

Joseph K. Liu  
Ron Steinfeld (Eds.)

LNCS 9722

# Information Security and Privacy

21st Australasian Conference, ACISP 2016  
Melbourne, VIC, Australia, July 4–6, 2016  
Proceedings, Part I

1  
Part I

 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, Zürich, 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, 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>

Joseph K. Liu · Ron Steinfeld (Eds.)

# Information Security and Privacy

21st Australasian Conference, ACISP 2016  
Melbourne, VIC, Australia, July 4–6, 2016  
Proceedings, Part I

*Editors*

Joseph K. Liu  
Monash University  
Melbourne, VIC  
Australia

Ron Steinfeld  
Monash University  
Melbourne, VIC  
Australia

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-319-40252-9              ISBN 978-3-319-40253-6 (eBook)  
DOI 10.1007/978-3-319-40253-6

Library of Congress Control Number: 2015940421

LNCS Sublibrary: SL4 – Security and Cryptology

© Springer International Publishing Switzerland 2016

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.

Printed on acid-free paper

This Springer imprint is published by Springer Nature  
The registered company is Springer International Publishing AG Switzerland

# Preface

This volume contains the papers presented at ACISP 2016: the 21st Australasian Conference on Information Security and Privacy held during July 4–6, 2016, in Melbourne.

This year we received a record high number of submissions: 176. Each submission was reviewed by an average of 2.9 Program Committee members. The committee decided to accept 52 full papers and eight short papers. In addition, we also included eight invited papers in order to widen the coverage to different areas of cyber security such as smart cities security and bitcoin security. We would like to extend our sincere thanks to all authors who submitted their papers to ACISP 2016.

The program included two excellent and informative keynote addresses. One of them was from Prof. Elisa Bertino, of Purdue University in the USA. Another was from Prof. Chris Mitchell, of Royal Holloway, University of London in the UK. Furthermore, our program also included eight invited talks from eight international well-known researchers in cyber security. They were Prof. Ed Dawson from Queensland University of Technology, Australia; Prof. Willy Susilo from University of Wollongong, Australia; Prof. Xun Yi from RMIT, Australia; Prof. Yu Yu from Shanghai Jiao Tong University, China; Prof. Wenlei Zhou from Deakin University, Australia; Dr. Surya Nepal from Data61, Australia; Prof. Jinjun Chen from University of Technology Sydney, Australia; and Dr. Jonathan Oliver from Trend Micro, Australia.

We would like to thank the 86 Program Committee members (from 22 different countries) as well as the external reviewers for their volunteer work of reading and discussing the submissions. We also deeply thank the general chair, Prof. Yang Xiang, publication co-chairs, Dr. Dong Seong Kim and Dr. Kaitai Liang, publicity chair, Dr. Nalin Asanka, and the Web chair, Dr. Yu Wang. This conference would not have been successful without their great assistance. Last but not least, we would like to thank EasyChair for providing a user-friendly interface for us to manage all submissions and proceeding files.

July 2016

Joseph K. Liu  
Ron Steinfeld

# Organization

## Program Committee

Cristina Alcaraz	University of Malaga, Spain
Myrto Arapinis	University of Edinburgh, UK
Claudio Ardagna	Università degli Studi di Milano, Italy
David Aspinall	University of Edinburgh, UK
Giuseppe Ateniese	Sapienza University of Rome, Italy
Man Ho Au	Hong Kong Polytechnic University, HKSAR China
Joonsang Baek	Khalifa University of Science, Technology and Research, UAE
Zubair Baig	Edith Cowan University, Australia
Lynn Batten	Deakin University, Australia
Colin Boyd	Norwegian University of Science and Technology (NTNU), Norway
Serdar Boztas	RMIT University, Australia
Alvaro Cardenas	University of Texas at Dallas, USA
Aniello Castiglione	University of Salerno, Italy
Jinjun Chen	University of Technology Sydney, Australia
Liqun Chen	Hewlett-Packard Laboratories, UK
Xiaofeng Chen	Xidian University, China
Ray Cheung	City University of Hong Kong, HKSAR China
Kim-Kwang Raymond Choo	University of South Australia, Australia
Christophe Doche	Macquarie University, Australia
Ernest Foo	Queensland University of Technology, Australia
Steven Galbraith	Auckland University, New Zealand
David Galindo	SCYTL Secure Electronic Voting, Spain
Felix Gomez Marmol	NEC Laboratories Europe, Germany
Swee-Huay Heng	Multimedia University, Malaysia
Andreas Holzer	University of Toronto, Canada
Xinyi Huang	Fujian Normal University, China
Mitsugu Iwamoto	University of Electro-Communications, Japan
Sanjay Jha	University of New South Wales, Australia
Akinori Kawachi	The University of Tokushima, Japan
Dong Seong Kim	University of Canterbury, New Zealand
Howan Kim	Pusan National University, South Korea
Steve Kremer	Inria Nancy - Grand Est, France
Marina Krotofil	European Network for Cyber Security, The Netherlands

Noboru Kunihiro	The University of Tokyo, Japan
Mirosław Kutylowski	Wrocław University of Technology, Poland
Junzuo Lai	Jinan University, China
Gaëtan Leurent	Inria, France
Jin Li	Guangzhou University, China
Yingjiu Li	Singapore Management University, Singapore
Zhen Li	Institute for Infocomm Research, Singapore
Kaitai Liang	Aalto University, Finland
Joseph Liu	Monash University, Australia
Shengli Liu	Shanghai Jiao Tong University, China
Zhe Liu	University of Waterloo, Canada
Javier Lopez	University of Malaga, Spain
Jiqiang Lu	Institute for Infocomm Research, Singapore
Rongxing Lu	Nanyang Technological University, Singapore
Kazuhiko Minematsu	NEC Corporation, Japan
Chris Mitchell	Royal Holloway, University of London, UK
Yi Mu	University of Wollongong, Australia
Udaya Parampalli	The University of Melbourne, Australia
Mathias Payer	Purdue University, USA
Christian Payne	Murdoch University, Australia
Thomas Peyrin	Ingenico, Singapore
Josef Pieprzyk	Queensland University of Technology, Australia
Michalis Polychronakis	Columbia University, USA
Kui Ren	State University of New York at Buffalo, USA
Reza Reyhanitabar	NEC Laboratories Europe, Germany
Carsten Rudolph	Monash University, Australia
Sushmita Ruj	Indian Statistical Institute, India
Joerg Schwenk	Ruhr-Universität Bochum, Germany
Jun Shao	Zhejiang Gongshang University, China
Taeshik Shon	Ajou University, South Korea
Haya Shulman	Technische Universität Darmstadt, Germany
Anna Squicciarini	Penn State University, USA
Ron Steinfeld	Monash University, Australia
Chunhua Su	Osaka University, Japan
Willy Susilo	University of Wollongong, Australia
Shaohua Tang	South China University of Technology, China
Juan Tapiador	Universidad Carlos III de Madrid, Spain
Mahesh Tripunitara	University of Waterloo, Canada
Craig Valli	Edith Cowan University, Australia
Frederik Vercauteren	K.U. Leuven - ESAT/COSIC, Belgium
Triet D. Vo-Huu	Northeastern University, USA
Petros Wallden	The University of Edinburgh, UK
Cong Wang	City University of Hong Kong, HKSAR China
Yu Wang	Deakin University, Australia
Sheng Wen	Deakin University, Australia
Qianhong Wu	Beihang University, China



Guomin Yang  
 Yanjiang Yang  
 Wun-She Yap  
 Xun Yi  
 Yong Yu

Tsz-Hon Yuen  
 Aaram Yun

University of Wollongong, Australia  
 Huawei, Singapore  
 Universiti Tunku Abdul Rahman, Malaysia  
 RMIT University, Australia  
 University of Electronic Science and Technology,  
 China  
 Huawei, Singapore  
 Ulsan National Institute of Science and Technology,  
 South Korea

## Additional Reviewers

Alamer, Ahmed  
 Aono, Yoshinori  
 Behnia, Rouzbeh  
 Boura, Christina  
 Chattopadhyay, Anupam  
 Chen, Wei  
 Cheng, Yao  
 Chin, Ji-Jian  
 Cui, Hui  
 Dai, Tianxiang  
 El Ioini, Nabil  
 Gaudenzi, Filippo  
 Ghodosi, Hossein  
 Ghosh, Satrajit  
 Gotfryd, Karol  
 Guasch, Sandra  
 Han, Shuai  
 Hanzlik, Lucjan  
 He, Kai  
 He, Shuangyu  
 Higo, Haruna  
 Hirano, Takato  
 Hongjun, Wu  
 Imine, Abdessamad  
 Jahan, Mosarrat  
 Javali, Chitra  
 Kim, Jun Young  
 Kluczniak, Kamil  
 Koshiha, Takeshi  
 Lauer, Sebastian  
 Li, Fagen  
 Liang, Zhi  
 Liu, Weiran

Liu, Ximeng  
 Luykx, Atul  
 Majcher, Krzysztof  
 Morozov, Kirilina  
 Murphy, Sean  
 Myers, David  
 Nieto, Ana  
 Nishimaki, Ryo  
 Nuida, Koji  
 Peikert, Chris  
 Peters, Thomas  
 Rahman, Anisur  
 Reparaz, Oscar  
 Roenne, Peter  
 Saito, Teruo  
 Sakzad, Amin  
 Schneider, Thomas  
 Sengupta, Binanda  
 Seo, Hwajeong  
 Shani, Barak  
 Shibusaki, Kyoji  
 Sinha Roy, Sujoy  
 Späth, Christopher  
 Sun, Li  
 Sun, Shifeng  
 Szepieniec, Alan  
 Tan, Hailun  
 Tan, Syhyuan  
 Tso, Raylin  
 Velichkov, Vesselin  
 Vivek, Srinivas  
 Vizár, Damian  
 Wang, Jianfeng

Wang, Qin  
Wei, Xiaochao  
Yau, Wei-Chuen  
Ye, Jun  
Yu, Xingjie  
Yu, Zuoxia

Zhang, Lei  
Zhao, Chuan  
Zhao, Minghao  
Zheng, Haibin  
Zhong, Lin  
Zhou, Xiuwen

# Contents – Part I

## Invited Papers

I Know Where You All Are! Exploiting Mobile Social Apps for Large-Scale Location Privacy Probing . . . . .	3
<i>Shuang Zhao, Xiapu Luo, Bo Bai, Xiaobo Ma, Wei Zou, Xinliang Qiu, and Man Ho Au</i>	
MUSE: Towards Robust and Stealthy Mobile Botnets via Multiple Message Push Services . . . . .	20
<i>Wei Chen, Xiapu Luo, Chengyu Yin, Bin Xiao, Man Ho Au, and Yajuan Tang</i>	
A Survey on the Cyber Attacks Against Non-linear State Estimation in Smart Grids . . . . .	40
<i>Jingxuan Wang, Lucas C.K. Hui, S.M. Yiu, Xingmin Cui, Eric Ke Wang, and Junbin Fang</i>	
Towards Bitcoin Payment Networks . . . . .	57
<i>Patrick McCorry, Malte Möser, Siamak F. Shahandasti, and Feng Hao</i>	
Statistical Disclosure Control for Data Privacy Using Sequence of Generalised Linear Models . . . . .	77
<i>Min Cherng Lee, Robin Mitra, Emmanuel Lazaridis, An Chow Lai, Yong Kheng Goh, and Wun-She Yap</i>	
Energy-Efficient Elliptic Curve Cryptography for MSP430-Based Wireless Sensor Nodes . . . . .	94
<i>Zhe Liu, Johann Großschädl, Lin Li, and Qiuliang Xu</i>	

## National Security Infrastructure

A Comparison Study of Wireless Network Security in Several Australasian Cities and Suburbs . . . . .	115
<i>Alastair Nisbet and Andrew Woodward</i>	
On the Guessability of Resident Registration Numbers in South Korea . . . . .	128
<i>Youngbae Song, Hyounghick Kim, and Jun Ho Huh</i>	

**Social Network Security**

Towards Privacy-Preserving Data Mining in Online Social Networks:  
Distance-Grained and Item-Grained Differential Privacy. . . . . 141  
*Shen Yan, Shiran Pan, Yuhang Zhao, and Wen-Tao Zhu*

**Bitcoin Security**

Fair Client Puzzles from the Bitcoin Blockchain . . . . . 161  
*Colin Boyd and Christopher Carr*

**Statistical Privacy**

Privacy-Preserving  $k$ -Nearest Neighbour Query on Outsourced Database . . . . 181  
*Rui Xu, Kirill Morozov, Yanjiang Yang, Jianying Zhou,  
and Tsuyoshi Takagi*

Reversible Data Hiding for Encrypted Images Based on Statistical Learning . . . 198  
*Zhen Li and Wei Wu*

**Network Security**

An Ensemble Learning Approach for Addressing the Class Imbalance  
Problem in Twitter Spam Detection. . . . . 215  
*Shigang Liu, Yu Wang, Chao Chen, and Yang Xiang*

**Smart City Security**

Putting the User in Control of the Intelligent Transportation System . . . . . 231  
*Catalin Gosman, Tudor Cornea, Ciprian Dobre, Florin Pop,  
and Aniello Castiglione*

**Digital Forensics**

Exploring the Space of Digital Evidence – Position Paper . . . . . 249  
*Carsten Rudolph*

**Lightweight Security**

Towards Lightweight Anonymous Entity Authentication for IoT  
Applications. . . . . 265  
*Yanjiang Yang, Haibin Cai, Zhuo Wei, Haibing Lu,  
and Kim-Kwang Raymond Choo*

Hybrid MQ Signature for Embedded Device . . . . . 281  
*Shaohua Tang, Bo Lv, and Wuqiang Shen*

**Secure Batch Processing**

Batch Verifiable Computation with Public Verifiability for Outsourcing  
 Polynomials and Matrix Computations. . . . . 293  
*Yujuan Sun, Yu Yu, Xiangxue Li, Kai Zhang, Haifeng Qian,  
 and Yuan Zhou*

Accelerating Oblivious Transfer with Batch Multi-exponentiation . . . . . 310  
*Yang Sun, Qianhong Wu, Jingwen Liu, Jianwei Liu, Xinyi Huang,  
 Bo Qin, and Wei Hu*

**Pseudo Random/One-way Function**

CTM-sp: A Family of Cryptographic Hash Functions from Chaotic Tent  
 Maps. . . . . 329  
*Xun Yi, Xuechao Yang, Yong Feng, Fengling Han,  
 and Ron van Schyndel*

One-Key Compression Function Based MAC with Security Beyond  
 Birthday Bound . . . . . 343  
*Avijit Dutta, Mridul Nandi, and Goutam Paul*

**Cloud Storage Security**

Towards Efficient Fully Randomized Message-Locked Encryption . . . . . 361  
*Tao Jiang, Xiaofeng Chen, Qianhong Wu, Jianfeng Ma, Willy Susilo,  
 and Wenjing Lou*

Secure and Traceable Framework for Data Circulation. . . . . 376  
*Kaitai Liang, Atsuko Miyaji, and Chunhua Su*

Public Cloud Data Auditing with Practical Key Update and Zero  
 Knowledge Privacy . . . . . 389  
*Yong Yu, Yannan Li, Man Ho Au, Willy Susilo,  
 Kim-Kwang Raymond Choo, and Xinpeng Zhang*

**Password/QR Code Security**

Exploiting the Error Correction Mechanism in QR Codes for Secret Sharing . . . . 409  
*Yang-Wai Chow, Willy Susilo, Guomin Yang, James G. Phillips,  
 Ilung Pranata, and Ari Moesriami Barmawi*

Password Requirements Markup Language. . . . . 426  
*Moritz Horsch, Mario Schlipf, Johannes Braun,  
 and Johannes Buchmann*

**Functional Encryption and Attribute-Based Cryptosystem**

Leakage-Resilient Functional Encryption via Pair Encodings . . . . . 443  
*Zuoxia Yu, Man Ho Au, Qiuliang Xu, Rupeng Yang, and Jinguang Han*

Secret Handshakes with Dynamic Expressive Matching Policy . . . . . 461  
*Lin Hou, Junzuo Lai, and Lixian Liu*

Ciphertext-Policy Attribute-Based Encryption with Key-Delegation Abuse  
Resistance . . . . . 477  
*Yinhao Jiang, Willy Susilo, Yi Mu, and Fuchun Guo*

Chosen Ciphertext Secure Attribute-Based Encryption with Outsourced  
Decryption . . . . . 495  
*Cong Zuo, Jun Shao, Guiyi Wei, Mande Xie, and Min Ji*

Accountable Large-Universe Attribute-Based Encryption Supporting Any  
Monotone Access Structures . . . . . 509  
*Yinghui Zhang, Jin Li, Dong Zheng, Xiaofeng Chen, and Hui Li*

A Cloud-Based Access Control Scheme with User Revocation and  
Attribute Update . . . . . 525  
*Peng Zhang, Zehong Chen, Kaitai Liang, Shulan Wang, and Ting Wang*

**Author Index** . . . . . 541

## Contents – Part II

### Signature and Key Management

One-Round Strong Oblivious Signature-Based Envelope . . . . .	3
<i>Rongmao Chen, Yi Mu, Willy Susilo, Guomin Yang, Fuchun Guo, and Mingwu Zhang</i>	
Proxy Signature with Revocation . . . . .	21
<i>Shengmin Xu, Guomin Yang, Yi Mu, and Sha Ma</i>	
On the Relations Between Security Notions in Hierarchical Key Assignment Schemes for Dynamic Structures . . . . .	37
<i>Arcangelo Castiglione, Alfredo De Santis, Barbara Masucci, Francesco Palmieri, and Aniello Castiglione</i>	

### Public Key and Identity-Based Encryption

Content-Based Encryption . . . . .	57
<i>Xiaofen Wang and Yi Mu</i>	
Provably Secure Threshold Paillier Encryption Based on Hyperplane Geometry . . . . .	73
<i>Zhe Xia, Xiaoyun Yang, Min Xiao, and Debiao He</i>	
Identity-Based Group Encryption . . . . .	87
<i>Xiling Luo, Yili Ren, Jingwen Liu, Jiankun Hu, Weiran Liu, Zhen Wang, Wei Xu, and Qianhong Wu</i>	
Edit Distance Based Encryption and Its Application . . . . .	103
<i>Tran Viet Xuan Phuong, Guomin Yang, Willy Susilo, and Kaitai Liang</i>	
Proxy Re-encryption with Delegatable Verifiability . . . . .	120
<i>Xiaodong Lin and Rongxing Lu</i>	
Efficient Completely Non-Malleable and RKA Secure Public Key Encryptions . . . . .	134
<i>Shi-Feng Sun, Udaya Parampalli, Tsz Hon Yuen, Yu Yu, and Dawu Gu</i>	

**Searchable Encryption**

Verifiable Searchable Encryption with Aggregate Keys for Data Sharing in Outsourcing Storage . . . . . 153  
*Tong Li, Zheli Liu, Ping Li, Chunfu Jia, Zoe L. Jiang, and Jin Li*

Public Key Encryption with Authorized Keyword Search . . . . . 170  
*Peng Jiang, Yi Mu, Fuchun Guo, and Qiaoyan Wen*

Linear Encryption with Keyword Search . . . . . 187  
*Shiwei Zhang, Guomin Yang, and Yi Mu*

**Broadcast Encryption**

Generic Anonymous Identity-Based Broadcast Encryption with Chosen-Ciphertext Security . . . . . 207  
*Kai He, Jian Weng, Man Ho Au, Yijun Mao, and Robert H. Deng*

Anonymous Identity-Based Broadcast Encryption with Revocation for File Sharing . . . . . 223  
*Jianchang Lai, Yi Mu, Fuchun Guo, Willy Susilo, and Rongmao Chen*

**Mathematical Primitives**

Partial Key Exposure Attacks on RSA with Multiple Exponent Pairs . . . . . 243  
*Atsushi Takayasu and Noboru Kunihiro*

A New Attack on Three Variants of the RSA Cryptosystem . . . . . 258  
*Martin Bunder, Abderrahmane Nitaj, Willy Susilo, and Joseph Tonien*

Generalized Hardness Assumption for Self-bilinear Map with Auxiliary Information . . . . . 269  
*Takashi Yamakawa, Goichiro Hanaoka, and Noboru Kunihiro*

Deterministic Encoding into Twisted Edwards Curves . . . . . 285  
*Wei Yu, Kunpeng Wang, Bao Li, Xiaoyang He, and Song Tian*

