to a host) USB packet arrives, it will be filtered by the rules of Type 1 filtering policy. If there are no rules or the result of matching the rules is

permitted, it goes to the communication stage. In the enumeration process, the data of a USB packet will be extracted. This process contains many USB protocol information and technical skills. The specific process will be shown together with the upstream and downstream filtering algorithms in our low-cost simulation framework for USB packet filtering the technical report. After matching rules of Type 2 policy, the filter will forward or drop the packet according to the matching result. If the packet is in the functional communication, the rules of Type 3 and Type 4 will be matched in sequence as shown in Fig. 4.

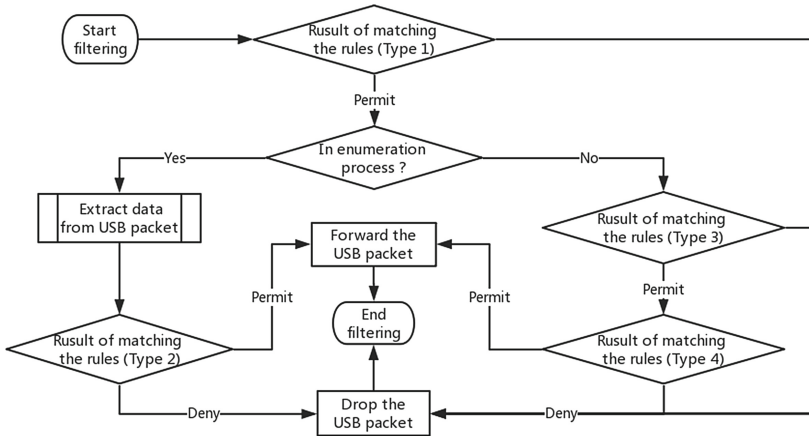


Fig. 4. Upstream filtering process

As we described in USB security issues, host broadcast downstream USB packets may be at risk of eavesdropping. To reduce this kind of risk, the downstream USB packet filtering is involved in the filtering process. In the downstream USB packet filtering, the packets from the host are filtered according to the destination. If the destination is one of the downstream USB devices of the filter, the packet will be forwarded, otherwise it will be dropped, as illustrated in Fig. 5. No additional filtering policy is required in this process.

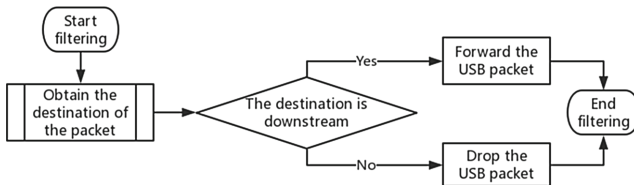


Fig. 5. Downstream filtering process

4.3 Experimental Result

We implement and deploy the USB communication simulation framework using Java on Eclipse platform with JDK 1.8 installed on the computer with Windows 10 64-bit OS, Intel Core i5-7200u cup and 8G memory.

In order to compare the efficiency of various filtering policies and the impact of increased filtering rules on filtering time, we simulate the change in filter processing time with linearly increasing filter rules for each type of filtering policies mentioned in Sect. 3.1. Different from traditional network packet-filter firewalls, the protected object of USB packet filtering is a personal computer rather than a private network area. The number of rules required for USB packet filtering is much less than the number of rules for network packet filtering. Therefore, for each type of filtering policies, we increment each group of filter rules by 5 until 25. Since the time for filtering a single packet is too short to be measured accurately, we use the Ellisys hardware to capture the packets in a period of USB communication with a USB device as the basis of the experiment. In our experiment, we choose the first 10000 packets of the communication between the host and the USB device, which contain the enumeration process and functional communication, as the object to filter. The experimental result is presented in Table 1.

Table 1. Process time for each type of filtering policy

Process Time (ms) \ Number of Rules		Number of Rules				
		5	10	15	20	25
Policy Type						
	Type 1	565	926	1230	1410	1718
	Type 2	9	20	33	45	61
	Type 3	108	175	233	281	318
	Type 4	99	171	237	290	351

The table shows the filtering process time (millisecond) of each type of filtering policies by a different number of rules. For each type of filtering policy, we can find that as the number of filtering rules increases, the process time of filtering also increases linearly. When there are fewer filtering rules, the filtering efficiency will be higher and the USB communication delay will be less.

After longitudinal comparison, the Type 1 filtering policy has the highest cost of time for filtering processing under the same conditions, because it needs to filter every packet from a USB device. Although high processing time means more communication delay, it can completely prevent USB devices from accessing hosts during unauthorised periods, thereby maximising host security. Type 2 filtering policy requires the least time because it only happens during the enumeration process. In each enumeration process, the attributes in the Type 2 policy only need to be verified once. Type 3 and Type 4 filtering policies occur during the functional communication of USB communication. Each time a USB

device sends functional data to a host, they will be executed. This is the reason why their filtering process time is close.

Through the experimental results, the efficiency of different types of filtering policy can be observed. It can help technicians to optimise filtering rules better: use more efficient filtering rules to achieve the same filtering effect.

5 Conclusion and Future Work

In this paper, we introduced the attacks that exploit the vulnerabilities of USB and presented the advantages of USB packet filtering compared to URB filtering. We then proposed the universal formalised flexible USB packet filtering policies which leave good scalability for future filtering of contents of USB functional communication packet, and a low-cost USB communication simulation framework for their early stage of conception and evaluation. The USB packet filtering approach can make up for the deficiencies of traditional anti-virus software, and give the operating system more comprehensive protection. The USB communication data can be verified before they enter the operating system to expand the scope of protection and minimise the risk of being attacked. Our simulation framework offers a handy, easy-operation means to evaluate the concept of filtering policies.

In the future, we will continue our research work in three areas. First, we will continue to improve our simulation to adapt to more complex USB communication. Besides, we will design USB packet filtering policies to filter the contents of the USB packet in the functional communication stage of different USB devices. Last but not least, after improving and evaluating the USB packet filtering policies, implementation in real communication environment will be necessary. We will try to implement our filtering on special hardware such as the Raspberry Pi.

Acknowledgements. This work was funded by the Research Pole of the “Pôle d’excellence cyber” with the support of the French Ministry of the Armed Forces and the Brittany Region.

References

1. Ellisys: Ellisys. <https://www.ellisys.com/>
2. Hirofuchi, T., Kawai, E., Fujikawa, K., Sunahara, H.: USB/IP-a peripheral bus extension for device sharing over IP network. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, pp. 42–42. USENIX Association (2005)
3. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering DAC, MAC and RBAC. In: Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 41–55. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31540-4_4
4. Li, S., Jia, X., Lv, S., Shao, Z.: Research and application of USB filter driver based on windows kernel. In: 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, IITSI, pp. 438–441. IEEE (2010)

5. Loe, E.L., Hsiao, H.C., Kim, T.H.J., Lee, S.C., Cheng, S.M.: SandUSB: an installation-free sandbox for USB peripherals. In: 2016 IEEE 3rd World Forum on Internet of Things, WF-IoT, pp. 621–626. IEEE (2016)
6. Microsoft: USB Request Blocks (URBs). <https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/communicating-with-a-usb-device>
7. Nohl, K., Lell, J.: BadUSB-on accessories that turn evil. Black Hat, USA (2014)
8. OASIS: eXtensible Access Control Markup Language (XACML) v3.0. <https://www.oasis-open.org/standards#xacmlv3.0>
9. Tanenbaum, A.S., Woodhull, A.S.: Operating Systems: Design and Implementation, vol. 2. Prentice-Hall Englewood Cliffs, Upper Saddle River (1987)
10. Tian, D.J., Scaife, N., Bates, A., Butler, K., Traynor, P.: Making USB great again with USBFILTER. In: USENIX Security Symposium (2016)
11. USB-IF: Universal Serial Bus specification 2.0. <http://www.usb.org/home>
12. Yuan, E., Tong, J.: Attributed based access control (ABAC) for web services. In: Proceedings of the 2005 IEEE International Conference on Web Services, ICWS 2005. IEEE (2005)

Short Paper Session III: Applied Cryptography



FPGA-Based Assessment of Midori and GIFT Lightweight Block Ciphers

Carlos Andres Lara-Nino^(✉) , Arturo Diaz-Perez ,
and Miguel Morales-Sandoval 

CINVESTAV Tamaulipas campus, Victoria, Tamaulipas, México
`clara@tamps.cinvestav.mx`

Abstract. Lightweight block ciphers are today of paramount importance to provide security services in constrained environments. Recent studies have questioned the security properties of PRESENT, which makes it evident the need to study alternative ciphers. In this work we provide hardware architectures for Midori and GIFT, and compare them against implementations for PRESENT and GIMLI under fair conditions. The hardware description for our designs is made publicly available.

Keywords: Midori · GIFT · PRESENT · GIMLI · FPGA

1 Introduction

For years, the lightweight block cipher PRESENT [6] has been in the spotlight as the most ideal solution for providing confidentiality under constrained environments. However, recent findings [5, 11] call into question the security properties of the scheme. It is clear that the study of alternatives which offer resilience against birthday attacks and linear or differential cryptanalysis is necessary.

In 2015, a possible NIST standard for lightweight cryptography was first mentioned. Over the course of two years NIST published a report detailing the scope and state of the art in lightweight cryptography [10] and the standardization works seem to be in progress. This hints to the fact that lightweight cryptographic primitives are key components in the development of future technologies and applications. Generating solid and reproducible implementation results and benchmarking is undoubtedly primordial for any future standards.

In this work we evaluate hardware realizations of the cryptographic algorithms Midori and GIFT, which are believed to be secure. We compare these implementations against State of the Art architectures for GIMLI and PRESENT. Our main contributions are:

1. Novel architectural designs for the Midori and GIFT block ciphers following area-reduction strategies.
2. The first implementation results for GIFT and the first area-oriented results for Midori in FPGA.

3. All the proposed designs (VHDL) are available at <https://www.tamps.cinvestav.mx/~hardware/>.

The rest of the paper is structured as follows. Section 2 describes the different architectures for the selected lightweight algorithms, which are implemented and evaluated. Section 3 describes our experimental setup. Section 4 presents our findings. Section 5 concludes this work.

2 Methods

In this section we focus on encryption functions since these can be used to encrypt and decrypt data under the CTR mode of operation [7]. We use 128-bit key sizes for all the block ciphers.

For each lightweight block cipher we study its *iterative* and *serial* architectures [8]. We define two types of serial architectures. The first type (serial-1) targets a reduction in the number of 4-bit substitution boxes (SBOX) from $n/4$ to two. The second type of architecture (serial-2) seeks to reduce not only the number of substitution boxes, but also the width of other transformations when possible.

2.1 PRESENT

The PRESENT block cipher follows a Substitution-Permutation Network (SPN) construction. It has a block size of 64-bit and supports key sizes of 80-bit and 128-bit. The specification for its encryption function is presented in [6].

We first study the basic implementation of the block cipher with IO ports of 8-bit described in [8]. That hardware realization of PRESENT requires 17 substitution boxes (SBOX), 77 XOR gates, and 192 Flip-Flops (FF). In regards to latency, 16 cycles are required to input the plaintext and the cipher key, 31 cycles to encrypt the data, and 8 cycles to produce the output. In total this sums 55 cycles.

In the serial-1 architecture for PRESENT found in [8], the the main optimization involves reducing the number of substitution boxes to two. The substitution boxes used in the key generation are also removed and the number of XOR gates is reduced. The trade-off is an increment in the number of cycles required to encrypt the data. The implementation of this design requires 2 SBOX, 21 XOR gates, and 192 FF. With this design 303 latency cycles are needed to encrypt a data block.

The serial-2 PRESENT architecture under study is the one reported in [9]. In that design the main strategy was outlined as reducing the whole datapath to 16-bit, which is a quarter of its block size. The hardware realization for this design involves the use of 6 SBOX, 21 XOR gates, and 192 FF. The total latency of the design is of 136 cycles.

2.2 Midori

Midori is a lightweight block cipher “that is optimized with respect to the energy consumed by the circuit per bit in encryption or decryption operation” [2]. This block cipher operates over data blocks of 64 or 128 bits. A key size of 128-bit is used in both versions of the algorithm. Midori also has an SPN structure.

The iterative architecture for Midori created in this work is presented in Fig. 1 (left). This design can describe both Midori-64 and Midori-128 realizations. It follows the algorithm specification closely but uses 8-bit IO ports. In hardware, this architecture requires 16 SBox (which are of 4-bit for Midori-64 and of 8-bit for Midori-128), an n -bit transformation which can be simplified as $n/2$ XOR gates (MixColumn), an n -bit XOR layer, 16 XOR gates for the key mechanism, $r - 2$ 16-bit round constants, and $n + 128$ FF. In total the iterative architectures for Midori-64 and Midori-128 have a latency of 41 and 53 cycles, respectively.

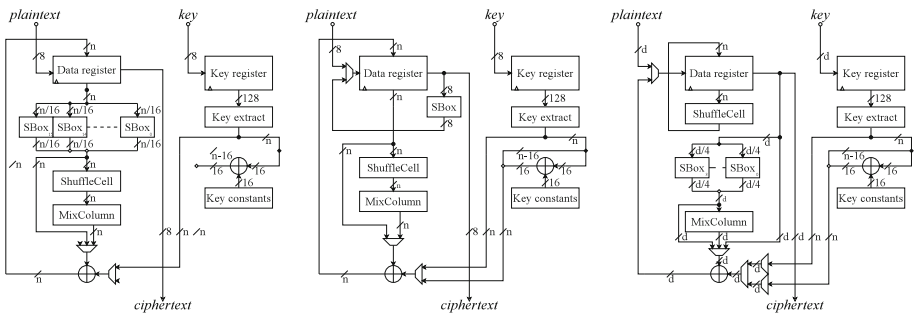


Fig. 1. Iterative (left), serial-1 (center), and serial-2 (right) architectures for Midori-64 and Midori-128

Figure 1 (center) illustrates the serial-1 architecture created in this work for Midori-64 and Midori-128. This version focuses on reducing the SBOX count. For Midori-64, the SBox illustrated represents two 4-bit SBOX. For Midori-128, eight 8-bit permutations are also allocated inside the SBox. These permutations work together with two 4-bit SBOX to produce the output of the substitution layer. Two of the four permutations are selected depending on the position in the state of the data nibble being processed. The hardware realization of this design uses two 4-bit SBOX, the $n/2$ XOR gates simplification of MixColumn, an n -bit XOR layer, the 16 XOR gates used in the key generation, $r - 2$ 16-bit round constants, and $128 + n$ FF. This Midori-64 architecture has a latency of 169 cycles while the Midori-128 design requires a total of 373 cycles.

The serial-2 architecture developed in this work for Midori is shown in Fig. 1 (right). In this design the datapath width d is reduced to 16-bit for Midori-64 and 32-bit for Midori-128. In both cases the operations which can be serialized are the substitution layer, the MixColumn step, and the key addition. The n -bit permutation is performed during an extra cycle in the round. In order

to achieve this design we modified the Midori algorithm so that the SubCell and ShuffleCell operations are swapped. This allows pushing the ShuffleCell step from the i iteration back to the $i - 1$ iteration. From this, the serializable steps of the algorithm are now grouped at the beginning of the round and can be processed together in 4 cycles. The non serializable part is left at the end of the round and performed in the extra cycle. The cost of this modification only affects Midori-128 due to the 8-bit permutations used inside the SBox which have to be shuffled. For implementing Midori-64 this design requires four 4-bit SBOX, an 8 XOR gates version of MixColumn, 32 XOR gates, 14 16-bit round constants, and 192 FF. The latency of this design amounts to 96 cycles. For implementing Midori-128 the hardware requirements are four 8-bit SBox, a 16 XOR gates version of MixColumn, 48 XOR gates, 18 16-bit round constants, and 256 FF. A latency of 112 cycles is required to encrypt the data with this architecture.

2.3 GIFT

The block cipher GIFT is said to be a direct improvement to PRESENT “that provides a much increased efficiency in all domains (smaller and faster)” and also patches security weaknesses of the latter. Two specifications of the algorithm were presented in [3] for block sizes of 64 and 128-bit. A key size of 128-bit is used in both versions of the algorithm.

The iterative architecture created for GIFT is presented in Fig. 2 (left). This design is a direct implementation of the specification with 8-bit IO ports. For GIFT-64 or GIFT-128 the design requires $n/4$ 4-bit SBOX, $n/2 + 6$ XOR gates, a NOT gate, and $n + 134$ FF. The latency for GIFT-64 is 52 cycles and the latency for GIFT-128 is 72 cycles.

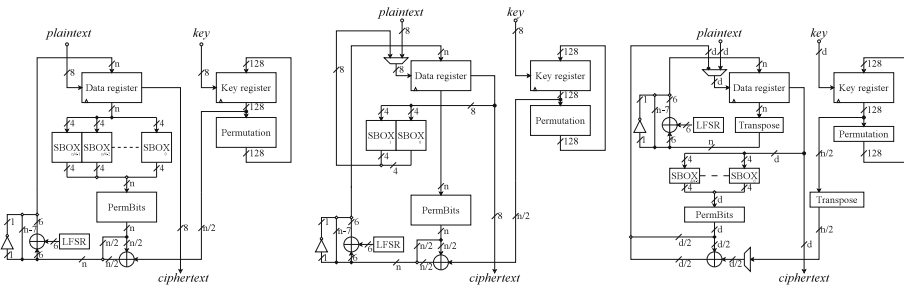


Fig. 2. Iterative (left), serial-1 (center), and serial-2 (right, the value d equals $n/4$) architectures for GIFT-64 and GIFT-128

Figure 2 (center) presents our serial-1 architecture for GIFT. This design uses 8-bit IO ports and has a serialized application of the substitution layer based on two 4-bit SBOX. The architecture illustrated describes both GIFT-64 and GIFT-128. In the case of GIFT-64 the implementation requires two 4-bit SBOX,

38 XOR gates, a NOT gate, and 198 FF. For this version 276 latency cycles are required. For GIFT-128, two 4-bit SBOX, 70 XOR gates, a NOT gate, and 262 FF are used. In this case the latency is of 712 cycles.

Our serial-2 architecture for GIFT, shown in Fig. 2 (right), was created by serializing the substitution, permutation, and key addition layers. The datapath width d was adjusted to 16-bit for GIFT-64 and to 32-bit for GIFT-128. The reduction of the substitution layer is straightforward for GIFT. We used a regular pattern found in the original permutation to reduce the permutation layer width to a quarter of its original width. However, by using this reduction an additional transposition of the state is required. Let us use a 2-D representation of the state as described in [3]. The new reduced permutation will yield a transposed version of the 2-D state, arranged in 16 $n/16$ -bit nibbles. Thus, the additional permutation is a shuffling of the state in 4-bit nibbles for GIFT-64 and 8-bit nibbles for GIFT-128. This strategy is similar to that used in [9] for PRESENT. The *small* permutation is applied on a serialized manner while the *transposition* is applied over the state during an additional cycle. The round key also needs to be shuffled to accommodate for this intermediate result. In order to serialize the key addition step, we separated the addition of the keying materials and the addition of the round constants. The keying materials are derived from the key register, shuffled, and serialized, before being applied to the state. The round constants are applied to the state during the additional cycle while the key register is updated. Based on this architecture, the implementation of GIFT-64 requires four 4-bit SBOX, 14 XOR gates, a NOT gate, and 198 FF. The implementation of GIFT-128 uses eight 4-bit SBOX, 22 XOR gates, a NOT gate, and 262 FF. The total latency for GIFT-64 and GIFT-128 is of 152 and 208 cycles, respectively.

2.4 GIMLI

GIMLI is a 384-bit permutation “designed to achieve high security with high performance across a broad range of platforms”. According to its creators, this permutation can be easily used to build high-security block ciphers. We have included this algorithm into our review since its authors claim it was designed for “energy-efficient hardware” and “compactness”. The specification for this function is presented in [4]. Since the implementations provided in [4] do not implement a block cipher, a secret key is not used.

In the iterative implementation for GIMLI provided in [4] a block size of 384-bit is used. The application of the parallel SP-box requires two 384-bit permutations, 768 XOR gates, 256 AND gates, and 128 OR gates. The Big-Swap and the Small-Swap can be seen as 384-bit permutations. Finally, 37 XOR gates are used for the addition of the round constants. This architecture has a latency of 120 cycles.

A serial-1 architecture for GIMLI was also retrieved from [4]. The main strategy for reducing resources consists on serializing the application of the SP-box layer. In this instance, 96-bit of the state are processed in parallel so that four cycles are required for each application of the SP-box layer. The other

transformations are applied to the state in a fifth cycle, which is present for half of the rounds. The application of the serialized SP-box requires two 96-bit permutations, 192 XOR gates, 64 AND gates, and 32 OR gates. The Big-Swap and the Small-Swap can still be represented as 384-bit permutations and 37 XOR gates are also used for the addition of the round constants. A latency of 204 cycles is required for this design.

2.5 Summary

Table 1 provides a summary of the different architectures discussed in this section.

Table 1. Summary of the different designs reviewed in this section

Label	Alg	State (bits)	Key (bits)	Rounds	Ref	Arch.	Latency (cycles)	Hardware resources			
								SBOX	Gates	Const.	FFs
C01	PRESENT	64	128	31	[8]	Iterative	55	17	77	1	192
C02					[8]	Serial-1	303	2	21	1	192
C03					[9]	Serial-2	136	6	21	1	192
C04	Midori-64	64	128	16	Ours	Iterative	41	16	112	14	192
C05					Ours	Serial-1	169	2	112	14	192
C06					Ours	Serial-2	96	4	40	14	192
C07	Midori-128	128	128	20	Ours	Iterative	53	32	208	18	256
C08					Ours	Serial-1	373	2	208	18	256
C09					Ours	Serial-2	112	8	64	18	256
C10	GIFT-64	64	128	28	Ours	Iterative	52	16	39	0	198
C11					Ours	Serial-1	276	2	39	0	198
C12					Ours	Serial-2	152	4	15	0	198
C13	GIFT-128	128	128	40	Ours	Iterative	72	32	71	0	262
C14					Ours	Serial-1	712	2	71	0	262
C15					Ours	Serial-2	208	8	23	0	262
C16	GIMLI	384	-	24	[4]	Iterative	120	0	1189	2	384
C17					[4]	Serial-1	204	0	325	2	384

3 Experimental Evaluation

The different designs in Table 1 are used as configurations for our experimental evaluation. The VHDL description for the PRESENT implementations is the one used in [8] and [9]. The hardware descriptions for the different Midori and GIFT architectures were created in this work. Lastly, the VHDL description for the GIMLI architectures is the one used in [4].

All the configurations were implemented for the xc6slx16-3csg324 FPGA using ISE Design Suite 14.2 and for the xc7a15t-1cpg236c FPGA using Vivado Design Suite 2017.3 Version. The synthesis process was configured with *Area* as

optimization goal in both instances. The use of RAM/ROM elements was disabled for all the implementations. We provide Post-Place & Route area results in terms of slices (SLC), Look-Up-Tables (LUT), and Flip-Flops (FF) for all the configurations in the two implementation platforms.

In regards to performance, we report the total latency (LAT), the maximum achievable frequency (Fmax) from the Post-Place & Route report, the runtime (Time), and the throughput (Thr) for each configuration. The throughput was calculated for operational frequencies of 100 KHz and Fmax as $\text{Thr} = (\text{state size} \times \text{Freq})/\text{LAT}$.

A power analysis for the xc6slx16-3csg324 FPGA was performed using the Xilinx XPower Analyzer tool version 14.2 for operational frequencies of 100 KHz and Fmax. The power estimations were obtained after place and route using Xilinx XPower Analyzer 14.3 with HIGH overall confidence level. This analysis used the Post-Place & Route Design file (ncd), a Physical Constraints file (pcf) specific for the evaluation target, and a Simulation Activity file (saif) generated from a Post-Place & Route simulation in Isim. The Simulation Run Time was of 100 ms for all the 100 KHz instances and of 100 μs for all the Fmax instances. From this evaluation we report the quiescent and dynamic power for each design. The power dissipation and the performance at 100 KHz and Fmax were then used to calculate the energy consumption for each configuration.

We use three efficiency (EFF) metrics to evaluate the different configurations. The first figure represents the relation between performance and area and is given in Kbps per SLC. The second figure represents the relation between energy and area and is given in μJ per SLC. Lastly, the third efficiency indicator represents the relation between the energy spent and the bits processed and is expressed in nJ per bit. These metrics are expected to indicate the prowess of the configurations for different trade offs, which might be attractive for different application scopes.

4 Results

The area and performance results for the implementations in the xc6slx16-3csg324 FPGA are presented in Table 2. The results for the power analysis and energy consumption calculations for the different configurations implemented in the xc6slx16-3csg324 FPGA are provided in Table 3. The area results in the xc7a15t-1cpg236c FPGA are shown in Fig. 3.

4.1 Discussion

The iterative architectures presented for Midori and GIFT offer a good balance between area and performance. While iterative implementations are generally more efficient, serial architectures can be used in cases where further area reduction is needed.

The first type of serial architectures described (S1: reduction of the SBOX count) offers a reduction in the hardware resources over the iterative architectures for all the block ciphers reviewed. But the latency is the least favorable for

Table 2. Area and performance results for the xc6slx16-3csg324 FPGA using operational frequencies of 100 KHz and Fmax.