**Symmetric Cipher**

Improved Rebound Attacks on AESQ: Core Permutation of CAESAR Candidate PAEQ . . . . . 301  
*Nasour Bagheri, Florian Mendel, and Yu Sasaki*

Efficient Beyond-Birthday-Bound-Secure Deterministic Authenticated Encryption with Minimal Stretch . . . . . 317  
*Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel*



Improved (related-key) Attacks on Round-Reduced KATAN-32/48/64 Based on the Extended Boomerang Framework . . . . .	333
<i>Jiageng Chen, Je Sen Teh, Chunhua Su, Azman Samsudin, and Junbin Fang</i>	
Authenticated Encryption with Small Stretch (or, How to Accelerate AERO) . . . . .	347
<i>Kazuhiko Minematsu</i>	
Impossible Differential Cryptanalysis of 14-Round Camellia-192 . . . . .	363
<i>Keting Jia and Ning Wang</i>	
Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA . . . . .	379
<i>Ling Song, Zhangjie Huang, and Qianqian Yang</i>	
On the Security of the LAC Authenticated Encryption Algorithm . . . . .	395
<i>Jiqiang Lu</i>	
Linear Hull Attack on Round-Reduced Simeck with Dynamic Key-Guessing Techniques . . . . .	409
<i>Lingyue Qin, Huaifeng Chen, and Xiaoyun Wang</i>	
<b>Short Papers-Public Key and Identity-Based Encryption</b>	
Reducing the Key Size of the SRP Encryption Scheme . . . . .	427
<i>Dung Hoang Duong, Albrecht Petzoldt, and Tsuyoshi Takagi</i>	
<b>Short Papers-Biometric Security</b>	
Biometric Access Control with High Dimensional Facial Features. . . . .	437
<i>Ying Han Pang, Ean Yee Khor, and Shih Yin Ooi</i>	
Security Analysis on Privacy-Preserving Cloud Aided Biometric Identification Schemes . . . . .	446
<i>Shiran Pan, Shen Yan, and Wen-Tao Zhu</i>	
<b>Short Papers-Digital Forensics</b>	
Interest Profiling for Security Monitoring and Forensic Investigation . . . . .	457
<i>Min Yang, Fei Xu, and Kam-Pui Chow</i>	
<b>Short Papers-National Security Infrastructure</b>	
Pseudonymous Signature on eIDAS Token – Implementation Based Privacy Threats. . . . .	467
<i>Mirosław Kutylowski, Lucjan Hanzlik, and Kamil Kluczniak</i>	

**Short Papers-Mobile Security**

A Feasible No-Root Approach on Android. . . . . 481  
*Yao Cheng, Yingjiu Li, and Robert H. Deng*

**Short Papers-Network Security**

Improved Classification of Known and Unknown Network Traffic Flows  
Using Semi-supervised Machine Learning . . . . . 493  
*Timothy Glennan, Christopher Leckie, and Sarah M. Erfani*

**Short Papers-Pseudo Random/One-way Function**

A Noiseless Key-Homomorphic PRF: Application on Distributed Storage  
Systems . . . . . 505  
*Jhordany Rodriguez Parra, Terence Chan, and Siu-Wai Ho*

**Author Index** . . . . . 515

## **Invited Papers**

# I Know Where You All Are! Exploiting Mobile Social Apps for Large-Scale Location Privacy Probing

Shuang Zhao<sup>1,5</sup>, Xiapu Luo<sup>3,4</sup>, Bo Bai<sup>1,5</sup>, Xiaobo Ma<sup>2,3,4</sup> (✉), Wei Zou<sup>1,5</sup>, Xinliang Qiu<sup>1,5</sup>, and Man Ho Au<sup>3,4</sup>

<sup>1</sup> Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

{zhaoshuang,baibo,zouwei,qiuxinliang}@iie.ac.cn

<sup>2</sup> MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an, China

<sup>3</sup> Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

xma.cs@xjtu.edu.cn

<sup>4</sup> The Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen, China

<sup>5</sup> Beijing Key Laboratory of Network Security and Protection Technology, Beijing, China

{csxluo,csallen}@comp.polyu.edu.hk

**Abstract.** Mobile social apps have been changing the way people interact with each other in the physical world. To help people extend their social networks, Location-Based Social Network (LBSN) apps (e.g., Wechat, SayHi, Momo) that encourage people to make friends with nearby strangers have gained their popularity recently. They provide a “Nearby” feature for a user to find other users near him/her. While seeing other users, the user, as well as his/her coarse-grained relative location, will also be visible in the “Nearby” feature of other users. Leveraging this observation, in this paper, we model the location probing attacks, and propose three approaches to perform large-scale such attacks on LBSN apps. Moreover, we apply the new approaches in the risk assessment of eight popular LBSN apps, each of which has millions of installation. The results demonstrate the severity of such attacks. More precisely, our approaches can collect a huge volume of users’ location information effectively and automatically, which can be exploited to invade users’ privacy. This study sheds light on the research of protecting users’ private location information.

## 1 Introduction

With the wide adoption of mobile devices with built-in positioning systems (e.g., GPS), LBSNs (Location-based Social Networks) are becoming increasingly popular, especially among the young. Nowadays, millions of people are using various LBSN apps to share interesting location-embedded information with others in their social networks, while simultaneously expanding their social networks with the new interdependency derived from their locations [1].

These LBSN apps can be roughly divided into two categories (I and II). LBSN apps of category I encourage users to share location-embedded information with their friends, such as Foursquare and Google+. Foursquare, which has achieved more than 55 million users worldwide since 2009 [2], allows users to *check-in* at some interesting places and then share the check-in locations with their friends. Google+, besides sharing check-in locations, even allows users to share their real-time locations with pre-specified users (e.g., their families) [3].

LBSN apps of category II concentrate on location-based social network discovery. Such LBSN apps allow users to search and interact with strangers around, and make new friends. Salient examples of this category include Wechat, Momo, SayHi, Skout and so forth. WeChat, which now has more than 540 million monthly active users around the world [4], has a feature called “Nearby”. This feature allows users to get a list of other users nearby as well as their coarse-grained relative locations. People can use this feature to discover strangers (and be discovered by others simultaneously), and then make friends with strangers of interest. Some apps (e.g., Facebook and Sina Weibo) that were not originally designed for LBSNs are now also upgraded to this category. For example, Facebook Places was announced in 2010 to bring similar LBSN features into Facebook [5]. Sina Weibo, a Twitter-like microblog app in China, has also come up with a “Nearby” feature to let users discover nearby people, microblogs and hot places.

While using LBSN apps of category I to check-in or share locations with friends, users are likely to *explicitly* publish their locations to their social networks [6]. On the contrary, while using LBSN apps of category II to discover nearby users, users will get information *without explicitly* making their locations public. As a matter of fact, when a user (using LBSN apps of category II) searches nearby users, the user’s exact location (e.g., GPS coordinates) will be uploaded to the app server, and then exposed (usually after obfuscation as needed) to nearby users by the app server. At first glance, the users’ exact locations would be secure as long as the app server is securely managed. However, there remains a risk of location privacy leakage when at least one of the following two potential threats happens. First, the location exposed to nearby users by the app server is not properly obfuscated. Second, the exact location can be deduced from (obfuscated) locations exposed to nearby users.

In this paper, we systematically investigate these two threats using typical LBSN apps of category II. We find that existing LBSN apps<sup>1</sup> are vulnerable to these threats, which could be exploited by the adversary to perform automated and efficient large-scale location probing attacks. Such attacks could reveal the location of any user that uses the “Nearby” feature. We propose a series of novel methods to probe the location privacy of people using different LBSN apps and show that our location probing methods are general and applicable to the vast majority of existing LBSN apps.

---

<sup>1</sup> Unless otherwise mentioned, LBSN apps investigated in the remainder of this paper belong to category II by default.

Our work is different from existing work in twofold. First, we are able to probe locations of any user whereas in existing works such as [7–9] the attacker can only probe the locations of his/her friends. Second, we propose three general approaches for location probing, and discuss the scenarios for using different approaches, whereas existing studies usually focus on specific situations. For example, [10] uses Android virtual machines to carry out proof-of-concept attacks via Wechat, Skout and Momo, and [11] discusses the possibilities of launching the sniffing attacks if the HTTP traffic of LBSN apps can be intercepted and manipulated.

To the best of our knowledge, we are the first to carry out a large-scale experiment to evaluate the practical efficiency of such attack in real world. Our contributions include:

- **Track location information flows and evaluate the risk of location privacy leakage in popular LBSN apps.** We analyze the location information flows from many aspects including location accuracies, transport protocols, packet contents, etc. in popular LBSN apps such as Wechat, Momo, Mitalk, SayHi, Skout, MeetMe and Weibo, and find out that most of them have high risk of location privacy leakage.
- **Propose three general attack methods for location probing and evaluate them via different LBSN apps.** We propose three general attack methods to probe and track users' location information, which can be applied to the majority of existing LBSN apps. We also discuss the scenarios for using different attack methods, and demonstrate these methods on SayHi, Mitalk and Wechat, separately.
- **Recommendation of count-measures.** We suggest some count-measures against this new threat of privacy leakage in LBSNs.

The rest of the paper proceeds as follows. Section 2 presents an overview of location-based social networks and LBSN apps. Section 3 details three general attack approaches with examples. After recommending some countermeasures in Sect. 4, we conclude the paper in Sect. 5.

## 2 Overview of LBSN Apps

Most LBSN apps have two features: check-in and social network discovery. We focus on the latter because people's check-in locations are shared with their friends and therefore they are not regarded as private information.

The workflow of social network discovery in LBSN apps is elaborated in Fig. 1. The following steps will be performed in the scenario where a user searches for people nearby at location  $l_0$  and time  $t_0$ .

- **Step (a):** The mobile app sends a request including the user's current location  $l_0$  which is obtained by GPS or online SDKs (e.g., Google SDK [12], Baidu Location SDK [13]) and the authorization token to the server. The authorization token is provided by the server as a unique identifier as long as the user logs into the mobile app.

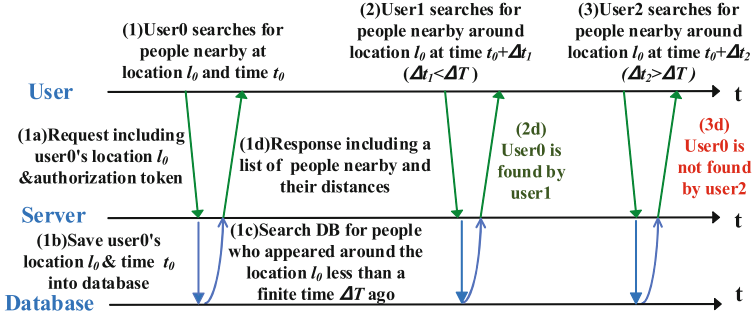


Fig. 1. The workflow of social network discovery in LBSN apps

- **Step (b):** Once the request from the user is received, the server saves the user’s location  $l_0$ , time  $t_0$  and other information into the database for further use, such as letting the user be visible to others.
- **Step (c):** The server searches the database which contains the request time and locations of all the users who have ever searched for nearby people. Then, it finds out a list of users who are not in the friend list of the user ( $user0$ ) and have appeared around the location  $l_0$  (within a distance of  $\Delta D$ ) less than a finite time  $\Delta T$  ago. Given a user as  $u$ , the people in user  $u_i$ ’s social network as  $U_{fi}$ , and the distance between two locations  $l_i, l_j$  as  $D_{l_i, l_j}$ , the nearby users queried from the database for user  $u_0$  can be described as:

$$\{u, l, t | u \notin U_{f0}, D_{l, l_0} \leq \Delta D, t \geq t_0 - \Delta T\} \tag{1}$$

- **Step (d):** The server sends a response to the mobile app with the queried results. For the purpose of privacy protection, the results returned by most LBSN app servers only contain essential user information  $u$  and coarse-grained distances  $l$ , because if the accurate distances are provided, a user’s exact location can be calculated by trilateration position methods easily [14]. Finally, the mobile app displays these results to the user.

Figure 2 shows the displayed results in typical LBSN apps: Wechat, Mitalk, Momo, Weibo, SKout, SayHi, Badoo and LOVOO. The displayed user information generally contains nickname, gender and other information (e.g., personalized signatures). In particular, Wechat, Mitalk and Weibo provide distances to an accuracy of 100 meters, and Momo and SayHi do so to an accuracy of 10 meters. However, LOVOO provides distances accurate to within 0.1 miles, which is the least accurate.

The user can view detailed information (e.g., publicly available photos) of nearby strangers, send greetings to them, and finally make new friends to extend the user’s own social network.

Figure 1 also presents two other scenarios to show in what circumstances a user can be found by others. In one scenario where  $user1$  searches for people nearby at a place which is close to the location of  $user0$  ( $l_0$ ) for a short while

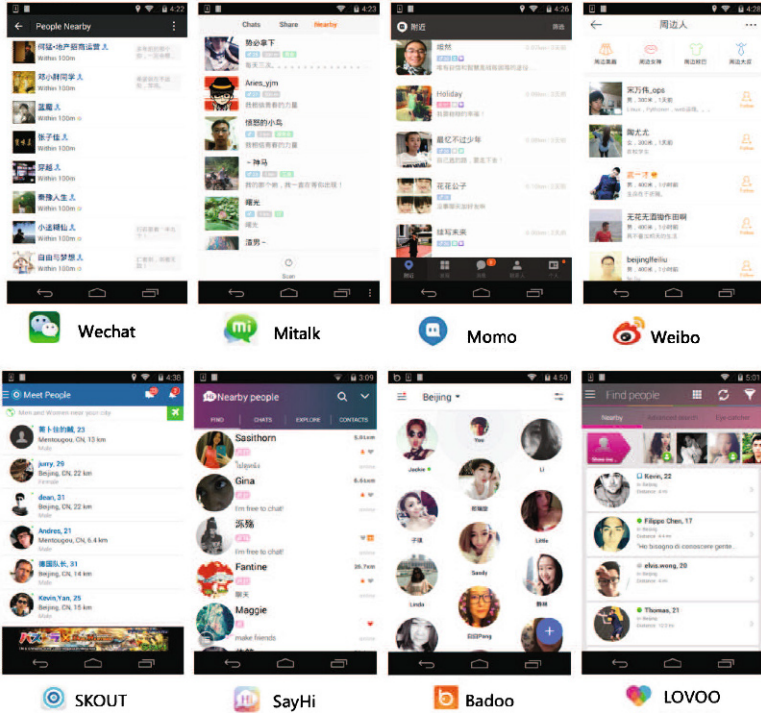


Fig. 2. Search nearby people in LBSN apps

( $\Delta t_1, t_1 < \Delta T$ ) after  $user_0$  searches for people nearby, according to Eq. 1,  $user_0$  can be found by  $user_1$  because  $D_{l_0, l_1} \leq \Delta D$  and  $t_0 \geq t_1 - \Delta T$ . As for the other scenario where  $user_2$  searches for people nearby after a long while ( $\Delta t_2, t_2 > \Delta T$ ),  $user_0$  cannot be found.

If an attacker  $u_A$  (whose friend list  $U_{f_A}$  is empty) is able to send a fake location  $l_A$  to the server in Step (a), he will get a response containing the users  $u_i$  around  $l_A$  and their distances  $D_{l_i, l_A}$  in Step (d). By changing the value of  $l_A$  constantly, the attacker can probe the users at any location.

In order to perform the location probing attack, we need to address the following challenging issues.

- **How to forge the request with fake locations:** We need to intercept the request in Step (a), and tamper the value of current location  $l$ . For securing data transportation, some LBSN apps use techniques like SSL authentication and data encryption, making request forgery a challenging task. Therefore, we need to try all possible ways to break or bypass these protection techniques.
- **How to perform a large-scale probing effectively and economically:** We need to use as few resources as possible (e.g., 1 PC) to probe thousands of locations for large-scale attacks. Because the location information of the users will be cached for a while ( $\Delta T$ ) in Step (c), using too many probers to probe at



different locations synchronously is both resource-consuming and unnecessary. But if the time span of probing two nearby locations is too long (e.g., longer than  $\Delta T$ ), some data may be missed. For example, a user appeared at location  $l_0$  at time  $t_0$ , his location information can be probed only if the prober happens to probe at a location near  $l_0$  between time  $t_0$  and  $t_0 + \Delta T$ .

### 3 Location Privacy Probing via LBSN Apps

This section presents some general paradigms for location privacy probing via popular LBSN apps. We first look deeply into some popular LBSN apps and examine the security through their transport protocols, request encryptions, response data, etc. Then, we propose and demonstrate three general methods for location privacy probing, which can be applied to the majority of existing LBSN apps.

#### 3.1 Examining Popular LBSN Apps

We install 8 popular LBSN apps including LOVOO, MeetMe, Mitalk, Momo, SayHi, Skout, Wechat and Weibo into an Android mobile phone, and use a web debugging proxy named Fiddler [15] to intercept and examine the network traffic between the apps and their servers. We set up a proxy with Fiddler 4 on a computer, and configure the proxy settings in the Android mobile phone to access Internet through our proxy. Then, all the HTTP/HTTPS traffic of the LBSN apps can be intercepted and monitored by Fiddler 4. Figure 3 shows the user interface of Fiddler 4. We see a list of intercepted HTTP/HTTPS requests on the left side of the user interface, including *Protocol*, *Host*, *URL*, etc. On the right side, there are two windows showing the details of the selected request and decoded response respectively.

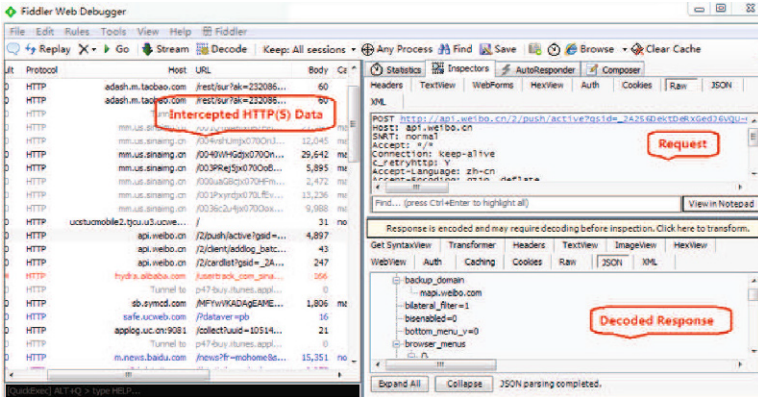


Fig. 3. Intercept and monitor network traffic with Fiddler 4

We examine the security of the intercepted network traffic from different aspects.

- **Transport protocols:** The content in HTTP requests can be easily intercepted and manipulated to launch the request forgery attacks. HTTPS (HTTP over TLS/SSL) can provide data encryption to prevent the data from being tampered [16]. It is worth noting that TLS/SSL can be configured either one-way or two-way. In one-way TLS/SSL communication, the server is required to present the certificate to the client, but the client is not required to present the certificate to the server, meaning that the server will accept the request from anyone. In this case, the HTTPS request can still be forged using a local self-signed certificate [17]. In two-way SSL communication, both the server and the client are required to present the certificates, which makes request interception a hard task. Therefore, we consider the HTTP and HTTPS with one-way SSL protocols are insecure, and HTTPS with two-way SSL protocol is safe.
- **Request encryptions:** Another way for data protection is to encrypt some of the parameters in the HTTP request. For example, a checksum or signature can prevent the request from being tampered effectively, as long as the encryption algorithm can not be cracked easily.
- **Response data:** The response data should not contain more information than what the app client needs. If the response data contains much more information (e.g., more accurate location than which is displayed in the app, the last time the person appeared), it will bring a risk of information leakage.

The analysis results are shown in Table 1. We can see that most apps use HTTP or HTTPS protocol with one-way SSL for data transportation and have no encrypted parameters in the requests. In this case, we can forge the HTTP/HTTPS requests to query nearby people at any location. Mitalk and LOVOO encrypt parameters (checksum and signature) and therefore the request can be forged only if we can crack the encryption algorithms and figure out the value of checksum or signature parameters. If the requests are too difficult to forge while the data is transported via HTTPS with two-way SSL or the encryption algorithm is irreversible, we can also use mobile phone emulators and automated testing methods to simulate user actions to get people nearby at fake locations. The detailed demonstrations of these three methods are shown in Sects. 3.2, 3.3 and 3.4.

### 3.2 Forging Requests

For LBSN apps, the request for searching people nearby contains parameters which are used to locate the user. The attacker can search people at any location by intercepting and tampering the location parameters. We demonstrate the attack in the following steps:

**Step 1: Request Interception.** We use Fiddler as a web proxy to intercept the HTTP/HTTPS traffic between LBSN apps and their servers. For HTTP traffic in

**Table 1.** Examination results of popular LBSN apps

APP	Downloads(million)			Location Accuracy in APP	Transport protocol	Request encryption	Location accuracy & Other information in response data
	Google play <sup>1</sup>	App Store	360 Android market <sup>2</sup>				
LOVOO	14	8.4	0.001	0.1 mi radius	HTTPS with one-way SSL	signature	100m radius&last time
MeetMe	14	8.9	0.001	100 m radius	HTTP with plaintexts	none	100 m radius
Mitalk	0.8	1.8	17	100 m radius	HTTP with plaintexts	checksum	10 m radius
Momo	1.8	26	168	10 m radius	HTTPS with one-way SSL	none	1 m radius
SayHi	12	3.3	0.04	10 m radius	HTTP with plaintexts	none	0.00001° coordinate (≈ 0.1m)
Skout	23	24	0.06	1000 m radius	HTTP with plaintexts	none	0.01 m radius
Wechat	169	43	455	100 m radius	HTTPS with two-way SSL	N/A	N/A
Weibo	10	23	456	100 m radius	HTTP with plaintexts	none	0.00001° coordinate (≈ 1m) & last time

<sup>1</sup> Most people in Chinese Mainland download Android apps from third-party markets because Google Play is inaccessible there.

<sup>2</sup> One of the largest Android third-party markets in China provided by Qihoo 360 Company.

plaintext, we can directly get the contents of the requests and responses. Fiddler can also decrypt HTTPS traffic with one-way SSL, as long as a local self-signed certificate is generated and installed into the mobile phone. If certificate and public key pinning [18] is used in the LBSN app, reverse engineering work should be performed to replace the hard-coded key of the app with the one generated by Fiddler.

Some of the intercepted requests of different LBSN apps are as follows:

- **MeetMe:**

GET <http://friends.meetme.com/mobile/boost/0?placement=meet&targetGender=b&latitude=38.988088&longitude=-76.977333&orderBy=distance&includeFriends=t&onlineOnly=f&pageSize=30>

- **SayHi:**

GET <http://r.x-vv.com/ft?s=L99uLTNuWxp1RJ&gt=true&ii=1&ts=0&of=0&lc=116.2085999,39.9726842>

- **Weibo:**

GET [http://api.weibo.cn/2/place/nearby\\_users?gender=0&sourcetype=findfriend&offset=0&s=a5516ad4&c=android&lat=39.83178&lng=116.290966&gsid=u078d0a32pkzvoOr0ElvflVM8j&&page=1&sort=1&count=20](http://api.weibo.cn/2/place/nearby_users?gender=0&sourcetype=findfriend&offset=0&s=a5516ad4&c=android&lat=39.83178&lng=116.290966&gsid=u078d0a32pkzvoOr0ElvflVM8j&&page=1&sort=1&count=20)

- **Momo:**

POST <https://api.immomo.com>  
Count=20&lat=39.83178&lng=116.290966&index=0

- **Skout:**

Set current city name:

POST <http://and.skout.com/api/1/me/location>

Get nearby people:

GET [http://and.skout.com/api/1/lookatme?application\\_code=3456025fd1e4ec43hec488b84fd700f4\&area=city\&limit=20\&start=0\&rand\\_token=3dcbf32a-9966-4b6b-9c18-441be07b12e1](http://and.skout.com/api/1/lookatme?application_code=3456025fd1e4ec43hec488b84fd700f4\&area=city\&limit=20\&start=0\&rand_token=3dcbf32a-9966-4b6b-9c18-441be07b12e1)

- **Badoo:**

POST <https://eu1.badoo.com/jsss/mjinba.phtml?v=2>

<jsondata>

In these requests, the location parameters  $\{latitude, longitude\}$ ,  $lc$ ,  $\{lat, long\}$ ,  $\{lat, lng\}$  indicate the location of the user who is searching nearby people.

**Step 2: Request Forgery.** We forge HTTP or HTTPS with one-way SSL requests by modifying the values of the location parameters in the intercepted requests to search nearby people at any location. We develop a program to automatically probe nearby people at random locations repeatedly. In order to avoid the alarm of anomaly detections, the program sleeps for a short while after each probing.

**Step3: Response Parsing.** For most of the LBSN apps, the responses of searching nearby people are in JSON format because it is more efficient than XML and other data interchange formats [19]. We can extract useful information such as the person’s id, name, distance or geo-coordinate by comparing the response data with the information displayed in the app.

Figure 4 shows the displayed results and the JSON-formatted response of searching nearby people in SayHi. SayHi provides distance values to an accuracy of 0.01 km. As shown in Fig. 4, although we can see that the user *Sasithorn* is 5.01 km away, we cannot figure out the exact location of *Sasithorn* only using this information. However, we find that the geo-coordinate of *Sasithorn* (116.339193, 39.9923481) is in the JSON-formatted response data.

Besides the geo-coordinate of the user, the JSON-formatted response of Weibo also exposes the time when the user was located in that place for the last time (*last\_at* field). Figure 5 demonstrates a real-world example, which indicates that the user with ID *2753134315* was at the location (116.30042, 40.02080) at 01:09:58, Sep 27th, 2015.

### 3.3 Encryption Cracking

Some LBSN apps use data encryption techniques other than HTTPS protocol to secure the data traffic. They add encrypted parameters such as checksum or signature into the requests for data tampering detection. Take Mitalk for example, the intercepted request of searching nearby people in Mitalk is shown in Fig. 6, in which *latitude* and *longitude* represent the searching location. The JSON-formatted response contains an “ok” code and a list of persons around the searching location. However, when we try to modify the value of *latitude*, *longitude* or any other parameter in the request, the response indicates errors with code 401.

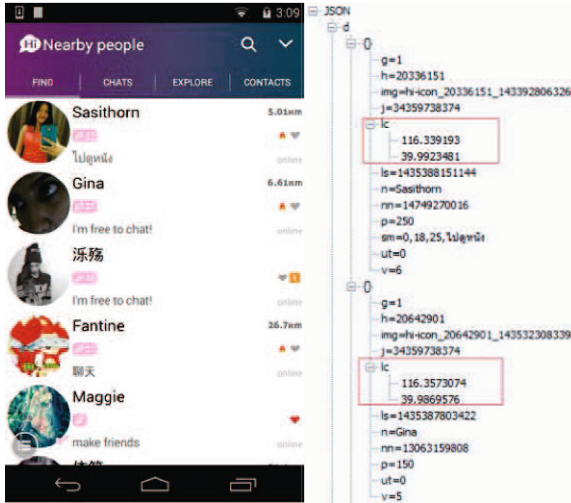


Fig. 4. Displayed and JSON results of searching nearby people in SayHi



Fig. 5. Displayed and JSON results of searching nearby people in Weibo

After a series of experiments, we figure out that the parameter  $s$  in the request is generated by a customized algorithm and it represents the checksum of all other parameters. The server will recalculate the checksum and compare it to the value of  $s$  when it receives a request. If the values don't match (i.e., one or more of the parameters might be tampered), an error message will be returned.

We perform reverse engineering to crack the algorithm of generating parameter  $s$ . We decompile the APK of Mitalk into Java using tools including apk-tool [20], dex2jar [21] and Jd-gui [22], and find out the generation procedure of  $s$ , which is shown in Fig. 7(a).

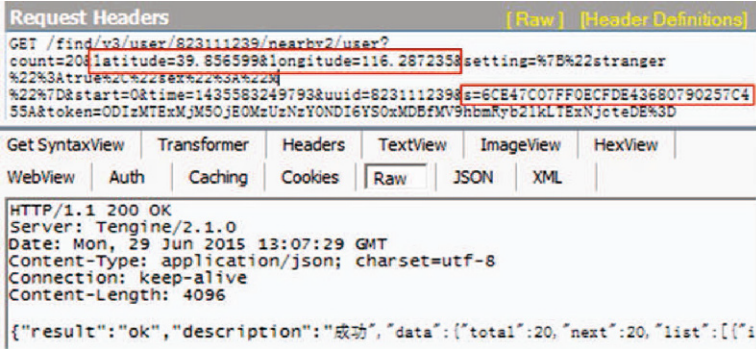


Fig. 6. Intercepted request and response of Mitalk

```
public static String s(List<NameValuePair> paramList, String paramString)
{
    paramList.add(new BasicNameValuePair("time", String.valueOf(System.currentTimeMillis())));
    Collections.sort(paramList, new Eb());
    StringBuilder localStringBuilder = new StringBuilder();
    Iterator localIterator = paramList.iterator();
    for (int m = 1; localIterator.hasNext(); m = 0)
    {
        NameValuePair localNameValuePair = (NameValuePair)localIterator.next();
        if (m == 0) {
            localStringBuilder.append("&");
        }
        localStringBuilder.append(localNameValuePair.getName()).append("=").append(localNameValuePair.getValue());
    }
    localStringBuilder.append("&").append(paramString);
    return com.kizami.channel.d.f.g.c(new String(com.kizami.channel.d.f.g.p.localStringBuilder.toString()).getBytes());
}
```

(a) The Decompiled Generation Procedure of Parameter  $s$ 

Source	Tag	Text
com-string/jumbo v1, "FLOWER"	D:FLOWER	9990FAA17282AC8C5267E140C96857
com-string/jumbo v1, p1, Landroid/util/Log;=>v4(Ljava/lang/String;Ljava/lang/String	D:FLOWER	F7CE178FA18D1A3924A52B77F4C9A084
com-string/jumbo v1, p1, p1, Lcom/kizami/channel/common/network/azr=>v4(Ljava/util/List	D:FLOWER	count=100&read_time=1411458647811
com-string/jumbo v2, "FLOWER"	D:FLOWER	uid=539405118&f7CE178FA18D1A3924
com-string/jumbo v2, "FLOWER"	D:FLOWER	38251743D61D8315CA082B910252822C
com-string/jumbo v2, "FLOWER"	D:FLOWER	F7CE178FA18D1A3924A52B77F4C9A084
com-string/jumbo v2, "FLOWER"	D:FLOWER	comctxs=-1&time=1411459705330&uid
com-string/jumbo v2, "FLOWER"	D:FLOWER	7F4C9A084
com-string/jumbo v2, v0, Landroid/util/Log;=>v4(Ljava/lang/String;Ljava/lang/String	D:FLOWER	21530623E7057C3A23C0922CC8603B50
com-string/jumbo v2, v0, Lorg/apache/http/message/BasicNameValuePair;	D:FLOWER	F7CE178FA18D1A3924A52B77F4C9A084
com-string/jumbo v2, "s"	D:FLOWER	count=100&read_time=1411458757433
	D:FLOWER	uid=539405118&f7CE178FA18D1A3924
	D:FLOWER	1AA1843607746DC3FC0C18E8720751CC

(b) Monitor  $paramString$  in LogcatFig. 7. Cracking encrypted parameter  $s$  of Mitalk

From the figure we conclude that the encrypted parameter  $s$  is calculated according to Eq. 2:

$$s = b(\text{name1} = \text{value1} \& \dots \& \text{nameN} = \text{valueN} \& \text{paramString}) \quad (2)$$

In the equation,  $\text{name1}$  to  $\text{nameN}$  are the alphabetical parameters of the request excluding  $s$ . Meanwhile,  $\text{paramString}$  is a fixed value. In order to get the value of  $\text{paramString}$ , we firstly disassemble the APK file of Mitalk into smali codes, and insert some debug codes to let the app print the value of  $\text{paramString}$  into *logcat* (which is an Android logging system for collecting debug outputs) while running. Then, we repackage [23] the APK file and install it into an Android phone. When

```

public static char[] b(byte[] var0, int var1, int var2)
{
    int var3 = (1 + var2 * 3) / 3;
    char[] var4 = new char[3 * ((var2 + 2) / 3)];
    int var5 = var1 + var2;

    int var10;
    for (int var6 = 0; var1 < var5; var1 = var10)
    {
        int var7 = var1 + 1;
        int var8 = 255 & var0[var1];
        int var9;
        if (var7 < var5)
        {
            int var21 = var7 + 1;
            var9 = 255 & var0[var7];
            var7 = var21;
        }
        else
        {
            var9 = 0;
        }
        int var11;
        if (var7 < var5)
        {
            var10 = var7 + 1;
            var11 = 255 & var0[var7];
        }
        else
        {
            var10 = var7;
            var11 = 0;
        }

        int var12 = var8 / 4;
        int var13 = (var8 & 3) << 4 | var9 / 16;
        int var14 = (var9 & 15) << 2 | var11 / 48;
        int var15 = var11 & 48;
        int var16 = var2 + 1;
        var4[var6] = b[var12];
        int var17 = var16 + 1;
        var4[var16] = b[var13];
        char var18;
        if (var17 < var3)
        {
            var18 = b[var14];
        }
        else
        {
            var18 = (char)0;
        }
        var4[var17] = var18;
        int var19 = var17 + 1;
        char var20;
        if (var19 < var3)
        {
            var20 = b[var15];
        }
        else
        {
            var20 = (char)0;
        }
        var4[var19] = var20;
        var6 = var19 + 1;
    }
    return var4;
}

```

Fig. 8. Decompiled function *b* from *com.xiaomi.channel.d.f.a.b*

we launch the repackaged Mitalk app and login with an account, the value of *paramString* can be watched in a logcat viewer, as shown in Fig. 7(b).

From *com.xiaomi.channel.d.f.a.b*, the function *b* in Eq. 2 can be decompiled. The decompiled function is shown in Fig. 8. We can calculate the value of *s* using Eq. (2) to bypass the data tampering detection of the server, and then use the same method in Sect. 3.2 to search nearby people at any location.

### 3.4 Emulator Simulation

Some LBSN apps like Wechat and LOVOO use HTTPS with two-way SSL protocol or use advanced encryption techniques. In this case, it will be too difficult, if not impossible, to intercept and forge the requests. Under these circumstances, we use mobile phone emulators and automated testing tools to simulate user's actions to probe nearby people at any location.

We demonstrate the method on Wechat using Android emulator [24] and *uiautomator*, which is a testing framework for Android [25]. We create an automated functional UI test case using *uiautomator*, which will automatically press a series of buttons to launch Wechat app and search nearby people in it. As soon as the results are displayed on the screen, the test case will inspect the UI to find the layout hierarchy and read information we need such as usernames and distances through the properties of specific UI components. The UI and the corresponding layout hierarchy of Wechat are shown in Fig. 9. The algorithm of the testcase is shown in Algorithm 1.

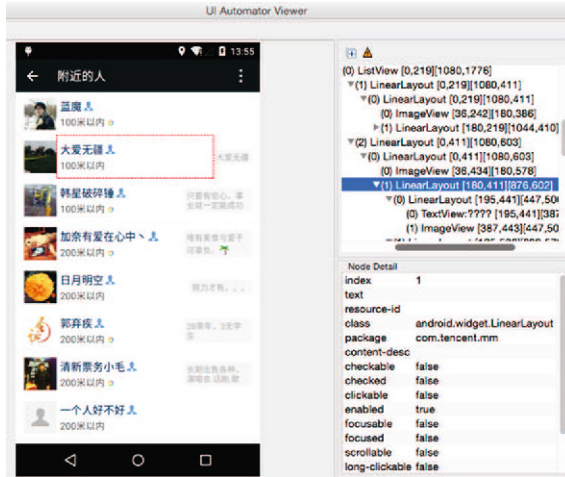


Fig. 9. Inspect the layout hierarchy of Wechat with UIAutomator

---

**Algorithm 1.** Searching and Reading People Nearby in Wechat

---

- 1: Press *HOME* button to return to home screen
  - 2: Find the icon with text “Wechat”, click it and wait for new window
  - 3: Find the tab with text “Discover”, click it and wait for new window
  - 4: Find the button with text “People Nearby”, click it and wait for new window
  - 5: **if** Text “Unable to load your location data” is found **then**
  - 6:   Return error
  - 7: **end if**
  - 8: Get the listview  $L$  with resource id “com.tencent.mm:id/atf”
  - 9: Read the properties of each listitem in  $L$  to *Result*
  - 10: **if**  $L$  is scrollable **then**
  - 11:   Scroll down  $L$  for next page
  - 12:   Goto 1
  - 13: **else**
  - 14:   Return *Result*
  - 15: **end if**
- 

In our experiments, we first send fake geo-coordinates to the emulator using a GPS command *geo fix* in the emulator’s control console, and then launch the testcase in the emulator to get nearby people at the fake location. By repeating the above two steps, we can probe nearby people at any location.

### 3.5 Location Tracking

As long as a large volume of data is collected, it is likely that a specific person would be probed multiple times at different places. Then, we can mark the





**Fig. 10.** Location tracking via different LBSN apps

location and the time when the person appeared on a map to track his/her locations.

For some LBSN apps such as Weibo and SayHi, we can get the geo-coordinates of a targeted person directly, and hence we mark the exact locations of the person with *points* on a map, as shown in Fig. 10(a). For some other apps such as Wechat and Momo, we can only get the coarse-grained locations which are determined by the probed location and the distances from the targeted person to the probed location. In this case, we mark the approximate locations of the person with *circles* on a map, as shown in Fig. 10(b). The red points indicate the locations of the probers, and the circles denote the possible locations of the probed users.

According to the trilateration positioning method [14], if a point lies on two circles at the same time, we can narrow down the possible locations to the intersections of the two circles. If a point lies on three or more circles, we can narrow down the possibilities to a unique point. Figure 10(c) shows that at nearly the same time, a user is probed by 5 probers (red points) and another user is probed by 3 probers. The locations of these two users can be deduced precisely to *Point1* and *Point2*.

### 3.6 Risk Evaluation

We next evaluate the overall risk induced by the popular LBSN apps. Table 2 shows the overall risk evaluation of popular LBSN apps. More than half of the apps (i.e., five out of eight) are easy to exploit. Meanwhile, half of the apps (i.e., four out of eight) can expose people’s location privacy with high accuracy. Weibo and SayHi have the highest risks, because they can be exploited for location probing easily and meanwhile can leak people’s geo-coordinate directly. LOVOO and Wechat have a relatively low risk of being exploited for large-scale location probing, because the efficiency of emulator simulation is much lower than forging requests, and they only expose people’s coarse-grained locations.

**Table 2.** Risk evaluation of popular LBSN apps

APP	Possible exploit method	Exploit difficulty	Accuracy of leaked data
LOVOO	Encryption cracking & Forging requests or emulator simulation	Difficult	Medium
MeetMe	Forging requests	Easy	Medium
Mitalk	Encryption cracking & Forging requests or emulator simulation	Difficult	High
Momo	Forging requests	Easy	High
SayHi	Forging requests	Easy	Very high
Skout	Forging requests	Easy	Low
Wechat	Emulator simulation	Difficult	Medium
Weibo	Forging requests	Easy	Very high

## 4 Recommendations on Counter-Measures

In this section, we discuss some possible counter-measures against the threat of location privacy leakage via LBSN apps.

Firstly, we point out that using HTTP protocol with plaintexts for data transportation is extremely unsafe, because it is vulnerable to both request forgery and MITM (man-in-the-middle) attacks. Besides, since the one-way TLS/SSL authorization does not require the server to check the validity of the certificate from the client, a self-signed local certificate can be generated and used to parse the plain content from the TLS/SSL traffic, which makes HTTPS protocol with one-way TLS/SSL unsafe to use. Other misuses of TLS/SSL in the development of the apps such as allowing all hostnames, trusting all certificates, SSL stripping and lazy SSL usage [26] will also make the apps vulnerable.

Also, anti-probing and anomaly detection methods should be used by the service providers to distinguish automatic probers from normal human users. It is not efficient enough to simply limit the quota for searching nearby people of each user just as what Momo does, because it can be bypassed easily by using multiple probing accounts and devices. A witty designed machine behavior model should be studied and applied for better detection and protection. Last but not least, in the client/server (C/S) model, while responding to the request, the response data volume should be small without more extra information than the app client needs, while leaving the data filtering and omitting work to the client will bring the risk of information leakage.

## 5 Conclusion

In this paper, we pointed out that mobile social apps will introduce location privacy leakage when they provide the functionality of searching nearby people.

We examined the risks of location leakage in popular LBSN apps and find out that they can be exploited for launching the location probing attacks. Moreover, we proposed three general methods for conducting such attacks via LBSN apps.