Cipher	Ref.	Conf.	Size (bits)				Resources			LAT	Fmax	Time (μ s)		Thr (Mbps)		EFF (Kbps/SLC)	
			State	Key	IO	DP	FF	LUT	SLC	(Cycles)	(MHz)	100KHz	Fmax	100KHz	Fmax	100KHz	Fmax
PRESENT	[8]	C01	64	128	8	64	200	202	56	55	145.35	550	0.38	0.12	169.13	2.08	3020.24
		C02	64	128	8	8	203	157	45	303	131.87	3030	2.30	0.02	27.85	0.47	618.99
	[9]	C03	64	128	16	16	201	220	61	148	159.21	1480	0.93	0.04	68.85	0.71	1128.65
Midori-64	Ours	C04	64	128	8	64	200	356	118	41	166.17	410	0.25	0.16	259.38	1.32	2198.17
		C05	64	128	8	8	203	262	109	169	141.56	1690	1.19	0.04	53.61	0.35	491.83
		C06	64	128	16	16	202	268	96	96	157.80	960	0.61	0.07	105.20	0.69	1095.86
Midori-128	Ours	C07	128	128	8	128	264	549	157	53	157.95	530	0.34	0.24	381.47	1.54	2429.75
		C08	128	128	8	8	269	390	115	373	139.31	3730	2.68	0.03	47.81	0.30	415.72
		C09	128	128	32	32	267	482	155	112	86.07	1120	1.30	0.11	98.37	0.74	634.64
GIFT-64	Ours	C10	64	128	8	64	205	189	58	52	218.10	520	0.24	0.12	268.43	2.12	4628.17
		C11	64	128	8	8	209	151	44	276	225.53	2760	1.22	0.02	52.30	0.53	1188.56
		C12	64	128	16	16	208	235	67	152	219.44	1520	0.69	0.04	92.40	0.63	1379.06
GIFT-128	Ours	C13	128	128	8	128	270	290	93	72	189.93	720	0.38	0.18	337.66	1.91	3630.75
		C14	128	128	8	8	275	286	81	712	144.15	7120	4.94	0.02	25.92	0.22	319.94
		C15	128	128	32	32	273	256	66	208	217.63	2080	0.96	0.06	133.92	0.93	2029.16
GIMLI	[4]	C16	384	-	8	384	394	587	174	120	121.34	1200	0.99	0.32	388.30	1.84	2231.62
		C17	384	-	8	32	397	493	164	204	148.88	2040	1.37	0.19	280.24	1.15	1708.76

Table 3. Power and energy results for the xc6slx16-3csg324 FPGA using operational frequencies of 100 KHz and Fmax.

Cipher	Ref.	Conf.	POW@100KHz (mW)		ENE@100KHz	EFF@100KHz		POW@Fmax (mW)		ENE@Fmax	EFF@Fmax	
			Quiescent	Dynamic	(nJ)	(nJ/SLC)	(nJ/bit)	Quiescent	Dynamic	(nJ)	(nJ/SLC)	(nJ/bit)
PRESENT	[8]	C01	21.51	0.50	12105.50	216.17	189.15	21.82	31.22	20.07	0.36	0.31
		C02	21.51	0.55	66841.80	1485.37	1044.40	21.68	17.48	89.98	2.00	1.41
	[9]	C03	21.51	0.49	32560.00	533.77	508.75	21.89	37.67	55.37	0.91	0.87
Midori-64	Ours	C04	21.51	0.50	9024.10	76.48	141.00	22.00	48.36	17.36	0.15	0.27
		C05	21.51	0.48	37163.10	340.95	580.67	21.69	18.07	47.47	0.44	0.74
		C06	19.90	0.47	19555.20	203.70	305.55	20.14	24.72	27.29	0.28	0.43
Midori-128	Ours	C07	21.51	0.52	11675.90	74.37	91.22	22.28	75.15	32.69	0.21	0.26
		C08	21.51	0.49	82060.00	713.57	641.09	21.76	24.98	125.14	1.09	0.98
		C09	19.90	0.53	22881.60	147.62	178.76	20.38	48.13	89.15	0.58	0.70
GIFT-64	Ours	C10	21.51	0.49	11440.00	197.24	178.75	21.94	42.29	15.31	0.26	0.24
		C11	21.51	0.46	60637.20	1378.12	947.46	21.68	17.63	48.11	1.09	0.75
		C12	19.90	0.46	30947.20	461.90	483.55	20.10	20.99	28.46	0.42	0.44
GIFT-128	Ours	C13	21.51	0.49	15840.00	170.32	123.75	22.03	51.32	27.81	0.30	0.22
		C14	21.51	0.48	156568.80	1932.95	1223.19	21.65	14.24	177.27	2.19	1.38
		C15	19.90	0.46	42348.80	641.65	330.85	20.20	30.79	48.73	0.74	0.38
GIMLI	[4]	C16	21.51	0.58	26508.00	152.34	69.03	21.74	23.39	44.63	0.26	0.12
		C17	21.51	0.57	45043.20	274.65	117.30	21.73	21.99	59.91	0.37	0.16

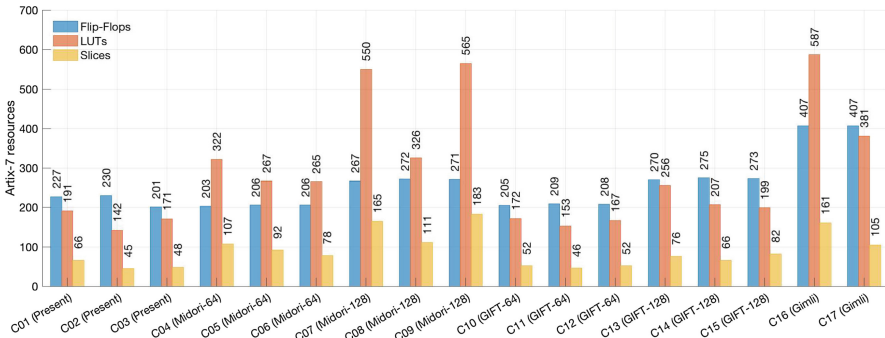


Fig. 3. Area results of lightweight block ciphers using the xc7a15t-1cpq236c FPGA. Results obtained after place and route

every instance. The second type of serial architectures (S2: general reduction of the datapath) offers better performance than the S1 type. The hardware profile seems to vary from design to design. For PRESENT, the serial-2 architecture (C03) appears to be ineffective compared to C01 in the xc6slx16-3csg324 FPGA. However, the improvement for this design (C03) is palpable when implemented on the xc7a15t-1cpg236c FPGA. Other instances where the serial-2 architecture is advantageous for area occur for Midori-64 and GIFT-128 in the xc6slx16-3csg324 FPGA and for Midori-64 in the xc7a15t-1cpg236c FPGA.

The iterative architectures consistently achieved the smaller energy consumption figures. However, the second type of serial architectures dissipated the least power for Midori and GIFT at low operational frequencies (100 KHz). While low energy consumption is a desirable trait for extending the lifetime of battery-powered applications such as WSN motes, low power dissipation is required in passive devices such as RFID tags.

Even though high operational frequencies lead to increased power dissipation, the execution times obtained from the frequency increment, and the resulting energy consumption, are greatly improved. For throughput, the variation from 100 KHz to Fmax is generally of three orders of magnitude, which coincides with the reduction of the execution time. The frequency increment causes the power dissipation to double for all the configurations, but due to the delay reduction the final energy consumption is also reduced three orders of magnitude for almost all the configurations. This experiment presents evidence that constrained devices can benefit from high operational frequencies, however, the application scope shall ultimately dictate the operational frequency to be used.

From the results it is possible to note how small IO buffers can be a burden for an implementation. It is known that most constrained devices can not afford to implement wide interfaces. But if the IO width selected is too small, the port interfacing will take longer than the data processing itself. This is more evident with primitives with large block sizes such as Midori-128, GIFT-128 and GIMLI.

The efficiency results allow drawing specific comparisons among the different configurations. From the performance per slice comparison it is possible to note that the iterative architectures (C01, C04, C07, C10, C13, C16) are consistently more efficient compared to the serial realizations. From this set, the iterative implementations of the GIFT block cipher, in both 64 (C10) and 128 bits (C13) instances, resulted to be the most efficient. The results are consistent for both operational frequencies used.

In terms of energy per slice, the minimal energy expenditure per slice is observed for the iterative realization of Midori-64 (C04) and Midori-128 (C07). The maximum energy per slice was observed for the serial architectures of GIFT (C14) and PRESENT (C02), these designs both follow the approach of reducing the number of substitution boxes in the design. In this case the behavior for both operational frequencies is similar even though the difference of three orders of magnitude is noticeable.

Both implementations for the GIMLI permutation (C16, C17) obtained the smaller expenditures in the energy per bit efficiency results. These were followed

by the iterative implementations of GIFT-128 (C13) and GIFT-64 (C10). The same pattern can be discerned for both operational frequencies used.

4.2 Comparison with the State of the Art

In the literature we found one work which implements the Midori block cipher in FPGA [1]. In that reference the authors propose fault-diagnosis schemes for Midori-128 and compare them with the “Original Midori128 Encryption” in an xc7vx330t FPGA. Results in SLC, maximum frequency, power, and throughput are provided for four Midori-128 implementations. Since a different FPGA platform is used and not all the information is available (latency, synthesis criteria) it is difficult to have a fair comparison. In regards to area, the implementations in [1] cost from 155 to 171 SLC while our designs for Midori-128 in the xc6slx16-3csg324 FPGA cost from 112 to 162 SLC. In performance, our fastest implementation of Midori-128 can reach up to 433 Mbps while the range in [1] is 42.52 to 47.41 Gbps. The power requirements for our designs range from 20.42 mW to 22.02 mW while the more modest design in [1] requires 340 mW. Its clear that our implementations were created following different design goals. While the results in [1] were obtained for improved security and high performance, our implementations seek to provide low implementation size and energy consumption.

No FPGA implementations for GIFT were found in our review.

5 Conclusions

In this paper we have studied cryptographic algorithms which can substitute the use of PRESENT and might be considered for future standardization. Even though the modern constructions are efficient, they can not improve the resource requirements of PRESENT for secure state sizes.

We have provided lightweight hardware architectures for the Midori and GIMLI block ciphers. The proposed designs exhibit varying trade-offs which can be attractive for different applications. In order to increase the usability of our work the hardware descriptions for these architectures are made public.

To the best of our knowledge, we have obtained the first FPGA results for the GIFT block cipher and the first area-optimized implementations for Midori.

Acknowledgments. This work was supported by CONACyT under grant number 336750 and CINVESTAV. This work was also funded by “Fondo Sectorial de Investigación para la Educación”, CONACyT México, through the project number 281565.

References

1. Aghaie, A., Kermani, M.M., Azarderakhsh, R.: Fault diagnosis schemes for low-energy block cipher Midori benchmarked on FPGA. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **25**(4), 1528–1536 (2017)
2. Banik, S., et al.: Midori: a block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) *ASIACRYPT 2015*. LNCS, vol. 9453, pp. 411–436. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_17
3. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: a small present. In: Fischer, W., Homma, N. (eds.) *CHES 2017*. LNCS, vol. 10529, pp. 321–345. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_16
4. Bernstein, D., et al.: GIMLI : a cross-platform permutation. In: Fischer, W., Homma, N. (eds.) *CHES 2017*. LNCS, vol. 10529, pp. 299–320. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_15
5. Bhargavan, K., Leurent, G.: On the practical (in-)security of 64-bit block ciphers: collision attacks on HTTP over TLS and OpenVPN. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 456–467. CCS 2016. ACM, New York (2016)
6. 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
7. Dworkin, M.: Recommendation for block cipher modes of operation. Technical report, NIST, Information Technology Laboratory, December 2001
8. Hanley, N., O'Neill, M.: Hardware comparison of the ISO/IEC 29192–2 block ciphers. In: 2012 IEEE Computer Society Annual Symposium on VLSI, pp. 57–62, August 2012
9. Lara-Nino, C.A., Diaz-Perez, A., Morales-Sandoval, M.: Lightweight hardware architectures for the present cipher in FPGA. *IEEE Trans. Circuits Syst. I: Regul. Pap.* **64**(9), 2544–2555 (2017)
10. McKay, K.A., Bassham, L., Turan, M.S., Mouh, N.: Report on lightweight cryptography. Technical report, NIST, Information Technology Laboratory, March 2017
11. Zhang, G., Liu, M.: A distinguisher on PRESENT-like permutations with application to SPONGENT. *Sci. China Inf. Sci.* **60**(7), 072101 (2017)



Simpler CCA Secure PKE from LPN Problem Without Double-Trapdoor

Haitao Cheng^{1,2}, Xiangxue Li^{1,3}(✉), Haifeng Qian¹, and Di Yan⁴

¹ Department of Computer Science and Technology,
East China Normal University, Shanghai, China
{xxli,hfqian}@cs.ecnu.edu.cn

² National Engineering Laboratory for Wireless Security, Xi'an University of Posts
and Telecommunications, Xi'an, China

³ Westone Cryptologic Research Center, Beijing, China

⁴ Department of Computer Science and Engineering,
Shanghai Jiaotong University, Shanghai, China

Abstract. The first CCA secure public key encryption (PKE) on the learning parity with noise (LPN) assumption was invented by Döttling et al. (ASIACRYPT 2012). At PKC 2014, Kiltz et al. gave a simpler and more efficient construction, where a double-trapdoor technique was introduced to handle the decryption queries in game simulation. Different from the technique, we build in the standard model the CCA secure PKE on a variant of Extended Knapsack LPN problem (which is provably equivalent to the standard LPN problem). We abstract out an ephemeral key from the LPN assumption, which can then be used to encrypt the underlying plaintext when equipped with several typical classes of cryptographic primitives. Thanks to these techniques, the decryption queries can be correctly answered (yet without relying on a double-trapdoor mechanism) during security reduction from LPN. The resulting simple proposal appears more modular and efficient.

Keywords: Post quantum cryptography · Low-noise LPN
Extended Knapsack LPN

1 Introduction

In cryptography and learning theory, the Learning Parity with Noise (LPN) problem has become a well-known problem. The two versions of LPN have been pointed out to be polynomially equivalent [10]. The decisional one with parameter $0 < \mu < 1/2$ (noise rate), $m = \text{poly}(n)$, $n \in \mathbb{N}$ posulates that $(\mathbf{A}, \langle \mathbf{A}, \mathbf{s} \rangle + \mathbf{e})$ is pseudorandom given \mathbf{A} (i.e., computationally indistinguishable from uniform randomness), where $\mathbf{A} \in \{0, 1\}^{m \times n}$, $\mathbf{s} \in \{0, 1\}^n$ are chosen uniformly at random, $\mathbf{e} \in \{0, 1\}^m$ is distributed to \mathcal{B}_μ^m , (i.e., concatenation of m independent copies of the Bernoulli distribution \mathcal{B}_μ such that $\Pr[\mathcal{B}_\mu = 1] = \mu$), $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors and ‘+’ denotes the XOR operation. The computational

version assumes that it is computationally infeasible to find out the random secret binary vector $\mathbf{s} \in \{0, 1\}^n$ from those noisy linear samples.

LPN Hardness. The computational LPN problem is deemed as a well-known NP-complete problem “decoding random linear codes” [2], which makes LPN be a promising candidate for post-quantum cryptography. Furthermore, the simplicity of LPN makes it more suitable for weak-power devices (e.g., RFID tags) than other post-quantum candidates such as LWE [17]. The best known algorithms for solving constant noise (noise parameter $0 < \mu < 1/2$) LPN problem require $2^{O(n/\log n)}$ time and samples [4, 12]. When given only polynomially many $\text{poly}(n)$ samples, the time complexity goes up to $2^{O(n/\log \log n)}$ [13], and even $2^{O(n)}$ when given only linearly many $O(n)$ samples [14, 19]. Under low-noise rate i.e., the noise rate $\mu = O(n^{-c})$ (typically $c = 1/2$), the best LPN solvers need only $2^{O(n^{1-c})}$ time when given $O(n)$ samples [3, 19].

1.1 Related Work

PKE with CPA security. Retrospectively, Alekhnovich [1] constructed the first CPA-secure public-key encryption scheme from low-noise LPN (i.e., noise rate $\mu = 1/\sqrt{n}$). Inspired by the schemes of Regev [17] and Gentry et al. [9], Döttling et al. proposed an alternative one [8]. The work of Yu and Zhang [20] in 2016 made a breakthrough in solving the open problem of constructing public-key primitives based on constant-noise LPN problem. In their IND-CPA scheme, they used a variant assumption called LPN on Squared-Log Entropy and gave a tight requirement of secret key’s distribution.

PKE with CCA security. IND-CCA security [16] is one of the strongest known notions of security for public-key encryption schemes. Döttling et al. [8] constructed the first CCA-secure PKE scheme from low-noise LPN by using the correlated products approach of [18]. But the complexity of that scheme was hundreds of times worse than Alekhnovich’s scheme. Kiltz et al. [11] gave a more efficient CCA-secure construction by means of the techniques from LWE-based encryption in [15] with some technical changes. Specifically, they used a double-trapdoor mechanism, together with a trapdoor switching lemma so that there is always an available trapdoor to answer the decryption queries in game simulation. In [20], Yu and Zhang constructed the first constant-noise LPN problem based CCA-secure scheme which uses a tag-based encryption technique.

1.2 Our Contributions

In this work, we propose a simple and efficient PKE scheme which is IND-CCA secure from low-noise LPN. We build a neat construction with noise rate $\mu \approx O(\sqrt{1/n})$.

With an IND-CPA secure private-key scheme and a collision resistant hash function H we plug the $H(\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}, \mathbf{H}_t)$ into $\text{Enc}'_{\mathbf{k}}(\mathbf{m})$ where $\mathbf{k} = H(\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}, \mathbf{H}_t)$ becomes a secret key of the Enc' algorithm of an IND-CPA-secure private-key

scheme Π' . Intuitively, based on the indistinguishability of LPN samples, it holds that the scheme is IND-sTag-CCA secure (see Definition 4) and can be efficiently transformed into a CCA-secure encryption scheme [5, 11, 20].

2 Preliminaries

2.1 Notations and Definitions

We use capital letters (e.g., X, Y) for random variables and distributions, standard letters (e.g., x, y) for values. Vectors are used in the column form and denoted by bold lower-case letters (e.g., \mathbf{a}). We treat matrices as the sets of its column vectors and denote them by bold capital letters (e.g., \mathbf{A}). For a binary string x , $|x|$ refers to the Hamming weight of x . We use \mathcal{B}_μ to denote the Bernoulli distribution with parameter μ , i.e., $\Pr[\mathcal{B}_\mu = 1] = \mu$, $\Pr[\mathcal{B}_\mu = 0] = 1 - \mu$, while \mathcal{B}_μ^n denotes the concatenation of n independent copies of \mathcal{B}_μ . For $n, \ell \in \mathbb{N}$, U_n (resp., $U_{\ell \times n}$) denotes the uniform distribution over $\{0, 1\}^n$ (resp., $\{0, 1\}^{\ell \times n}$) and independent of any other random variables in consideration. $X \sim D$ denotes that random variable X follows distribution D . We use $s \leftarrow S$ to denote sampling an element s according to distribution S . For random variables X and Y , the statistical distance between them is defined by $\Delta(X, Y) = \frac{1}{2} \cdot \sum_x |\Pr[X = x] - \Pr[Y = x]|$. If for probability ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$, $\Delta(X_n, Y_n) \leq \text{negl}(n)$ holds, then X and Y are called statistically indistinguishable, denoted by $X \stackrel{s}{\sim} Y$. If for any PPT distinguisher \mathcal{D} , $|\Pr[\mathcal{D}(X_n) = 1] - \Pr[\mathcal{D}(Y_n) = 1]| \leq \text{negl}(n)$ holds then X and Y are called computationally indistinguishable, denoted by $X \stackrel{c}{\sim} Y$.

Collision Resistant Hash Function. A hash function family $\mathcal{H} = \{H : \mathcal{X} \rightarrow \mathcal{Y}\}$ is collision resistant if for any PPT adversary \mathcal{A} , it satisfies that $\text{Adv}_{\mathcal{H}, \mathcal{A}}^{cr}(n) = \Pr[H \stackrel{\$}{\leftarrow} \mathcal{H}, (x, x') \stackrel{\$}{\leftarrow} \mathcal{A}(H) : H(x) = H(x') \wedge x \neq x'] \leq \text{negl}(n)$.

2.2 Learning Parity with Noise

Definition 1 (Learning Parity with Noise). The *decisional* $\text{LPN}_{n,m,\mu}$ problem is hard if for every $m = \text{poly}(n)$ we have $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \stackrel{s}{\sim} (\mathbf{A}, \mathbf{b})$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{s} \sim U_n$, $\mathbf{e} \sim \mathcal{B}_\mu^m$ and $\mathbf{b} \sim U_m$ while the secret length is n and the noise rate is $0 < \mu < 1/2$. The *computational* $\text{LPN}_{n,m,\mu}$ problem is hard if for every $m = \text{poly}(n)$ and every PPT algorithm \mathcal{D} we have $\Pr[\mathcal{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) = \mathbf{s}] = \text{negl}(n)$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{s} \sim U_n$ and $\mathbf{e} \sim \mathcal{B}_\mu^m$.

Definition 2 (Knapsack LPN-KLPN). The knapsack LPN problem is hard if for $m > n$ samples we have $(\mathbf{A}, \mathbf{A}^\top \mathbf{t}) \stackrel{s}{\sim} (\mathbf{A}, \mathbf{b})$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{t} \sim \mathcal{B}_\mu^m$, $\mathbf{b} \sim U_n$.

With a standard hybrid argument technique, we have results on the ℓ -fold LPN and ℓ -fold KLPN that $(\mathbf{A}, \mathbf{A}\mathbf{S} + \mathbf{E}) \stackrel{s}{\sim} (\mathbf{A}, \mathbf{B}_1)$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{S} \sim U_{n \times \ell}$, $\mathbf{E} \sim \mathcal{B}_\mu^{m \times \ell}$ and $\mathbf{B}_1 \sim U_{m \times \ell}$; $(\mathbf{A}, \mathbf{T}^\top \mathbf{A}) \stackrel{s}{\sim} (\mathbf{A}, \mathbf{B}_2)$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{T} \sim \mathcal{B}_\mu^{m \times \ell}$ and $\mathbf{B}_2 \sim U_{\ell \times n}$.

Definition 3 (Extended Knapsack LPN-EKLPN). *The Extended Knapsack LPN problem is hard if for $m > n$ samples we have $(\mathbf{A}, \mathbf{A}^\top \mathbf{t}, \mathbf{e}, \mathbf{t}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b}, \mathbf{e}, \mathbf{t}^\top \mathbf{e})$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{b} \sim U_n$, $\mathbf{t}, \mathbf{e} \sim \mathcal{B}_\mu^m$.*

Lemma 1. *Assume that the Extended Knapsack LPN problem is hard then we have $(\mathbf{A}, \mathbf{A}^\top \mathbf{t}, \mathbf{e}, \mathbf{t}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{A}^\top \mathbf{t}', \mathbf{e}, \mathbf{t}'^\top \mathbf{e})$.*

Proof. From Definition 3 we have $(\mathbf{A}, \mathbf{A}^\top \mathbf{t}, \mathbf{e}, \mathbf{t}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b}, \mathbf{e}, \mathbf{t}^\top \mathbf{e})$. From Definition 2 we have $(\mathbf{A}, \mathbf{A}^\top \mathbf{t}') \stackrel{c}{\sim} (\mathbf{A}, \mathbf{b})$ where $\mathbf{A} \sim U_{m \times n}$, $\mathbf{t}, \mathbf{t}', \mathbf{e} \sim \mathcal{B}_\mu^m$. By combining these two equations, we immediately obtain $(\mathbf{A}, \mathbf{A}^\top \mathbf{t}, \mathbf{e}, \mathbf{t}^\top \mathbf{e}) \stackrel{c}{\sim} (\mathbf{A}, \mathbf{A}^\top \mathbf{t}', \mathbf{e}, \mathbf{t}'^\top \mathbf{e})$.

The Extended Knapsack LPN to standard LPN problem reduction can be referenced to [7].

3 CCA Secure PKE from Low-Noise LPN

In this section, we construct a CCA-secure PKE from low-noise LPN problem. Technically, we construct a tag-based PKE against selective tag and chosen ciphertext attacks from LPN, which can be transformed into a standard CCA-secure PKE by using known techniques [5, 11, 20].

3.1 Tag-Based Encryption

A tag-based encryption (TBE) scheme with tag-space \mathcal{T} and message-space \mathcal{M} consists of three PPT algorithms $\mathcal{TB}\mathcal{E} = (\text{KeyGen}, \text{Enc}, \text{Dec})$. The randomized key generation algorithm KeyGen takes the security parameter n as input, outputs a public key pk and a secret key sk , denoted as $(pk, sk) \leftarrow \text{KeyGen}(1^n)$. The randomized encryption algorithm Enc takes pk , a tag $\mathbf{t} \in \mathcal{T}$, and a plaintext $\mathbf{m} \in \mathcal{M}$ as input, outputs a ciphertext C , denoted as $C \leftarrow \text{Enc}(pk, \mathbf{t}, \mathbf{m})$. The deterministic algorithm Dec takes sk and C as inputs, outputs a plaintext \mathbf{m} , or a special symbol \perp , which is denoted as $\mathbf{m} \leftarrow \text{Dec}(sk, \mathbf{t}, C)$. For correctness, we require that for all $(pk, sk) \leftarrow \text{KeyGen}(1^n)$, any tag \mathbf{t} , any plaintext \mathbf{m} and any $C \leftarrow \text{Enc}(pk, \mathbf{t}, \mathbf{m})$, the equation $\text{Dec}(sk, \mathbf{t}, C) = \mathbf{m}$ holds with overwhelming probability.

We consider the following game between a challenger \mathcal{C} and an adversary \mathcal{A} .

Init. The adversary \mathcal{A} takes the security parameter n as input, and outputs a target \mathbf{t}^* to the challenger \mathcal{C} .

KeyGen. The challenger \mathcal{C} computes $(pk, sk) \leftarrow \text{KeyGen}(1^n)$, gives the public key pk to the adversary \mathcal{A} , and keeps the secret key sk .

Phase 1. The adversary \mathcal{A} can make decryption queries polynomial times for any pair (\mathbf{t}, C) , with a restriction that $\mathbf{t} \neq \mathbf{t}^*$, and the challenger \mathcal{C} returns $\mathbf{m} \leftarrow \text{Dec}(sk, \mathbf{t}, C)$ to \mathcal{A} accordingly.

Challenge. The adversary \mathcal{A} outputs two equal length plaintexts $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$. The challenger \mathcal{C} randomly chooses a bit $b^* \stackrel{\$}{\leftarrow} \{0, 1\}$, and returns the challenge ciphertext $C^* \leftarrow \text{Enc}(pk, \mathbf{t}^*, \mathbf{m}_{b^*})$ to the adversary \mathcal{A} .

Phase 2. The adversary can make more decryption queries as in Phase 1.

Guess. Finally, \mathcal{A} outputs a guess $b \in \{0, 1\}$. If $b = b^*$, the challenger \mathcal{C} outputs 1, else outputs 0.

Advantage. \mathcal{A} 's advantage is defined as $\text{Adv}_{\mathcal{TBE}, \mathcal{A}}^{\text{ind-stag-cca}}(1^n) \stackrel{\text{def}}{=} |\Pr[b = b^*] - \frac{1}{2}|$.