Using the new attack methods, we evaluated the overall risk induced by popular LBSN apps and had many interesting observations. This study shows that the current methods for location privacy protection in LBSN apps are insufficient and new protection mechanisms are desired to address such risk.

**Acknowledgements.** This work is supported in part by the Hong Kong GRF/ECS (No. PolyU 5389/13E) and the HKPolyU Research Grant (G-YBJX), and in part by the National Natural Science Foundation (No. 61202396, 61221063, U1301254), Postdoctoral Science Foundation (2015M582663), Postdoctoral Science Foundation of Shaanxi Province, 863 High Tech Development Plan (2012AAA011003), 111 International Collaboration Program, the Fundamental Research Funds for the Central Universities (3115200112), Shenzhen City Science and Technology R&D Fund (JCYJ20150630115257892), and Technology Innovation Project of Chinese Academy of Sciences (Y5X0011716, Y5X0011516), of China.

## References

1. Zheng, Y.: Tutorial on location-based social networks. In: Proceedings of the 21st International Conference on World Wide Web, WWW, vol. 12 (2012)
2. Foursquare Inc. About us. <https://foursquare.com/about>
3. Hattersley, M.: Google+ Companion. Wiley, Indianapolis (2012)
4. Inc, S.: Number of active wechat messenger accounts 2010–2015. <http://www.statista.com/statistics/255778/number-of-active-wechat-messenger-accounts/>
5. O’Dell, J.: A field guide to using facebook places. Mashable. com, p. 23 (2012)
6. Zhao, X., Li, L., Xue, G.: Checking in without worries: location privacy in location based social networks. In: 2013 Proceedings IEEE INFOCOM, pp. 3003–3011. IEEE (2013)
7. Carbutar, B., Sion, R., Potharaju, R., Ehsan, M.: Private badges for geosocial networks. *IEEE Trans. Mob. Comput.* **13**(10), 2382–2396 (2014)
8. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1082–1090. ACM (2011)
9. Cheng, Z., Caverlee, J., Lee, K., Sui, D.Z.: Exploring millions of footprints in location sharing services. *ICWSM* **2011**, 81–88 (2011)
10. Li, M., Zhu, H., Gao, Z., Chen, S., Le, Y., Shangqian, H., Ren, K.: All your location are belong to us: breaking mobile social networks for automated user location tracking. In: Proceedings of the 15th ACM International Symposium on Mobile ad hoc Networking and Computing, pp. 43–52. ACM (2014)
11. Patsakis, C., Zigomitos, A., Solanas, A.: Analysis of privacy and security exposure in mobile dating applications. In: Boumerdassi, S., Bouzefrane, S., Renault, É. (eds.) *Mobile, Secure, and Programmable Networking*. LNCS, vol. 9395, pp. 151–162. Springer, Switzerland (2015)
12. Rogers, R., Lombardo, J., Mednieks, Z., Meike, B.: *Android Application Development: Programming with the Google SDK*. O’Reilly Media Inc, Sebastopol (2009)

13. Baidu Inc. Baidu location sdk. <http://api.map.baidu.com/lbsapi/cloud/geosdk.htm>
14. Murphy, W., Hereman, W.: Determination of a position in three dimensions using trilateration and approximate distances. Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95, **7**, p. 19 (1995)
15. Lawrence, E.: Fiddler: Web Debugging Proxy (2007)
16. Hickman, K., Elgamal, T.: The ssl protocol. Netscape Communications Corp, vol. 501 (1995)
17. Rudrappa, N.: Defeating SSL certificate validation for android applications
18. Evans, C., Palmer, C., Sleevi, R.: Public key pinning extension for http. Technical report (2015)
19. Nurseitov, N., Paulson, M., Reynolds, R., Izurieta, C.: Comparison of JSON and XML data interchange formats: a case study. *Caine* **2009**, 157–162 (2009)
20. Winsniewski, R.: Android-apktool: a tool for reverse engineering android APK files (2012)
21. Alll, B., Tumbleson, C.: Dex2jar: Tools to work with android. dex and java. class files
22. Dupuy, E.: Jd-gui: Yet another fast java decompiler. <http://java.decompiler.free.fr/?q=jdgui/accessible>
23. Berthome, P., Fecherolle, T., Guilloteau, N., Lalande, J.F.: Repackaging android applications for auditing access to private data. In: 2012 Seventh International Conference on Availability, Reliability and Security (ARES), pp. 388–396. IEEE (2012)
24. Android Developers: Using the android emulator (2012)
25. Android Developers: Uiautomator (2013)
26. Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., Smith, M.: Why eve and mallory love android: an analysis of android SSL (in) security. In: Proceedings of the 2012 ACM conference on Computer and communications security, pp. 50–61. ACM (2012)

# MUSE: Towards Robust and Stealthy Mobile Botnets via Multiple Message Push Services

Wei Chen<sup>1</sup>, Xiapu Luo<sup>2,3</sup>, Chengyu Yin<sup>1</sup>, Bin Xiao<sup>2</sup>,  
Man Ho Au<sup>2</sup>, and Yajuan Tang<sup>4</sup>(✉)

<sup>1</sup> School of Computer,  
Nanjing University of Posts and Telecommunications, Nanjing, China

<sup>2</sup> Department of Computing,  
The Hong Kong Polytechnic University, Hung Hom, Hong Kong

<sup>3</sup> The Hong Kong Polytechnic University Shenzhen Research Institute,  
Shenzhen, China

<sup>4</sup> College of Engineering, Shantou University, Shantou, China  
yjtang@stu.edu.cn

**Abstract.** Exploiting unique features in mobile networks and smart-phones, mobile botnets pose a severe threat to mobile users, because smartphones have become an indispensable part of our daily lives and carried a lot of private information. However, existing mobile botnets usually rely on a single command and control channel (e.g., a push server or an SMS server) to disseminate commands, which can become the bottleneck or a single point of failure, without considering the robustness. In this paper, we propose MUSE, a novel multiple push service-based botnet, which can significantly outperform existing push-styled mobile botnets in terms of robustness, controllability, scalability, and stealthiness. Although the basic idea of using multiple push services seems straightforward, we explore the design space of exploiting such services and tackle several challenging issues to overcome the limitations of existing push-styled mobile botnets. We have implemented MUSE by exploiting ten popular push services and evaluated it through extensive experiments. The results demonstrate not only MUSE's feasibility but also its advantages, such as stealthiness, controllability etc.

## 1 Introduction

With the rapid advent of mobile Internet access, mobile devices are used by millions of users due to their portable Internet access and the increasing powerful computing capabilities. Botnets have also evolved from the traditional PC-based form to smartphones based form, because smartphones usually lack of sufficient security mechanisms and store a lot of private personal information. The mobile botnet is a collection of the compromised mobile devices that can remotely receive commands from botmaster through C&C channels. It can also upload the sensitive information in the compromised devices, thus bringing serious threats to the property, privacy, and security of smartphone users.

## 1.1 Current Mobile Botnets

In recent years, the mobile botnets have become an important threat to the cyber security with the development of mobile Internet. For example, in 2010, the GEINIMI botnet appeared on Android platform. Soon iKee.B was reported as the first botnet in jailbroken iPhones. In 2011, AnserverBot, an Android bot, began to use public blogs as C&C [24]. This approach was different from previous C&C channels, which usually depended on the traditional HTTP channel. The Bmaster botnet controlled thousands of Android devices [13]. By controlling lots of compromised mobile devices, the botmaster can harvest private information [25] and launch various malicious activities, such as DDoS attacks against a cellular network core [19].

Since message push services eliminate the direct communication between bots and botmaster and provide well-maintained infrastructures to deliver messages, recent studies suggested constructing push-styled mobile botnets [10, 23]. However, existing research has not fully explored the capability of push services for building advanced mobile botnet. For example, Zhao et al. proposed using Google’s push service (i.e., C2DM: Google’s Cloud to Device Messaging Service) to build mobile botnets targeting on Android platform [23]. However, the botnet in [23] is not robust enough because its bots could be identified through the same embedded API key. If the API key was blocked by the defender, the whole botnet would fail. Lee et al. designed Punobot [10], a mobile botnet built upon Google Cloud Message for Android (GCM), the new version of C2DM. Although Punobot avoids using the same API key in all bots by exploiting a vulnerability in GCM’s registration process, it still has the single-point-of-failure problem and does not explore the design space of employing push services for mobile botnet. In this paper, we propose MUSE, a novel push-styled mobile botnet exploiting multiple diverse push services (e.g., GCM, JPush, etc.) to provide better robustness, controllability, scalability, and stealthiness.

## 1.2 Goals and Challenges

In this paper, we explore the design space of mobile botnets based on message push services because such services provide great convenience and flexibility for bots to receive messages. Although a few studies proposed mobile botnets based on push services, none of them could achieve all of the following features:

1. **Robustness.** How to empower the botmaster to maintain the control of bots even after substantial push service accounts have been blocked by the defender? Some push services do not provide global services. If these services have taken down in certain area, how to keep the botnet survive?
2. **Controllability.** When multiple push servers are organized together, the complexity of controlling the botnet increases. How to allow the botmaster to quickly disseminate commands to bots?
3. **Scalability.** How to scale up the botnet when numerous new mobile bots join the botnet?

4. **Stealthiness.** How to prevent (or make it harder for) defenders from detecting bots via their communication traffic patterns? How to make botnet traffic similar to normal traffic?

We propose MUSE, a novel multiple push service-based botnet, which can significantly outperform existing push-styled mobile botnets in terms of robustness, controllability, scalability, and stealthiness. MUSE involves a set of practical methods to address these challenges. More precisely, besides using multiple push services to avoid the single point of failure, we propose a hybrid structure to connect bots, push services, and the botmaster together. Such structures can not only scale up the MUSE botnet but also make it very difficult to dismantle the botnet. We design a set of mechanisms to improve a MUSE botnet’s stealthiness and performance in terms of controllability and scalability. First, we use a dynamic weight round robin algorithm for push server selection. It can help botmaster avoid performing excessive operations on certain servers. This algorithm also ensures that the C&C traffic is distributed among different servers. Second, since push services only support text messages, we employ high-order mimic functions to transform binary data into English text. Third, the LEACH protocol [6] is applied for dynamically selecting servant bots to construct intermediate layer. Finally, when implement MUSE by integrating ten push services, we discuss incremental update techniques for Android bots and bi-direction communication channels with HTTP Restful API.

In summary, we make the following contributions:

- We exploit multiple push services to construct robust and stealthy mobile botnets. Our design, MUSE, avoids the single point of failure problem in mono-push-server structure and has flexible structures to improve the robustness and scalability.
- We propose a set of mechanisms to improve a MUSE botnet’s stealthiness and performance in terms of controllability and scalability. More precisely, at the botmaster side, MUSE disperses the botnet traffic among different push servers, which makes mobile botnet stealthy. MUSE uses three independent channels to enhanced upstream C&C channels. It is possible for all mobile bots to establish bi-direction communication channels.
- We have implemented MUSE employing ten popular push services and conducted extensive evaluations on it. The incremental update method is proposed to reduce update traffic consumption. The experimental results show that MUSE can effectively enhance mobile botnets’ stealthiness and performance.

### 1.3 Paper Organization

Section 2 reviews the related work. Section 3 introduces MUSE’s architecture. Section 4 discusses the design of MUSE, including commands dissemination and servant bots selection. Sections 5 and 6 present the implementation and the evaluations of MUSE, respectively. Section 7 concludes the paper with future work.

## 2 Related Work

Although botnets in wired network have been extensively examined in the past eight years [9, 16, 17], mobile botnets just emerged along with the rapid advent of mobile Internet and the prosperity of smartphones. Since mobile devices have lower computation ability and users are sensitive to resource consumption, researchers proposed mobile botnets with special functions and abilities on smartphones.

Pieterse and Olivier [15] designed a hybrid C&C structure for mobile botnet to avoid single point of failure. They used the short message service (SMS), HTTP and bluetooth to build communication channels. Singh et al. [18] used bluetooth as a C&C channel to send commands from one bot to another. This mobile botnet was limited by the radio range of Bluetooth. Anagnostopoulos et al. [1] introduced two botnet architectures that consist only of mobile devices. One applied mobile HTTP proxy to build C&C and the other used DNS protocol as a covert channel. Zeng et al. [22] proposed a P2P botnet structure using SMS. With the P2P topological structure, botmaster could reduce the number of SMS messages required for bots communications and decrease the time delay for delivering SMS commands. Hua et al. [7] used SMS to construct a mobile botnet and evaluated the performance in different types of network topology. Eslahi et al. [5] made a survey about the current available data set and samples of mobile botnets. They tried to implement a mobile botnet test bed to collect data for further research. According to the Kademlia P2P network structure, Mulliner and Seifert [14] utilized an SMS-HTTP hybrid approach to established a P2P mobile botnet, in which the C&C was divided into HTTP and SMS parts. Karim et al. [8] investigated mobile botnet attacks and compared existing mobile botnets on commercial as well as open source mobile operating system platforms. Cui et al. [4] presented the URL Flux-based mobile botnet model for Android. The botnet commands were hidden into specified pictures and uploaded to the blogs. Bot visited the corresponding blogs to extract the commands from pictures. They also proposed a Botnet Triple-Channel Model with three independent sub-channel to enhance the upstream channel [3]. BTM botnet had a high-performance upstream channel, which would be attractive for botmaster.

The research work most relevant to ours is push-styled mobile botnets proposed by Zhao et al. [23]. Such botnets exploited push services to establish C&C channel and the commands were disseminated by Google's C2DM. They showed that the message push service can deliver commands to bots timely and effectively. The botnet command dissemination can be hidden into the legitimate push traffic to some extent. Lee et al. [10] followed Shuang's work and proposed a mobile botnet which exploited GCM as its C&C channel, which reduced battery power consumption, traffic cost and money cost. These works depended on the reliability and availability of a single push server. If the defender blocks the suspicious botnet accounts on the push server, the botnet will be paralyzed. Some push services are not available in some regions, such as GCM in China. In these regions, the push service becomes the single point of failure in the botnet. In our previous work [2], we addressed the single point of failure problem by



using multiple push servers to substitute one single server. But we did not consider how to construct a sophisticated structure with the multiple push servers. The proposed method used KING algorithm to measure path delay between bots and push servers. Then bots were clustered into different groups by DBSCAN algorithm according to path delay. A static weight round-robin algorithm was applied to select a push server to disseminate commands for each bot group. The dynamic performance of push servers was not considered in the push server selection. In this paper, we propose a hybrid structure, which is more flexible and stealthy. The robustness of MUSE is mathematically analyzed. To support this hybrid structure, we discuss the servant bot selection and bot bi-direction communication channel. MUSE also involves a set of practical methods to implement the botnet. For example, incremental update is applied to reduce update traffic consumption and messages for command transmission are preprocessed to fulfill the requirement of push services that only support text messages.

### 3 Architecture of MUSE

In this section, we describe the architecture of MUSE starting from a basic scenario(i.e., Fig. 1a) where one push service is used to advanced ones including flat structure(i.e., Fig. 1b) and hybrid structure(i.e., Fig. 1c).

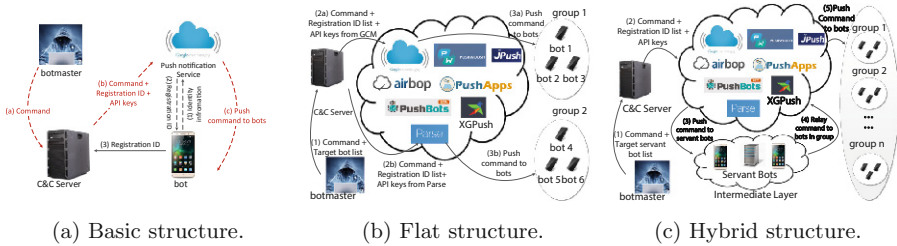


Fig. 1. MUSE's architecture.

#### 3.1 Basic Structure

As shown in Fig. 1a, the basic structure involves one push service. When a new bot joins the botnet, it first registers itself to the push service and then reports its registration ID, generated by the push service, to the botmaster. To disseminate commands to bots, the botmaster constructs a push request, which contains the command, the registration IDs, and required API Keys. After the botmaster sends the request to the push server, the push server will deliver the command to bots when they are online. Otherwise, the push server will store the command and send it later. The C2DM botnet [23] and the Punobot [10] can be considered as variants of the basic scenario.

The mobile botnet with push notification service eliminates the needs for bots to access the C&C server to retrieve the botnet commands intermittently. In addition, push-styled botnet avoids direct communication between the botmaster and the bot. However, the basic scenario has some limitations. The botnet depends on the availability of the push server. Some push services are not available in some regions, such as Google’s GCM in China. The push service may become the bottleneck for the whole botnet. What’s more, if the push server is blocked by the defender, the botnet will be paralyzed. MUSE can overcome these limitations by using the simple flat structure introduced in Sect. 3.2 or the advanced hybrid structure introduced in Sect. 3.3.

### 3.2 Flat Structure

As shown in Fig. 1b, the flat architecture enables the botmaster to send commands via different push servers to bots. Although push services may have diverse features, a botnet can exploit them by performing the following steps:

1. The bot obtains registration IDs from a push service. For example, by changing the identification information, such as the email address for GCM [10], a bot could have several registration IDs from one push service for improving its resiliency. Therefore, even if one registration ID is blocked by the defender, the bot can still receive messages from the push server using other registration IDs.
2. The bot sends all registration IDs to the botmaster. Since the amount of such information is small, a bot can easily deliver them to botmaster through email, SMS, and various covert channels [11].
3. The botmaster registers itself as one or more server applications to obtain accounts in each push service. After getting bots’ registration IDs, the botmaster can send push messages that contain commands to bots according to their registration IDs and the corresponding push service accounts.
4. When a push service receives messages from the botmaster, it will dispatch them to bots according to their registration IDs.

In the flat structure, a push service controls several groups of bots and each bot can receive messages from more than one push service. Therefore, even if one push service is down or bots associated with one push service are blocked, bots can still receive commands through other push services. Moreover, since the botnet traffic is distributed among multiple servers, MUSE introduces much less additional traffic to normal connections with push services than the basic scenario does, thus improving the stealthiness. Note that in the flat structure there are no connections among different push services. To further improve a MUSE botnet’s robustness, stealthiness, and flexibility, we propose a hybrid structure so that the botmaster can dynamically change the botnet’s structure (e.g., hierarchy structure, P2P structure, etc.).

### 3.3 Hybrid Structure

As shown in Fig. 1c, the hybrid structure has one or more intermediate layer that includes a set of compromised smartphones or dedicated servers running emulator. We call them servant bots, and they usually have good capability, such as powerful CPU, stable network connection, and large bandwidth. Besides joining multiple push services to receive commands, servant bots can relay messages received from one push service to other push services through their APIs.

With the hybrid structure, the botmaster can easily re-organize the structure of bots. For example, the botmaster can construct a hierarchy structure by letting servant bots serve as internal nodes of a tree and normal bots as leaf nodes. Figure 2a shows the hierarchy structure, where C&C server becomes the root of the tree and servant bots work as internal nodes. All bots become the leaf nodes and internal nodes send messages to its children through push services.

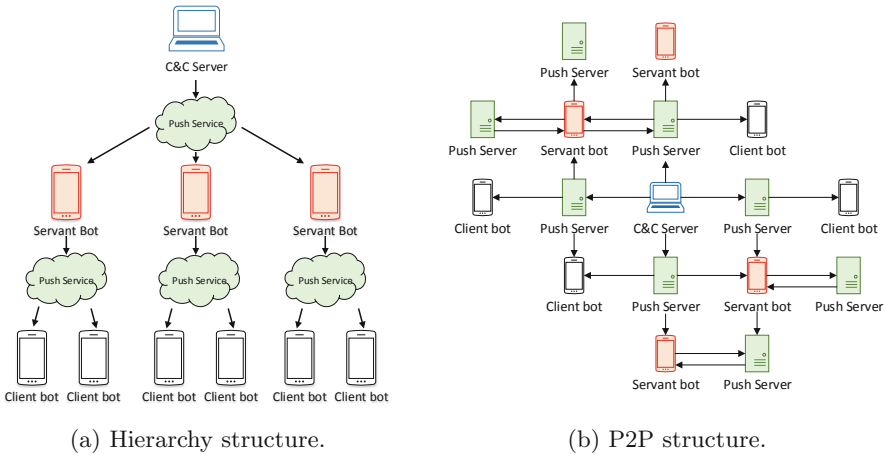


Fig. 2. Change to different structures.

The botmaster can also build a hybrid P2P botnet [20] where push services act as super nodes. As shown in Fig. 2b, servant bots make push services completely connected. A servant bot can connect to multiple push services and each push service can send commands to multiple client bots. When a new bot joins botnet, it can send registration information to multiple servants bots. With the P2P structure, the commands can spread over the botnet in a short time.

With different structures, a botnet becomes more robust, because bots have more approaches to receive commands, and more stealth, because bots can also easily change their traffic patterns.

### 3.4 Robustness Analysis

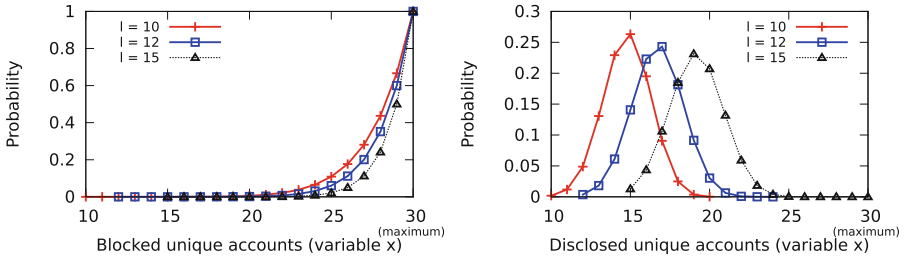
We conduct an analytical study on the robustness of botnet. Assume that there are  $n$  push servers and botmaster registers  $m$  accounts on each push server.

These accounts are shared by all bots. Each bot maintains an account list and suppose there are  $l$  accounts in each bot list. In MUSE’s implementation, each bot randomly chooses  $l$  accounts from all shared  $n \times m$  accounts. The account list can be set in the manifest file in Android application. It is not difficult for bots to use different account lists in practice.

One defense method against MUSE is to catch some bots and retrieve the account information by reverse engineering bot malware. Then, they can block these accounts by informing push services providers. We assume that defenders catch  $b$  bots and get  $b \times l$  accounts information. Since the accounts are shared by bots, some accounts may be in more than one bots. We let  $x$  stand for the number of unique accounts.

Note that only when all accounts in a bot list are blocked by the defenders, this bot will lose the connection to the botmaster. If there is one account left, the bot can still receive information from the botmaster to update its account list. When  $x$  accounts are blocked, the probability of a bot being blocked is:

$$P(x) = \frac{C(x, l)}{C(n \times m, l)}, \quad x \in [l, \max(b \times l, m \times n)] \tag{1}$$



(a) Blocked probability with different accounts number. (b) Disclosed probability when bots are captured.

**Fig. 3.** Robustness mathematical analysis. (Color figure online)

Figure 3a shows the analytical result of botnet robustness. The parameters are set as  $n = 10, m = 3, b = 2$ . It shows that when the blocked account  $x$  is not close to maximum value, the botnet can still be safe. Moreover, if the botmaster uses more push servers ( $n$ ) and applies more accounts ( $m$ ), the botnet will be more robust.

Now we analyze how many unique accounts( $x$ ) appear in all caught accounts( $b \times l$ ). When  $b$  bots are caught,  $b \times l$  accounts on these bots are blocked. The number of unique accounts  $x$  is:

$$N(x) = C(m \times n, x) \sum_{k=0}^{x-1} (-1)^k C(x, k) (x - k)^{b \times l} \tag{2}$$

The probability of  $x$  unique accounts being caught among  $b \times l$  accounts is:

$$P_n(x) = \frac{N(x)}{\sum_{x=l}^{\min(b \times l, m \times n)} N(x)} \quad (3)$$

Figure 3b shows the probability of disclosing unique accounts when bots are captured. The parameters are set as  $n = 10, m = 3, b = 2$ . The result shows that it is not easy for defender to disclose all accounts when a small part of bots are captured.

## 4 Design of MUSE C&C

The C&C mechanism is the core part of the botnet. One task of C&C is to efficiently disseminate botnet commands to bots. It is also important to hide and protect command and control traffic to avoid detection. In this section, we describe the design of MUSE’s C&C mechanism for command dissemination with the consideration of controllability and stealthiness.

To enhance the controllability, bots are initially categorized into different groups according to their locations. We propose a new algorithm to help the botmaster select a push server for sending commands to different bot groups. To scale up the botnet, a hybrid architecture has been introduced in Sect. 3.3. Here, we discuss how to select servant bots from bot groups to construct intermediate layer. Finally, we employ high-order mimic functions to transform binary data into English text because push services only support text messages.

### 4.1 Push Server Selection

Push servers usually provide services for global users and deploy distributed system of servers in multiple data centers across the Internet, which is similar to CDN (Content Distribution Network) technology. The goal of distributed deployment is to push content to end-users with high availability and high performance.

For individual smartphone, different push servers have different performance. To improve the controllability of a botnet, a push server with best performance is preferred to disseminate commands. In order to simplify command dissemination, the bots from the same region are categorized into the same group. The botmaster selects one push server for each group to send commands.

During push server selection, several factors should be considered, including delay, quota, and load balance. The push server with least delay is preferred to perform commands dissemination. However, some push servers have message quotas limitation. What’s more, if only one push sever is used to send commands, botnet may be exposed due to anomaly heavy pushing overhead. Sophisticated load balance technology can improve stealthiness of MUSE.

In order to maintain better utilization of push server, botmaster can establish a dynamic weight vector to record server status. We use a dynamic feedback mechanism to adjust the server weights. By constantly monitoring server load

information, the feedback mechanism can calculate an integrated load balance value. When a server load value is low, the feedback mechanism increases its weight to expand the number of connections. The dynamic feedback mechanism can maintain weight variation within a specific range to achieve good server utilization. Within a fixed period of time  $T$ , two parameters are measured to adjust the weigh vector:

*Input<sub>j</sub>*: it indicates pushing contribution ratio for a push server  $j$ . The value equals the ratio of received push requests  $N_j$  to the average received push requests for all push servers. This parameter is related with the request number from botmaster and is independent of push server itself.

$$Input_j = \frac{N_j \times n}{\sum_{k=1}^n N_k} \quad (4)$$

*Left<sub>j</sub>*: it indicates the remaining push ability for server  $j$ .  $M_j$  means the number of requests sent by a push server  $j$ . It is divided by the maximum quota of a push server and is reset to 0 in the next day. The remaining request quota is related with push server performance. The maximum quota can be learned from push server development documents. In particular, if the maximum number of requests quota is unlimited, then  $Max_j$  can be set to a predefined value. If there is no quota left, then the server is directly set unusable.

$$Left_j = \frac{M_j}{Max_j} \quad (5)$$

With the above two parameters, the feedback mechanism updates the *weight* of server  $j$  every  $T$  time using the Eq. 6.

$$weight_j = weight_j + \alpha \times \sqrt[3]{1 - Input_j - Left_j} \quad (6)$$

$\alpha$  is a feedback parameter. The *weight* value is adjusted in the range of  $[default, \beta \times default]$ . The *default* is initialized by path delay measurement. The path delay can be used as the major factor to evaluate the performance of push server. The  $\beta$  is a range parameter. In practice, when all *weights* are less than the *default*, all servers are idle. On the other hand, if all *weights* exceed  $\beta \times default$ , it means the all servers are overloaded. Botmaster should reduce the number of push requests.

After getting weight vector of push servers, we apply a weight round-robin (WRR) scheduling algorithm to schedule servers for pushing commands. The push server with higher weight will be assigned more push requests than those with lower weight. Different from original WRR algorithm, the proposed scheduling algorithm dynamically adjusts the weight after each round according to Eq. 6.

The improved dynamic WRR algorithm is presented in Fig. 4. Here variable  $i$  indicates index of the current server used for pushing and is initialized with -1.  $cw$  is the current weight in scheduling and is initialized with 0.  $max(weight\_vector)$  is the maximum value in weight vector.  $gcd(weight\_vector)$  is the greatest common divisor of weight vector.  $Input_j, Left_j, N_i, M_i$  are global variables that appeared in Eqs. 4-6.

---

```

Dynamic Weight Round-Robin(weight_vector, last_select, last_cw, push_num)
cw = last_cw ;
i = last_select;
while true do
  i = (i + 1) mod n; // n is the number of push servers
  if i == 0 then
    for each weightj ∈ weight_vector do
      //dynamically update weight in vector
       $Input_j = \frac{N_j \times n}{\sum_{k=1}^n N_k}$ 
       $Left_j = \frac{M_j}{Max_j}$ 
       $weight_j = weight_j + \alpha \times \sqrt[3]{1 - Input_j - Left_j}$ 
      if weightj < default then
        weightj = default
      end if
      if weightj > β × default then
        weightj = β × default
      end if
    end for
    cw = cw - cw_gcd(weight_vector);
    if cw ≤ 0 then
      cw = max(weight_vector);
      if cw == 0 then
        return null;
      end if
    end if
  end if
  if weight_vector[i] ≥ cw then
    Ni = Ni + push_num;
    Mi = Mi + push_num;
    return i, cw;
  end if
end while

```

---

**Fig. 4.** Dynamic Weight Round-Robin algorithm

## 4.2 Servant Bots Selection

To scale up the botnet, a hybrid architecture is proposed in Sect. 3. Here we discuss how to select servant bots from bot groups to construct intermediate layer. The servant bots construct the intermediate layer and enhance the scalability of MUSE. Servant bots consist of static servant and dynamic servant. Static servant bots may be computers or smartphones firmly controlled by the botmaster. Dynamic servant bots can be selected from each group and changed to avoid being detected.

We adopt the LEACH(Low Energy Adaptive Clustering Hierarchy) protocol [6] to select servant bots. It was designed to lower the energy consumption required to create and maintain clusters for improving the life time of a wireless

sensor network. We apply LEACH to improve the stealthiness because a servant bot should avoid working too long time by dynamically substitution. Each bot in the group uses a stochastic algorithm at each round to determine whether it will become a servant bot in this round.

To select a servant bot, each bot determines a random number between 0 and 1. If the number is less than a threshold  $T(n)$ , the  $bot_n$  becomes a servant bot for the current round. The threshold is set as follows:

$$T(n) = \frac{P}{1 - P \times (r \bmod \frac{1}{P})} \quad \forall n \in G \quad (7)$$

$$T(n) = 0, \quad \forall n \notin G \quad (8)$$

with  $P$  as the servant bot probability;  $r$  as the number of the current round and  $G$  as the set of bots that have not been servant bots in the last  $1/P$  rounds.  $G$  can be reset over a period of time since we do not require every bot becomes a servant exactly once with  $1/P$  rounds, which is different from the original LEACH.

### 4.3 Message Preparation for Command Transmission

Since Android applications can use dynamic Java class loading to install new functions, a botmaster can send either text commands or binary codes (i.e., dex file) to bots. To evade the detection and fulfill the requirement of push services that only support text messages, MUSE adopts the procedure shown in Fig. 5 to prepare messages for command transmission.

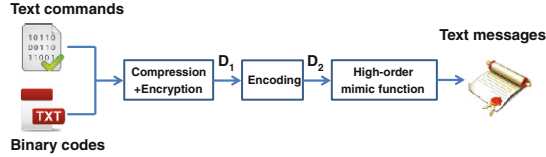


Fig. 5. Procedure of message preparation.

To prevent defenders from injecting fake commands, the botmaster generates a pair of public/private keys,  $\langle K_{pr}, K_{pu} \rangle$ , and embeds the public key  $K_{pu}$  in the bot program. After compromising a mobile device, the bot program randomly generates a symmetric encryption key  $K_s$ , encrypts it with  $K_{pu}$ , and sends  $\{K_s\}_{K_{pu}}$ , which denotes the encrypted  $K_s$  using key  $K_{pu}$ , to the botmaster. Note that only the botmaster can obtain  $K_s$  from  $\{K_s\}_{K_{pu}}$  using its private key  $K_{pr}$ . After that,  $K_s$  will be used along with  $\langle K_{pr}, K_{pu} \rangle$  to encrypt/decrypt data transferred between the bot and the botmaster. Moreover, the bot will periodically update  $K_s$  or do it after receiving a certain instruction from the botmaster.

To deliver commands or binary codes through text messages, the botmaster first compresses them, and encrypts them along with a sequence number (i.e.,  $I$ )



and a nonce (i.e.,  $\sigma$ ) using  $K_s$ . The output is concatenated with  $\{\sigma \oplus I \oplus K_s\}_{K_{pr}}$  to generate  $D_1$ . After that, the encoding module adds forward error correction (FEC) code to  $D_1$  and then divides  $D_1$  into blocks. The FEC allows bots to recover the data even if some blocks are lost. Each block is encoded through Base64 in order to represent binary data using ASCII. Moreover, the botmaster uses high-order mimic functions [21] to transform each block in  $D_2$  into English text with similar statistical properties in order to evade the detection. Finally, the botmaster sends these English text messages to bots through push services. After receiving the text message, a bot first recovers  $D_2$  using the corresponding inverse function and then extracts the binary data through base64 and FEC decoding. Using the embedded public key  $K_{pu}$  and  $K_s$ , the bot can extract the original data, authenticate it and verify its integrity. A bot employs a similar approach to send collected data to the botmaster.

## 5 Implementation

We have implemented MUSE by integrating ten push services including GCM, JPush, XGPush, ZYPush, GeXinPush, Airbop, Parse, Pushwoosh, Pushbots and PushApps. Since the last five push services rely on GCM services, we only use the first six push services in the experiments. The push services' properties are listed in Table 1. Most of popular push services are free and some of them also provide premium services to paid users. All push services in the Table 1 support statistical reports, including online users, amount of messages sent, and amount of messages read. Moreover, we recruited 15 volunteers and installed MUSE bots into their Android phones.

**Table 1.** Properties of push services

Push services	Registration parameters	Frequency limit	Size limit	Connection server
GCM	SenderId	token bucket scheme	4 KB	HTTP POST, CCS
Airbop	SenderId, Appkey, Appsecret	N/A	4 KB	HTTP POST, CCS
Parse	ApplicationID, ClientKey	N/A	N/A	HTTP PUT, POST
PushWoosh	ApplicationCode, GCMSenderID	N/A	N/A	HTTP POST
PushBots	ApplicationID, GCMSenderID	N/A	N/A	HTTP POST
PushApps	AppToken, GCMSenderID	N/A	N/A	HTTP POST
JPush	Appkey	<600/min	1 KB	HTTP POST
GexinPush	AppID, AppSecret, Appkey	N/A	2 KB	HTTP OpenService
XinGePush	AccessID, AccessKey	<20/min for broadcast	4 KB	HTTP Post
ZYPush	Appkey	N/A	1 KB	HTTP Post

Thus, we construct a real-life, small scale MUSE botnet with 15 bots. The Android application of MUSE bot can receive messages from different push servers as shown in the Fig. 6a. We then experiment four different attack commands: `contactinfo`, `gpsinfo`, `pic-upload` and `applist`. These commands require bots to upload contact list, GPS location information, photographs and applications list to a specified URL.

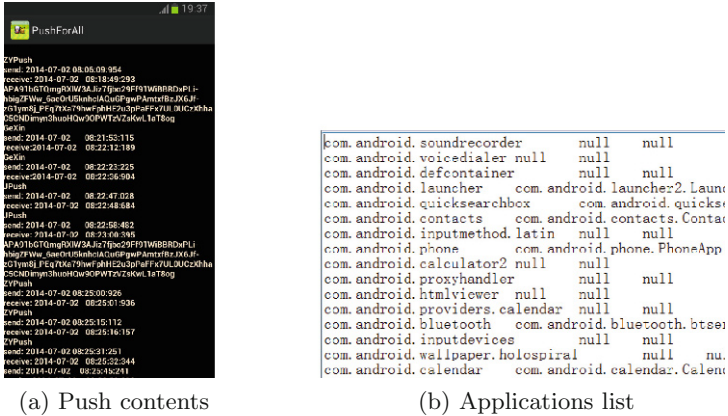


Fig. 6. Attack demonstration

## 5.1 Incremental Update

Once some accounts are blocked by defenders, botmaster prefers to update the push service accounts in bots. They can apply for new accounts to continue pushing services. As we analyzed in Sect. 3.4, botmaster can disseminate commands, including update information, to bots even if there is only one available account left. A simple update method is to recompile the latest version of the bot program with new accounts. Then botmaster dispatches the latest version to bots. This method will increase the network traffic dramatically, which can be easily detected by anomaly detection methods. Therefore, it requires a light-weight update method with low traffic consumption. We apply an incremental update method, which uses patch files to update bots. These accounts information can be compiled as a patch file with limited size. The patch file can implement incremental update using technology like Smart App update provided by Google.

The principle of incremental update is simple. Developers compare the old version with the new version and get the difference. Then they make update patch file and publish it to users. After downloading the patch, users need to integrate the old version and the update patch to generate the new version. In addition, when developers make the patch, the latest version should be compared with each previous version because users may use different old versions.

Botmaster can force all bots to update to the latest version, which can decrease the maintenance costs for servers and clients. The bots can use two methods to configure push service accounts. One is to put the encrypted accounts in the class files and the other one is to store them in the manifest file. The latter is more efficient than the former, because the botmaster can perform efficient incremental update by sending a patch file that only changes the manifest file. Note that some push services do not support the latter update mechanism, such as Parse, Pushbots.

**Table 2.** Incremental Update

Push services	Update file	App size	Patch size
Pushwoosh	AndroidManifest.xml	362 KB	4 KB
Parse	PushActivity.java	581 KB	500–540 KB
PushBots	PushActivity.java	322 KB	280–300 KB
PushApps	PushActivity.java	331 KB	280–300 KB
ZYPush	AndroidManifest.xml	416 KB	4 KB
XGPush,GeXinPush, JPush(Integrated)	AndroidManifest.xml	1407 KB	10 KB

The size of patch file is compared with that of the original application file in the Table 2. It shows that incremental update with manifest file can dramatically decrease the update cost. But if the upgrade patches use the java file (i.e., bytecode generated from java files), the patch sizes are very large, which are almost as same as the original files.

## 5.2 Bot Communications

Most of push services support push API for app developers. For example, some push servers provide REST API (Representational State Transfer), which defines a set of push services as web services. Application servers or smartphones can use HTTP methods explicitly to perform pushing operations. It means a smartphone can not only receive messages, but also send pushing messages with REST support. It provides a mechanism for bots to send messages to botmaster or other bots through push services. Current push-styled botnets [10, 23] only support one-way communications, which is from botmaster to bots. In MUSE, bots can use bi-direction communications to send feedback reports to botmaster. This also enables bots to act as servants to construct botnet intermediate layer.

In the hybrid structure shown in Fig. 2, there are three types of nodes: C&C server, servant bot and client bot. We can establish a communication channel for each two of them and propose three channels according to Botnet Triple-Channel Model (BTM) [3]. In MUSE, there are three types of channels:

1. One-way channel from C&C server to client bot. Like the conventional message push mobile botnet [10, 23], C&C server directly uses push services to disseminate commands to client bots with this one-way channel.
2. Bi-direction channel between servant bot and client bot. We find that the SDKs provided by most push services are designed for PC instead of smartphone. Fortunately, they also provide HTTP Restful APIs, and hence mobile applications can send HTTP requests to realize the push function. This functionality enables all mobile bots to establish bi-direction communication channel. In other words, client bots can not only receive commands but also submit push requests.
3. One-way channel from bots to C&C server. Since there is no direct communication channel from bots to C&C server, one possible approach is to use

third-party servers, such as email or network disk [12], to upload data collected on victims. This channel can also utilize a short URL or URL-Flux and other existing technologies to enhance the robustness and stealthiness. After bots upload data to file servers, C&C server then downloads data from them.

In BTM, traditional C&C channel was substituted by three independent sub-channels, denoting as Command Download Channel (CDC), Registration Channel (RC) and Data Upload Channel (DUC), respectively. CDC is responsible for commands distribution. RC is responsible for collecting fundamental registration information. DUC is responsible for collecting and storing the uploaded information. In MUSE, BTM can be completely enhanced with our proposed bi-direction channels. In particular, push servers are used to construct CDC. File servers can help bots upload registration information as RC and steal sensitive information on victims as DUC. The three channels are listed and compared with BTM in Table 3.

**Table 3.** Triple channel model

Channel	BM $\leftrightarrow$ SB <sup>a</sup>	BM $\leftrightarrow$ CB <sup>b</sup>	SB $\leftrightarrow$ CB <sup>c</sup>
Registration channel	File servers		
Command download channel	Push servers		
Data upload channel	File servers		File servers

<sup>a</sup>Between botmaster and servant bot.