Definition 4 (IND-sTag-CCA.) We say that a TBE scheme \mathcal{TBE} is IND-sTag-CCA secure if for any PPT adversary \mathcal{A} , its advantage is negligible in n .

3.2 The Construction

Our TBE scheme \mathcal{TBE} is constructed by using the following parameters and building blocks. Let k be the security parameter, $n = \Theta(k^2)$, $m \in \mathbb{Z}$ such that $m \geq 2n$. A constant $0 < c < \frac{1}{6}$ (recall that we set $6c < \alpha < 1$) defining: The Bernoulli parameter $\mu = \sqrt{c/m}$ and the bounding parameter $\beta = 2\sqrt{cm}$ to check consistency during decryption. A generator matrix $\mathbf{G} \in \mathbb{Z}_2^{m \times n}$ of a binary linear error-correcting code $\mathcal{C} = \mathcal{C}(\mathbf{G})$ and has efficient decode algorithm $\text{Decode}_{\mathbf{G}}$ correcting up to αm errors (we refer to [11] for details about error-correcting code). Let the tag-space $\mathcal{T} = \mathbb{F}_{2^n}$. We use a matrix representation $\mathbf{H}_{\mathbf{t}} \in \{0, 1\}^{n \times n}$ for finite field elements $\mathbf{t} \in \mathbb{F}_{2^n}$ [5, 6, 11] such that $\mathbf{H}_{\mathbf{0}} = \mathbf{0}$, $\mathbf{H}_{\mathbf{t}}$ is invertible for any $\mathbf{t} \neq \mathbf{0}$, and $\mathbf{H}_{\mathbf{t}_1} + \mathbf{H}_{\mathbf{t}_2} = \mathbf{H}_{\mathbf{t}_1 + \mathbf{t}_2}$. A family of collision resistant hash functions $\mathcal{H} := \{\mathbf{H} : \mathbb{Z}_2^m \times \mathbb{Z}_2^m \times \mathbb{Z}_2^n \times \mathbb{Z}_2^{n \times n} \rightarrow \mathbb{Z}_2^\ell\}$. Let $\Pi' = (\text{Enc}', \text{Dec}')$ be a private-key encryption scheme for messages $\mathbf{m} \in \{0, 1\}^{\ell'}$ ($\ell' \ll n$, say $\ell' = 128$ typically). We present the construction of $\mathcal{TBE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ with message space $\{0, 1\}^{\ell'}$ in Fig. 1.

$(pk, sk) \stackrel{\$}{\leftarrow} \text{KeyGen}(1^k) :$ $\mathbf{A} \stackrel{\$}{\leftarrow} U_{m \times n}.$ $\mathbf{T} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^{m \times m}.$ $\mathbf{B} := \mathbf{T}\mathbf{A}.$ Return $pk := (\mathbf{A}, \mathbf{B}),$ $sk := \mathbf{T}.$	$c \stackrel{\$}{\leftarrow} \text{Enc}(pk, \mathbf{t}, \mathbf{m}) : // \mathbf{m} \in \{0, 1\}^{\ell'}, \mathbf{t} \in \mathbb{F}_{2^n}$ Parse $pk = (\mathbf{A}, \mathbf{B}).$ $\mathbf{s} \stackrel{\$}{\leftarrow} U_n.$ $\mathbf{e}_1 \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^m.$ $\mathbf{T}' \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^{m \times m}.$ $\mathbf{c}_1 := \mathbf{A}\mathbf{s} + \mathbf{e}_1.$ $\mathbf{c}_2 := (\mathbf{G}\mathbf{H}_{\mathbf{t}} + \mathbf{B})\mathbf{s} + \mathbf{T}'\mathbf{e}_1.$ $\mathbf{k} = \mathbf{H}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}, \mathbf{H}_{\mathbf{t}}).$ $\mathbf{c}_3 := \text{Enc}'_{\mathbf{k}}(\mathbf{m}).$ Return $c := (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3).$	$\mathbf{m} \leftarrow \text{Dec}(sk, \mathbf{t}, c) :$ Parse $sk = \mathbf{T}.$ Parse $c := (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3).$ $\mathbf{y} := \mathbf{c}_2 - \mathbf{T}\mathbf{c}_1.$ $\mathbf{H}_{\mathbf{t}}\mathbf{s} = \mathbf{b} := \text{Decode}_{\mathbf{G}}(\mathbf{y}).$ Compute $\mathbf{s} = \mathbf{H}_{\mathbf{t}}^{-1}\mathbf{b}$, and check whether $ \underbrace{\mathbf{c}_1 - \mathbf{A}\mathbf{s}}_{\mathbf{e}_1} \leq \beta \wedge \underbrace{\mathbf{c}_2 - (\mathbf{G}\mathbf{H}_{\mathbf{t}} + \mathbf{B})\mathbf{s}}_{\mathbf{T}'\mathbf{e}_1} \leq \frac{\alpha m}{3}.$ If yes, compute $\mathbf{k} = \mathbf{H}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}, \mathbf{H}_{\mathbf{t}}),$ $\mathbf{m} = \text{Dec}'_{\mathbf{k}}(\mathbf{c}_3)$, otherwise let $\mathbf{m} = \perp.$ Return $\mathbf{m}.$
---	---	--

Fig. 1. IND-sTag-CCA secure \mathcal{TBE} from low-noise LPN

3.3 Correctness

Lemma 2 (Chernoff Bound [11, 20]). *For any $0 < \mu < 1$ and any $\delta > 0$, we have $\Pr[|\mathcal{B}_\mu^m| > (1 + \delta)\mu m] < e^{-\frac{\min(\delta, \delta^2)}{3}\mu m}$, in particular, for $\delta = 1$ $\Pr[|\mathcal{B}_\mu^m| > 2\mu m] < e^{-\mu m/3}$.*

Obviously, for the chosen $\mathbf{e}_1 \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^m$, the Chernoff Bound yields: $\Pr[|\mathbf{e}_1| > \underbrace{\beta}_{=2\mu m}] < e^{-\mu m/3} = 2^{-\Theta(\sqrt{m})}$.

Theorem 1 (Correctness). *Let parameters be chosen as in our construction then with overwhelming probability over the choice of the public and secret keys and for all $\mathbf{m} \in \{0, 1\}^{\ell'}$, $\text{Dec}(sk, c)$ outputs \mathbf{m} correctly over $c \leftarrow \text{Enc}(pk, \mathbf{m})$.*

Proof. The scheme’s correctness requires the following:

1. $|(\mathbf{T}' - \mathbf{T})\mathbf{e}_1| \leq \alpha m$ (to let $\text{Decode}_{\mathbf{G}}$ reconstruct \mathbf{s} from $\mathbf{y} = \mathbf{c}_2 - \mathbf{T}\mathbf{c}_1$).
2. $|\mathbf{c}_1 - \mathbf{A}\mathbf{s}| \leq \beta \wedge |\mathbf{c}_2 - (\mathbf{G}\mathbf{H}_t + \mathbf{B})\mathbf{s}| \leq \frac{\alpha m}{3}$.

For the decryption algorithm we require that the Hamming weight of the inner-product of a matrix $\mathbf{T} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^{m \times m}$ and a vector $\mathbf{e}_1 \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^m$ is upper bounded by $\frac{1}{3}\alpha m$ with overwhelming probability. We firstly analyze the inner-product of a vector $\mathbf{t} \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^m$ and the vector $\mathbf{e}_1 \stackrel{\$}{\leftarrow} \mathcal{B}_\mu^m$ whose Hamming weight is at most β described as above. Since $|\mathbf{e}_1| \leq \beta$, a necessary condition for $\mathbf{t}^\top \mathbf{e}_1 = 1$ is that $\mathbf{t}[i] = 1$ for at least one of the i ’s where $\mathbf{e}_1[i] = 1$. By a simple XOR-Lemma, it holds that $\mu' = \Pr[\mathbf{t}^\top \mathbf{e}_1 = 1] \leq \beta \mu = 2c$.

By the Chernoff Bound (1) and with $\delta = \alpha/(3\mu') - 1$ (where $\mu' \leq 2c < \alpha/3$) $\Pr [|\mathbf{T}\mathbf{e}_1| > \frac{1}{3}\alpha m] = \Pr [|\mathbf{T}\mathbf{e}_1| > (1 + \delta)\mu' m] < e^{-\frac{\min(\delta, \delta^2)}{3}\mu' m}$.

Since $\delta\mu' = \alpha/3 - \mu' \geq \alpha/3 - 2c > 0$ and $\delta = \alpha/(3\mu') - 1 \geq \alpha/(6c) - 1 > 0$ are lower bounded by constants and therefore $\Pr [|\mathbf{T}\mathbf{e}_1| > \frac{1}{3}\alpha m] < e^{-\frac{\min(\delta, \delta^2)}{3}\mu' m} = 2^{-\Theta(m)}$.

Finally, in the ciphertext of our construction we have $|\mathbf{c}_1 - \mathbf{A}\mathbf{s}| = |\mathbf{e}_1| \leq \beta \wedge |\mathbf{c}_2 - (\mathbf{G}\mathbf{H}_t + \mathbf{B})\mathbf{s}| = |\mathbf{T}'\mathbf{e}_1| \leq \frac{1}{3}\alpha m$ holds with overwhelming probability $1 - 2^{-\Theta(\sqrt{m})}$. In the decryption operation, $\mathbf{y} = \mathbf{c}_2 - \mathbf{T} \cdot \mathbf{c}_1 = (\mathbf{G}\mathbf{H}_t + \mathbf{B}) \cdot \mathbf{s} + \mathbf{T}'\mathbf{e}_1 - \mathbf{T}(\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) = \mathbf{G}\mathbf{H}_t \cdot \mathbf{s} + (\mathbf{T}' - \mathbf{T}) \cdot \mathbf{e}_1$ it is sufficient to bound the error item $|(\mathbf{T}' - \mathbf{T})\mathbf{e}_1|$. It holds that $|(\mathbf{T}' - \mathbf{T})\mathbf{e}_1| \leq |\mathbf{T}'\mathbf{e}_1| + |\mathbf{T}\mathbf{e}_1| \leq \frac{2}{3}\alpha m < \alpha m$. Therefore, the decoding-procedure $\text{Decode}_{\mathbf{G}}$ will successfully recover \mathbf{s} .

In all, the message \mathbf{m} can be decrypted with overwhelming probability. \square

3.4 Security

Theorem 2. *Assume that the LPN problem is hard, \mathbf{H} is a collision resistant hash function and Π' is an IND-CPA-secure private-key encryption scheme then our TBE scheme TBE in Fig. 1. is IND-sTag-CCA secure.*

Proof. Let \mathcal{A} be any PPT adversary that can attack our scheme $\mathcal{TB}\mathcal{E}$ with advantage ε . We show that ε must be negligible in n . We continue the proof by using a sequence of games, where the first game is the real game, while the last is a random game in which the challenge ciphertext contains one component from an IND-CPA secure private-key encryption. Thus if \mathcal{A} can win in the last game he breaks the IND-CPA secure private-key encryption as well which violates the assumption. The security of $\mathcal{TB}\mathcal{E}$ can be established by showing that \mathcal{A} 's advantage in any two consecutive games are negligibly close.

Game 1. This is the IND-sTag-CCA experiment. The challenger \mathcal{C} honestly runs the adversary \mathcal{A} with the security parameter k and obtains a target tag \mathbf{t}^* from \mathcal{A} . Then, it simulates the IND-sTag-CCA security game for \mathcal{A} as follows:

KeyGen. First uniformly choose a collision resistant hash function $H \xleftarrow{\$} \mathcal{H}$ and matrices $\mathbf{A} \xleftarrow{\$} U_{m \times n}$, $\mathbf{T} \xleftarrow{\$} \mathcal{B}_\mu^{m \times m}$. Then, compute $\mathbf{B} = \mathbf{TA} \in \{0, 1\}^{m \times n}$.

Finally, \mathcal{C} sends $pk = (\mathbf{A}, \mathbf{B})$ to the adversary \mathcal{A} , and keeps $sk = \mathbf{T}$ to itself.

Phase 1. While receiving a decryption query $c = (\mathbf{t}, (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3))$ from adversary \mathcal{A} , the challenger \mathcal{C} directly returns \perp if $\mathbf{t} = \mathbf{t}^*$. Otherwise it first computes $\mathbf{y} = \mathbf{c}_2 - \mathbf{T} \cdot \mathbf{c}_1 = (\mathbf{GH}_t + \mathbf{B}) \cdot \mathbf{s} + \mathbf{T}'\mathbf{e}_1 - \mathbf{T}(\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) = \mathbf{GH}_t \cdot \mathbf{s} + (\mathbf{T}' - \mathbf{T})\mathbf{e}_1$. Then the challenger reconstructs $\mathbf{b} = \mathbf{H}_t \mathbf{s}$ from the error $(\mathbf{T}' - \mathbf{T})\mathbf{e}_1$ by using the error correction property of \mathbf{G} and computes $\mathbf{s} = \mathbf{H}_t^{-1} \mathbf{b}$. Then the challenger \mathcal{C} checks that whether it satisfies that $|\mathbf{c}_1 - \mathbf{As}| \leq \beta \wedge |\mathbf{c}_2 - (\mathbf{GH}_t + \mathbf{B})\mathbf{s}| \leq \frac{1}{3}\alpha m$. If yes it computes $\mathbf{k} = H(\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}, \mathbf{H}_t)$, $\mathbf{m} = \text{Dec}'_{\mathbf{k}}(\mathbf{c}_3)$ otherwise lets $\mathbf{m} = \perp$. Finally it returns \mathbf{m} to \mathcal{A} .

Challenge. After receiving two equal length plaintexts $\mathbf{m}_0, \mathbf{m}_1 \in \{0, 1\}^{\ell'}$ from the adversary \mathcal{A} , the challenger \mathcal{C} first randomly chooses a bit $b^* \xleftarrow{\$} \{0, 1\}$, and $\mathbf{s} \xleftarrow{\$} U_n, \mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_\mu^m, \mathbf{T}' \xleftarrow{\$} \mathcal{B}_\mu^{m \times m}$. Then, it calculates $\mathbf{c}_1^* := \mathbf{As} + \mathbf{e}_1 \in \{0, 1\}^m, \mathbf{c}_2^* := (\mathbf{GH}_{t^*} + \mathbf{B})\mathbf{s} + \mathbf{T}'\mathbf{e}_1 \in \{0, 1\}^m, \mathbf{k} = H(\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{s}, \mathbf{H}_{t^*}) \in \{0, 1\}^\ell, \mathbf{c}_3^* := \text{Enc}'_{\mathbf{k}}(\mathbf{m}_{b^*}) \in \{0, 1\}^{\ell'}$, and returns the challenge ciphertext $(\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{c}_3^*)$ to the adversary \mathcal{A} .

Phase 2. The adversary can make more decryption queries and the challenger \mathcal{C} responds to \mathcal{A} as in Phase 1.

Guess. Finally, \mathcal{A} outputs a guess $b \in \{0, 1\}$. If $b = b^*$, the challenger \mathcal{C} outputs 1, else outputs 0.

Let W_i be the event that \mathcal{C} outputs 1 in Game i for i in $\{1, 2, 3\}$.

Game 2. This Game is identical to Game 1 except that the challenge phase is changed as follows:

Challenge. After receiving two equal length plaintexts $\mathbf{m}_0, \mathbf{m}_1 \in \{0, 1\}^{\ell'}$ from the adversary \mathcal{A} , the challenger \mathcal{C} first randomly chooses a bit $b^* \xleftarrow{\$} \{0, 1\}$, and $\mathbf{s} \xleftarrow{\$} U_n, \mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_\mu^m$. Then, it calculates $\mathbf{c}_1^* := \mathbf{As} + \mathbf{e}_1 \in \{0, 1\}^m, \mathbf{c}_2^* := (\mathbf{GH}_{t^*} + \mathbf{B})\mathbf{s} + \mathbf{T}\mathbf{e}_1 \in \{0, 1\}^m, \mathbf{k} = H(\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{s}, \mathbf{H}_{t^*}) \in \{0, 1\}^\ell, \mathbf{c}_3^* := \text{Enc}'_{\mathbf{k}}(\mathbf{m}_{b^*}) \in \{0, 1\}^{\ell'}$, and returns the challenge ciphertext $(\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{c}_3^*)$ to the adversary \mathcal{A} .

Lemma 3. $|\Pr[W_1] - \Pr[W_2]| \leq \text{negl}(n)$

Proof. The only difference between Game 1 and Game 2 is that \mathcal{C} replaces $\mathbf{c}_2^* := (\mathbf{G}\mathbf{H}_{\mathbf{t}^*} + \mathbf{B})\mathbf{s} + \mathbf{T}'\mathbf{e}_1$ in Game 1 with $\mathbf{c}_2^* := (\mathbf{G}\mathbf{H}_{\mathbf{t}^*} + \mathbf{B})\mathbf{s} + \mathbf{T}\mathbf{e}_1$ in Game 2. Next, we introduce a sequence of games $\{\text{Game}_{1,i}\}_{i \in [0,m]}$ between Game 1 and Game 2 to replace \mathbf{T}' in the \mathbf{c}_2^* row by row. Firstly, we define $\mathbf{T} = (\mathbf{t}_1, \dots, \mathbf{t}_m)^\top$, $\mathbf{T}' = (\mathbf{t}'_1, \dots, \mathbf{t}'_m)^\top$.

- $\text{Game}_{1,i}$, $i \in [m]$. This game is a hybrid of Game 1 and Game 2: the challenger \mathcal{C} replaces \mathbf{t}'_i with \mathbf{t}_i in \mathbf{c}_2^* during the challenge phase and keeps the remaining rows as in $\text{Game}_{1,i-1}$. Let $\text{Game}_{1,0}$ be Game 1. Obviously, $\text{Game}_{1,m}$ is identical to Game 2.

It suffices to show that $|\Pr[W_{1,i}] - \Pr[W_{1,i-1}]| \leq \text{negl}(n)$ for any $i \in [m]$. The hardness of the EKLPN problem ensures that the probability for adversary \mathcal{A} to distinguish $\text{Game}_{1,i}$ from $\text{Game}_{1,i-1}$ is negligible. Otherwise we can construct an algorithm \mathcal{B} to solve EKLPN problem. Precisely, \mathcal{B} is constructed by simulating $\text{Game}_{1,i}$ or $\text{Game}_{1,i-1}$ for \mathcal{A} . \mathcal{B} is given a quadruple $(\mathbf{A}, (\bar{\mathbf{t}}_i^\top \mathbf{A})^\top, \mathbf{e}_1, \bar{z}_i)$, where \bar{z}_i is either $\bar{\mathbf{t}}_i^\top \mathbf{e}_1$ or $\bar{\mathbf{t}}'_i \mathbf{e}_1$. \mathcal{B} 's behavior is as follows.

KEYGEN. \mathcal{B} picks $\mathbf{H} \xleftarrow{\$} \mathcal{H}$, $\mathbf{T}_i = (\mathbf{t}_1, \dots, \mathbf{r}_i, \dots, \mathbf{t}_m)^\top$ and then \mathcal{B} sets $\mathbf{B} = (\mathbf{A}^\top \mathbf{t}_1, \dots, \mathbf{A}^\top \bar{\mathbf{t}}_i, \dots, \mathbf{A}^\top \mathbf{t}_m)^\top$. Finally, \mathcal{B} sends $pk = (\mathbf{A}, \mathbf{B})$ to the adversary \mathcal{A} , and keeps $sk = \mathbf{T}_i$ to itself. Note that the i^{th} row in \mathbf{T}_i is chosen randomly and the i^{th} row in \mathbf{B} is independent of it.

PHASE 1. While receiving a decryption query $c = (\mathbf{t}, (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3))$ from adversary \mathcal{A} , \mathcal{B} directly returns \perp if $\mathbf{t} = \mathbf{t}^*$. Otherwise it first computes $\mathbf{y} = \mathbf{c}_2 - \mathbf{T}_i \cdot \mathbf{c}_1 = (\mathbf{G}\mathbf{H}_{\mathbf{t}} + \mathbf{B}) \cdot \mathbf{s} + \mathbf{T}'\mathbf{e}_1 - \mathbf{T}_i(\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) = \mathbf{G}\mathbf{H}_{\mathbf{t}} \cdot \mathbf{s} +$

$$\underbrace{\begin{pmatrix} 0 \\ \vdots \\ (\bar{\mathbf{t}}_i^\top - \mathbf{r}_i^\top)\mathbf{A}\mathbf{s} \\ \vdots \\ 0 \end{pmatrix}}_{\Delta_i} + \begin{pmatrix} (\mathbf{t}'_1 - \mathbf{t}_1)\mathbf{e}_1 \\ \vdots \\ (\mathbf{t}'_i - \mathbf{r}_i)\mathbf{e}_1 \\ \vdots \\ (\mathbf{t}'_m - \mathbf{t}_m)\mathbf{e}_1 \end{pmatrix}, \mathbf{H}_{\mathbf{t}}\mathbf{s} = \text{Decode}(\mathbf{y}).$$

Let $\mathbf{y} = \mathbf{G}\mathbf{H}_{\mathbf{t}}\mathbf{s} + \Delta_i$, where $|\Delta_i| \leq \frac{2}{3}\alpha m + 1 < \alpha m$, $\text{Decode}_{\mathbf{G}}$ also can handle correct \mathbf{s} from \mathbf{y} . Then \mathcal{B} checks that whether it satisfies that $|\mathbf{c}_1 - \mathbf{A}\mathbf{s}| \leq \beta \wedge |\mathbf{c}_2 - (\mathbf{G}\mathbf{H}_{\mathbf{t}} + \mathbf{B})\mathbf{s}| \leq \frac{1}{3}\alpha m$. If yes it computes $\mathbf{k} = \mathbf{H}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}, \mathbf{H}_{\mathbf{t}})$, $\mathbf{m} = \text{Dec}'_{\mathbf{k}}(\mathbf{c}_3)$ otherwise lets $\mathbf{m} = \perp$. Finally it returns \mathbf{m} to \mathcal{A} . Therefore, the decryption oracle can behave correctly.

CHALLENGE. After receiving two equal length plaintexts $\mathbf{m}_0, \mathbf{m}_1 \in \{0, 1\}^{\ell'}$ from the adversary \mathcal{A} , \mathcal{B} first randomly chooses a bit $b^* \xleftarrow{\$} \{0, 1\}$, and $\mathbf{s} \xleftarrow{\$} U_n, \mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_{\mu}^m$. Then, it calculates $\mathbf{c}_1^* := \mathbf{A}\mathbf{s} + \mathbf{e}_1 \in \{0, 1\}^m, \mathbf{c}_2^* = (\mathbf{G}\mathbf{H}_{\mathbf{t}^*} + \mathbf{B})\mathbf{s} + (\mathbf{e}_1^\top \mathbf{t}_1, \dots, \mathbf{e}_1^\top \mathbf{t}_{i-1}, \bar{z}_i, \mathbf{e}_1^\top \mathbf{t}'_{i+1}, \dots, \mathbf{e}_1^\top \mathbf{t}'_m)^\top \in \{0, 1\}^m, \mathbf{k} = \mathbf{H}(\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{s}, \mathbf{H}_{\mathbf{t}^*}) \in \{0, 1\}^{\ell}, \mathbf{c}_3^* := \text{Enc}'_{\mathbf{k}}(\mathbf{m}_{b^*}) \in \{0, 1\}^{\ell'}$, and returns the challenge ciphertext $(\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{c}_3^*)$ to the adversary \mathcal{A} .

PHASE 2. The adversary can make more decryption queries and \mathcal{B} responds to \mathcal{A} as in Phase 1.

GUESS. Finally, \mathcal{A} outputs a guess $b \in \{0, 1\}$. If $b = b^*$, \mathcal{B} outputs 1, else outputs 0.

If $\bar{z}_i = \bar{\mathbf{t}}_i^{\top} \mathbf{e}_1$, then \mathcal{B} simulates the behavior of the challenger in $\text{Game}_{1,i-1}$ exactly. Hence, $\Pr[W_{1,i-1}] = \Pr[\mathcal{B}(\mathbf{A}, (\bar{\mathbf{t}}_i^{\top} \mathbf{A})^{\top}, \mathbf{e}_1, \bar{\mathbf{t}}_i^{\top} \mathbf{e}_1) = 1]$.

If $\bar{z}_i = \bar{\mathbf{t}}_i^{\top} \mathbf{e}_1$, then \mathcal{B} simulates the behavior of the challenger in $\text{Game}_{1,i}$ exactly. Hence, $\Pr[W_{1,i-1}] = \Pr[\mathcal{B}(\mathbf{A}, (\bar{\mathbf{t}}_i^{\top} \mathbf{A})^{\top}, \mathbf{e}_1, \bar{\mathbf{t}}_i^{\top} \mathbf{e}_1) = 1]$.

Therefore, for $i \in [m]$, we have $|\Pr[W_{1,i-1}] - \Pr[W_{1,i}]| \leq \text{negl}(n)$.

Game 3. This Game is identical to Game 2 except that the challenger \mathcal{C} replaces $\mathbf{B} = \mathbf{TA}$ with $\mathbf{B}' = \mathbf{B} - \mathbf{GH}_{\mathbf{t}^*} \in \{0, 1\}^{m \times n}$ in the key generation phase.

Lemma 4. $\Pr[W_3] = \Pr[W_2]$.

Proof. The only difference between Game 2 and Game 3 is that \mathcal{C} replaces $\mathbf{B} = \mathbf{TA}$ in Game 2 with $\mathbf{B}' = \mathbf{B} - \mathbf{GH}_{\mathbf{t}^*}$ in Game 3. This means that the public key in Game 3 has the same distribution in Game 2. Thus we have $\Pr[W_3] = \Pr[W_2]$.

Game 4. This Game is identical to Game 3 except that the challenger \mathcal{C} replaces $\mathbf{c}_1^* = \mathbf{As} + \mathbf{e}_1 \in \{0, 1\}^m$ with $\mathbf{c}_1^* = \mathbf{u} \in \{0, 1\}^m$ in the challenge phase. Note that in Game 2, $\mathbf{c}_2^* = (\mathbf{GH}_{\mathbf{t}^*} + \mathbf{B})\mathbf{s} + \mathbf{Te}_1 = \mathbf{GH}_{\mathbf{t}^*}\mathbf{s} + \mathbf{Tc}_1^*$. Therefore, in Game 3 we have $\mathbf{c}_2^* = (\mathbf{GH}_{\mathbf{t}^*} + \mathbf{B}')\mathbf{s} + \mathbf{Te}_1 = \mathbf{Tc}_1^*$.

Lemma 5. $|\Pr[W_4] - \Pr[W_3]| \leq \text{negl}(n)$.

Proof. Since the only difference between Game 3 and Game 4 is that \mathcal{C} replaces $\mathbf{c}_1^* = \mathbf{As} + \mathbf{e}_1 \in \{0, 1\}^m$ in Game 3 with $\mathbf{c}_1^* = \mathbf{u} \in \{0, 1\}^m$ in Game 4, we can construct a distinguisher \mathcal{D} that distinguishes the distributions $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ and (\mathbf{A}, \mathbf{u}) (where $\mathbf{u} \stackrel{\$}{\leftarrow} U_m$) with advantage $\text{adv}(n)$ (assuming that \mathcal{A} distinguishes 3 and Game 4 with non-negligible $\text{adv}(n)$), contradicting the assumption. Thus we have $|\Pr[\mathcal{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})] - \Pr[\mathcal{D}(\mathbf{A}, \mathbf{u})]| = |\Pr[W_3] - \Pr[W_4]| = \text{adv}(n)$, which contradicts the assumption. This means that we have $|\Pr[W_3] - \Pr[W_4]| \leq \text{negl}(n)$.

Lemma 6. $\Pr[W_4] = \frac{1}{2} + \text{negl}(n)$.

Proof. This lemma follows from that the challenge ciphertext $(\mathbf{c}_1^*, \mathbf{c}_2^*)$ in game 4 is uniformly distributed. From \mathcal{A} 's view, \mathbf{s} is perfectly hidden since \mathbf{c}_1^* is uniformly distributed. The collision resistant hash function implies that it's nearly impossible for \mathcal{A} to guess \mathbf{k} correctly. Combining with the IND-CPA secure private-key encryption scheme it ensures that the advantage of the adversary \mathcal{A} is negligible.

Note that the security requirement of private-key encryption scheme Π' is IND-CPA secure, for example an one-time pad scheme, since the replacement of

the pseudorandomness with randomness makes the challenge ciphertext perfectly random thus it is impossible for adversary to guess correctly with probability more than $1/2$. Meanwhile it answers the decryption queries correctly. In all, we have $\Pr[W_1] = \frac{1}{2} + \text{negl}(n)$, such that $\varepsilon = \text{negl}(n)$. Thus we complete the proof.

Acknowledgement. The work was supported by the National Cryptography Development Fund (Grant No. MMJJ20180106).

References

1. Alekhnovich, M.: More on average case vs approximation complexity. In: 44th Annual Symposium on Foundations of Computer Science, pp. 298–307. IEEE, Cambridge, October 2003
2. Berlekamp, E., McEliece, R.J., van Tilborg, H.: On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theor.* **24**(3), 384–386 (1978)
3. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Rogaway, P. (ed.) *CRYPTO 2011*. LNCS, vol. 6841, pp. 743–760. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_42
4. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* **50**(4), 506–519 (2003)
5. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.* **36**(5), 1301–1328 (2006). <https://doi.org/10.1137/S009753970544713X>
6. 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
7. Döttling, N.: Low noise LPN: KDM secure public key encryption and sample amplification. In: Katz, J. (ed.) *PKC 2015*. LNCS, vol. 9020, pp. 604–626. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_27
8. Döttling, N., Müller-Quade, J., Nascimento, A.C.A.: IND-CCA secure cryptography based on a variant of the LPN problem. In: Wang, X., Sako, K. (eds.) *ASIACRYPT 2012*. LNCS, vol. 7658, pp. 485–503. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_30
9. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pp. 197–206. ACM, Victoria, 17–20 May 2008
10. Katz, J., Shin, J.S.: Parallel and concurrent security of the HB and HB + Protocols. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_6
11. Kiltz, E., Masny, D., Pietrzak, K.: Simple chosen-ciphertext security from low-noise LPN. In: Krawczyk, H. (ed.) *PKC 2014*. LNCS, vol. 8383, pp. 1–18. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_1
12. Leveil, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) *SCN 2006*. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006). https://doi.org/10.1007/11832072_24

13. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX/RANDOM -2005. LNCS, vol. 3624, pp. 378–389. Springer, Heidelberg (2005). https://doi.org/10.1007/11538462_32
14. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107–124. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_6
15. 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
16. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_35
17. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) STOC, pp. 84–93. ACM (2005)
18. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_25
19. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989). <https://doi.org/10.1007/BFb0019850>
20. Yu, Y., Zhang, J.: Cryptography with auxiliary input and trapdoor from constant-noise LPN. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 214–243. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_9



PoS: Constructing Practical and Efficient Public Key Cryptosystems Based on Symmetric Cryptography with SGX

Huorong Li^{1,2,3}, Jingqiang Lin^{1,2(✉)}, Bingyu Li^{1,2,3}, and Wangzhao Cheng^{1,2,3}

¹ Data Assurance and Communication Security Research Center, Beijing, China

² State Key Laboratory of Information Security,
Institute of Information Engineering, CAS, Beijing, China

linjingqiang@iie.ac.cn

³ School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

Abstract. Public key cryptosystems (PKCs) often rely on public key algorithms based on mathematical problems that currently admit no efficient solution, such as integer factorization and discrete logarithm, which are relatively computationally costly compared with most symmetric key algorithms of equivalent security. In this paper, we propose PoS, constructing practical and efficient PKCs based on symmetric cryptography with SGX. To achieve this, we separate private and public operations into dedicated private and public SGX enclaves, hosted on the recipient and sender sides respectively, and leverage the security mechanisms provided by SGX to make symmetric keys shared between private enclave and public enclave, while being kept secret from the sender, by limiting the shared keys within public enclave and not exportable. PoS gains security guarantees when the security assumptions of SGX and symmetric cryptography stand. To demonstrate the practicality and efficiency of the PKCs constructed on PoS, we have constructed, implemented, and benchmarked two PKCs based on PoS, PKE-PoS and IBE-PoS. The evaluation results show that both PKE-PoS and IBE-PoS gain excellent performance: the performance of PKE-PoS is up to 195 times of that of RSA-2048, and the performance of IBE-PoS is up to 4 orders of magnitude higher than that of Boneh-Franklin IBE.

Keywords: Software Guard Extension
Public key cryptosystems · Identity-based encryption
Cryptographic keys

1 Introduction

Public key cryptosystems (PKCs) are fundamental security ingredients in applications and protocols. They underpin various Internet standards, such as Transport Layer Security (TLS), S/MIME, PGP, and GPG. PKCs often adopt public

key algorithms relying on cryptographic algorithms based on mathematical problems, such as integer factorization, discrete logarithm, and elliptic curve, all of which currently admit no efficient solution and are computationally intensive. Public key algorithms known so far are relatively computationally costly compared with most symmetric key algorithms of equivalent security.

Intel Software Guard Extensions (SGX) [1–3] enables execution of user-level sensitive code in an isolated environment, called *enclave*. The processor identifies and distinguishes enclaves in hardware-level by the measurement of the enclaves, which typically is the hash of enclave code. *Isolation* provided by SGX prevents other enclaves, other processes, and privileged code such as the OS and hypervisor, from reading or modifying the memory of an enclave at runtime. To protect enclave data across executions, SGX provides a security mechanism called *sealing* that allows each enclave to encrypt and authenticate data for persistent storage, with an enclave specific key derived from both processor-specific secrets and enclave measurement. SGX-enabled processors are equipped with certified cryptographic keys that can issue remotely verifiable *attestation* statements. A statement typically includes enclave measurement, fingerprint of enclave issuer, as well as user custom data, e.g., temporary public key. A verified attestation statement indicates that the enclave with given measurement is securely running on a SGX-enabled platform as expected. Through these security mechanisms, namely isolation, sealing, and attestation, SGX hardens the security of applications and services.

In this paper, we propose PoS, constructing practical and efficient PKCs based on symmetric cryptography with SGX. PoS provides two type cryptographic interfaces: private interfaces for signing/decrypting on the recipient side, and public interfaces for verifying/encrypting on the sender side. PoS allows PKCs to adopt symmetric key algorithms to implement those interfaces. To achieve this, as paradoxical as it may seem, a big challenge is that making symmetric keys shared between the sender side and the recipient side while being kept secret from the sender. We leverage the security mechanisms provided by SGX to solve such a challenge: **Firstly**, we separate private and public interfaces into two different dedicated SGX enclaves, private enclave and public enclave, hosted on the recipient and sender sides respectively. The two enclaves are isolated from each other, as well as privileged code like the OS and hypervisor, in hardware-level, making the code and data of cryptographic operations free from any modification and misuse. **Secondly**, we construct a secure private channel by leveraging remote attestation provided by SGX and provision symmetric keys to public enclave through it. The provisioned symmetric keys are limited within public enclave and never be exported. Isolation provided by SGX ensures that only public enclave can access the keys while preventing access from any other entities including the sender, other enclaves, or privileged code.

The security of PoS relies on SGX and symmetric cryptography and PoS gains security guarantees when the security assumptions of SGX and symmetric cryptography stand. To demonstrate the practicality and efficiency of the PKCs based on PoS, we have constructed PKE-PoS and IBE-PoS: PKE-PoS acts as traditional public key encryption (PKE) systems like RSA, while IBE-PoS

acts as traditional identity-based encryption (IBE) systems like Boneh-Franklin IBE [4].

We have implemented both PKE-PoS and IBE-PoS with AES-256-GCM and HMAC-SHA256, and evaluated the performance of them. We also benchmarked most commonly used PKCs, such as RSA-2048, RSA-4096, DSA-1024, DSA-2048, ECDSA-p224, ECDSA-p256, ECDSA-p384, and IBE systems with most commonly used schemes, including Boneh-Franklin scheme (BF) [5] and NTRU lattices-based scheme (GPV) [6]. All experiments were conducted on SGX-enabled computers. The benchmark results show that PoS-based PKCs gain very excellent performance, far better than any other ones: the performance of PKE-PoS signing and decryption operations is up to 195 times of that of RSA-2048 while the performance of PKE-PoS verifying and encryption operations is about 5.4 times of that of RSA-2048; the performance of IBE-PoS decryption is about 745 times of that of GPV scheme and is more than 4 orders of magnitude higher than that of BF scheme; the performance of IBE-PoS encryption is about 390 times of that of GPV scheme and is about 4 orders of magnitude higher than that of BF scheme.

Contributions. In summary, our main contributions are:

- We propose propose PoS, constructing practical and efficient PKCs based on symmetric cryptography with SGX. To the best of our knowledge, PoS is the first one that allows constructing practical PKCs based on symmetric cryptography.
- We have constructed two PKCs on our PoS, PKE-PoS and IBE-PoS, and demonstrated their practicality.
- We have implemented PKE-PoS and IBE-PoS and evaluated the efficiency of them on SGX-enabled computers. The results show that PoS-based PKCs gain excellent performance and are far more efficient than traditional PKCs.

2 Security Assumptions

We consider that PKCs based on our PoS will always employ provably secure symmetric key algorithms and symmetric keys of sufficient key length. We also consider that SGX security mechanisms, namely *isolation*, *sealing*, and *attestation*, provided by the processor will not be compromised. Our PoS does not aim to protect SGX enclaves against side-channel attacks [7, 8] and rollback attacks [3, 9]. Such attacks can be mitigated by existing solutions, e.g., [10, 11] for side-channel attacks and [12, 13] for rollback attacks. Our PoS is compatible with those solutions.

3 Principles of PoS

3.1 Overview

PoS consists of two dedicated SGX enclaves, **private enclave** and **public enclave**, hosted on the recipient and sender sides respectively. Both enclaves are isolated from the untrusted components of the host processes, as well as the

OS and hypervisor. Private enclave only provides private interfaces, i.e., *sign* and *decrypt*, while public enclave only provides public interfaces, i.e., *verify* and *encrypt*. The *attestation* and *isolation* mechanisms provided by SGX make sure that enclaves act as expected, and any modification to the code and data of the enclaves will be prevented.

In PoS, a symmetric key is used as a private key in private enclave while used as a public key in public enclave. Symmetric keys are provisioned to public enclave, limited within public enclave, and marked as not exportable, so that access from any other entities including other enclaves, specially private enclave, privileged code, and the sender are prevented by SGX. As such, the provisioned symmetric keys can only be used to perform cryptographic operations specified by public enclave, namely encrypt and verify. The sender is able to encrypt/verify messages to/from the recipient through the public interfaces while the shared symmetric keys are kept secret from the sender.

3.2 Typical Cryptographic Interfaces

PoS defines two type cryptographic interfaces: private interfaces, i.e., *sign* and *decrypt*, which use symmetric keys as private keys, and public interfaces, i.e., *verify* and *encrypt*, which use symmetric keys as public keys. For *encrypt* and *decrypt*, they can be typically implemented the same as in traditional symmetric key encryption and decryption of secure symmetric key algorithms, e.g., AES. For *sign* and *verify*, they can typically adopt secure Message Authentication Code (MAC) algorithms, e.g., HMAC. PKCs constructed based on PoS should implement those cryptographic operation interfaces.

PoS allows PKCs to extend additional interfaces that needed, e.g., key generation interfaces. The only requirement is that all those additional interfaces should comply with PoS policies, i.e., porting into different enclaves according whether the symmetric keys are used as private keys or public keys and not exporting the shared symmetric keys in enclaves.

3.3 Provisioning Symmetric Keys

Since PoS adopts symmetric key algorithms, to make the sender able to encrypt/verify messages to/from the recipient through the public interfaces, symmetric keys should be shared to public enclave. Before symmetric keys are provisioned, a secure private channel over untrusted network is constructed, leveraging remote *attestation* mechanism provided by SGX. To achieve that, public enclave generates a pair of temporary keys (sk_{tmp}, pk_{tmp}) and a signed remote attestation statement, $Q = Quote(pk_{tmp})$, which includes the measurement of public enclave, fingerprint of enclave issuer, and pk_{tmp} as the user custom data. Once verifying Q successfully, one accepts pk_{tmp} with assurance, uses it to protect the symmetric keys to be provisioned to public enclave.

The shared symmetric keys are limited within public enclave and not exportable, so that they are kept secret from any other entities.

3.4 Sealing Symmetric Keys

To protect symmetric keys across executions if necessary, PoS leverages the *sealing* mechanism provided by SGX to encrypt and authenticate symmetric keys for persistent storage. SGX ensures that the keys sealed by an enclave can only be extracted by the enclave that seals them. To make the sealed symmetric keys resist rollback attacks, one can, for example, employ SGX Monotonic Counter [12].

4 Constructing PKE on PoS

In this section, we construct PKE-PoS, a public key encryption (PKE) system based on PoS, employing a symmetric key algorithm, e.g., AES.

Like traditional PKE systems, keys are generated on the recipient side in PKE-PoS, except that a symmetric key (noted as mk) is generated within the private enclave. To enable a sender to encrypt/verify a message to/from a recipient, mk is provisioned to public enclave online. The key distribution procedure is as shown in Fig. 1, leveraging SGX remote attestation mechanism to construct a secure private channel. On success, mk is shared between public enclave and private enclave and can be sealed for persistent storage across executions. Note that mk is used as a public key within public enclave on the sender side while used as a private key within private enclave on the recipient side.

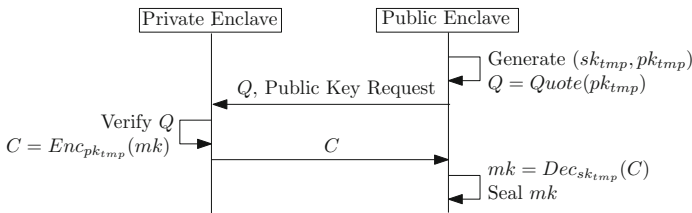


Fig. 1. Public key distribution in PKE-PoS

PKE-PoS implements *encrypt* and *decrypt* interfaces the same as in traditional symmetric key encryption and decryption provided by a secure symmetric key algorithm, e.g., AES-256-GCM, while *sign* and *verify* interfaces adopt a MAC algorithm, e.g., HMAC-SHA256.

5 Constructing IBE on PoS

Identity-based public key encryption (IBE) [4] allows an entity's public key to be derived from an arbitrary identification value (ID), such as name or email address. In this section, we construct IBE-PoS, an identity-based encryption (PKE) system based on PoS, employing symmetric key algorithms, e.g., AES,

Blowfish, Twofish, ChaCha, etc. Formally, IBE-PoS is composed by four algorithms [4], namely **Setup**, **Extract**, **Encrypt**, and **Decrypt**, as usual IBE cryptosystems:

Setup. A PKG takes a security parameter l , generates a master key mk , and outputs public system parameters $params$. l specifies the key length of user’s private keys. $params$ includes the security parameter l , a selected symmetric key algorithm as well as encryption mode, and a key derivation function (KDF) that used to extract the recipient’s private keys.

Extract. The PKG takes $params$, mk , and recipient’s ID as inputs, and then extracts a symmetric key $sk = KDF(mk, ID)$ of length l .

Encrypt. A sender takes $params$, recipient’s ID, and a message as inputs, and then outputs the ciphertext. This is done within public enclave. $params$ is retrieved from the PKG. Specially, mk is also provisioned to public enclave along with $params$. Figure 2 shows the procedure of public enclave requesting $params$ and mk from the PKG. Public enclave first extracts $sk = KDF(mk, ID)$ of length l , encrypts the message with sk , using the symmetric key algorithm and encryption mode specified in $params$, and then outputs the ciphertext. Note that mk is kept secret from the sender. Both mk and $params$ can be sealed for persistent storage across executions.

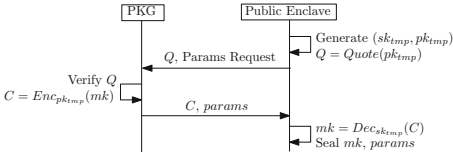


Fig. 2. System parameter request

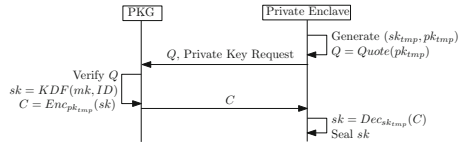


Fig. 3. Private key request

Decrypt. A recipient takes $params$, sk and ciphertext, and then outputs the message. $params$ and sk are requested from the PKG. Figure 3 shows the procedure of a recipient to request a private key from the PKG, the same as usual IBE systems: the PKG accepts the recipient’s private key request, and after successfully authenticating the recipient in some way, extracts the private key $sk = KDF(mk, ID)$ for the recipient, and provisions sk to private enclave. Private enclave then decrypts the ciphertext with sk , using the symmetric key algorithm and encryption mode specified in $params$, and then output the message. Note that mk is **not** provisioned to private enclave. Both sk and $params$ can be sealed for persistent storage across executions.

6 Implementation

We implemented PKE-PoS and IBE-PoS for the experimental purposes. The code of PKE-PoS and IBE-PoS is built on top of the popular OpenSSL library, which incorporates a multitude of cryptographic functions and large-number

arithmetic primitives, and Intel SGX SDK. Both PKE-PoS and IBE-PoS are implemented on Linux. Currently, PKE-PoS employs AES-256-GCM for encryption and decryption, and HMAC-SHA256 for signatures; IBE-PoS employs AES-256-GCM for encryption and decryption, and HKDF [14] with HMAC-SHA256 for key derivation.

7 Evaluation

The security of PoS relies on SGX and symmetric cryptography and PoS gains security guarantees when the security assumptions of SGX and symmetric cryptography stand. In this section, we focus on performance evaluation of PKE-PoS and IBE-PoS.

7.1 Experiment Setup

Our experiments were conducted on a Intel NUC6 with an SGX-enabled i3-6100U processor and 8 GB DRAM. The processor is designed for low power (15 W) usage, whose maximum frequency is 2.3 GHz. The operating system was Ubuntu 16.04 with Linux kernel version 4.13.0.

We benchmarked the speed of PKE-PoS and some commonly used traditional PKCs, including RSA-2048, RSA-4096, DSA-1024, DSA-2048, ECDSA-p224, ECDSA-p256, and ECDSA-p384, using *openssl speed* command. Also, we benchmarked the speed of IBE-PoS and compared it with most commonly used implementations of IBE schemes, including Boneh-Franklin scheme (BF) [5] and NTRU lattices-based scheme (GPV) [6]. All the benchmarks took data of same length as input.

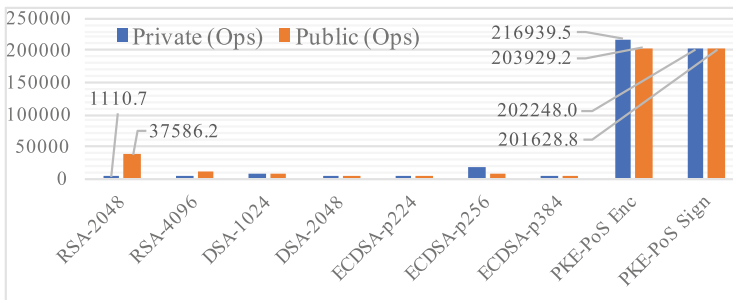


Fig. 4. Speed benchmarks for PKCs

7.2 Speed Benchmarks

Figure 4 shows the benchmark results of PKE-PoS, RSA, DSA, and ECDSA systems. We can see that the performance of PKE-PoS is far better than that of RSA, DSA, and ECDSA—compared with RSA-2048, the performance of

PKE-PoS decryption is more than 195 times of that of RSA-2048, while that is more than 182 times for signing, and for public operations, i.e., encryption and verifying, the performance of PKE-PoS is more than 5 times of that of RSA-2048.

Table 1. Comparing IBE-PoS with most commonly used implementations of IBE schemes, Boneh-Franklin scheme (BF) and NTRU lattices-based scheme (GPV).

Scheme	GPV	BF	IBE-PoS
Encryption	3.67 ms	204.85 ms	4.929 us
Decryption	1.82 ms	42.41 ms	4.663 us

Table 1 gives the speed benchmarks of GPV, BF, and our IBE-PoS schemes. It shows that decryption and encryption of our IBE-PoS are very fast, far better than GPV and BF schemes—the performance of IBE-PoS decryption is about 745 times of that of GPV scheme and is more than 4 orders of magnitude higher than that of BF scheme; the performance of IBE-PoS encryption is about 390 times of that of GPV scheme and is about 4 orders of magnitude higher than that of BF scheme.

Besides, we can calculate that for each operation of PKE-PoS and IBE-PoS, it takes more than 10000 cycles, far more time-consuming than that of original symmetric key algorithms, which takes only several dozen cycles. Take decryption operation of PKE-PoS for an example, which is the most efficient, it takes about 10600 cycles (recall that maximum frequency of the processor on our platform is 2.3 GHz). This is as expected, since enclave context switching as well as initialization of symmetric key algorithms would introduce overhead. The overhead of enclave context switching could be approximated to the time it takes for an empty enclave call, about 8914 cycles. In addition, it takes several hundred or thousand cycles for original symmetric key algorithms to setup key or Initialization Vector (IV), e.g., 1107 cycles for AES-GCM. As such, we can see that the time spent in a PKE-PoS decryption operation, 84.1% is spent in enclave context switching, 10.4% is spent in setup key or IV, while only 5.5% is spent in decryption. Nevertheless, as shown in Fig. 5, such overhead becomes

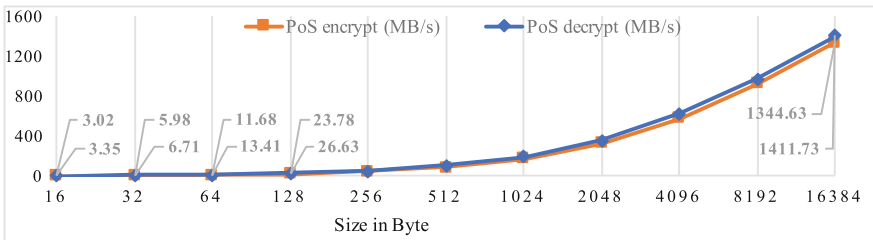


Fig. 5. The throughput of encryption in PoS-based systems

less significant when the size of data processed per operation increases as the throughput of encryption and decryption in PoS-based systems increases fast when the data size increases.

8 Discussion

In principle, PKE-PoS can employ any secure symmetric key algorithm with any encryption mode for *encrypt* and *decrypt* interfaces, and any secure MAC algorithm for *sign* and *verify* interfaces. However, there would be a risk that the sender might be able to forge a valid signature through *encrypt* interface in some cases. For example, CMAC adopts CBC mode and one is able to generate a CMAC signature for any message by encrypting the message in CBC mode with IV set to zero, and thus if PKE-PoS happens to employ a symmetric key algorithm with CBC mode, the sender would be able to forge signatures. As such, PKE-PoS should make sure that the selected symmetric key algorithm and MAC algorithm do not adopt same encryption mode. A preferable solution is to adopt an HMAC algorithm for *sign* and *verify* interfaces, instead of encryption based MAC algorithms like CMAC and GMAC.

A disadvantage of PKE-PoS is that requiring an online key distribution. A sender in PKE-PoS has to retrieve recipient's public key online from private enclave on the recipient side. This can be mitigated by introducing a provision enclave hosted on a public service, which keeps online—private enclave provisions symmetric keys to the provision enclave and then can go offline, while public enclave can request the symmetric keys after successfully authenticating to the public service if required. The provision enclave can also be incorporated into a certificate authority, making PKE-PoS compatible with PKI.

9 Related Work

Prior works have proposed many cryptosystems, leveraging system and network security mechanisms to equip cryptosystems with dedicated security features.

IB-MKD [15] employed a key distribution center (KDC) equipped with an RSA key pair to distribute message keys without the need to revoke any keys. HIBE [16] introduced lower-level PKGs to reduce the workload of the centralized PKG, in a way that a root PKG need only generate private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level. [17] proposed solutions to mitigate key escrow problem in an IBE system, by employing some distributed PKGs to generate the private keys, so that the keys are still secure when the PKGs are partially compromised. In contrary, RIKE [18] integrated the 'inherent key escrow' of IBE into PKIs to enable key escrow in PKIs while keeping the complete compatibility with PKI certificates.

Intel SGX provides preferable security mechanisms, namely isolation, sealing, and attestation, in protecting applications on the untrusted OS. VC3 [19] employed SGX to provide shielded execution to protect the confidentiality and integrity of the code and data of a program from the hosting untrusted platform.