<sup>b</sup>Between botmaster and client bot.

<sup>c</sup>Between servant bot and client bot.

## 6 Evaluation

### 6.1 Stealthiness

The stealthiness is critical for the botnet to circumvent the detection. We first use simulation to evaluate the stealthiness of MUSE. In this experiment, six push servers are used to construct MUSE C&C and 600 bots are divided into 3 groups. Botmaster will send 200 commands to each bot. There are totally 120,000 push messages sent from the push servers. The size of each push message containing commands is assumed to be 1 KB. When perform pushing operation, the botmaster selects one push server for each bot group according to the proposed dynamic WRR algorithm. The weight vector for six push servers is initialized with path delay measurement results. In each round, the botmaster disseminates 10 consecutive commands via six push servers and the traffic volume of each server is recorded in Fig. 7. By observing the traffic volume of six push servers in each round, we find that push requests disperse among six servers according to their weight. It is worth noting that since the pushed botnet traffic varies in different rounds, it is not easy for the defender to find botnet traffic patterns.

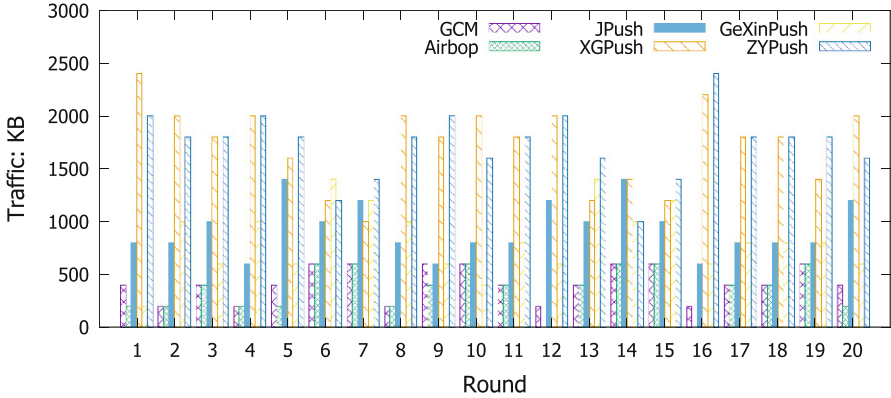


Fig. 7. Botnet traffic disperse among different push servers (Color figure online).

At the same time, the dynamic WRR algorithm can achieve good load balancing. Moreover, MUSE can make full use of all push services according to their performance, thus improving the scalability of the botnet.

### 6.2 Controllability

The controllability means whether the bot can receive commands efficiently from the botmaster. The smaller delay of disseminations will lead to the better controllability of botnet. Controllability is important for the botnet, especially during period of performing some synchronized attacks.

We first measure the delay of GCM-based mono-push-server botnet for a whole day in 6 different regions. The measurement is performed every hour and there are total 24 rounds for one day. The delay specifies how long it takes for a push command to travel across the network from botmaster to a bot. We repeat each experiment 5 times and take statistical measures.

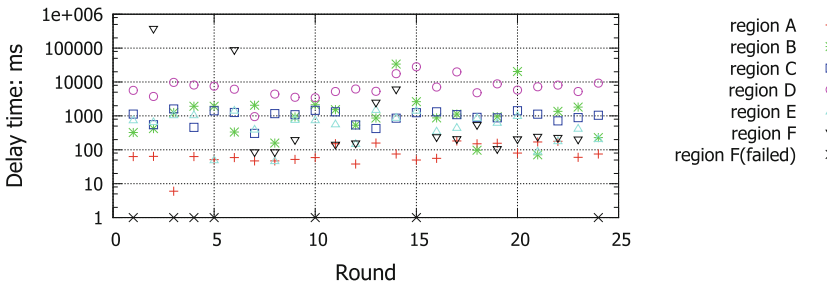


Fig. 8. Command delay for mono-push-server botnet (Color figure online).

Figure 8 shows the performance of mono-push-server botnet is not stable in some regions. Especially in region F (Nanjing, China), some messages delay is longer than 10s and even 7 messages are lost (note the logarithmic scale on the y-axis). The poor pushing performance is fatal for a mono-push-server botnet because it means the commands can not be reliably sent to bots.

Then we evaluate delay of command dissemination for MUSE. The experiment is conducted in region F where GCM performance is not stable. We compare the dynamic weight round-robin algorithm, which is implemented in MUSE, with the other two scheduling algorithms: round-robin scheduler and random scheduler. The experiments are conducted once per hour for a whole day and 24 rounds results are recorded. In each round the botmaster sends 10 commands with three different scheduling algorithms. As illustrated in the Fig. 9, the performance of the weight round-robin is superior to the other two. In general, MUSE with weigh round-robin algorithm can disseminate commands efficiently and outperform the mono-push-server botnet in stability and controllability.

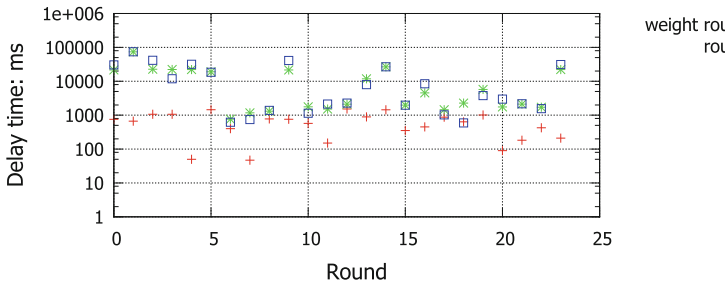


Fig. 9. Command delay for MUSE (Color figure online).

## 7 Conclusion

We propose MUSE to construct mobile botnets using multiple message push servers. It can avoid the single point of failure problem in mono-push-server structure and efficiently improve the robustness. MUSE has new architectures and group management to improve the scalability and controllability of botnet. Moreover, to keep stealthy, we design new algorithms for botmaster to select push servers and servant bots. We have implemented MUSE and the experimental results demonstrate MUSE's capability. In future work, we will investigate how to defend against such mobile botnets.

**Acknowledgement.** This work was supported in part by the Hong Kong GRF/ECS (No. PolyU 5389/13E), the National Natural Science Foundation of China (No. 61202396, 61202353, 61272084), the HKPolyU Research Grant (G-YBJX).

## References

1. Anagnostopoulos, M., Kambourakis, G., Gritzalis, S.: New facets of mobile botnet: architecture and evaluation. *Int. J. Inf. Secur.* 1–19 (2015)
2. Chen, W., Yin, C., Zhou, S., Yan, X.: Cloud-based mobile botnets using multiple push servers. In: 2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 183–189. IEEE (2015)
3. Cui, X., Fang, B., Liao, P., Liu, C.: Advanced triple-channel botnets: model and implementation. In: Proceedings of CCS (2012)
4. Cui, X., Fang, B., Yin, L., Liu, X., Zang, T.: Andbot: towards advanced mobile botnets. In: Proceedings of LEET (2011)
5. Eslahi, M., Rostami, M.R., Hashim, H., Tahir, N.M., Naseri, M.V.: A data collection approach for mobile botnet analysis and detection. In: 2014 IEEE Symposium on Wireless Technology and Applications (ISWTA), pp. 199–204, September 2014
6. Handy, M., Haase, M., Timmermann, D.: Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In: Proceedings of IEEE MWCN (2002)
7. Hua, J., Sakurai, K.: Botnet command and control based on short message service and human mobility. *Comput. Netw.* **57**(2), 579–597 (2013)
8. Karim, A., Shah, S.A.A., Salleh, R.: New perspectives in information systems and technologies. In: Rocha, Á., Correia, A.M., Tan, F.B., Stroetmann, K.A. (eds.) *Mobile Botnet Attacks: A Thematic Taxonomy*, vol. 2, pp. 153–164. Springer International Publishing, Cham (2014). ISBN=978-3-319-05948-8
9. Khattak, S., Ramay, N., Khan, K., Syed, A., Khayam, S.: A taxonomy of botnet behavior, detection, and defense. *IEEE Commun. Surv. Tutor.* **16**(2), 898–924 (2014)
10. Lee, H., Kang, T., Lee, S., Kim, J., Kim, Y.: Punobot: mobile botnet using push notification service in android. In: Kim, Y., Lee, H., Perrig, A. (eds.) *WISA 2013*. LNCS, vol. 8267, pp. 124–137. Springer, Heidelberg (2014)
11. Luo, X., Chan, E., Zhou, P., Chang, R.: Robust network covert communications based on TCP and enumerative combinatorics. *IEEE Trans. Dependable Secure Comput.* **9**(6), 890–902 (2012)
12. Luo, X., Zhou, H., Yu, L., Xue, L., Xie, Y.: Characterizing mobile \*-box applications. *Comput. Netw.* **103**, 228–239 (2016)
13. Mullaney, C.: Android.Bmaster: a million-dollar mobile botnet (2012). <http://goo.gl/sxpoNN>
14. Mulliner, C., Seifert, J.P.: Rise of the iBots: owning a telco network. In: Proceedings of IEEE MALWARE (2010)
15. Pieterse, H., Olivier, M.: Design of a hybrid command and control mobile botnet. In: Proceedings of the 8th International Conference on Information Warfare and Security, ICIW 2013, p. 183. Academic Conferences Limited (2013)
16. Rodríguez-Gómez, R., Maciá-Fernández, G., García-Teodoro, P.: Survey and taxonomy of botnet research through life-cycle. *ACM Comput. Surv.* **45**(4), 45 (2013)
17. Silva, S., Silva, R., Pinto, R., Salles, R.: Botnets: a survey. *Comput. Netw.* **57**(2), 378–403 (2013)
18. Singh, K., Sangal, S., Jain, N., Traynor, P., Lee, W.: Evaluating bluetooth as a medium for botnet command and control. In: Kreibich, C., Jahnke, M. (eds.) *DIMVA 2010*. LNCS, vol. 6201, pp. 61–80. Springer, Heidelberg (2010)
19. Traynor, P., Lin, M., Ongtang, M., Rao, V., Jaeger, T., McDaniel, P., La Porta, T.: On cellular botnets: measuring the impact of malicious devices on a cellular network core. In: Proceedings of ACM CCS (2009)

20. Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer-to-peer botnet. *IEEE TDSC* **7**(2), 113 (2010)
21. Wu, Z., Gianvecchio, S., Xie, M., Wang, H.: Mimimorphism: a new approach to binary code obfuscation. In: *Proceedings of ACM CCS* (2010)
22. Zeng, Y., Shin, K.G., Hu, X.: Design of SMS commanded-and-controlled and P2P-structured mobile botnets. In: *Proceedings of WiSec* (2012)
23. Zhao, S., Lee, P., Lui, J., Guan, X., Ma, X., Tao, J.: Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service. In: *Proceedings of ACSAC* (2012)
24. Zhou, Y., Jiang, X.: An analysis of the AnserverBot trojan (2011). <http://goo.gl/Dz8qda>
25. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: *Proceedings of IEEE Symposium on Security and Privacy* (2012)



# A Survey on the Cyber Attacks Against Non-linear State Estimation in Smart Grids

Jingxuan Wang<sup>1</sup>, Lucas C.K. Hui<sup>1</sup>, S.M. Yiu<sup>1(✉)</sup>,  
Xingmin Cui<sup>1</sup>, Eric Ke Wang<sup>2</sup>, and Junbin Fang<sup>3</sup>

<sup>1</sup> Department of Computer Science, The University of Hong Kong, Hong Kong, China  
{jxwang, hui, smyiu, xmcui}@cs.hku.hk

<sup>2</sup> Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China  
wk\_hit@hitsz.edu.cn

<sup>3</sup> Department of Optoelectronic Engineering, Jinan University, Guangzhou, China  
junbinfang@gmail.com

**Abstract.** It is well-known that critical infrastructures would be targets for cyber attacks. In this paper, we focus on smart grids. In a smart grid system, information from smart meters would be used to perform a state estimation in real time in order to maintain the stability of the system. A wrong estimation can lead to disastrous consequences (e.g. suspension of electricity supply or a big financial loss). Unfortunately, quite a number of recent results showed that attacks on this estimation process are feasible by manipulating readings of only a few meters. In this paper, we focus on *nonlinear* state estimation which is a more realistic model and widely employed in a real smart grid environment. We summarize and categorize all possible attacks, and review the mechanisms behind. We also briefly talk about the countermeasures. We hope that the community would be able to come up with a better protection scheme for smart grids.

**Keywords:** Cyber-physical system · Security · Smart grids · Nonlinear state estimation

## 1 Introduction

A Cyber-Physical System (CPS) is a highly integrated system, which incorporates cyber components (i.e. computation, communication networks) and physical components (i.e. physical dynamics). Smart grid system is a typical example of CPS, with sensors (i.e. smart meters) as the physical entities and the SCADA (Supervisory Control And Data Acquisition) control system as the computational unit. Smart grid is envisioned to be the power grid infrastructure in the future and has been employed in many countries already. A smart grid can adjust its power flows of electricity in real time, even when an electrical transmission path is interrupted, to meet the requirements from both power suppliers and customers. Smart grid can also encourage competitions among power suppliers, which would lead to a more efficient means of energy production and a drop in the price of electricity [2].

In modern smart grid architectures, information from smart meters is sent to the SCADA software system via PLCs (programmable logic controllers) or RTUs

(remote terminal units) [18]. SCADA software can then process the received data and report the results to the operators for further decisions. One of the critical processes is the *state estimation*, which provides important information to maintain the system in a stable and secure state. Within the monitoring procedure, there is a bad data detection routine that will try to identify bad measurements and remove them from the system before performing the state estimation. There are two widely used power flow models when considering state estimation: alternative current (AC) power flow model and direct current (DC) power flow model. An AC power flow model is formulated by nonlinear equations with two types of variables: bus voltage magnitudes and phase angles, whereas a DC power flow model uses a linear model to approximate the AC power flow model [7, 25]. AC model is more realistic and widely employed in real smart grid systems.

However, to maintain the stability and security of a smart grid system is not trivial, despite the fact that safe-guard mechanisms have been employed, cyber attacks leading to large-scale blackouts still occurred. Examples include [18] in North American (2003) and [4] in Pakistan (2015). In these cases, attacking only a few components of legacy devices can already trigger the failure of the whole power system and lead to economic losses.

Smart grid security is a complicated subject and can be classified into at least five categories: Process Control Systems (PCS) security, smart meter security, state estimation security, smart grid communication protocol security and smart grid simulation for security analysis [2]. This paper focuses on **power system state estimation**. In the process of state estimation, it is supposed to have a bad data detection procedure to identify problematic readings from smart meters in order to protect the system from erroneous readings or attacks of injected data. However, existing mechanisms are not able to distinguish bad data from normal data if bad data is injected in a carefully planned effort.

To our best knowledge, this is the first survey paper to summarize such cyber attacks in the AC model. We first summarize the attacks, which affect the state estimation in smart grids and then show how to extend some attacks, which have already been implemented in the DC model, to the AC model. At the end, we briefly review existing countermeasures in the AC model.

## 1.1 Related Work

Although we focus on AC model in this paper, we will discuss some works in DC model, which is less complicated than the AC model, but could provide insights for the study of the AC model. Therefore, some significant works in the DC model are also reviewed. Vulnerability analysis of state estimation refers to the inherent weaknesses of state estimation to detect injected bias on SCADA data, which has attracted considerable interests in recent years [9, 10, 13–15, 17, 19, 26, 27, 30–32, 34–36]. In [15], they divide these analyses into two major directions: *single-period attacks* and *multi-period attacks*. Here, we further classify the related works into two categories (*unobservable state attacks* and *unobservable topology attacks*) (see Table 1).

**Table 1.** Classification of attacks against state estimation

Classification	(DC Model)	(AC Model)
<b>Single-period attacks</b>		
Unobservable state attacks	[14, 17, 19, 26, 27, 35]	[10, 31, 32]
Unobservable topology attacks	[9, 13, 30]	—
<b>Multi-period attacks</b>		
Multi-period attacks with unobservable state attacks	[15]	[15]
Multi-period attacks with unobservable topology attacks	[34, 36]	—

1. *Single-period Attacks*: A single attack or a series of attacks, which can be launched simultaneously. Single-period attacks can be divided into two types: unobservable state attacks and unobservable topology attacks.

Single-period attacks work on injecting bias of system states by manipulating the meter readings (measurements) [10, 14, 17, 19, 26, 27, 31, 32, 35]. [19] first proposed the false data injection attacks, which could introduce bias assuming that the attacker could obtain the knowledge of the power system topology. By studying the mathematical properties of the system matrix in the DC model, attack vectors were constructed, which could work in the system without being detected. Results following [19] provided insights into the vulnerability of power grid. [26] proposed to build attacks with incomplete information (i.e. admittances in transmission lines) due to the attacking ability and [35] constructed a false data attack with no knowledge of the system topology, which was more realistic in practice. The false data attacking strategies implemented in the AC model (nonlinear state estimation model) were discussed in [10, 32]. The relationship between the attackers and the control center was studied in [17, 27]. In [14], data framing attack was proposed and formulated as a quadratically constrained quadratic problem (QCQP) in the DC model. [31] extended data framing attacks in the AC model by maximizing the mean energy of normalized residues. The attacks presented in above works could be pigeonholed as the unobservable state attacks since only measurement states were manipulated.

Besides the state attacks, [9, 13, 30] considered unobservable topology attack, which achieved the goal of introducing system state bias by manipulating both the system topology (i.e. perturbing breakers) and measurements. [30] proposed a type of topology attack called leverage point attack, which made use of the properties of leverage points. [9] proposed an attack framework by manipulating breaker statuses, which overcame the difficulty involved in the modification of meter readings. However, works on topology attacks [9, 13, 30] only considered the DC model. A more realistic attack model, which could be applied in the AC model, should be considered. In this paper, some attack mechanisms in the DC model are extended to the AC model.

2. *Multi-period Attacks*: A series of attacks, which are launched sequentially and was first considered in [15, 34, 36]. [34, 36] studied this vulnerability by adopting a cascading failure simulator. However, the relationship between the

attack sequence and the performance (i.e. injecting bias on the system states) was not given. If this relation can be obtained, the strategy of constructing such attacks with good performance will be more efficient. In [15], the dynamic attack strategy where the attacker injected bias on the system states for real-time economic dispatch was investigated.

Consequence analysis refers to the analysis of effects caused by the attacks. Most related research works focused on the DC model while only a few have addressed the nonlinearity effects, which include [11, 15, 20, 29]. [11] studied the worst case impacts on behalf of the real-time locational marginal price. [29] gave the impacts of integrity attacks on real-time price signals received by consumers. [20] proposed a dynamic game between the attacker and the operator, in which the strategy is taken by both sides to maximize their profits. In [15], the potential damages of attacks were given.

The design of security measures refers to the detections of malicious attacks and protection schemes for the system. Among the more recent relevant works, we can divide them into two categories: *protection-based* and *detection-based*. The protection-based method is to protect some critical sensors, which includes [3, 5, 12, 13, 16, 33] in the AC model. In practice, the cost of protecting meters will be very high when a grid system contains millions of meters. The detection-based method is to develop the exceptional detection mechanisms, which includes [28] in the AC model.

## 1.2 Notations and Definitions

The notations that appear in the rest of this paper are listed in Table 2.

**Definition 1** [1]. *A critical measurement is one whose elimination from the measurement set will result in an unobservable system.*

**Definition 2** [1]. *A redundant measurement is the measurement, which is not critical.*

**Definition 3** (Definition 2.2, [13]). *Given the measurement  $z$  from a topology  $\mathcal{G}$ , an attack vector  $a$  to modify  $\mathcal{G}$  to  $\bar{\mathcal{G}}$  is said to be unobservable if  $z + a = h(x + c, \bar{\mathcal{G}})$ .*

**Definition 4** (Definition 3.1, [22]). *A leverage point of regression is a point which is far away from the bulk of the data points in the factor space, and its corresponding measurement is referred to as a leverage measurement.*

## 1.3 Organization

The rest of the paper is organized as follows. Section 2 gives the related background information, such as the power system network and measurement model, state estimation, attack models and security mechanisms are presented. Section 3 focuses on single-period attacks. We review the attack mechanisms and extend

**Table 2.** Notations

Notations	Description
$\tau$	Leverage score threshold
$p$	Network parameter
$s$	Breaker sensor measurements
$\mathcal{G}$	An undirected graph $\{\mathcal{N}, \mathcal{E}\}$ that represents the system circuit topology
$\mathcal{N}$	Set of buses
$\mathcal{E}$	Set of connected transmission lines
$z$	Vector of meter measurements
$x$	Vector of system state
$m$	Number of meter measurements
$n$	Number of system states
$v$	Vector of measurement errors
$h$	Nonlinear measurement functions
$H$	Jacobian matrix of $h(x)$
$\Sigma$	Covariance matrix of the measurement residues
$r$	Vector of measurement residuals
$K$	Hat matrix
$\mathcal{L}$	Set of leverage scores
$a$	Attack vector injected into the meter measurements
$b$	Attack vector injected into the breaker measurements
$w$	Weight on edge or node
$e_{ij}$	Edge from bus $i$ to bus $j$
$n_i$	Node on bus $i$
$H_P$	Jacobian matrix through PCA analysis
$S_A$	Set of meter measurements that will be manipulated by the attacker
$S_F$	Set of measurements without malicious data
$R_F$	$ S_F  \times m$ matrix with only the rows corresponding to $S_F$
$P_j$	Real power injection on bus $j$
$Q_j$	Reactive power injection on bus $j$
$\mathcal{A}$	Set of unobservable state attack vector $a$
$\mathcal{B}$	Set of unobservable topology attack vector $b$
$H_1$	Submatrix of $H$ with only the columns corresponding to the meters in $S_A$
$l$	Number of transmission lines

some attacks, which have already been implemented in the DC model, but not in the AC model. In Sect. 4, multi-period attacks are discussed. Security measures against existing attacks in the AC model are reviewed in Sect. 5. Finally, Sect. 6 concludes the paper and raises open problems in the related areas.

## 2 Smart Grid State Estimation

This section begins with a brief background on the network/measurement models, state estimation and bad data detection schemes in power grid system. Then we will describe how the state estimation works for estimating system states and how the bad data detection routine identifies bad data.

### 2.1 Network and Measurement Models

A power grid system is a network of connected buses by transmission lines [1]. When the system is under operation, control center will receive three types of data: network parameter, network topology data, and meter data. In this paper, the network parameter is denoted as  $p$ , which includes the branch susceptance data, the variances of meter measurements errors, etc. The topology data is a set of binary measurements from the breakers, which can be denoted as  $s = [s_1, s_2, \dots, s_l]^T$ . With the help of  $s$ , control center can observe the topology  $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ , where  $\mathcal{N}$  is the set of buses and  $\mathcal{E}$  is the set of connected transmission lines [13]. The meter data is denoted as  $z$ , which includes voltage magnitude measurements, power injection measurements, and power flow measurements.

In the absence of attacks and measurements noise, the meter measurements  $z = [z_1, z_2, \dots, z_m]^T$  are related to the system states  $x = [x_1, x_2, \dots, x_n]^T$ , where  $m$  is the number of measurements and  $n$  is the number of system states. The basic measurement model can be represented as

$$z = h(x, \mathcal{G}) + v, \quad (1)$$

where  $h_i(x, \mathcal{G})$  is a nonlinear function relating  $z_i$  to  $(x, \mathcal{G})$  and  $v = [v_1, v_2, \dots, v_m]^T$  denotes the vector of measurement errors. Furthermore,  $E(v) = 0$  and covariance matrix  $\Sigma = \text{diag}\{\sigma_1^2, \sigma_2^2, \dots, \sigma_m^2\}$ , where  $\sigma_i^2$  is the variances of  $v_i$ .

A power grid is said to be *observable* if the state vector (denoted as  $x$ ) can be uniquely estimated from the inputs (i.e.  $p, s, z$ ) [1]. Without loss of generality, we assume the system is observable for a period of time that its operation is normal.

### 2.2 State Estimator

A control center is used to monitor the devices in the power system. The control center will first observe the topology  $\hat{\mathcal{G}}$  as soon as it receives the network parameter  $p$  and network topology data  $s$ . When  $\hat{\mathcal{G}}$  is obtained, the system state can be estimated by the weighted least square (WLS) criterion, with the help of  $\hat{\mathcal{G}}$  and sensor measurements  $z$ . Then the estimated system state  $\hat{x}$  will be sent to the bad data detector. If it is suspected to be manipulated with bad data,  $\hat{x}$  will be forwarded to bad data identification scheme for further processing; otherwise, the system is said to be free of bad data and will be sent out of the state estimation process for power distributions. The above process is called a *generalized state estimation (GSE)* [24].

*State Estimator.* For state estimator using the AC power flow model, system states are commonly estimated by the iterative weighted least-square criterion (WLS) [1].

$$\begin{aligned} & \text{Min} \sum_{i=1}^m \Sigma_{ii}^{-1} r_i^2 \\ & \text{s.t. } r_i = z_i - h_i(\hat{x}, \hat{\mathcal{G}}), \quad i = 1, \dots, m. \end{aligned} \quad (2)$$

Considering the measurement functions are nonlinear, iterative algorithms are used. For every iteration,  $\Delta x$  can be obtained from  $\Delta z = H\Delta x + e$ , where  $H$  is the Jacobian matrix of  $h(\hat{x})$  and  $\Delta z$  is the residual vector. Then, the WLS estimator will be given by  $\Delta \hat{x} = G^{-1}H^T R^{-1}\Delta z$  and the estimated value of  $\Delta \hat{z}$ :

$$\Delta \hat{z} = H\Delta \hat{x} = K\Delta z \quad (3)$$

where  $K$  is called the *hat* matrix, for putting a hat on  $\Delta z$ . The diagonal elements of  $K$  is called the *leverage scores*, which can be collected in set  $\mathcal{L} = \{K_{ii} \mid i = 1, \dots, m\}$ .

### 2.3 Adversary Model

To launch an attack, an adversary must first exploit entry points, upon which the attacker can inject errors on the smart grid infrastructure (i.e. breaker sensor, meters). In practice, such adversary actions can be accomplished by seeking backdoors in the network perimeter and hijacking the VPN [23].

Let  $\bar{z}$  and  $\bar{s}$  represent the observed measurements and breaker sensor measurements, which may contain malicious data, respectively. The adversary model can be described as follows [13, 19],

$$\begin{aligned} \bar{z} &= z + a; \\ \bar{s} &= s + b \pmod{2}, \end{aligned} \quad (4)$$

where  $a$  and  $b$  are the attack vectors injected into the meter measurements and the breaker sensor measurements, respectively.

### 2.4 Bad Data Detection and Identification

*Bad Data Detection.* Bad measurements may be introduced due to minor physical errors or malicious attacks. The process of validating the topology and meter data is called bad data detection (BDD). The ‘‘residual principle’’ and  $\chi^2$ -Test is widely used [1]. After solving the WLS estimation problem, we can test whether

$$\sum_{i=1}^m \Sigma_{ii}^{-1} r_i^2 \geq \chi_{(m-n), p}^2, \quad (5)$$

where  $p$  is the value from the Chi-squares distribution table corresponding to confidence with probability  $p$  and  $(m - n)$  degrees of freedom. If yes, the bad

data alarm is triggered, and the identification can be accomplished by further processing residuals; otherwise, the system is said to be free of bad data.

*Iterative Bad Data Identification and Removal (BDIR)*. If BDD detects the existence of at least one meter with bad data, BDIR is invoked to identify the bad data entries and remove them from the system [8]. A widely used criterion for determining a bad data entry is the Largest Normalized Residual ( $r_N^{max}$ ) Test. At the  $k$ -th iteration, the state estimator uses ( $z^{(k)}$ ) as input. The measurement residual of the  $i$ -th measurement can be normalized by

$$r_i^{N(k)} = \frac{|r_i^{(k)}|}{\sqrt{\Sigma_{ii}^{(k)}}}. \quad (6)$$

The normalized residual vector follows a standard normal distribution with zero mean and unit covariance. Hence the meter with the largest  $r_i^{N(k)}$  will be treated as a meter with bad data and then be removed from the system.

### 3 Single-Period Attacks

As is described in Sect. 1, the research works done against power grid state estimation covered two major directions of attacks: single-period attacks and multi-period attacks. We talk about the single-period attacks in this section.

For single-period attacks, unobservable state attacks and unobservable topology attacks are considered. Unobservable state attacks aim at injecting errors on meter measurements, and unobservable topology attacks aim at injecting errors on both meter measurements and breaker measurements. Table 1 lists the research works done on single-period attacks.

#### 3.1 Type I. Unobservable State Attacks

This subsection reviews attack mechanisms for unobservable state attacks in the AC model. As explained in Sect. 1, state attack assumes that one attacker can access a set of measurements and can control/manipulate them, whereas the breaker data is intact. In other words, the attacker can only manipulate measurements ( $z$ ) and cannot modify the system topology ( $\mathcal{G}$ ). Examples of unobservable state attacks include false data injection attack [10, 32] and data framing attack [14].

**False Data Injection Attack (FDIA)**. FDIA can potentially inject errors on system states by only modifying the sensor data. Assuming that the meter data ( $z$ ) to be sent to the control center is known, and it can be altered to values specified by the attacker. Moreover, the attacker should obtain the topology of the system  $h(\cdot)$ . The data collected by the control center will be used to estimate the system states ( $x$ ) and make operational decisions with possibly far-ranging consequences. The main idea of constructing attack vector  $a$  is to set  $a$  to a



calculated sparse  $m \times 1$  vector. If the breaker data is intact, the attack model will become,

$$\bar{z} = h(x + c) + a + v, \quad (7)$$

where  $c$  is the error injected after adding the attack  $a$ .

[10, 32] worked on false data injection attacks in the AC model. The work in [10] proposed the attacking strategy with the topographical analysis. They came up with a procedure for determining  $S_A$ , which is the set of meter measurements that will be manipulated by the attacker. In their strategy, the power grid is treated as a weighted graph, in which an edge denotes a transmission line and a node represents a bus. After assigning the weights ( $w$ ) on edges ( $e_{ij}$ )/nodes ( $n_i$ ), the adversary can choose a subgraph, which can minimize the number of measurements to be attacked:

$$\min_{S_A} \left( \sum_{e_{ij} \in S_A} w(e_{ij}) + \sum_{n_i \in S_A} w(n_i) + w_p \right), \quad (8)$$

where  $w_p = 1$ , if all system states are measured at bus  $p$ , otherwise  $w_p = 0$ . The meters in the obtained subgraph constitute to  $S_A$ . Having determined which measurements to alter, to what values they are changed (nonzero elements in  $a$ ) can be systematically calculated by,

$$a_i = h_i(x + c) - z_i \quad \text{if } i \in S_A. \quad (9)$$

Moreover, the algebraic condition given in Eq.(9) can be transformed to an observability condition [32].

**Theorem 1** (Theorem 1, [32]). *A set of unobservable attacks  $\mathcal{A}$  exists iff there exists another attack vector  $a_0$  (either  $a_0 \in \mathcal{A}$  or  $a_0 \notin \mathcal{A}$ ), which can make the system observable.*

From the attacker's perspective, a preferable approach is to launch the attack without the knowledge of the topology or the meter measurements, which is easier for the attackers. Yu et al. in [35] solved this problem by proposing a principal component analysis (PCA) approach without the knowledge of topology. The main idea is to transform the meter measurements into a linear combination of a vector with uncorrelated components, denoted as  $H_P$ . Then the unobservable attack can be calculated by the product of this linear combination and an arbitrary vector  $c$  [35]:

$$a = H_P c. \quad (10)$$

One caveat of this approach is that the attack model is constructed in the DC model. However, in the AC model, we cannot simply compute the unobservable attack vector  $a$  in the way of Eq. (10) since  $Hc \neq h(x + c) - h(x)$ . One open problem is whether FDIA can be constructed in the AC model without the topology knowledge.

**Data Framing Attacks (DFA).** Data framing attack aims at misleading the control center about the source of a state attack. It was first proposed in the DC model by [14] and then be extended to the AC model by [31].

Unlike FDIA, DFA does not intend to bypass BDD. Instead it tries to confuse BDIR to remove “clean” measurements (those without malicious data) while keeping the malicious ones within the system, which finally will perturb the estimated system states. After  $i$  iterations of BDIR,  $2i$  measurements will be removed out of the system. Specifically, in the  $k$ -th process of BDIR, if  $P_j$  is identified as a bad measurement, the rows of  $P_j$ ,  $Q_j$  and the rows of  $h(\cdot)$  that correspond to  $P_j$  and  $Q_j$  will be removed<sup>1</sup>. And the updated measurement vector and measurement functions, denoted as  $z^{(k+1)}$  and  $h^{(k+1)}$ , will be returned for the next iteration. Above all, the attack model can be represented as,

$$\bar{z}^{(k)} = h^{(k)}(x + c) + a^{(k)} + v^{(k)}, \quad (11)$$

where  $a^{(k)}$ ,  $v^{(k)}$  are the  $(m - 2k) \times 1$  vectors and  $h^{(k)}(\cdot)$  is a vector of  $(m - 2k)$  nonlinear functions.

Kim and Tong in [14] proposed to maximize the expected energy of normalized residues at framed meters ( $S_F$ ). Framed meters are the meters, which are not manipulated by the adversary. Thus, the measurements in  $S_F$  will have a higher probability to be recognized as “bad” (with malicious data). Quadratically constrained quadratic program (QCQP) framework can be used to find the solutions of constructing DFA. Although [14] maximized the expected energy of normalized residues, they did not consider to minimize the number of meters to be attacked, which may require more resources on behalf of the adversary. Besides, the solution in [14] cannot be directly adapted to the AC model. This is because [14] simplified  $\Sigma$  as a constant matrix. However, in the AC model,  $\Sigma$  varies with  $\frac{\partial h(x)}{\partial x}$ , which will lead to inaccurate results.

Wang et al. in [31] extended the work in [14] to the AC model. In order to construct DFA in the AC model, the attacker should obtain the knowledge of meter measurements and the topology. An upper bound for the residual of the leverage measurements is first explored:

**Theorem 2** (Theorem 1, [31]). *For any leverage measurement in the power grid system, its residual yields an upper bound  $(1 - \tau) \sum_{j=1}^m (z_j + a_j)^2$ , where  $\tau$  is the leverage score threshold.*

Theorem 2 implies that when the parameter  $\tau$  closes to 1, the residual of the leverage measurement varies towards 0. Thus, the errors on these measurements will have a lower probability to be detected. At the same time, the risk of being recognized as a “bad” meter (by BDD) is transferred to other redundant meters. The strategy is to add all leverage measurements in  $S_A$ , and then the problem of finding an optimal attack vector  $a$  can be formulated as follows,

---

<sup>1</sup> As given in Table 2,  $P_j$  and  $Q_j$  refer to the real power injection and reactive power injection on bus  $j$ .

$$\begin{aligned}
\max_a \quad & \sum_{i=1, i \notin S_A}^m (1 - \mathcal{L}_i^{(N)}) \sum_{j=1}^m (z_j + a_j)^2 \\
\text{s.t.} \quad & \|a\|_2^2 = \lambda, a \in \mathbb{R}(H_1) \cap \mathcal{A}
\end{aligned} \tag{12}$$

where  $N$  is the total number of iterations in the state estimation process,  $\lambda \in \mathcal{R}$ ,  $H_1$  denotes the submatrix of  $H$  obtained by retaining only the columns corresponding to the meters in  $S_A$  and  $\mathcal{A} = \{a \in \mathcal{R}^m : a_i \neq 0, i \in S_A\}$ . Note that the upper bound described in Theorem 2 can be further reduced by adding more constraints that are easy to implement in practice.

### 3.2 Type II. Unobservable Topology Attacks

Different from unobservable state attacks, unobservable topology attack is to disorder the topology estimate (by perturbing breakers), as well as inject errors on the meter measurements. Thus, it will lead to the disorder of state estimation. In other words, the topology attack also considers the manipulations of power network topology data.

Kim and Tong in [13] first pointed out the possibility of a topology attack. Definition 3 (in Sect. 1.2) provides the conditions, which guarantee the feasibility for searching an unobservable topology attack. Though the attacking strategy in [13] only considered the DC model, we can mathematically extend it to the AC power flow model. Suppose that the adversary modifies breaker measurements and meter measurements to  $\bar{\mathcal{G}}$  and  $\bar{z}$ , respectively. The condition for this topology attack is stated in Theorem 3. Though such nonlinear condition is difficult to check [13], an optimal attack  $a$  can still be obtained by heuristic methodologies.

**Theorem 3.** *Given the measurement data  $z$ , estimated topology  $\hat{\mathcal{G}} = \{\mathcal{N}, \mathcal{E}\}$  and nonlinear functions  $h(\cdot)$ . If  $a$  is the solution of the following problem in 13, which is undetectable.*

$$a = \underset{a \in \mathcal{M}}{\operatorname{argmin}} \left\| z + a - h(\hat{x}, \hat{\mathcal{G}}) \right\|^2, \tag{13}$$

where  $\mathcal{M} \triangleq \{h(\hat{x}, \hat{\mathcal{G}}) - h(\hat{x}, \hat{\mathcal{G}}) : \hat{\mathcal{G}} = \{\mathcal{N}, \bar{\mathcal{E}}\}, \bar{\mathcal{E}} \not\subseteq \mathcal{E}\}$  and  $\bar{\mathcal{E}}$  is the manipulated set of connected transmission lines.

*Proof.* Without loss of generality, we assume the attacker alters the estimated topology from  $\hat{\mathcal{G}}$  to  $\hat{\mathcal{G}}$ . For the line addition attack and other measurement availabilities, similar argument can be made. Suppose there exists an undetectable attack  $a$ , which satisfies  $a \in \mathcal{M}$ , the  $\mathcal{J}(\hat{x})$ -test statistic at the control center is upper bounded as,

$$\begin{aligned}
\left\| z + a - h(\hat{x}, \hat{\mathcal{G}}) \right\|^2 &= \left\| z + h(\hat{x}, \hat{\mathcal{G}}) - h(\hat{x}, \hat{\mathcal{G}}) - h(\hat{x}, \hat{\mathcal{G}}) \right\|^2 \\
&= \left\| z - h(\hat{x}, \hat{\mathcal{G}}) \right\|^2 \leq \chi_{m-n,p}^2.
\end{aligned}$$

Thus, the right hand of Eq. (13) has  $\chi_{m-n,p}^2$  distribution, which cannot be detected by BDD. Hence, the attack  $a$  is said to be undetectable.  $\square$

**Leverage Point Attacks.** Tan et al. in [30] presented one unobservable topology attack in the DC model, called leverage point attacks (LPA). The attacking strategy of constructing LPA is different from [13]. They used the key feature of leverage point, that is to say, the residue of the measurement corresponds with leverage point is very small even it is injected with a very large error [22]. Their attacking strategy is first to manipulate the network parameter  $p$ , which will increase  $K_{ii}$ . Then the premeditated error  $a_i$  added to this measurement  $z_i$  can not be detected. The principles, which will increase the value of  $K_{ii}$ , are discussed in Theorem 4.

**Theorem 4** (Theorem 3, [30]). *Given the hat matrix  $K = H(H^T H)^{-1} H^T$ , the  $i$ -th diagonal element of  $K$  satisfies the following condition,*

$$(1 - K_{ii})^2 \leq \frac{\left\| \begin{bmatrix} H_p \\ H_f \end{bmatrix} \right\|_2^2}{\|H_i^T\|_2^2}, \tag{14}$$

where  $H_i$  is the  $i$ -th row of  $H$  and  $H$  is partitioned as  $[H_p H_i H_f]^T$ .

The attacking strategy in [30] can also be applied in the AC model by adding one consideration: the matrix  $K$  varies with the system states.

The unobservable topology attacks in [13,30] consider to inject the malicious data on both breakers and meter measurements. These attacks may be formidable to achieve as they require a high attacking ability for the adversary. To solve this problem, Deka et al. in [9] proposed the topology attack only by manipulating breaker statuses, which overcomes the difficulty involved in the modification of meter readings. The linear functions (DC model) are considered in this work. However, it will lead to inaccuracies when considering the nonlinear models. Thus, a more realistic attack mechanism, which collaborates with the nonlinear functions (in AC model), is a possible open problem.