REM [20], a new blockchain mining framework, leveraged the partially decentralized trust model inherent in SGX to achieve Proof-of-Useful-Work (PoUW), reducing the waste of PoW (Proof-of-Work). [21] used SGX to design a proof of luck consensus protocol and constructed a blockchain based on such a protocol.

10 Summary and Future Work

In this paper, we presented PoS, constructing practical and efficient PKCs based on symmetric cryptography with SGX. PoS makes this possible by sharing symmetric keys between dedicated SGX enclaves and limiting the keys within the enclaves, so that the keys are kept secret from other processes, privileged code like the OS, and the sender. PoS gains security guarantees when the security assumptions of SGX and symmetric cryptography stand. We have constructed two PKCs based on PoS to demonstrate the practicality and efficiency of PoS-based systems. The evaluation results have shown that PoS-based systems gain excellent performance, far better than traditional PKCs.

Our PoS can also be used to construct other PKCs, such as an IBE system supporting identity-based signatures with non-repudiation. A more formal analysis of semantic security of PoS remains for our future work.

Acknowledgments. We thank all reviewers for their helpful feedback. This work was partially supported by the National Natural Science Foundation of China (No. 61772518), the Cyber Security Program of National Key R&D Plan of China (No. 2017YFB0802100), and the National Cryptography Development Fund (No. MMJJ20170213).

References

1. Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., Del Cuvillo, J.: Using innovative instructions to create trustworthy software solutions, p. 11 (2013)
2. McKeen, F., et al.: Innovative instructions and software model for isolated execution. *HASP@ ISCA* **10**, 1 (2013)
3. Costan, V., Devadas, S.: Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016/86 (2016)
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
5. Boneh-Franklin IBE scheme. <https://github.com/SEI-TTG/id-based-encryption>
6. Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014*. LNCS, vol. 8874, pp. 22–41. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_2
7. Brassier, F., Müller, U., Dmitrienko, A., Kostianen, K., Capkun, S., Sadeghi, A.-R.: Software grand exposure: SGX cache attacks are practical. *arXiv preprint arXiv:1702.07521*, p. 33 (2017)

8. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 640–656. IEEE (2015)
9. Strackx, R., Piessens, F.: Ariadne: a minimal approach to state continuity. In: USENIX Security (2016)
10. Gruss, D., Lettner, J., Schuster, F., Ohrimenko, O., Haller, I., Costa, M.: Strong and efficient cache side-channel protection using hardware transactional memory. In: USENIX Security Symposium (2017)
11. Seo, J., et al.: SGX-shield: enabling address space layout randomization for SGX programs. In: Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA (2017)
12. Intel SGX Monotonic Counter. <https://software.intel.com/en-us/sgx-sdk-dev-reference-sgx-create-monotonic-counter-ex>
13. Matetic, S., et al.: ROTE: Rollback Protection for Trusted Execution. IACR Cryptology ePrint Archive 2017/48 (2017)
14. Krawczyk, H., Eronen, P.: HMAC-based extract-and-expand key derivation function (HKDF) (2010)
15. Khurana, H., Basney, J.: On the risks of IBE. In: International Workshop on Applied PKC, pp. 1–10 (2006)
16. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_34
17. Kate, A., Goldberg, I.: Distributed private-key generators for identity-based cryptography. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 436–453. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_27
18. Zhang, N., Lin, J., Jing, J., Gao, N.: RIKE: using revocable identities to support key escrow in PKIs. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 48–65. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31284-7_4
19. Schuster, F., et al.: VC3: trustworthy data analytics in the cloud using SGX. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 38–54. IEEE (2015)
20. Zhang, F., Eyal, I., Escriva, R., Juels, A., Van Renesse, R.: REM: Resource-Efficient Mining for Blockchains. IACR Cryptology ePrint Archive 2017/179 (2017)
21. Milutinovic, M., He, W., Wu, H., Kanwal, M.: Proof of luck: an efficient blockchain consensus protocol. In: Proceedings of the 1st Workshop on System Software for Trusted Execution, p. 2. ACM (2016)

Short Paper Session IV: Applied Cryptography



Application of Public Ledgers to Revocation in Distributed Access Control

Thanh Bui^(✉) and Tuomas Aura

Aalto University, Helsinki, Finland
thanh.bui@aalto.fi

Abstract. Blockchain and similar public ledger structures provide a new way to publish information in distributed systems. This paper investigates their application to distributed access control and, more specifically, to distributed group membership management where entities are represented by public keys and authorization is encoded into signed certificates. We show that public ledgers with their fundamental security properties make it easier to handle revocations in distributed settings. We also present an efficient ledger design for membership revocation.

1 Introduction

Various mechanisms have been proposed to control access in distributed environments where resources, stakeholders, and users are geographically and organizationally distributed [2, 4, 5]. Typically they are key-oriented, in which entities are represented by their public keys and authorizations are encoded into signed certificates. With these certificates, an entity can delegate some of its authority to another. The certificates can form a chain or even a complicated hierarchy that reflects the underlying relations between entities. When attempting access, a subject presents a certificate chain to the verifier. The verifier then verifies each certificate to determine the subject's access rights.

With these systems, all entities can distribute rights to the resources that they control and act as authorities for those who depend on them for the resources, which are not possible with centralized approaches (e.g. access control lists). They, however, are not flexible in terms of revocation. Let us demonstrate the issues with the following example of group membership management.

“A multi-organizational group has a sub-group of admins and a sub-group of members. The admins have the authority to add new admins and members and to remove existing ones. Alice, as an admin, adds Bob to the admins. Bob then adds his assistant Carol as an admin. Later, when Alice is transferred to a new group, Carol removes Alice from the list of admins.”

The hierarchy of certificates mandates a fixed hierarchical structure on the entities such that revoking access rights of an entity will effectively invalidate those of everyone below it in the hierarchy. It means that revoking Alice automatically causes the removal of Bob and Carol. Also, typically access rights of

an entity can only be revoked by the one granting them or specific authorized entities. This is an issue because an admin in the scenario should be able to revoke any member, not just those it added to the group.

Blockchain [9] and similar public ledger structures [1, 6, 8, 10–12] provide a new way to publish information in a distributed system. They solve several fundamental problems in the design of secure distributed systems, such as global time in the form of a strict linear order of past events and globally consistent and immutable view of the history. The goal of this paper is to apply these ideas to solving the revocation issues of distributed access control. We use *distributed group membership management* as the main case study because it provides more challenging revocation scenarios than most access control systems. As our main contribution, we show how a public ledger, as an abstraction, can help to solve the problems. We also present a ledger design that enables efficient verification and management of distributed access control. To demonstrate the feasibility and efficiency of the solution, we implemented a prototype of distributed group membership management using the public ledger design.

2 Background

This section gives background and introduces the concept of distributed group membership management, as well as related literature.

2.1 Public Ledger

A public ledger is basically a public log of *events*. It can be totally distributed with no central points of trust. Prominent examples of distributed ledgers are Bitcoin [9] and other cryptocurrencies. Typically in these cryptocurrencies, transactions are communicated as signed messages in a peer-to-peer (P2P) network with no single party responsible for any critical operation. The transactions are mined into blocks, which are appended into an append-only log, called blockchain. The systems can use various consensus mechanisms [3, 9, 13] to achieve global consistency of ledger content among participants.

While Bitcoin and its variants have achieved a large degree of success, more centralized ledgers have been proposed in the literature [1, 6, 8, 10–12]. Most of them aim to provide transparency into the web PKI. The content of these ledgers is maintained by an untrusted third party (UTP) instead of the P2P network, and they rely on auditors (or monitors) to audit the UTP's behaviors. The ledger design presented in this paper follows this approach.

Regardless of the architecture, a public ledger typically has the following properties:

- **Immutability:** The ledger is *append-only*. It must not be possible to edit the entered information or to roll back the ledger to an earlier state.
- **Consistency:** Different parties will not receive conflicting information from the ledger. Together with immutability, this ensures that different parties have the same view of the current state and history of the ledger.

- **Constraints:** The ledger may define constraints that must hold at all times. In cryptocurrencies, an important constraint is the over-spending prevention.
- **Inclusiveness:** The ledger eventually accepts all entries sent to it, except those violating defined constraints. Inclusiveness is important for access control because revocations must not be blocked by those maintaining the ledger.
- **Linear order:** The ledger might define a linear order or have some built-in concept of time on all data entries. In Bitcoin, for example, a new block is generated roughly every 10 min.

2.2 Distributed Access Control

In distributed access control systems [2,4,5], entities are typically represented by public keys (PK) and access credentials are in the form of signed certificates. These certificates bind the keys to attributes or access rights:

$$C = \langle PK_{issuer}, PK_{subject}, auth, validity, Sig_{issuer} \rangle \quad (1)$$

The certificate is signed by the *issuer*, who gives to the *subject* an *authorization*, i.e. access rights, for a *validity* period. With these certificates, an entity can delegate some of its authority to another. As a result, the certificates may form a chain or even a complicated hierarchy that reflects the underlying relations between their issuers and subjects. When a subject requests access to a resource, it presents a certificate chain to the *verifier*. The verifier then determines whether the subject is permitted to perform the requested operation by recursively verifying each certificate in the chain.

Revocation. Certificate revocations are usually communicated with signed messages, which are similar to certificates but convey negative information. The verifier usually needs some independent means to check the availability of revocations because nothing prevents the subject from excluding the negative information that would cause it to be denied access.

There are some general principles of these revocations. First, *revocations apply to certificates, not to keys*. Key revocation would be equivalent to revoking all certificates issued to or by the key. Second, if any certificate in the chain is revoked, the chain as a whole becomes invalid. This means that revoking the access rights of an entity effectively invalidates those of everyone below it in the hierarchy. Third, typically only the issuer of a certificate can revoke it. Alternatively, the certificate itself could identify the authorized revocation key.

2.3 Distributed Group Membership Management

The example in Sect. 1 illustrates the group management system that we aim to define and implement, but in a distributed setting where signed messages (i.e. certificates and revocations) are the way to communicate. The distributed access control solution in Sect. 2.2 does not work here because Alice needs to be revoked by someone below her in the hierarchy and revoking Alice should not

automatically cause the removal of Bob and Carol. In the system, each entity belong to one or more groups and is assigned *roles* in each group. Since we are interested in systems of devices and services as well as users, we will use the words *leaders* and *members*, respectively, in place of admins and users. Only the leaders have the authority to add new leaders and members and to remove existing ones. These two roles are assigned and revoked separately.

The main difference between membership revocation and certificate revocation in Sect. 2.2 is that *membership revocations apply to keys instead of certificates*. A revocation will invalidate a key's role in a group regardless of how many certificates have previously been issued to assign it. Also, a group leader has the authority to revoke any members in its group no matter which leader added them. Moreover, a key's role is valid until it is explicitly revoked. It means that a certificate remains valid even after its issuer is revoked.

To implement such revocation scheme, we aim to achieve the following. First, *all events in the system (i.e. add-member and revoke-member) are consistent and propagated reliably*. This is important because we do not want different parties to receive conflicting information. Second, it must be possible to determine the *linear order of events* related to a group. This helps to answer the question: Is key *PK* a current leader when it issues revocation *Rev*? Third, the system must enable *efficient verification and management* of group membership. In this paper, we will use public ledgers to achieve these goals.

2.4 Related Work

The widely publicized compromises of certifiers, such as Comodo and DigiNotar, have motivated quite a few proposals for monitoring the security of the web PKI with the help of public log servers. Our ledger design is inspired by these.

Certificate Transparency (CT) [8] suggests public logs of all web certificates to bring transparency to the CA operations. The log is structured as an append-only Merkle hash tree, in which new records are added to the right of the tree. Revocation Transparency (RT) [7] was suggested as a supplement for CT. It uses a sparse Merkle tree for storing revocations, which enables proofs of existence and non-existence for the revocations. Enhanced certificate transparency [10] replaced the sparse tree of RT with a shallower hash tree to handle revocation more efficiently. In AKI [6], certificates and revocations are entered into an integrity log server (ILS). It maintains an *ordered Merkle tree*, where the data in the leaf nodes is sorted by the domain name. ARPKI [1] is a redesign of AKI which provides more prudent security guarantees by combining multiple X.509 certificates from different CAs to one certificate, and by establishing quorum among a fixed group of n global ILSs. PoliCert [12] is a similar solution with focus on domain-specific certification policies. PKISN [11] timestamps certificates and revocations with the public log so that certificates issued by a CA before the CA's certificate is revoked do not become invalid. This is done by maintaining two trees that are similar to those in CT and AKI, respectively.

All of the schemes presented above enforce a fixed hierarchical structure on the entities, which is not what we want for group management. Specifically,

except PKISN, revoking an entity’s certificate in these systems automatically invalidates those of everyone below it in the hierarchy. PKISN, however, still does not allow an entity to be revoked by those below it in the hierarchy.

3 Definitions

This section defines the building blocks of distributed group membership management.

Each user or other entity in the system is represented by a public-key pair $[PK, SK]$. The public key PK is used to identify the entity.

Group. Any user can create a group by generating a key pair $[PK_O, SK_O]$ and giving the group a name. The group is then identified by the combination of the public key and the name: $G = G(PK_O, name)$.

Add-Member. Adding a member to a group is represented by a *certificate*, which specifies role R of the member and is issued by a leader’s key PK_L :

$$C_U = \langle PK_L, PK_U, G, R, \text{“add”}, t_L, Sig_L \rangle_t. \quad (2)$$

These signed documents represent *add-member* events that occurred at a specific time t , when they were issued. They contain the signer’s timestamp t_L . For now, let us assume that we have a global view of the system and know the time and order of the events. Of course, there are no reliable global clocks in a distributed system, and we cannot trust the signer’s timestamp t_L for ordering the events. Below, we will resolve this problem by defining t as the order in which the certificates or revocations were added to the public ledger.

To prove membership in a group, the entity presents to the verifier a certificate chain $CH = C_1 \dots C_n$, which starts from the group’s key. The certificate chains are typically maintained in a distributed manner, so that the leader gives to the new member not only the new certificate but also the preceding chain.

Revoke-Member. Revoking a member is represented by a signed *revocation*, which is an event similar to the certificates:

$$Rev_U = \langle PK_L, PK_U, G, R, \text{“revoke”}, t_L, Sig_L \rangle_t. \quad (3)$$

Member-Role Relation. We define who is a group member or leader by iterating through the events in the global time order. By definition, there is an initial event e_0 at time zero which makes the key PK_O leader of the group $G = G(PK_O, name)$ for all keys PK_O and all names. Consider globally all the events related to the group G and sort them by their timestamps t : $e_0 \dots e_N$. An event e_t is *authorized* if its issuer is a leader in the group after the previous event e_{t-1} in the global time order. The group membership after the event differs from the membership before the event as follows: If the event e_t is authorized and it is a member certificate, then its subject PK_U has new role R in the group G after the event. If the event is authorized and it is a revocation, then PK_U does not have role R in the group G after the event. These rules determine the group members and their roles through the global history of events. The current state of the system is the role assignments after the latest event in the global history.

4 Group Membership Revocation with Public Ledger

We now consider how the public ledger, as an abstraction, can enable secure membership revocation. Section 5 will present the ledger design in details.

4.1 Requirements of the Ledger

Naturally, we assume that the ledger content is append-only and globally consistent. In addition, we make the following requirements on the ledger.

First, events are stored with an index value, which is the cryptographic hash $H(G, R, PK_U)$ of the group, the role and the subject's public key. Anyone can query the ledger for the list of events with a given index. The ledger also provides a proof that the list is complete. With this index, the verifier can easily query events that are related to an entity in a specific group.

Second, the ledger acts as a time-stamping service that gives all events a global linear order. In particular, the ledger gives each event a *sequence number*, and this number is considered to be the global timestamp t . The ledger must assign these numbers sequentially.

Third, to detect outdated messages and replay attacks, the issuer of an event must include in the signed certificate or revocation the latest ledger sequence number that it knows t_L . The ledger requires t_L in the event to be greater than the time of the last event recorded for the same index $H(G, R, PK_U)$. This way, updates from leaders will be rejected if they are not based on the latest status of the key in the group and role.

Fourth, the ledger must check the validity of an event before entering it into the ledger. Besides cryptographic checks, this requires the issuer to prove to the ledger with a certificate chain that it is a current leader of the group. Note that validation of events is needed to prevent attackers from filling the ledger with invalid events. It is not necessary for the correctness of membership management.

Finally, the ledger must store the certificate chains that authorize revocations. This is needed only for efficiency reasons. Specifically, for a new revocation Rev , the ledger stores the certificate chain CH_{rev} that authorizes Rev . Later, the ledger is able to present CH_{rev} as an easily checkable proof that the revocation Rev was valid. This will enable membership verification with computation and communication complexity $O(n)$, where n is the length of the longest certificate chain in the ledger. Without storing CH_{rev} , the ledger might have to traverse the whole graph of relations between the group's members to rebuild the chain. Note that CH_{rev} can be stored either in the ledger or in a secondary storage.

4.2 Fundamental Processes

This section presents the fundamental processes of the system. First, we define the *checkChain* subprocess, which is needed by various processes.

checkChain subprocess: A verifier checks whether a certificate chain $CH = C_1 \dots C_n$ authorizes PK_U to role R in group $G = G(PK_O, name)$ as follows.

1. Perform general checks on the chain: (1) the certificates form a chain so that the issuer of the next certificate is always the subject of the previous one, and the signatures are valid, (2) the root key is PK_O , (3) all the certificates delegate the leader role in G , except the last certificate, which delegates the role R , and (4) the subject of the last certificate is PK_U .
2. For each certificate C_i in CH , retrieve from the ledger its ledger timestamp t_i and the list of events with the index $H(G, \text{"leader"}, PK_{i-1})$, where PK_{i-1} is the issuer of C_i . Retrieve also a proof of completeness of each list. From the list retrieved for each i , check that the leader role of the issuer PK_{i-1} has not been revoked between t_{i-1} and t_i (where $t_0 = 0$). For $PK_U = PK_n$, check that its role R has not been revoked after t_n .
3. Return “success” or “fail” depending on whether all the above checks succeed. If any one of the issuers PK_i has been revoked, return the revocation Rev in addition to the “fail” status.

addMember: A group leader PK_L adds PK_U to role R in group G as follows.

1. Issue a certificate:

$$C_U = \langle PK_L, PK_U, G, R, \text{"add"}, t_L, Sig_L \rangle.$$

2. Submit C_U into the public ledger with index $H(G, R, PK_U)$. Send to the ledger also a certificate chain CH that authorizes PK_L as a group leader.
3. Check that C_U has been given a timestamp $t > t_L$ and that it has been entered into the ledger.
4. If the ledger declines the certificate because CH is not valid, it must tell the reason. The only expected reason is that there is a revocation Rev' in the ledger making the chain CH invalid. In that case, the ledger provides Rev' . Retrieve from the ledger the chain CH' that authorized Rev' , and check the correctness of the server’s reason with the *checkChain* process on CH' .
5. If the ledger does not respond or refuses to accept C_U without a valid reason, raise an alarm. (This is implementation-specific, and we do not define it in this paper. A natural choice is to report the case to related parties.)

revokeMember(): How a group leader PK_L revokes role R of PK_U in group G is similar to the *addMember* process above. The main difference is that instead of a certificate, a revocation is issued and submitted into the ledger by the leader:

$$Rev_U = \langle PK_L, PK_U, G, R, \text{"revoke"}, t_L, Sig_L \rangle.$$

Also, the ledger needs to store the chain CH_{rev} that authorizes the revocation.

verifyMember(): The verifier checks whether certificate chain CH proves that PK_U has role R in group G as follows.

1. Check CH with the *checkChain* process. If *checkChain* returns “success”, then PK_U has role R in G .

2. If *checkChain* returns “fail” and includes revocation *Rev* as the reason, retrieve from the ledger the certificate chain CH_{rev} that authorized *Rev*. Check CH_{rev} with the *checkChain* process. If this second call to *checkChain* returns “success”, then *CH* does *not* give PK_U the role *R* in *G*.
3. If the second call to *checkChain* returns “fail”, raise an alarm because the ledger is storing the revocation *Rev* without a valid authorization for it.
4. If the ledger does not respond or returns a syntactically or cryptographically invalid revocation record, raise an alarm.

4.3 Security Considerations

Proposition 1. *The processes described in Sect. 4.2 enable the verifier to determine correctly whether a key is a member or leader of a group.*

We present informal reasoning to support Proposition 1: In Sect. 3, we defined the semantics of group membership in the global history membership events. The certificate chain *CH* in step 4.2 of the *verifyMember* process is a subset of the global history: one path of leader certificates from the initial event e_0 at time zero to the present time. The verifier first checks that, if we look at the subset alone, it would prove the membership. Now, the only events outside the subset that could change this outcome are revocations. Step 4.2 of the *checkChain* subprocess checks that no effectively timed revocations exist in the global history. Thus, if *CH* actually authorizes the membership of PK_U in *G*, the process returns “success”. On the other hand, if there exists a revocation in the global history that invalidates *CH*, then the ledger provides as evidence a such revocation *Rev* and the chain CH_{rev} that authorized *Rev*. The verifier calls *checkChain* again to check that CH_{rev} is valid. Again, the only events outside the subset of $CH \cup \{Rev\} \cup CH_{rev}$ that could change the outcome are further revocations that invalidate CH_{rev} . The ledger must prove that no such revocations exist in the ledger. When the ledger does this, the verifier knows that *CH* does not authorize role *R* in *G* for PK_U . The only reason why this might not happen is if the ledger actually contains a revocation that invalidates CH_{rev} , but that would be irrefutable evidence of the ledger’s misbehavior because it stores *Rev* without having an authorizing certificate chain for it.

While the processes implement the desired semantics, it does not mean that malicious entities cannot do anything harmful. For example, if a compromised leader acts fast, it can revoke everyone and thus destroy the group.

5 Ledger Design

This section shows how the public ledger that was used as an abstract service in Sect. 4 can be implemented at a reasonable cost.

The architecture of the public ledger that we propose to use for membership revocation resembles that of log-based solutions surveyed in Sect. 2.4. We have one untrusted third party (UTP) that maintains the ledger and multiple

independent *auditors* that monitor its honesty. It is assumed that organizations that deploy the access control solution or independent watchdogs will perform the role of auditors. Our design goal is to leave heavy work to the UTP, while keeping the workload of the auditors and clients relatively small.

Ledger Data Structure. The ledger content is structured as a single *Merkle prefix tree*, which is basically a binary tree where each path down the tree corresponds to a unique bit string x . Each bit in x represents either a left or right turn on the way down. We denote by V_x the node that corresponds to x .

The events are stored in the leaves indexed by the hash value $H(G, R, PK_U)$. The events related to the same group G , role R and subject key PK_U are bundled together into an append-only list. The most recent record is at the end of the list. Each event is additionally stored with a ledger-assigned *sequence number* t , which we also call *timestamp* because it is used in place of global time. Note that the timestamps create internal constraints to the ledger data structure, and auditors are needed to enforce their sequential assignment. We use SHA-256 for the hash function H but only extend each branch of the tree down to the lowest level where no two input $\langle G, R, PK_U \rangle$ map to the same path.

The UTP calculates a hash value h_x for each node V_x in the tree as follows. The hash of a *leaf node* is a cumulative hash of the event list stored in the leaf node, including their sequence numbers, and the leaf node's full index. The value of a non-leaf node is the hash of its two children. However, if either the left or right branch of the tree does not continue, the child value is zero.

The hash value of the root of the tree, denoted by h_{root} , summarizes the whole tree. Periodically (e.g. once a minute), the UTP appends the latest value of h_{root} into a hash chain along with the latest ledger and UTC timestamps:

$$h_{block}^i = H(h_{block}^{i-1}, h_{root}, t_{latest}, t_{UTC}) \quad (4)$$

The values of this hash chain are signed and published to the auditors. The hash chain ensures that if the UTP ever forks or modifies the history, clients can detect its malicious behavior by comparing notes with the auditors.

Ledger Operations. The UTP must accept valid certificates and revocations from clients, allow the verifier to query presence of objects in the ledger by their index values, and send event data to the auditors for auditing.

To check for the presence or absence of an event in the ledger, the UTP follows the path determined by the event's index in the Merkle prefix tree. On the way down, it accumulates a *proof* as a list of the hashes of the siblings of the path. If the search down the tree reaches a leaf node V_x and the index stored in the leaf node matches the full index, the UTP appends the hashes of events and timestamps stored in the leaf to the list. This list is the *proof of presence*. On the other hand, if the index stored in the leaf node does not match the full index, or if the search down the tree terminates at a non-existing branch, the so-far accumulated list of hash values becomes the *proof of absence*. These are the proofs of completeness received by the client in Sect. 4.

When a client submits a revocation or a certificate to the ledger, the UTP verifies it before adding it into the tree. The main part of the verification process

is to check that the issuer of the new event is a current group leader. This check is similar to *checkChain* subprocess in Sect. 4.2. The only difference is that the UTP does not need to prove presence or absence as it trusts its own information.

If the submitted event e is valid, the UTP must immediately return a signed *proof of delivery* (POD) before writing any data to the tree:

$$POD = Sig_{UTP}(e, h_{block}^i, t_{latest}, t_{UTC}) \quad (5)$$

The purpose of the latest block hashes and the ledger and UTC timestamps is to bind the receipt to the various notions of time in the system. The POD is effectively a promise that the UTP will include the certificate in the ledger as soon as it is technically possible. It will be used as proof of UTP misbehavior if it fails to enter the event into the ledger. The user then has to wait until the next block update before it can verify that the event has been included.

Auditor Operations. The task of the auditors is to verify the following: (1) the append-only property of the ledger and (2) the timestamp t for new events grows monotonically. To do so, an auditor receives from the UTP a stream of proofs of updates to the ledger. A *proof of update* is simply a proof of absence followed by a proof of presence. The proofs enable the auditor to compute the root hash before and after each update. Since the two proofs differ only for a very small part, sending the two does not take much more space than one. Furthermore, the UTP only needs to send the hashes of the events without any details since they are sufficient for the auditor to calculate the root hash. With these proofs, the auditor does not need to save any ledger data. It simply checks that the root hashes form an unbroken sequence between two consecutive block hashes.

6 Implementation and Evaluation

We implemented a prototype of the ledger and group membership system to demonstrate its feasibility. It was written with Python (2.7.11) and the M2Crypto (0.25.1) cryptography library. We used RSA-2048 keys as the entity identities. We simulated the group management with 10^6 users in the system, and groups are gradually formed randomly among them. Membership certificates and revocations were randomly issued among the users. Since the workload of the client is relatively small, we only evaluated the performance of the ledger and the auditor. In all the experiments, except for the estimation of memory usage, we inserted 10^7 certificate and revocation entries into the ledger, of which 10% was revocations, before executing the performance measurements.

Memory Usage. The experiments showed that the ledger consumed less memory when the revocation rate increased: with 10^7 entries, the ledger used on average 8.94 GB, 8.8 GB and 8.66 GB of memory when the fractions of revocations were 10%, 20% and 30%, respectively. The reason is that revocations do not expand the Merkle tree as certificates do. While the ledger needs to store the certificate chains that authorized the revocations, pointers to the certificates in the ledger data are sufficient to represent the chains.

Cost of Verifying Certificate Chains. When the client submits an event to the ledger, the ledger only verifies the authorizing certificate chain of the event (with *checkChain*) and appends the event to a queue so that it can sequentially insert them later when it signs the root. We measured the performance of the UTP server when the length of the certificate chains was relatively large ($L = 50$). Since the verification does not involve updating the ledger, it is easy to parallelize. The UTP server was able to perform on average 9385 verifications per second. We expect the typical length of the chains to be shorter than 50, which will result in proportionally better performance.

Cost of Adding Events. To insert an event to the Merkle prefix tree, the hash values of all the nodes on the path from the updated leaf to the root need to be recalculated. Our UTP server took, on average, 265 ms to insert a new entry to the tree and to collect the proof of update for the second-type auditors. The average proof of update was less than 1 KB in size.

Cost of Auditing. Since the auditor verifies the proofs of update step by step without keeping any ledger data, it does not require any significant storage space. Regarding the bandwidth, it needs to receive the logarithmic-size proofs of update, each of which was less than 1 KB in our experiments but grows logarithmically with the size of the ledger. Our auditor implementation was able to process on average 10526 updates per second.

7 Discussion

This section describes possible extensions to the proposed revocation solution.

Certificate Validity Times and Temporary Revocations. It is tempting to include a UTC validity period in the membership certificates, similar to that in X.509. Handling such validity periods needs to be done with care, however. In X.509, the verifier is only interested in whether the certificates are valid at the verification time, which it can do by comparing with a relatively accurate clock. In distributed group management, however, the verifier needs to know the order of past events, which cannot be determined from the verifier's clock. Moreover, even small differences in clocks or message propagation time could lead to different interpretations of a revocation. The solution is to let the UTP to decide the order of all events, including the expiry of certificates. It means that the certificate expiry event should be entered into the ledger when it occurs.

This further calls our attention to the question whether a revocation can have a validity period. Such a temporary revocation of access rights can be useful, for example, when there is uncertainty about whether a device is lost. We can implement temporary revocation by adding an optional UTC validity period to the signed revocation message. Just like with the certificates, the UTP should add the expiry events to the linear order of the ledger when they occur.

Role Inheritance and RBAC Support. We have considered only two kinds of roles, *leader* and *member*, which are assigned independently. In practice, more

roles could be defined, such as owner or guest. This calls into question the wisdom of having any fixed roles at all. Also, there could be inheritance between the roles to make their assignment easier. Indeed, the group-management solution presented in this paper is structured so that it can be extended with inheritance, role hierarchies and possibly other role-based access control features. We leave them as future work to focus on revocation, the main topic of this paper.

8 Conclusion

In this paper, we demonstrate that public ledger is a suitable abstraction for solving problems of distributed access control, and in particular those that arise from revocation and other negative permissions. As the main result, we present a ledger-based design for distributed group membership management and its experimental implementation.

References

1. Basin, D., Cremers, C., Kim, T.H.J., Perrig, A., Sasse, R., Szalachowski, P.: ARPKI: attack resilient public-key infrastructure. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 382–393. ACM (2014)
2. Blaze, M., Keromytis, A.D.: The KeyNote trust-management system version 2. RFC 2704, IETF (1999)
3. Buterin, V.: What proof of stake is and why it matters. Bitcoin Mag. (2013)
4. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI certificate theory. RFC 2693, IETF (1999)
5. Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V.: dRBAC: distributed role-based access control for dynamic coalition environments. In: International Conference on Distributed Computing Systems, pp. 411–420. IEEE (2002)
6. Kim, T.H.J., Huang, L.S., Perring, A., Jackson, C., Gligor, V.: Accountable key infrastructure (AKI): a proposal for a public-key validation infrastructure. In: International Conference on World Wide Web, pp. 679–690 (2013)
7. Laurie, B., Kasper, E.: Revocation transparency. Google Research, September 2012
8. Laurie, B., Langley, A., Kasper, E.: Certificate transparency. RFC 6962, IETF (2013)
9. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
10. Ryan, M.D.: Enhanced certificate transparency and end-to-end encrypted mail. In: Proceedings of NDSS (2014)
11. Szalachowski, P., Chuat, L., Perrig, A.: PKI safety net (PKISN): addressing the too-big-to-be-revoked problem of the TLS ecosystem. In: IEEE European Symposium on Security and Privacy (2016)
12. Szalachowski, P., Matsumoto, S., Perrig, A.: PoliCert: secure and flexible TLS certificate management. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 406–417. ACM (2014)
13. 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



Micropaying to a Distributed Payee with Instant Confirmation

Peifang Ni^{1,2,3}, Hongda Li^{2,3}(✉), and Dongxue Pan^{1,2,3}

¹ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

² Data Assurance and Communication Security Research Center, Beijing, China
{nipeifang, lihongda, pandongxue}@iie.ac.cn

³ State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China

Abstract. Micropayment means the value of transaction is small, i.e., payment worths a few pennies. To achieve instant micropayments, Hearn and Spilman introduced a notion of *payment channel*. In this paper, we formally discuss the robustness requirements of a scheme that is suitable for micropayments, consider the explicit value of penalty and user privacy leakage. More precisely, we propose a micropayments scheme for decentralized blockchain-based payment system based on the notion of *payment channel*, which enables a payee to receive funds at several unsynchronized points of sale and penalize the double-spenders, with instant confirmation.

Keywords: Micropayments · Bitcoin · Blockchain · Double-spending
User privacy

1 Introduction

There are two main techniques to handle transactions with small value. *Payment channel* [1] is emerging in bitcoin community [15, 16] that needs two transactions being confirmed in blockchain network: creating channel transaction and closing channel transaction. *Probabilistic payment* [5, 20] lets payee receive a macro-value with a given probability and a micro-value for each transaction in expectation.

Decentralized Micropayments. In decentralized system, all participants achieve an agreement together via consensus mechanism, i.e., proofs-of-work. Realizing micropayments in decentralized system brings us new challenge to balance efficiency and security.

Double Spending. Micropayments scheme, which requires payee responding to payer in short time and just doing local confirmation, is easy suffering from

The work is supported by National Key R&D Program of China (No. 2017YFB0802500).

double-spending attack that payer reuses a valid voucher *cert* to different unsynchronized payees repeatedly before being detected.

User Privacy. User privacy is not only concealment of identity, such as the pseudonym in bitcoin system. In this work, we also consider protecting user transaction message among the unsynchronized points of sale.

We here ask the following question:

Is that possible to strengthen micropayments scheme for decentralized blockchain-based payment system so that it can be secure even adversary reuses a voucher repeatedly before being detected and enhance user privacy among the unsynchronized points of sale?

1.1 Our Contributions

We give an affirmative answer to the above question. Most existing micropayments schemes [5, 14, 16, 20] focus on general setting, where payee is a single entity. In real word, it is usual that a merchant consists of several geographically distributed and unsynchronized points of sale. We mainly focus on the security of micropayments scheme in this complex setting.

General Setting. The first step, we assume that payee B is a single entity and accepts small payments as shown in *micropayment 1* (Fig. 1).

Complex Setting. Based on step one, We go further to explore a complex setting [19] as shown in *micropayment 2* (Fig. 2), where B consists of several geographically distributed and unsynchronized points of sale. We then propose our construction *micropayment 3* (Fig. 3) that solves the security problems of *micropayment 2*.

Robustness Requirements for Achieving Micropayments:

1. Basic requirements in general setting:
 - (1) **Instant Confirmation.** Micropayment requires quick response, i.e., payer will receive service as soon as he sends valid messages (voucher).
 - (2) **Small Transaction Fees.** Payer is unwilling to use a payments system to handle small transactions, which costs high transaction fess, since the fees maybe higher than the value of transaction.
2. Additional requirements in complex setting:
 - (1) **Preventing Double-Spending.** Security in the presence of reusing a voucher (*cert*), i.e., payer spends a *cert* to different points of sale with the risk of being detected and losing coins.
 - (2) **Protecting User Privacy.** Security in the presence of using the voucher provided by the last transaction in the current transaction, i.e., payer spends a voucher *cert* signed by last payee to current payee without disclosing the identity of last payee.

Expiry Time. To solve the above questions, we propose a notion of *expiry time*, which means that each voucher is valid during a given time.

Upper Bound of Penalty. We give a proper value of penalty, which means that it makes the malicious payer at a disadvantage for his dishonesty and is reasonable for honest payer. What's more, we get the upper bound of penalty as $p \leq (\hat{T}/\bar{T}) * u_2$ (more details are in Sect. 3.3).

For user privacy, we utilize *ring signature* during the process of *paying through channel* to break linkage between singer and signature.

1.2 Related Work

Many off-line micropayments schemes are proposed [8, 14, 18] with a trusted third party to sign a voucher for payer and punish cheaters. Bitcoin system is a peer-to-peer fully decentralized payment system introduced in [13]. Unlike traditional e-cash system [2, 4], where there is a central bank to handle transactions and detect cheaters. Decentralized system utilizes distributed public ledger *blockchain* to record all transactions.

Probabilistic payment was proposed in [12, 17] that allows payer to execute series of small transactions. Rivest [17] and Micali [12] proposed *lottery-based payment* to overcome the relative high fees of small transactions. [12, 20] are implementations of this idea. [5] presents a decentralized micropayment scheme by following the way of probability payment.

Creating payment channel was introduced in [1, 9]. [21] discusses two major questions about why we need micropayments. Further studies as [6, 16]. Constructing anonymity set [11] enhances privacy in some certain situations. [10] uses *TumbleBit*, a new unidirectional unlinkable payment hub, to allow payer to execute payment via an untrusted intermediary. These schemes are secure if the size of set is big enough and majority of participants are alive. [19] proposed a micropayment scheme in complex setting, but there are two problems obviously in this scheme: double-spending and user privacy leakage. More details are in Sect. 3.2.

1.3 Outline of the Paper

The rest of the paper is organized as follows. In Sect. 2, we give preliminaries in our construction. In Sect. 3, we show our detailed construction. In Sect. 4, security proofs are presented. Conclusion is in Sect. 5.

2 Preliminaries

In this section, we give the main techniques behind our construction and the definitions of security properties are presented in *game-based* fashion.

2.1 Techniques

Definition 1 (*Ring Signature*). A ring signature scheme is a triple of p.p.t. algorithms $\mathcal{RS} = (\text{Gen}, \text{Sign}, \text{Vrfy})$ [3]. Formally:

- $Gen(1^\lambda)$. Takes as input the security parameter λ , outputs a public key pk and a secret key sk .
- $Sign_{sk}(R, M)$. Outputs a signature σ on message M with respect to ring $R = (pk_1, \dots, pk_n)$.
- $Vrfy_R(M, \sigma)$. Takes as input a ring R , a message m , and a signature σ for M to return a single bit $b = 1/0$.

Definition 2 (*Accountable Assertion*). We recall the definition in [19] that consists of four algorithms $\Pi = (Gen, Assert, Verify, Extract)$:

- $(pk, sk, auxsk) \leftarrow Gen(1^\lambda)$: Outputs a key pair consisting of a public key pk and a secret key sk , and auxiliary secret information $auxsk$.
- $\tau / \perp \leftarrow Assert(sk, auxsk, ct, st)$: Takes as input a secret key sk , auxiliary secret information $auxsk$, a context ct , and a statement st and returns either an assertion τ or \perp to indicate failure.
- $b \leftarrow Verify(pk, ct, st, \tau)$: Outputs 1 if τ is a valid assertion of a statement st in the context ct under the public key pk .
- $sk / \perp \leftarrow Extract(pk, ct, st_0, st_1, \tau_0, \tau_1)$: Takes as input a public key pk , a context ct , two statements st_0, st_1 , two assertions τ_0, τ_1 and returns either the secret key sk or \perp to indicate failure.

2.2 Security Properties

According to our goals, the micropayments scheme should satisfy three security properties: *unforgeability*, *unlinkability* and *double-spending detection*. We show these security properties in the following three experiments as $\text{Exp}_{\Pi^m, \mathcal{A}}^{uf}(\lambda)$, $\text{Exp}_{\Pi^m, \mathcal{A}}^{ul}(\lambda)$ and $\text{Exp}_{\Pi^m, \mathcal{A}}^{ds}(\lambda)$.

Definition 3 (*Unforgeability, Unlinkability, double-spending detection*). Given a micropayments scheme Π^m in blockchain-based system, a p.p.t. adversary \mathcal{A} , security parameter λ and consider the followings:

Experiment $\text{Exp}_{\Pi^m, \mathcal{A}}^{uf}(\lambda)$

$\{(pk_i, sk_i)\}_1^n \leftarrow \mathcal{RS}.Gen(1^\lambda)$; $Q = \emptyset$; $R = \{pk_i\}_1^n$
 $cert^* \leftarrow \mathcal{A}^{O_{cert}(i, R, state)}(R)$; $i \in [n]$ is index of each sale
 $Q = Q \cup (., R, state)$, $cert^* = (state^*, \sigma^*)$; $(., R, state^*) \notin Q$
 if $Vrfy_R(state^*, \sigma^*) = 1$, then return 1, else return 0

Experiment $\text{Exp}_{\Pi^m, \mathcal{A}}^{ul}(\lambda)$

$\{(pk_i, sk_i)\}_1^n \leftarrow \mathcal{RS}.Gen(1^\lambda)$; $Q = \emptyset$; $R = \{pk_i\}_1^n$;
 $(cert^*, R, i) \leftarrow \mathcal{A}^{O_{link}(cert^*, R)}(R)$; $i \in [n]$ is index of each sale
 $Q = Q \cup (cert^*, R)$; $(cert^*, R) \notin Q$
 if B_i is the signer of $cert^*$, then return 1, else return 0

Experiment $\text{Exp}_{\Pi^m, \mathcal{A}}^{ds}(\lambda)$

$\{(pk_i, sk_i)\}_1^n \leftarrow \mathcal{RS.RK}(1^\lambda); (pk_{\mathcal{A}}, sk_{\mathcal{A}}, auxsk_{\mathcal{A}}) \leftarrow \Pi.Gen(1^\lambda)$
 $Q = \emptyset; R = \{pk_i\}_1^n; (service, cert') \leftarrow \mathcal{A}^{O_{spend}(tx, \tau, cert)}(R)$
 $Q = Q \cup (tx, \tau, cert); sk_{\mathcal{A}} \leftarrow \text{Extract}(pk_{\mathcal{A}}, Q)$
 if $\{(tx, \tau, cert), (tx', \tau', cert')\} \in Q \wedge (pk_{\mathcal{A}}, sk_{\mathcal{A}}) \notin \Pi.Gen(1^\lambda)$
 then return 1, else return 0

We define the advantage of \mathcal{A} in the above experiments as:

$$\begin{aligned}
 Adv_{\Pi^m, \mathcal{A}}^{ul}(\lambda) &= Pr[\text{Exp}_{\Pi^m, \mathcal{A}}^{ul}(\lambda) = 1] - \frac{1}{n} \\
 Adv_{\Pi^m, \mathcal{A}}^{uf}(\lambda) &= Pr[\text{Exp}_{\Pi^m, \mathcal{A}}^{uf}(\lambda) = 1] \\
 Adv_{\Pi^m, \mathcal{A}}^{ds}(\lambda) &= Pr[\text{Exp}_{\Pi^m, \mathcal{A}}^{ds}(\lambda) = 1]
 \end{aligned}$$

3 Micropayments System

In this section, we propose a scheme about achieving micropayments in decentralized blockchain-based system in three steps.

3.1 Micropay 1

Before showing the description of *micropay 1*, we assume that A has a bitcoin address pk_1 with value v and unforgeable digital signature scheme with algorithms $(Gen, Sign, Vrfy)$. We show this scheme in Fig. 1.

Security Analysis. In *micropay 1*, A succeeds to micropay to B with one security problem that B can get all knowledge of A 's purchase messages that breaks A 's privacy.

3.2 Micropay 2

Now we show a scheme in which B is a distributed entity by recalling the construction in [19]. We give a simple description in Fig. 2.

Security Analysis. In *micropay 2*, A succeeds to micropay to a distributed B , but with the following security problems:

- (1) During **Stage 1**, it does not specify the size of p , so that A can spend more than $d + p$ easily. For example, A reuses a $cert$ signed by B many times and spends $\{\{b_i\}_1^n | b_i < d\}$ to $\{B_i\}_1^n$ in time T' . Consequently $\sum_1^n b_i > d + p$, which makes penalty useless.
- (2) During **Stage 2**, A sends $cert$ signed by B_j to B_i . So B_i verifies $cert$ by doing $Vrfy(pk_{B_j}, cert^*) = 1$ and B_i gets knowledge that A has bought service from B_j , which breaks A 's privacy.

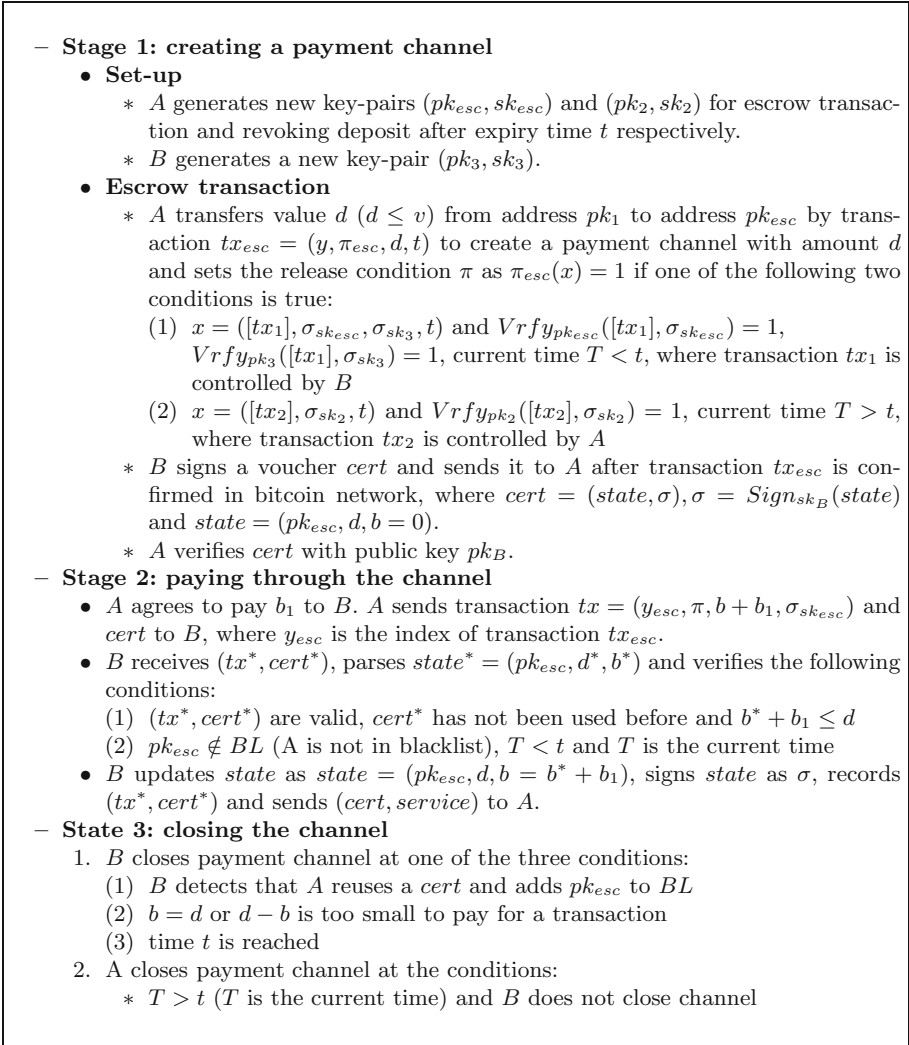


Fig. 1. A micropays to a single B

3.3 Micropay 3

To overcome the problems in *micropay 2*, we present *micropay 3*. In this scheme, we employ *expiry time* to control the number of a *cert* being reused and use *ring signature* scheme to hide A 's former purchase messages to current payee.

Notations. d, p is denoted the amount of deposit and penalty respectively. T is time that escrow transaction is locked and T' is the expiry time of voucher *cert*. Price of service provided by B_i is v_i and we let $u_1 = \min\{v_1, \dots, v_n\}, u_2 = \max\{v_1, \dots, v_n\}$. The average time of each transaction is denoted by \bar{T} , \bar{T} is time

slot that B collects all transactions recorded by each point B_i and \hat{T} is the working time of each point B_i within time \bar{T} . Let $T = l\bar{T} + T_{conf}$ ($l = 1, 2, \dots$) to ensure that B can close the payment channel before A revokes escrow transaction and T_{conf} is a safety margin to guarantee transactions broadcasted by B being confirmed on blockchain. The number of transactions that A can have is $\lfloor \frac{d}{u_2} \rfloor \leq k \leq \lfloor \frac{d}{u_1} \rfloor$ and $g(d, u_2)$ is a function to specify the remaining number of transactions that A can have. Function $f(d, u_2)$ denotes the total number that A can double-spend in the worst case.

Assumptions. (1) $k = \lfloor \frac{d}{u_2} \rfloor > 1$. (2) A can only have one transaction synchronously.

Upper Bound of Penalty: Preventing Double-Spending

Case 1

- (1) A broadcasts escrow transaction with value $d + p$.
- (2) B computes and signs $cert$ with expiry time $T' = \tilde{T} k$ for A .
- (3) A sends $cert^* = (state^*, \sigma^*)$ to B_i and gets an updated $cert'$ signed by B_i with expiry time $T' = \tilde{T} * (k^* - 1)$.

We let $g(d, u_2) \leftarrow g(d, u_2) - 1$. In the worst case, the number of $cert$ signed by B_i can be double-spent is $(k^* - 1 - 1)$, So $f(d, u_2) = \frac{d(d-u_2)}{2u_2^2}$ and we set $p = f(d, u_2) * u_2$ for $f(d, u_2) * \tilde{T} \leq \hat{T}$ or $p = (\hat{T} / \tilde{T}) * u_2$.

Case 2

- (1) A broadcasts escrow transaction with value $d + p$.
- (2) B computes and signs $cert_0$ with expiry time $T' = \tilde{T} \lceil P_0 k_0 \rceil$ ($0 < P_0 \leq 1, k_0 = k$) for A .
- (3) A sends $cert_{i-1} = (state_{i-1}, \sigma_{i-1})$ to B_i and gets $cert_i$ signed by B_i with expiry time $T' = \tilde{T} \lceil P_i k_i \rceil$ ($k_i = k_{i-1} - 1, 0 < P_i \leq 1$).

In case 2, we let $g_i(d, u_2) \leftarrow \lceil P_i(k_{i-1} - 1) \rceil$. So $f(d, u_2) = \sum_{i=0}^n (P_i k_i - 1)$ and we set $p = f(d, u_2) * u_2$ for $f(d, u_2) * \tilde{T} \leq \hat{T}$ or $p = (\hat{T} / \tilde{T}) * u_2$. In this case, double-spending attack can be prevented drastically when $P_i = \frac{1}{k_i}$ and expiry time of each $cert$ is \tilde{T} . But \tilde{T} is short in micropayments scheme, so A is required to keep on having transactions in case that the voucher expires. B_i can select a proper $P_i (P_i > \frac{1}{k_i})$ according to demands.

Ring Signature: Protecting User Privacy

The main idea that we apply ring signature scheme [3] in our scheme is as following. Payee in ring R generates a ring signature σ of $state$, which contains of n ciphertexts and a proof π . Proof π is produced by ZAP (the definition can be referred in [7]) to proof that one of ciphertexts is an encryption of a signature on the $state$ with respect to the ring members, that corresponds to σ . Finally, payee sends $cert$ to payer without actually exposing the signature. The formal construction is given in Appendix A.

Full Protocol Π^m . Based on the above analysis, we give our construction that with higher security in Fig. 3.

Assumptions: B and its points of sale B_i have corresponding key pairs (pk_B, sk_B) , $\{(pk_{B_i}, sk_{B_i})\}_1^n$ respectively. B collects transactions recorded by each B_i at time T' .

- **Stage 1: creating a payment channel**
 - A sets up bitcoin key pair (pk_A, sk_A) and accountable assertions keys $(apk = pk_A, ask = sk_A, auxsk)$ for non-equivocation contracts.
 - A creates payment channel with amount $d + p$ and expiry time t ($t > T'$).
 - B provides a signed voucher $cert = (state, \sigma)$, where $state = (t, d, k = 0, b = 0, B)$, $\sigma = Sign_{sk_B}(state)$, after escrow transaction is confirmed in network.
- **Stage 2: paying through the channel**
 - A agrees to pay b_i to B_i , then B_i selects a fresh nonce r and sends it to A .
 - A computes $\tau \leftarrow Assert(ask, auxsk, k, r)$ and sends $(tx, \tau, cert)$ to B_i .
 - B_i receives $(tx^*, \tau^*, cert^*)$, parses $state^* = (t^*, d^*, k^*, b^*, B_j)$ and verifies:
 - * $Vrfy(pk_{B_j}, state^*, \sigma^*) = 1$, $Vrfy(apk, k^*, r, \tau^*) = 1$
 - * tx^* is a valid transaction with amount $b^* + b_i$ and $b^* + b_i \leq d^*$
 - * $A \notin BL$ (A is not in blacklist) and $T < t^*$ (T is current time)
 - B_i updates $k = k^* + 1$, $b = b^* + b_i$, signs $\sigma = Sign_{sk_{B_i}}(state)$, where $state = (t^*, d^*, k, b, B_i)$, records tx^*, τ^* and sends $(service, cert)$ to A .
- **Stage 3: closing the channel**
 1. B collects all transactions recorded by each B_i at time T' and close the channel at one of the three conditions:
 - (1) Expiry time t is reached and A is honest
 - B signs and broadcasts the last tx that is sent by A to get funds
 - (2) B detects A 's dishonesty by τ
 - B extracts sk_A from two different assertions τ_1, τ_2 about k
 - B signs a transaction with $d + p$ from payment channel with sk_B, sk_A
 - (3) $b = d$ or $d - b$ is too small to pay a transaction
 - B signs and broadcasts the last tx that is sent by A to get funds
 2. A signs a transaction with $d + p$ from payment channel to closes the channel:
 - (1) $T > t$ (T is the current time) and B does not close the channel

Fig. 2. A micropays to a distributed B

4 Security Proofs

Theorem 1. *If the ring signature scheme $\mathcal{RS} = (Gen, Sign, Vrfy)$ is unforgeable and anonymous, the accountable assertion is extractable efficiently. Then, for any p.p.t. adversary \mathcal{A} and security parameter λ , the micropayments scheme Π^m is secure as defined in Sect. 3.3.*

Proof (Unforgeability). Suppose that Π^m does not achieve *unforgeability*, then it follows that there is a p.p.t. adversary \mathcal{A} that succeeds in experiment $Exp_{\Pi^m, \mathcal{A}}^{uf}(\lambda)$ with non-negligible probability. So there exists polynomial function $p(\cdot)$ such that for security parameter λ and holds that: $Pr[Exp_{\Pi^m, \mathcal{A}}^{uf}(\lambda) = 1] \geq 1 - \frac{1}{p(\lambda)}$. Using \mathcal{A} as a subroutine, we construct a p.p.t. adversary \mathcal{A}' . with input of $(R, state)$: (1) invoke \mathcal{A} with $(R, state)$ and \mathcal{A} outputs $cert = (state, \sigma)$ (2) if $\mathcal{RS}.Vrfy_R(cert) = 1$, then halt and output $cert$, otherwise output a uniformly selected number $r \in_R \{0, 1\}^\lambda$.

Cryptographic Primitive: accountable assertions, standard digital signature scheme, semantically-secure public-key encryption scheme, ZAP

Assumptions:

- (2) B has key-pair (pk_{SB}, sk_{SB}) for signing transactions and the initial $cert$.
- (3) Points of sale $\{B_i\}_1^n$ have corresponding key pairs (pk_{SB_i}, sk_{SB_i}) , (pk_{EB_i}, sk_{EB_i}) and set $pk_{B_i} := (pk_{SB_i}, pk_{EB_i})$.
- (4) A has key-pair (pk_A, sk_A) for signing transaction and auxiliary secret information $auxsk$ for accountable assertions.
- (5) B and its points of sale $\{B_i\}_1^n$ hold corresponding probability $\{P_i\}_0^n$.
- (6) Public parameters $PP = (pk_{SB}, R, pk_A, R_E, \{P_i\}_0^n, g(d, u_2), f(d, u_2), u_1, u_2, \hat{T}, \tilde{T}, \bar{T})$, $R = (pk_{B_1}, \dots, pk_{B_n})$, and $R_E := (pk_{EB_1}, \dots, pk_{EB_n})$.

– **Stage 1: creating a payment channel**

- A and B wish to execute micropayments with amount d .
- B computes p for A according to $f(d, u_2)$.
- A broadcasts escrow transaction with amount $b + p$, expiry time T and well defined release condition π to create payment channel with B .
- B provides A with $cert = (state, \sigma)$, where $\sigma = Sign'_{sk_{SB}}(state)$, $state = (pk_A, d, k = \lfloor \frac{d}{u_2} \rfloor, b = 0, T')$ and $T' = \tilde{T} g(d, u_2)$ after the escrow transaction is confirmed in bitcoin network.

– **Stage 2: paying through the channel**

- A pays v_i to B_i .
- B_i selects a fresh nonce r and sends it to A .
- A computes $\tau \leftarrow Assert(sk_A, auxsk, k, r)$ and sends $(tx, \tau, cert)$ to B_i .
- B_i receives $(tx^*, \tau^*, cert^*)$, parses $cert^* = (state^*, \sigma^*)$, $state^* = (pk_A, d^*, k^*, b^*, T'^*)$ and verifies the following conditions:
 - * $\mathcal{RS}.Verify_R(cert^*) = 1$ or $Verify'(pk_{SB}, cert^*) = 1$ for $k^* = \lfloor \frac{d}{u_2} \rfloor$.
 - * $Verify(pk_A, k^*, r, \tau^*) = 1$.
 - * tx^* is a valid transaction with amount $b^* + v_i$ and $b^* + v_i \leq d^*$.
 - * $A \notin BL(A \text{ is not in blacklist})$ and $t < T'^*$ for the current time t .
- B_i updates $k = k^* - 1, b = b^* + v_i, T' = \tilde{T}g(d, U_2)$, signs $\sigma = \mathcal{RS}.Sign_{sk_{SB_i}}(state, R)$, where $state = (pk_A, d^*, k, b, T')$, records (tx^*, τ^*) and sends $(service, cert = (state, \sigma))$ to A .

– **Stage 3: closing the channel**

1. B collects all transactions recorded by $\{B_i\}_1^n$ at time \bar{T} and close the channel at one of the three conditions:
 - (1) Expiry time T is reached and A is honest.
 - B signs and broadcasts the last tx that is sent by A to get funds.
 - (2) B detects A 's dishonesty by τ .
 - B extracts sk_A from two different assertions τ_1, τ_2 about k .
 - B signs a transaction with $d+p$ from payment channel with sk_B, sk_A .
 - (3) $b = d$ or $d - b < u_1$.
 - B signs and broadcasts the last tx that is sent by A to get funds.
2. A signs a transaction with $d + p$ from payment channel to closes the channel:
 - (1) $t > T$ (t is the current time) and B does not close the channel.

Fig. 3. A micropays to a distributed B with high security

When \mathcal{A} outputs a valid forgery $cert$, then \mathcal{A}' outputs the same forgery. Note that \mathcal{A}' outputs a valid signature whenever \mathcal{A} does.

since $cert = (state, \sigma) = (state, \mathcal{RS}.Sign_{sk}(R, state))$
implies $\mathcal{RS}.Verify_R(state, \sigma) = 1$

So \mathcal{A}' can forge a valid signature with respect to *ring signature scheme* (in Appendix A) with non-negligible probability, which contracts the unforgeability property of ring signature scheme. This completes the proof.

Proof (Unlinkability). Suppose that Π^m does not achieve *unlinkability*, then it follows that there is a p.p.t. adversary \mathcal{A} that succeeds in experiment $Exp_{\Pi^m, \mathcal{A}}^{ul}(\lambda)$ with non-negligible probability. So there exists polynomial function $p(\cdot)$ such that for security parameter λ and holds that: $Pr[Exp_{\Pi^m, \mathcal{A}}^{ul}(\lambda) = 1] \geq \frac{1}{n} + \frac{1}{p(\lambda)}$. Using \mathcal{A} as a subroutine, we construct a p.p.t. adversary \mathcal{A}' : (1) \mathcal{A}' selects two public keys pk_0, pk_1 and a valid $state$, (2) \mathcal{A}' is given $cert_b, b \in \{0, 1\}$ and invokes \mathcal{A} with $(cert_b, R)$, (3) \mathcal{A} outputs $(cert_b, R, b')$, then \mathcal{A}' halts with output b' .

Note that \mathcal{A}' outputs b' whenever \mathcal{A} does. By assumption we have that $Pr[\mathcal{A}'(cert_b, state, pk_0, pk_1) = b' : b = b'] \geq \frac{1}{2} + \frac{1}{p(\lambda)}$. So \mathcal{A}' can distinguish the signatures signed by different members of a ring with non-negligible probability, which contracts the anonymity property of ring signature scheme (in Appendix A). This completes the proof.

Proof (Double-Spending Detection). According to the extraction of accountable assertion that for any p.p.t. adversary \mathcal{A} , there exists a negligible function $negl(\cdot)$ such that for security parameter λ and holds that:

$$Pr[Extract(pk, ct, st_0, st_1, \tau_0, \tau_1) \neq sk \wedge \forall b \in \{0, 1\}, Verify(pk, ct, st_b, \tau_b) = 1 \wedge st_0 \neq st_1 : \tau_b \leftarrow Assert(sk, auxsk, ct, st_b)] < negl(\lambda).$$

Suppose that if there is a p.p.t. adversary \mathcal{A}' , which spends a $cert$ twice with the form: $(tx_0, \tau_0, cert)$ and $(tx_1, \tau_1, cert)$ without being detected. It follows that \mathcal{A} succeeds in experiment $Exp_{\Pi^m, \mathcal{A}'}^{ds}(\lambda)$ with non-negligible probability. So we have that for some polynomial function $p(\cdot)$ and security parameter λ : $Pr[Exp_{\Pi^m, \mathcal{A}'}^{ds}(\lambda) = 1] \geq 1 - \frac{1}{p(\lambda)}$. That implies: $Pr[sk_{\mathcal{A}'} \leftarrow Extract(pk_{\mathcal{A}'}, Q): (pk_{\mathcal{A}'}, sk_{\mathcal{A}'}) \notin \prod [Gen(1^\lambda)](tx_0, \tau_0, cert) \in Q \wedge (tx_1, \tau_1, cert) \in Q] \geq 1 - negl(\lambda)$.

So that contracts to the *extraction* property of accountable assertion. This completes the proof.

5 Conclusion

In this paper, we analysed previous works, extracted the robustness requirements for achieving micropayments in decentralized blockchain-based system and explored efficient solutions to achieve these requirements.

A Ring Signature

We now recall the construction of a ring signature in [3] and modify it to be suitable for our scheme.

- **Gen**(1^k): Payee B_i ($i \in \{1, 2, \dots, n\}$) generates key-pairs.
 1. Generate signing key-pair $(pk_{S_{B_i}}, sk_{S_{B_i}}) \leftarrow Gen'(1^k)$, encryption key-pair $(pk_{E_{B_i}}, sk_{E_{B_i}}) \leftarrow Gen(1^k)$
 2. Output public key $pk_{B_i} = (pk_{S_{B_i}}, pk_{E_{B_i}})$, secret key $sk_{B_i} = (sk_{S_{B_i}}, sk_{E_{B_i}})$.
- **Sign** $_{sk_{B_i}}$ (state,R): Payee B_i signs message *state* with secret sk_{B_i} in ring $R = \{pk_{B_1}, pk_{B_2}, \dots, pk_{B_n}\}$
 1. Set $R_E := \{pk_{E_{B_1}}, \dots, pk_{E_{B_n}}\}$, $state' = state|R$, where “|” denotes concatenation. B_i computes the signature $\sigma'_i \leftarrow Sign'_{sk_{S_{B_i}}}(state^*)$
 2. Choose random coins $\omega_1, \dots, \omega_n$: (1) computes $c_i = Enc_{R_E}^*(\sigma'_i, \omega_i)$ and (2) for $j \in \{1, \dots, n\} \setminus \{i\}$, computes $c_j = Enc_{R_E}^*(0^{|\sigma'_i|}, \omega_j)$
 3. For $i \in [n]$, let x_i denote the statement: “ $(pk_{S_{B_i}}, state^*, R_E, c_i) \in L$ ”, let $x := \bigvee_{i=1}^n x_i$ Compute the proof $\pi \leftarrow \mathcal{P}_r(x, (\sigma'_i, \omega_i))$
 4. The signature is $\sigma = (c_1, \dots, c_n, \pi)$ and return $cert := (state, \sigma)$.
- **Vrfy** $_R(cert)$: Payee B_j verifies *cert* in ring R
 1. Parse *cert* as $(state, \sigma)$, and set $state' := state|R$, $R_E := \{pk_{E_{B_1}}, \dots, pk_{E_{B_n}}\}$ and $\sigma := (c_1, \dots, c_n, \pi)$
 2. For $i \in [n]$, let x_i denote the statement: “ $(pk_{S_{B_i}}, state^*, R_E, c_i) \in L$ ” and set $x := \bigvee_{i=1}^n x_i$
 3. Output $V_r(x, \pi)$.

References

1. Bitcoinj: Working with micropayment channels. (2013). <https://bitcoinj.github.io/working-with-micropayments>
2. Baldimtsi, F., Chase, M., Fuchsbauer, G., Kohlweiss, M.: Anonymous transferable E-Cash. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 101–124. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_5
3. Bender, A., Katz, J., Morselli, R.: Ring signatures: stronger definitions, and constructions without random oracles. *J. Cryptol.* **22**, 114–138 (2008)
4. 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
5. Chiesa, A., Green, M., Liu, J., Miao, P., Miers, I., Mishra, P.: Decentralized anonymous micropayments. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 609–642. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_21
6. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Pelc, A., Schwarzmann, A.A. (eds.) SSS 2015. LNCS, vol. 9212, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21741-3_1
7. Dwork, C., Naor, M.: Zaps and their applications. In: Symposium on Foundations of Computer Science, p. 283 (2000)
8. Glassman, S., Manasse, M., Abadi, M., Gauthier, P., Sobalvarro, P.: The millicent protocol for inexpensive electronic commerce (1995)
9. Hearn, M., S.J.: Bitcoin contracts (2012). <https://en.bitcoin.it/wiki/Contract>

10. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: Tumblebit: an untrusted bitcoin-compatible anonymous payment hub. In: Network and Distributed System Security Symposium (2017)
11. Heilman, E., Baldimtsi, F., Goldberg, S.: Blindly signed contracts: anonymous on-blockchain and off-blockchain bitcoin transactions. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 43–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_4
12. Micali, S., Rivest, R.L.: Micropayments revisited. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 149–163. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45760-7_11
13. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Consulted (2008)
14. Pedersen, T.P.: Electronic payments of small amounts **24**(495), 59–68 (1996)
15. PeterTodd: near-zero fee transactions with hub-and-spoke micropayments (2014)
16. Poon, J., Dryja, T.: The bitcoin lightning network: SCALABLE off-chain instant payments
17. Rivest, R.L.: Electronic lottery tickets as micropayments. In: Hirschfeld, R. (ed.) FC 1997. LNCS, vol. 1318, pp. 307–314. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63594-7_87
18. Rivest, R.L., Shamir, A.: PayWord and MicroMint: two simple micropayment schemes. In: Lomas, M. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 69–87. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62494-5_6
19. Ruffing, T., Kate, A.: Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 219–230 (2015)
20. Shelat, A., Shelat, A.: Micropayments for decentralized currencies. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 207–218 (2015)
21. Wheeler, D.: Transactions using bets. In: Lomas, M. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 89–92. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62494-5_7



Revisiting Anonymous Two-Factor Authentication Schemes for Multi-server Environment

Ping Wang^{1,2,3,4}, Zijian Zhang^{1,3}, and Ding Wang^{2(✉)}

¹ School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen 518055, China

² School of EECS, Peking University, Beijing 100871, China

³ National Engineering Research Center for Software Engineering, Beijing, China

⁴ School of Software and Microelectronics, Peking University, Beijing 100260, China
{pwang, zhangzj, wangdingg}@pku.edu.cn

Abstract. Revealing the security flaws of existing cryptographic protocols is the key to understanding how to achieve better security. At ICICS'17, Xu *et al.* proposed an efficient two-factor authentication scheme for multi-server environment to cope with the vulnerabilities in Amin *et al.*'s scheme. However, in this paper, we reveal that Xu's new scheme actually is as vulnerable as Amin *et al.*'s scheme: anyone can impersonate any legitimate user. At FC'17, Wu *et al.* also developed an improvement over Irshad *et al.*'s scheme and this improved scheme is alleged to be practical and have a number of appealing merits. Yet, Wu *et al.*'s scheme still fails to achieve truly two-factor security (which is the most important goal of a two-factor scheme), and the leakage of a session-specific parameter will lead to the leakage of the user's long-term secret key.

Besides security, efficiency is another great concern. Recently, Leu-Hsieh showed that Lee *et al.*'s two-factor scheme fails to achieve truly two-factor security, and further suggested an enhanced anonymous scheme which is claimed to be robust against various attacks, while only using lightweight symmetric-key techniques. In this work, we show that Leu-Hsieh's enhanced scheme still fails to achieve truly two-factor security once again. Moreover, it cannot preserve user privacy. Our results invalidate any use of these three schemes for practical applications without further improvement, and underscore some new challenges (e.g., attacks arising from the leakage of session-specific parameters and from malicious insiders) in designing practical password authentication schemes.

Keywords: Password authentication · User anonymity
Smart card loss attack · Truly two-factor security

1 Introduction

User authentication plays a crucial part in ensuring that resources and services at the remote server can only be accessed by legitimate parties. In 1991, Chang *et al.* [5] suggested the first two-factor authentication scheme based on passwords and smart cards, and this influential study has given rise to a series of enhanced proposals with each diversified in aspects of usability [23], security [14] efficiency [9] and anonymity [20].

However, most of these schemes are designed for the single-server architecture, which means that the user needs to memorize n pairs of identity and password to login n different service servers. As the number of services increases rapidly, e.g., common users generally have 25–67 such pairs [12]. This is a great burden for use to maintain (memorize) such an amount of password pairs. Accordingly, a number of two-factor authentication protocols for multi-server architecture has been developed [13, 19, 22].

In a two-factor scheme¹ for multi-server architecture, there are three participants (i.e. a set of users, a control server CS and a set of service servers) involved. User U can login any service server under the same control server by using the same (identity, password)-pair. User U holds a memorable password and a smart card stored with some initial security parameters; The servers (including CS and service server S) only need to keep some secret key material of the system (but not the user). Since there is no need to keep a table with password-related verification information on the server side, the server is free from the threat of password dataset leaks and ameliorated from the burden of maintaining a large password dataset. This feature makes this type of schemes rather desirable, considering the incessant leakages of password databases from large websites [1].

The most important security goal of a two-factor authentication scheme is the so-called “two-factor security” [14]. This security concept essentially means that only the user that has the smart card as well as knows the correct password can be verified by the server. Nevertheless, past research [6, 16, 17, 20] have, again and again, proved that designing a two-factor authentication scheme with “two-factor security” for single-server architecture is a hard task, and the design of a truly two-factor scheme for multi-server architecture can only be harder.

In 2017, Amin *et al.* [4] developed an anonymous two-factor authentication scheme relying on the intractability of large integer factoring problem (i.e., RSA), and stated that their scheme is able to support “two-factor security” under the hypothesis that smart cards can be tampered. Later on, Xu *et al.* [22] found that Amin *et al.*’s scheme cannot resist against user impersonation attack if the parameters kept in the smart cards can be extracted, invalidating Amin *et al.*’s claim of ensuring “two-factor security”. Accordingly, Xu *et al.* [22] further proposed a new scheme based on the same cryptographic primitive (i.e., RSA) at ICICS’17. In addition, their scheme was “proved secure” in the random oracle

¹ As with [17, 23], in this work we mainly consider the most typical kind of two-factor schemes that are composed of password and smart card.

model. Surprisingly, we find that Xu *et al.*'s scheme [22] is subject to a damaging security hole: anyone can impersonate any legitimate user.

At FC'17, Wu *et al.* [19] demonstrated that various security drawbacks existed in both Irshad *et al.*'s [7] and Zhu's [24] schemes. More specifically, Irshad *et al.*'s scheme is vulnerable to stolen-verifier attack and insider attack, and provides no user anonymity; Zhu's scheme suffers from insider attack, provides no user anonymity, and has the de-synchronization problem in case the malicious attacker \mathcal{M} simply modify the third message flow. Wu *et al.* [19] also put forward an improved scheme and argued that their scheme is robust under the condition that the sensitive data in smart card has been revealed by \mathcal{M} . It should be noted that, recent rapid developments in side-channel attacks have proved that the sensitive information stored in general commercial smart cards could be extracted by power analysis [11] or reverse engineering [3]. Based on a weak yet realistic assumption, Wu *et al.*'s scheme [19] appears very practical.

However, as we will show, this scheme is prone to a much more serious problem (i.e., no truly two-factor security) than the original schemes (i.e., Irshad *et al.*'s [7] and Zhu's [24] schemes). Besides, Wu *et al.*'s scheme will leak the user's long-term secret key once a session-specific parameter is leaked. This is rather undesirable, because session-specific data is often less well protected than long-term keys, and the leakage of the former should not affect the latter. Our attack highlight the challenges arising from the leakage of session-specific data.

Besides robust security guarantees, protocol efficiency is also an important concern due to the resource-constrained nature of user devices. Leu-Hsieh [9] presented an anonymous two-factor scheme, which is claimed to ensure user privacy and robust security while only requiring a few lightweight hash operations. Unlike their claims, we show that their scheme still cannot provide truly two-factor security and user anonymity. In addition, forward secrecy cannot be attained. We note that Maitra *et al.* [10] have also analyzed Leu-Hsieh's scheme and presented some attacks, but their attacks are different from ours. Besides, Maitra *et al.* [10] further gave an improved scheme, which suffers some critical issues as pointed out in [13].

2 Adversary Models

Since a series of influential work [17, 21, 23], generally three assumptions are made about \mathcal{M} 's capabilities against two-factor authentication.

Assumption 1. \mathcal{M} completely manipulates the public channel (e.g., eavesdrop, delete, insert, modify or block any transcripts).

Assumption 2. \mathcal{M} can somehow obtain the victim's smart card and exploit side-channel attacks [3, 11] to extract sensitive data from the card memory.

Assumption 3. Users' passwords are selected from a very constrained space and \mathcal{M} can brutal force it. To increase usability, most schemes (e.g., the ones in [14, 20, 23]) allow the users to choose passwords at their discretion during registration

phase or password change phase. Generally, human beings are only capable of memorizing 5–7 different passwords, and tend to select popular passwords, use personal info to build passwords and reuse passwords. Therefore, user-chosen passwords follow the Zipf’s law [15] and come from a small space.

Note that, if all *Assumptions* 2 and 3 hold at the same time, then the attacker (with no need of other abilities) is able to impersonate any victim user and can trivially breach any scheme. Thus, it is common practice to do *not* assume that the attacker acquires a victim user’s both (all) authentication factors when analyzing security [6, 17, 20].

Also note that, an attacker might be an insider of the system, and it is practical for her to obtain both her own card and password. As shown in Sect. 5.2.3 of [16], such an attacker is really powerful and poses great threat to the security of the system. Overlooking the threats from this kind of attacker is likely to open large security loopholes. Many previous password authentication schemes employing smart cards (e.g., [8, 18, 21]) fail to achieve “two-factor security” or user un-traceability, when confronted with such a malicious insider. In this work, special attention are devoted to this kind of attacker and we show its perniciousness.

According to the abilities that are exploited by an attacker to launch a attack, four types of attackers can be further classified as follows:

- (I) **Basic attacker.** This attacker is only based on Assumption 1.
- (II) **Attacker with the target user’s smart card.** This attacker rest on the Assumption 1 and 2.
- (III) **Attacker with the target user’s password.** This attacker rests on the Assumption 1 and 3.
- (IV) **Attacker with her own smart card and password.** This attacker rests on the Assumption 1–3, being a malicious insider.

It is evident that the *basic attacker* is with the least capabilities, while the three remaining attackers are all realistic according to the aforementioned discussions. Consequently, any scheme aiming for practical use shall be able to withstand these four attackers. All the three schemes examined in this work are claimed to be secure under the above three assumptions. Actually, as we will show, this is not the case.

3 Cryptanalysis of Xu et al.’s Scheme

We first review Xu et al.’s scheme [22] proposed at ICICS’17, and then show that it is subject to a damaging security flaw: anyone can impersonate any legitimate user without guessing the victim’s password or obtaining the victim’s device.

3.1 A Brief Review of Xu et al.’s Scheme

Xu et al.’s scheme [22] is composed of four phases. For simplicity, the notations employed throughout this paper are listed in Table 1; We will comply with the abbreviations in Xu et al.’s scheme closely.

Table 1. Notations and abbreviations

Symbol	Description	Symbol	Description
U_i	i^{th} user	S_j	j^{th} server
RC	The register center	S_x	The foreign server
S_y	The home server	k_{xy}	Secret key shared by S_x and S_y
k_y	The secret key of S_y	\mathcal{M}	The malicious adversary
d	The secret key of RC	e	The public key of RC
ID_i	Identity of U_i	PW_i	Password of U_i
\Rightarrow	A secure channel	\oplus	Bitwise XOR operation
\rightarrow	A common channel	$h(\cdot)$	One-way hash function

Server Registration Phase. This phase proceeds as follows:

- Step 1. $S_j \Rightarrow RC: \{e_j, n_j, SID_j\}$. S_j computes $n_j = p_j \times q_j$, $\phi(n_j) = (p_j - 1)(q_j - 1)$ where both p and q are large prime numbers, then chooses a public key $e_j (1 < e_j < \phi(n_j))$ where $\gcd(\phi(n_j), e_j) = 1$, and computes $d_j \equiv e_j^{-1} \pmod{\phi(n_j)}$ as its private key.
- Step 2. $RC \Rightarrow S_j: \{Cer_j\}$. RC computes $Cer_j = h(e_j \parallel SID_j \parallel n_j)^d$.

User Registration Phase. This phase proceeds as follows:

- Step 1. $U_i \Rightarrow RC: \{ID_i\}$.
- Step 2. $RC \Rightarrow S_j: \{d_i\}$. RC computes $d_i = h(ID_i)^d \pmod{n_j}$.

Login and Authentication Phase. This phase proceeds as follows:

- Step 1. $U_i \rightarrow S_j$: a random number T_i .
- Step 2. $S_j \rightarrow U_i: \{e_j, n_j, Cer_j, A_j\}$ where $A_j = h(T_i)^{d_j}$.
- Step 3. $U_i \rightarrow S_j: \{PID_i, R_i, S_i, x\}$. If $Cer_j \pmod{n_j}$ equals $h(SID_j \parallel e_j \parallel n_j)$ and $A_j^{e_j}$ equals $h(T_i)$, U_i computes $PID_i = (ID_i \oplus a_i \parallel a_i)^{e_j} \pmod{n_j}$, $R_i = h(ID_i)^r \pmod{n_j}$, $x = h(m, R_i)$ and $S_i = d_i^{r-x}$ where a_i, r, m are three random numbers.
- Step 4. S_j computes $S_i^{e_j} = h(ID_i)^{r-x}$, $PID_i^{d_j} \pmod{n_j} = ID_i \oplus a_i \parallel a_i$, $ID'_i = ID_i \oplus a_i \parallel a_i$. If $S_i^{e_j} h(ID'_i)^x$ equals R_i , S_j authenticates U_i .

3.2 Flaws in Xu et al.’s Scheme

User Impersonation Attack. Xu et al. claimed that their “proposed scheme can provide proper mutual authentication”, but we show this is not the case: Anyone can impersonate any legitimate user without guessing password or accessing the victim’s device:

- Step 1.* \mathcal{M} chooses a random number $T_i \in_R (1, n_j]$;
- Step 2.* \mathcal{M} receives $\{e_j, n_j, Cer_j, A_j\}$ that comes from S_j ;

- Step 3.* \mathcal{M} chooses a random number $X \in_R (1, n_j]$;
Step 4. \mathcal{M} sets $S_i = X$, $R_i = X^{e_j} h(ID_i)^X$;
Step 5. $\mathcal{M} \rightarrow S_j: \{PID_i, R_i, S_i, x = X\}$, where PID_i is intercepted.

Note that the above attack will succeed, because $\{PID_i, R_i, S_i, x = X\}$ will be accepted by the the service server S_j . More specifically, according to attack Step 4, we have $S_i^{e_j} h(ID_i)^x = X^{e_j} h(ID_i)^X$, which equals R_i and passes Step 4 of login phase. This demonstrates that even a Type-I attacker (see Sect. 2) can completely break the scheme.

Poor Repairability. In Xu *et al.*'s scheme, there should be times that a user suspects (or realizes) that her smart card might be power analysed and the secret $d_i = h(ID_i)^d \bmod n$ has been leaked. However, even if U_i has detected this abnormality and changes her password to a new one, no means can be employed to deter \mathcal{M} from using the master secret d_i to login the server S_j . In other words, U_i cannot be easily repaired [17]. More detailedly, since $d_i = h(ID_i)^d \bmod n$ is uniquely defined by U_i 's identity ID_i and RC 's long-term private key d , RC is unable to update d_i for U_i unless either ID_i or d is updated. Nevertheless, because d is usually utilized for all legitimate users of the entire system rather than only one user U_i , it would be irrational and inefficient to change d to restore the security of a single user, i.e. U_i . Furthermore, since ID_i is typically bound with U_i in many application systems, it is also unreasonable to change ID_i to address the problem. In summary, the repairability of Xu *et al.*'s scheme constitutes a realistic issue.

4 Cryptanalysis of Wu *et al.*'s scheme

Here we first review Wu *et al.*'s scheme [19]. This scheme is an improvement over existing schemes aims to attain user anonymity lacked in [7, 24]. Wu *et al.*'s scheme can preserve user anonymity, however, we observe that it still remains feasible for an attacker to break "truly two-factor security". In addition, the scheme cannot provide sound repairability.

4.1 A Brief Review of Wu *et al.*'s scheme

Wu *et al.*'s scheme [19] is composed of four phases: initialization, registration, login and authentication, and one activity: password change. The notations and initial system parameters employed in Wu *et al.*'s scheme are same as employed in the scheme of Xu et al. (see Table 1).

Initialization Phase. Let k_{xy} ($1 \leq x, y \leq n, x \neq y$) be the common secret key of each pair of servers (S_x, S_y) ($x \neq y$), and s be their common parameter, k_y be the secret key of S_y .

User Registration. This phase proceeds as follows:

- Step 1. $U_i \Rightarrow S_y: \{ID_i, HPW_i\}$, where $HPW_i = h(PW_i \parallel b_i)$ and b is a random number.
- Step 2. $S_y \Rightarrow U_i: \{PID_i, B_1, B_2, s, h(\cdot)\}$. S_y selects PID_i , then computes: $B_{01} = h(PID_i \parallel k_y \parallel ID_{S_y})$, $B_1 = B_0 \oplus HPW_i$, $B_{02} = h(ID_i \parallel k_y \parallel ID_{S_y})$ and $B_2 = B_{02} \oplus h(ID_i \parallel HPW_i)$, and stores ID_i .
- Step 3. U_i inputs $(PID_i, B_1, B_2, B_3, s, h(\cdot))$ into mobile device, where $B_3 = b_i \oplus h(ID_i \parallel PW_i)$.

Login and Authentication Phase. This phase proceeds as follows:

- Step 1. $U_i \rightarrow S_x: M_1 = \{PID_i, C_1, C_2, C_3, C_5, SID_j, ID_{S_y}\}$. U_i inputs ID_i and PW_i , then the device calculates $b_i = B_3 \oplus h(ID_i \parallel PW_i)$ and $HPW_i = h(PW_i \parallel b_i)$, $C_1 = T_{r_U}(s)$, $C_2 = B_1 \oplus HPW_i \oplus N_U$, $C_3 = h(N_U) \oplus ID_i$, $C_4 = B_2 \oplus h(ID_i \parallel HPW_i)$ and $C_5 = h(C_1 \parallel N_U \parallel C_4)$, where r_U and N_U are two randomly chosen nonces.
- Step 2. $S_x \rightarrow S_y: M_2 = \{PID_i, C_1, C_2, C_3, C_5, C_6, C_7, SID_j\}$. S_x computes $C_6 = T_{r_{S_x}}(s)$ and $C_7 = h(C_6 \parallel k_{xy} \parallel SID_j)$, where r_{S_x} is a nonce.
- Step 3. There are further messages flows $S_y \rightarrow S_x: M_3 = \{C_8, C_9, C_{10}, C_{11}\}$ and $S_x \rightarrow U_i: M_4 = \{C_6, C_9 \sim C_{12}\}$, but they have little relevance to our discussions and are omitted.

4.2 Flaws in Wu *et al.*'s scheme

We now show the flaws of Wu *et al.*'s scheme [19]. Recall that the three assumptions listed in Sect. 2 are also explicitly made when Wu *et al.* analyzing Irshad *et al.*'s [7] and Zhu's [24] schemes.

Smart Card Loss Attack. Based on Wu *et al.*'s own security assumptions (i.e., the three ones in Sect. 2), we now cryptanalyze the security provisions of their scheme. More specifically, in what follows we assume that \mathcal{M} can extract the private data $\{B_1, B_2, B_3, h(\cdot)\}$ kept in U_i 's smart card, and can also eavesdrop the messages $\{PID_i, C_1, C_2, C_3, C_5, SID_j, ID_{S_y}\}$ exchanged between the parties. \mathcal{M} obtains U_i 's password PW_i as follows:

- Step 1.* Guesses the value of ID_i to be ID_i^* from dictionary space \mathcal{D}_{id} and the value of PW_i to be PW_i^* from dictionary space \mathcal{D}_{pw} ;
- Step 2.* Computes $b_i^* = B_3 \oplus h(ID_i^* \parallel PW_i^*)$, where B_3 is revealed from U_i 's card;
- Step 3.* Computes $N_u^* = C_2 \oplus B_1 \oplus h(PW_i^* \parallel b_i^*)$, where C_2 is intercepted from the channel and B_1 is revealed from U_i 's card;
- Step 4.* Computes $C_3^* = h(N_u^*) \oplus ID_i^*$;
- Step 5.* Verifies the correctness of (ID_i^*, PW_i^*) by checking if C_3^* equals the intercepted C_3 ;
- Step 6.* Repeats Steps 1–5 until the right (ID_i^*, PW_i^*) is found.

The time complexity of the attack is $\mathcal{O}(|\mathcal{D}_{id}| * |\mathcal{D}_{pw}| * 3T_H)$, where T_H is the running time for Hash operation. Recently, it has been found that user-chosen password follow the Zipf's law and the dictionary size is very restricted, e.g., $|\mathcal{D}_{id}| \leq |\mathcal{D}_{pw}| \leq 10^6$ [15]. Further, regarding the timings in Table 5 of [17], \mathcal{A} may figure out the password within 24.6 days on a common PC, or costs \$30.36 and spends 16.37h by using the Amazon EC2 C4.4X-large cloud computing service [2]. The above attack means that, once the smart card factor is breached, then the password factor will also be compromised. *This indicates that truly two-factor security cannot be achieved in Wu et al.'s scheme.*

Temporary Information Leakage Attack. As session-specific info are generally of large volume and deemed less sensitive than long-term secret keys, the former will be much less well protected than the latter and thus more easily leaked (e.g., through improper erasing, memory leakage or even poor implementations). Therefore, it is desirable that the security impact of the leakage of such session-specific info can be limited to just session-specific secret keys, but not the long-term secret keys.

However, in Wu et al.'s scheme [19], the leakage of session-specific information will make the long-term secret key dangerous:

- Step 1.* \mathcal{M} somehow obtains the session-specific N_u during one session;
Step 2. Computes $B_{01} = C_2 \oplus N_u = h(PID_i \parallel k_y \parallel ID_{S_y})$.

Note that, $B_{01} = h(PID_i \parallel k_y \parallel ID_{S_y})$ is just U_i 's long-term authenticator. After obtaining B_{01} , \mathcal{M} can further guess U_i 's passwords:

- Step 1.* Computes $ID_i = C_3 \oplus h(N_u)$, where C_3 is from open channel;
Step 2. Guesses the value of PW_i to be PW_i^* from space \mathcal{D}_{pw} ;
Step 3. Computes $C_2^* = B_{01} \oplus h(PW_i^* \parallel b_i^*) \oplus N_u$, where B_{01} is obtained as shown above;
Step 4. Verifies the correctness of PW_i^* by comparing if C_2^* equals C_2 ;
Step 5. Repeats Step 1–4 until the right value of PW_i^* is found.

The time complexity is $\mathcal{O}(|\mathcal{D}_{pw}| * 2T_H)$, which can be completed in 1.39s on a common PC according to the timings that $T_H \approx 0.693\mu s$ (see Table 5 of [17]). That is, the leakage of session-specific info will lead to the leakage of user identity and passwords. This is rather dangerous.

5 Cryptanalysis of Leu-Hsieh's Scheme

We now cryptanalyze Leu-Hsieh's scheme [9].

5.1 Review of Leu-Hsieh's Scheme

Due to space constraints, the details of the scheme are referred to [9].

5.2 Flaws in Leu-Hsieh's Scheme

No User Anonymity. With the concern of user privacy rising rapidly nowadays, user anonymity is becoming a primary feature to be considered in the design of authentication protocols, especially in wireless environments. In Leu-Hsieh's scheme, the exchanged messages are different in every session due to the use of fresh random nonces and user identity is dynamic in every session by hiding the true identity ID_i into shadow identities CID_i . In this way, anonymity service is claimed to be provided in [9] by arguing that an attacker \mathcal{M} "cannot distinguish between different sessions corresponding to a certain user and cannot obtain any clue to the real identity." However, Leu-Hsieh's scheme fails to consider that the attacker \mathcal{M} may be a malicious insider (i.e., a legitimate but malicious user – a type IV attacker, see Sect. 2). In the following we show that such a type IV attacker \mathcal{M} is able to breach U_i 's untraceability as follows:

Step 1. \mathcal{M} eavesdrops a login request $\{P_{ij}, N_i\}$ sent by U_i ;

Step 2. \mathcal{M} calculates $T_i = P_{ij} \oplus h(h(y) \| N_i \| SID_j)$, where $h(y)$ is shared among all users and service servers.

Note that, $T_i = h(R_i \| x)$ is specific to U_i and static in all of user U_i 's login sessions, and thus it can be used to link the different session participated by U_i , breaching the user untraceability.

The above procedure shows that, a type IV attacker (see Sect. 2) is capable of disclosing the activity of any legitimate user in the system without the sensitive info from user's smart card. Instead, \mathcal{M} only needs the sensitive info from her own knowledge. This is a much weaker condition as compared with the condition that \mathcal{M} needs the sensitive info from U_i 's smart card. This is contrary to Leu-Hsieh's claim that \mathcal{M} "cannot obtain any clue to the real identity." Thus, their scheme cannot preserve user anonymity and is not a true dynamic-ID based scheme. Our attack highlights the seriousness of threat arising from malicious insiders.

Smart Card Loss Attack I. We show that once \mathcal{M} obtains U_i 's smart card, a Type-I attacker \mathcal{M} can obtain U_i 's password PW_i as follows:

Step 1. \mathcal{M} extracts $\{V_i, H_i, h(\cdot)\}$ from U_i 's smart card.

Step 2. \mathcal{M} picks a candidate PW_i^* from the password dictionary \mathcal{D}_{pw} , and a candidate ID_i^* from the identity dictionary \mathcal{D}_{id} .

Step 3. \mathcal{M} computes $T_i^* = V_i \oplus h(ID_i^* \| h(b \| PW_i^*))$;

Step 4. \mathcal{M} computes $H_i^* = h(T_i^*)$;

Step 5. \mathcal{M} examines the validity of PW_i^* by comparing if the computed H_i^* is equal to H_i which is extracted from the card memory.

Step 6. \mathcal{M} goes to Step 2 until the right PW_i is obtained.

The time complexity is $\mathcal{O}(|\mathcal{D}_{id}| * |\mathcal{D}_{pw}| * (2T_H + T_X))$. Based on the results in [17], this attack is able to be carried out in a few days on a common computer. This attack has been given extensive attention in the literature [16, 17]. As shown in [13], Maitra *et al.*'s scheme [10], an improvement of Leu-Hsieh's scheme [9], suffers exactly the same issue.

Smart Card Loss Attack II. We further show that a Type-I attacker \mathcal{M} can obtain U_i 's password PW_i via another attacking procedure as follows:

Step 1. \mathcal{M} extracts $\{B_i, Z_i, V_i, b, h(\cdot)\}$ from U_i 's smart card by side channel attacks [3, 11].

Step 2. \mathcal{M} picks a candidate PW_i^* from the password dictionary \mathcal{D}_{pw} , and a candidate ID_i^* from the identity dictionary \mathcal{D}_{id} .

Step 3. \mathcal{M} computes $R_i^* = Z_i \oplus ID_i^* \oplus h(b\|PW_i^*)$;

Step 4. \mathcal{M} computes $O_i^* = h(b\|PW_i^*) \oplus ID_i^* \oplus R_i^*$;

Step 5. \mathcal{M} computes $A_i^* = h(T_i\|h(y)\|N_i)$, where N_i is from the open channel and $h(y)$ from a legitimate yet curious user/server;

Step 6. \mathcal{M} computes $Q_i^* = h(O_i^*\|A_i^*\|N_i)$;

Step 7. \mathcal{M} examines the validity of PW_i^* by comparing if the computed Q_i^* equals Q_i which is intercepted from the open channel.

Step 8. \mathcal{M} goes to Step 2 until the right PW_i is obtained.

The time complexity is $\mathcal{O}(|\mathcal{D}_{id}| * |\mathcal{D}_{pw}| * (4T_H + 3T_X))$. It can be carried out in a few days on a common computer according to the timings in [17]. Note that, our attack involves the parameter $h(y)$ but not $h(x\|y)$, which is different from Maitra et al.'s attack (see Sect. 6.3 of [10]). When compared with the above "smart card loss attack I", this attack is less effective as it requires that \mathcal{M} colludes with a malicious insider. Still, this attack invalidates the claim of achieving truly two-factor security in [9].

No Forward Secrecy. When analyzing their scheme, Leu and Hsieh do not consider (mention) forward secrecy. We now show that this desirable property cannot be preserved: Supposing an attacker \mathcal{M} manages to obtain the long-term keys $h(y)$ and $h(x\|y)$ from a compromised/malicious service server and eavesdropped the messages $\{CID_i, P_{ij}, Q_i, N_i, N_j\}$ exchanged during U_i and S authentication process from the public channel. For convenience of presentation, assume it is U_i 's m th login. \mathcal{M} can calculate U_i and S 's session key during the j th communication as follows:

Step 1. \mathcal{M} calculates $T_i = P_{ij} \oplus h(h(y)\|N_i\|SID_j)$, $A_i = h(T_i\|h(y)\|N_i)$, where $\{P_{ij}, N_i\}$ is from the open channel.

Step 2. \mathcal{M} computes $h(b \oplus PW_i \oplus R_i) = CID_i \oplus h(T_i\|A_i\|N_i)$ and $O_i = h(h(b \oplus PW_i \oplus R_i)\|h(x\|y))$, where $\{CID_i, N_i\}$ is intercepted.

Step 3. \mathcal{M} calculates $SK^m = h(O_i\|N_i\|N_j\|A_i\|SID_j)$, where $\{N_i, N_j\}$ is from the open channel.

Once the session key SK^m leaks, the entire m th communication will be leaked to \mathcal{M} . Maitra *et al.*'s scheme [10] suffers exactly the same issue.

6 Conclusion

Considerable efforts have been spent on designing an efficient, secure and privacy-preserving two-factor authentication scheme for multi-server environments under

the assumption that smart cards can be extracted. Very recently, Xu *et al.*, Wu *et al.* and Leu-Hsieh made three new attempts. However, through systematic evaluation we reveal that all of them are still subject to various serious defects. Most importantly, our results underscore some new challenges (e.g., attacks arising from the leakage of session-specific information and from malicious insiders) in devising a practical two-factor authentication scheme for multi-server environments.

Acknowledgments. We thank the anonymous reviewers. This research was in part supported by the National Key Research and Development Plan under Grants Nos. 2016YFB0800600 and 2017YFB1200700, and by the National Natural Science Foundation of China (NSFC) under Grant No. 61472016.

References

1. All Data Breach Sources, May 2018. <https://breachalarm.com/all-sources>
2. Amazon elastic compute cloud (Amazon EC2) (2018). <https://aws.amazon.com/ec2/pricing/>
3. Amiel, F., Feix, B., Villegas, K.: Power analysis for secret recovering and reverse engineering of public key algorithms. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 110–125. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77360-3_8
4. Amin, R., Islam, S., Khan, M.K., Karati, A., Giri, D., Kumari, S.: A two-factor rsa-based robust authentication system for multiserver environments. Secur. Commun. Netw. (2017). <https://doi.org/10.1155/2017/5989151>
5. Chang, C.C., Wu, T.C.: Remote password authentication with smart cards. IEE Proc.-Comput. Digit. Tech. **138**(3), 165–168 (1991)
6. Huang, X., Chen, X., Li, J., Xiang, Y., Xu, L.: Further observations on smart-card-based password-authenticated key agreement in distributed systems. IEEE Trans. Para. Distrib. Syst. **25**(7), 1767–1775 (2014)
7. Irshad, A., Ahmad, H.F., Alzahrani, B.A., Sher, M., Chaudhry, S.A.: An efficient and anonymous chaotic map based authenticated key agreement for multi-server architecture. KSII Trans. Internet Inf. Syst. **10**(12), 5572–5595 (2016)
8. Kumari, S., Khan, M.K., Li, X.: An improved remote user authentication scheme with key agreement. Comput. Electr. Eng. **40**(6), 97–112 (2014)
9. Leu, J.S., Hsieh, W.B.: Efficient and secure dynamic id-based remote user authentication scheme for distributed systems using smart cards. IET Inform. Secur. **8**(2), 104–113 (2014)
10. Maitra, T., Islam, S.H., Amin, R., Giri, D., Khan, M.K., Kumar, N.: An enhanced multi-server authentication protocol using password and smart-card: cryptanalysis and design. Secur. Commun. Netw. **9**(17), 4615–4638 (2016)
11. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Comput. **51**(5), 541–552 (2002)
12. Stobert, E., Biddle, R.: The password life cycle. ACM Trans. Priva. Secur. **21**(3), 13 (2018)
13. Wang, C., Xu, G., Li, W.: A secure and anonymous two-factor authentication protocol in multiserver environment. Secur. Commun. Netw. (2018). <https://doi.org/10.1155/2018/9062675>

14. Wang, D., Wang, P.: Two birds with one stone: two-factor authentication with security beyond conventional bound. *IEEE Trans. Depend. Secur. Comput.* **15**(4), 708–772 (2018)
15. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf’s law in passwords. *IEEE Trans. Inform. Foren. Secur.* **12**(11), 2776–2791 (2017)
16. Wang, D., Gu, Q., Cheng, H., Wang, P.: The request for better measurement: a comparative evaluation of two-factor authentication schemes. In: *Proceedings of ACM ASIACCS 2016*, pp. 475–486 (2016)
17. Wang, D., He, D., Wang, P., Chu, C.H.: Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment. *IEEE Trans. Depend. Secur. Comput.* **12**(4), 428–442 (2015)
18. Wang, Y.: Password protected smart card and memory stick authentication against off-line dictionary attacks. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) *SEC 2012. IAICT*, vol. 376, pp. 489–500. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30436-1_40
19. Wu, F., Xu, L., Li, X.: A new chaotic map-based authentication and key agreement scheme with user anonymity for multi-server environment. In: Hung, J.C., Yen, N.Y., Hui, L. (eds.) *FC 2017. LNEE*, vol. 464, pp. 335–344. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7398-4_35
20. Xie, Q., Wong, D.S., Wang, G., et al.: Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model. *IEEE Trans. Inform. Foren. Secur.* **12**(6), 1382–1392 (2017)
21. Xu, J., Zhu, W., Feng, D.: An improved smart card based password authentication scheme with provable security. *Comput. Stand. Inter.* **31**(4), 723–728 (2009)
22. Xu, Z., He, D., Huang, X.: Secure and efficient two-factor authentication protocol using RSA signature for multi-server environments. In: Qing, S., Mitchell, C., Chen, L., Liu, D. (eds.) *ICICS 2017. LNCS*, vol. 10631, pp. 595–605. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89500-0_51
23. Yang, G., Wong, D., Wang, H., Deng, X.: Two-factor mutual authentication based on smart cards and passwords. *J. Comput. Syst. Sci.* **74**(7), 1160–1172 (2008)
24. Zhu, H.: Flexible and password-authenticated key agreement scheme based on chaotic maps for multiple servers to server architecture. *Wirel. Personal Commun.* **82**(3), 1697–1718 (2015)

Author Index

- An, Wei 41
Anada, Hiroaki 530
Arita, Seiko 530
Aura, Tuomas 781
- Bahler, Lisa 497
Barengi, Alessandro 177
Baum, Carsten 303
Benaissa, Mohammed 161
Bkakra, Anis 232
Bösch, Christoph 20, 459
Bowen, Jonathan P. 192
Breuer, Peter T. 192
Bui, Thanh 781
- Cao, Zhenfu 215, 377
Chao, HanChieh 629
Chen, Lin 248
Chen, Sheng 248
Chen, Shiping 358
Chen, Xiaofeng 445
Chen, Zhili 248
Cheng, Haitao 513, 756
Cheng, Wangzhao 767
Cheng, Zhenyu 342
Chowdhury, Abdullahi 696
Coan, Brian 497
Cui, Xiang 57
Cuppens, Frédéric 232, 732
Cuppens-Boulahia, Nora 232, 732
- Dai, Rui 92
Di Crescenzo, Giovanni 497
Diaz-Perez, Arturo 745
Dong, Jiankuo 142
Dong, Xiaolei 215, 377
- Guo, Shanqing 75
Guo, Yun 358
- Han, Xiaohui 584
Han, Ya 647
Hashim, Shakirah 161
- Hou, Lin 271
Hu, Fangning 629
Huang, Mingjiang 664
Huang, Tao 107
- Ji, Xiaoshu 732
- Kai, Chen 551
Kammüller, Florian 611
Kamruzzaman, Joarder 696
Kargl, Frank 20, 459
Karmakar, Gour 696
Kopp, Henning 20, 459
Koshiha, Takeshi 287
Kunihiro, Noboru 598
- Lara-Nino, Carlos Andres 745
Le Guernic, Gurvan 732
Li, Bingyu 767
Li, Guoqiang 358
Li, Hongda 793
Li, Huorong 767
Li, Qing 721
Li, Shuhao 685
Li, Xiangxue 513, 756
Li, Yongqiang 647
Liang, Yunong 377
Liming, Wang 551
Lin, Dongdai 271
Lin, Huang 303
Lin, Jingqiang 142, 767
Lin, Xiaodong 3
Liu, Guangqi 584
Liu, Renzhang 271
Liu, Xiaoli 629
Liu, Zhiming 192
Lu, Yao 598
Luo, Xi 41
Lv, Heyang 57
- Ma, Hui 598
Mainardi, Nicholas 177
Man, Jianping 92

- Mimura, Mamoru 708
 Molva, Refik 393
 Morales-Sandoval, Miguel 745
- Ni, Peifang 793
 Ning, Zhang 551
- Oechsner, Sabine 303
 Önen, Melek 393
- Palomar, Esther 192
 Pan, Dongxue 793
 Pelosi, Gerardo 177
 Peng, Chengwei 685
 Peng, Liqiang 598
 Perera, Maharage Nisansala Sevewandi 287
 Peter, Andreas 20
- Qian, Haifeng 513, 756
 Qiu, Tian 271
- Rao, Wei 325
- Saha, Tapash 696
 Shen, Jiachen 377
 Sun, Daniel 358
- Tanaka, Hidema 708
 Tang, Dan 92
 Tang, Liu 92
 Tao, Xiaoling 445
 Tao, Yang 426
 Teşeleanu, George 124
 Tian, Miaomiao 248
 Tian, Zhihong 57
 Tu, Bibo 566, 721
- Van Rompay, Cédric 393
- Wang, Chenxu 325
 Wang, Chenyu 107
 Wang, Ding 805
 Wang, Jianfeng 445
- Wang, Lianhai 584
 Wang, Liming 41, 664
 Wang, Lin 721
 Wang, Mingsheng 647
 Wang, Ping 805
 Wang, Xin 409
 Wang, Xingfeng 478
 Wang, Ziyang 142
 Wu, Hongjun 107
- Xu, Chunxiang 3
 Xu, Jian 342
 Xu, Mingdi 629
 Xu, Shujiang 584
 Xu, Zhen 41
 Xue, Rui 409
- Yan, Di 513, 756
 Yang, Changsong 445
 Yin, Jie 57
 Yun, Kelly 409
 Yun, Xiaochun 342, 685
- Zeng, Yingpei 75
 Zhan, Sijia 92
 Zhang, Cong 629
 Zhang, Fan 629
 Zhang, Fangjiao 57
 Zhang, Kun 566
 Zhang, Rui 426, 598
 Zhang, Yan 664
 Zhang, Yongzheng 342, 685
 Zhang, Yuan 3
 Zhang, Zijian 805
 Zhao, Dawei 584
 Zhao, Liang 478
 Zhao, Zhiyuan 325
 Zhen, Xu 551
 Zheng, Fangyu 142
 Zheng, Meng 215
 Zhong, Hong 248
 Zhou, Jun 215
 Zhu, Min 566, 721