## 4 Multi-period Attacks

This section reviews attack mechanisms for multi-period attacks in the AC power flow model. A multi-period attack assumes that a sequence of attacks can be launched. Two kinds of attacks are reviewed in this section: sequential attacks and dynamic DoS attacks. We will summarize the characteristics and striking requirements for these attacks. Table 1 includes the research works done on multi-period attacks. A multi-period attack assumes that the attacks (either state attacks or topology attacks) can be injected sequentially according to a carefully designed time sequence. The adversary model can be represented as follows.

$$\begin{aligned} \bar{z}^{(t)} &= z^{(t)} + a^{(t)}, a^{(t)} \in \mathcal{A}; \\ \bar{s}^{(t)} &= s^{(t)} + b^{(t)} \pmod{2}, b^{(t)} \in \mathcal{B}, \end{aligned} \tag{15}$$

where  $a^{(t)}$  represents the meter data modification in the  $t$ -th period by the adversary and  $b^{(t)}$  represents the breaker sensor measurements modification in the  $t$ -th period by the adversary. Above all, the multi-period attacks can be divided into two forms:

- (1) *Unobservable state attacks in multi-periods*: the sequence of attacks is only composed of state attacks  $\mathbb{A}$ , where  $\mathbb{A} = \{a^{(1)}, a^{(2)}, \dots, a^{(t)} | a \in \mathcal{A}\}$ ;
- (2) *Unobservable topology attacks in multi-periods*: the sequence of attacks is only composed of topology attacks  $\mathbb{B}$ , where  $\mathbb{B} = \{b^{(1)}, b^{(2)}, \dots, b^{(t)} | b \in \mathcal{B}\}$ ;

**Dynamic DoS Attacks.** The dynamic DoS attacks, proposed in [15], belong to the first form. [15] aims at making the solution of real-time economic dispatch (RTED) [6] an empty set. Since RTED can find the optimal generation adjustments to satisfy demands of the next time period, lack of RTED solution will lead to no future generation schedules. It forces to use more high-price fast-ramping generators.

**Sequential Attacks.** Sequential attacks, proposed in [34, 36], belong to the second form. Their attacking strategy is to build a sequential cascading failure simulator. Suppose there are  $l$  transmission lines, then we can have  $C_l^k$  choices of a  $k$ -link attack sequence ( $k \leq l$ ) and implement these  $k$  attacks sequentially. When all attacks are launched, the simulator will quit, and the damage can be evaluated. The simulation results in [36] show that the combination of  $k$ -link sequential attacks will lead to larger damages than the single-period attacks in the DC model. However, they did not give simulations in the AC model, whether the AC model can render the same results needs to be verified in the future study.

## 5 Countermeasures Against Cyber Attacks

The work reviewed in this section covers countermeasures against single-period attacks and multi-period attacks. For each kind of attack, we will review the countermeasures based on protection-based and detection-based technique (see Table 3).

### 5.1 Countermeasures Against Single-Period Attacks

One protection-based countermeasure is to define a minimum set of line flow and bus injection sensors, which can make the grid observable. [3] gave a method to find a set of sensors. If these sensors can be well protected, no unobservable state attack can be launched. The placement of protected phasor measurement units (PMUs) was studied [16]. PMUs can be treated as sensors with a high-security level, which can provide more accurate measurements. Kim and Poor in [16] proposed a fast algorithm that strategically places PMUs at some selected buses against unobservable state attacks. The results in [16] show that PMUs placed

**Table 3.** Countermeasures against attacks in the AC model and work done

Classification	Work done (Protection-based)	Work done (Detection-based)
<b>Single-period attacks</b>		
Unobservable state attacks	[13],[3,12,16]*	[5,21,28,33]
Unobservable topology attacks	[13], [12]*	—
<b>Multi-period attacks</b>		
Unobservable state attacks in multi-periods	[13],[3,12,16]*	[28]
Unobservable topology attacks in multi-periods	[13], [12]*	—

\* Though the works were implemented in DC model, it is likely that they can be applied to AC model.

on a third of buses can prevent the unobservable state attacks. Furthermore, Kim and Tong in [13] presented a graph-theoretical condition that can protect the system free of unobservable topology attacks if the condition is satisfied. In particular, [12] proved that the protection-based strategy (implemented in the DC model) that prevents unobservable topology attacks can prevent unobservable state attacks. In particular, an interesting conclusion from [12] tells that any protection-based countermeasures that prevent unobservable topology attacks can also prevent unobservable state attacks. Given the complexity of the nonlinear models, whether the algorithms in [3,12,16] are effective in the AC models still needs to be explored.

Besides protection-based countermeasures, detection-based countermeasures, which can help figure out bad data, are also studied [5,21,28,33]. In current literature survey, most works focus on combating FDIA, which is one kind of unobservable state attacks. As FDIA makes use of the vulnerability, which the residual is calculated in  $l_2$ -norm within BDD mechanism, a good solution is to seek for improvements over normal BDD. Sou et al. in [28] proposed procedure, which is similar to the normal BDD described in Sect. 2.4, except that the decision threshold  $r$  is calculated based on the reactive power measurement residual [28]. Manandhar et al. in [21] used Euclidean detector instead of  $\chi^2$  detector in normal BDD. Alternatively, Gu et al. in [5] proposed a new type of BDD, which used Kullback-Leibler distance (KLD) indices between the measurement data in current time stamp and the history data. Xu et al. in [33] provided an iterative mixed  $l_1$  and  $l_2$  convex program to detect if there exists the bad data.

## 5.2 Countermeasures Against Multi-period Attacks

As is explained in Sect. 4, the attacker can manipulate measurements from a set of sensors over time, which will bring extensive damages, such as disorder the contingency analysis [36]. Thus, the studies of countermeasures against these attacks are very important.

Note that works of the protection-based countermeasures against single-period attacks can be applied to defend against multi-period attacks. The same

applies to detection-based countermeasures. In particular, Sou et al. in [28] studied the scenario of multiple-period attacks with state attacks. They proposed a protection-based countermeasure. Multiple reactive power measurement residuals in a series of time stamps are calculated in the form of a vector, denoted as  $R_Q$ . The alarm will be triggered if the maximum element in  $R_Q$  is larger than the chosen threshold.

However, when considering *hybrid attacks in multi-periods*, the combination of various defense methods (against unobservable topology attacks and state attacks) should be considered.

## 6 Conclusions

The power system state estimation model used in smart grid system is at risk of cyber attacks. We reviewed a variety of single-period attacks (i.e. state attacks, topology attacks) and the multi-period attacks, which is a series of single-period attacks being launched sequentially. Feasibility conditions for these cyber attacks are presented. In addition, we also review the countermeasures that aim at disabling existing cyber attacks. Especially, the discussions of associated cost are reviewed when considering the limited security resources (i.e. using as few PMUs as possible).

While a great amount of efforts has been made in working on cyber attacks against non-linear state estimation in smart grids, some problems are left for further studies.

1. Yu et al. in [35] proposed an approach in the DC model, which can solve the problem of constructing FDIA without the knowledge of topology. An open question is whether FDIA can be constructed in the AC model without obtaining the topology.
2. Most studies of unobservable topology attacks focus on the DC model. However, more attention should be paid to the more widely employed system with the AC model.
3. In the DC model, the combination of  $k$ -link sequential attacks can lead to larger damages than the single-period attacks [36]. However, the simulations in the AC model are not given. An open problem is whether the AC model can render the same results.
4. Existing countermeasures can defend one or two attacks. However, no work can act against all existing cyber attacks (i.e. hybrid attacks in multi-periods) in the AC model.

**Acknowledgments.** The work described in this paper was partially supported by the HKU Seed Fundings for Applied Research 201409160030; HKU Seed Fundings for Basic Research 201311159149 and 201411159122; National Natural Science Foundation of China (61572157, 61401176, 61402136), PRC; Shenzhen Strategic Emerging Industry Development Foundation (JCYJ2015040316 1923509 and JCYJ20150617155357681), PRC, National High Technology Research and Development Program of China (2015AA016008), Projects of International Cooperation

and Exchanges NSFC (61361166006), China, NSFCRGC Joint Research Scheme (N.HKU 72913), Hong Kong, Natural Science Foundation of Guangdong Province, China (2014A030310205, 2014A030313697), and Excellent Young Teachers Program of Guangdong High Education, China (YQ2015018), and China State Scholarship Fund.

## References

1. Abur, A., Exposito, A.G.: Power System State Estimation: Theory and Implementation. CRC Press, New York (2004)
2. Baumeister, T.: Literature review on smart grid cyber security. Collaborative Software Development Laboratory at the University of Hawaii (2010)
3. Bobba, R.B., Rogers, K.M., Wang, Q., Khurana, H., Nahrstedt, K., Overbye, T.J.: Detecting false data injection attacks on dc state estimation. In: Preprints of the First Workshop on Secure Control Systems, CPSWEEK, vol. 2010 (2010)
4. Burke, S., Schneider, E.: Enemy number one for the electric grid: mother nature. SAIS Rev. Int. Aff. **35**(1), 73–86 (2015)
5. Chaojun, G., Jirutitijaroen, P., Motani, M.: Detecting false data injection attacks in ac state estimation. IEEE Trans. Smart Grid **6**(5), 2476–2483 (2015)
6. Choi, D.H., Xie, L.: Ramp-induced data attacks on look-ahead dispatch in real-time power markets. IEEE Trans. Smart Grid **4**(3), 1235–1243 (2013)
7. Coffrin, C., Van Hentenryck, P.: A linear-programming approximation of AC power flows. INFORMS J. Comput. **26**(4), 718–734 (2014)
8. Cutsem, T.V., Ribbens-Pavell, M., Mili, L.: Hypothesis testing identification: a new method for bad data analysis in power system state estimation. IEEE Trans. power Apparatus Syst. **11**, 3239–3252 (1984)
9. Deka, D., Baldick, R., Vishwanath, S.: Attacking power grids with secure meters: the case for breakers and jammers. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 646–651. IEEE (2014)
10. Hug, G., Giampapa, J.A.: Vulnerability assessment of AC state estimation with respect to false data injection cyber-attacks. IEEE Trans. Smart Grid **3**(3), 1362–1370 (2012)
11. Jia, L., Thomas, R.J., Tong, L.: On the nonlinearity effects on malicious data attack on power system. In: Power and Energy Society General Meeting, IEEE 2012, pp. 1–8. IEEE (2012)
12. Kim, J., Tong, L.: On phasor measurement unit placement against state and topology attacks. In: 2013 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 396–401. IEEE (2013)
13. Kim, J., Tong, L.: On topology attack of a smart grid: undetectable attacks and countermeasures. IEEE J. Sel. Areas Commun. **31**(7), 1294–1305 (2013)
14. Kim, J., Tong, L., Thomas, R.J.: Data framing attack on state estimation. IEEE J. Sel. Areas Commun. **32**(7), 1460–1470 (2014)
15. Kim, J., Tong, L., Thomas, R.J.: Dynamic attacks on power systems economic dispatch. In: 2014 48th Asilomar Conference on Signals, Systems and Computers, pp. 345–349. IEEE (2014)
16. Kim, T.T., Poor, H.V.: Strategic protection against data injection attacks on power grids. IEEE Trans. Smart Grid **2**(2), 326–333 (2011)
17. Kosut, O., Jia, L., Thomas, R.J., Tong, L.: Malicious data attacks on smart grid state estimation: attack strategies and countermeasures. In: 2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 220–225. IEEE (2010)



18. Liscouski, B., Elliot, W.: Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations. A report to US Department of Energy, 40(4) (2004)
19. Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **14**(1), 13 (2011)
20. Ma, J., Liu, Y., Song, L., Han, Z.: Multiact dynamic game strategy for jamming attack in electricity market. *IEEE Trans. Smart Grid* **6**(5), 2273–2282 (2015)
21. Manandhar, K., Cao, X., Hu, F., Liu, Y.: Detection of faults and attacks including false data injection attack in smart grid using kalman filter. *IEEE Trans. Control Netw. Syst.* **1**(4), 370–379 (2014)
22. Mili, L., Phaniraj, V., Rousseeuw, P.J.: Least median of squares estimation in power systems. *IEEE Trans. Power Syst.* **6**(2), 511–523 (1991)
23. Mo, Y., Kim, T.H.J., Brancik, K., Dickinson, D., Lee, H., Perrig, A., Sinopoli, B.: Cyber-physical security of a smart grid infrastructure. *Proc. IEEE* **100**(1), 195–209 (2012)
24. Monticelli, A.: State Estimation in Electric Power Systems: A Generalized Approach. *Power Electronics and Power Systems*, 1st edn. Springer, New York (1999)
25. Purchala, K., Meeus, L., Van Dommelen, D., Belmans, R.: Usefulness of DC power flow for active power flow analysis. In: *Power Engineering Society General Meeting, IEEE 2005*, pp. 454–459. IEEE (2005)
26. Rahman, M.A., Mohsenian-Rad, H.: False data injection attacks with incomplete information against smart power grids. In: *Global Communications Conference (GLOBECOM)*, pp. 3153–3158. IEEE (2012)
27. Song, X., Willett, P., Zhou, S., Luh, P.B.: The mimo radar and jammer games. *IEEE Trans. Signal Process.* **60**(2), 687–699 (2012)
28. Sou, K.C., Sandberg, H., Johansson, K.H.: Data attack isolation in power networks using secure voltage magnitude measurements. *IEEE Trans. Smart Grid* **5**(1), 14–28 (2014)
29. Tan, R., Krishna, V.B., Yau, D.K., Kalbarczyk, Z.: Integrity attacks on real-time pricing in electric power grids. *ACM Trans. Inf. Syst. Secur.* **18**(2), 5 (2015)
30. Tan, S., Song, W.Z., Stewart, M., Long, L.: Lpattack: Leverage point attacks against state estimation in smart grid. In: *Global Communications Conference (GLOBECOM)*, pp. 643–648. IEEE (2014)
31. Wang, J., Hui, L.C., Yiu, S.: Data framing attacks against nonlinear state estimation in smart grid. In: *Global Communications Conference Workshop(GLOBECOM)*, pp. 1–6. IEEE (2015)
32. Wang, J., Hui, L.C., Yiu, S.: System-state-free false data injection attack for nonlinear state estimation in smart grid. *Int. J. Smart Grid Clean Energy* **4**(3), 169–176 (2015)
33. Xu, W., Wang, M., Cai, J.F., Tang, A.: Sparse error correction from nonlinear measurements with applications in bad data detection for power networks. *IEEE Trans. Sign. Process.* **61**(24), 6175–6187 (2013)
34. Yan, J., Tang, Y., Zhu, Y., He, H., Sun, Y.: Smart grid vulnerability under cascade-based sequential line-switching attacks. In: *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7. IEEE (2015)
35. Yu, Z.H., Chin, W.L.: Blind false data injection attack using pca approximation method in smart grid. *IEEE Trans. Smart Grid* **6**(3), 1219–1226 (2015)
36. Zhu, Y., Yan, J., Tang, Y., Sun, Y., He, H.: The sequential attack against power grid networks. In: *2014 IEEE International Conference on Communications (ICC)*, pp. 616–621. IEEE (2014)

# Towards Bitcoin Payment Networks

Patrick McCorry<sup>1</sup>✉, Malte Möser<sup>2</sup>, Siamak F. Shahandashti<sup>1</sup>, and Feng Hao<sup>1</sup>

<sup>1</sup> School of Computing Science, Newcastle University, Newcastle upon Tyne, UK  
patrick.mccorry@ncl.ac.uk, siamak.shahandashti@newcastle.ac.uk,  
feng.hao@newcastle.ac.uk

<sup>2</sup> Department of Information Systems, University of Münster, Münster, Germany  
malte.moeser@uni-muenster.de

**Abstract.** Bitcoin as deployed today does not scale. Scalability research has focused on two directions: (1) redesigning the Blockchain protocol, and (2) facilitating ‘off-chain transactions’ and only consulting the Blockchain if an adjudicator is required. In this paper we focus on the latter and provide an overview of Bitcoin payment networks. These consist of two components: payment channels to facilitate off-chain transactions between two parties, and the capability to fairly exchange bitcoins across multiple channels. We compare Duplex Micropayment Channels and Lightning Channels, before discussing Hashed Time-Locked Contracts which enable Bitcoin-based payment networks. Finally, we highlight challenges for route discovery in these networks.

## 1 Introduction

The Bitcoin community fears that Bitcoin [11], the ‘Money for the Internet’ and currently the most popular cryptocurrency, cannot scale to meet future demand. Today, the network can process 3.3–7 tps (transactions per second) due to an artificial cap of 1 MB blocks and recent research [5] highlights that 90 % of the network can only achieve an effective throughput of up to 27 tps. This is dwarfed by established payment providers such as Visa, which facilitates about 2,000 tps, with a peak capacity of 56,000 tps [21].

Bitcoin’s capacity limitations are increasingly felt by users in the form of delayed transaction processing and rising transaction fees. Users currently pay about 3 to 7 US cents per transaction (independent of the amount transferred). The costs of sending a transaction could continue to rise as competition for space in the Blockchain increases and the protocol’s monetary policy continually reduces the minting of new coins that rewards ‘miners’ for securing the network.

A simple short-term fix to increase Bitcoin’s capacity would be to increase the maximum block size, allowing more transactions to be included in each block. While multiple such proposals exist [1, 8, 12], none have actively been adopted as the community cannot agree if the size of blocks should be increased at all, incrementally, dynamically or whether an artificial cap is required at all.

Long-term research has focused on two directions to improve scalability:

1. Redesigning the underlying Blockchain protocol to support more transactions per second [2, 7, 9, 18], and
2. Facilitating ‘off-chain transactions’ where transactions are only committed to the Blockchain if an adjudicator is required [6, 14].

This paper explores the latter direction and investigates payment networks that facilitate off-chain transactions, using the Blockchain only as a settlement system. Payment networks are attractive as they require only two transactions to be committed to the Blockchain in order to represent all intermediary transactions, they allow bitcoins to be routed across a series of Payment Service Providers (PSPs) without any trust-assumptions, and they provide double-spending prevention as any payment requires the recipients cooperation. Most importantly, depositors using payment networks are guaranteed to receive their fair share of a channel’s balance at any point in time.

Payment networks can be split into two major components: payment channels that sends bitcoins between two parties off-chain and the capability to fairly exchange bitcoins across two or more of such channels. In this paper, we first present important concepts of Bitcoin and its underlying lock time rules that enable simple unidirectional and bidirectional payment channels (Sect. 2). Then, we explore and compare more complex payment channel constructions such as Duplex Micropayment Channels and Lightning Channels (Sect. 3). After that, we discuss Hashed Time-Locked Contracts that guarantee the fair exchange of bitcoins across two or more channels (Sect. 4) and highlight challenges for route discovery in these payment networks (Sect. 4.3).

## 2 Background

We first introduce the necessary background about Bitcoin and its transaction lock time rules used by payment networks. Then, we present how to establish and send bitcoins in basic payment channels with an untrusted counterparty.

### 2.1 Bitcoin

Bitcoin is a decentralized transaction system, based on a peer-to-peer network and a probabilistic consensus mechanism [11]. In Bitcoin, addresses are the equivalent of accounts in traditional payment systems, used to send and receive funds. An address is a cryptographic hash of a public key, and the corresponding private key is used to spend coins by digitally signing transactions. Addresses provide users with a degree of anonymity and it is recommended practice not to reuse them when receiving bitcoins to preserve the user’s privacy. Technically, bitcoins are not strictly sent to a Bitcoin address. Instead, bitcoins are associated with redemption criteria specified in a Forth-like script language. Authorising a transfer of coins requires satisfying these criteria. The two most popular script programs are the ‘Pay to public key hash’ that requires a signature corresponding

to an address, and the ‘Pay to script hash’ that, among others, enables multi-signature addresses which require a threshold of  $m$  signatures from  $n$  public keys.

Coins are transferred via transactions which have one or more inputs and one or more outputs. Each output specifies the number of bitcoins sent and the script program, whereas each input provides a reference to a previous transaction’s output and the redeem script (e.g., a corresponding signature) that satisfies the output’s spending conditions. Transactions cannot send more bitcoins than provided in the inputs, and there are no rules on how the bitcoins are split amongst the outputs. Also, transaction signers can choose to include a fee that is deducted from the available bitcoins to spend.

Transactions are serialized in blocks that form a public ledger called the Blockchain. Bitcoin’s blockchain is updated approximately every ten minutes, each update corresponding to a new block that contains recent transactions. For a block to be valid, it must be cryptographically linked with a previous block and provide a solution to a computationally difficult ‘puzzle’. Miners are rewarded for each block with a fixed amount of bitcoins and all transaction fees paid by transactions in the block. The chain of blocks that solves the most computationally difficult puzzles is accepted by the network as the genuine Blockchain and the position of a block in this chain is called ‘height’. Due to the lottery process of solving the puzzle, more than one block can be found at the same time, leading to uncertainty at the tip of the chain. A transaction is therefore regarded as ‘confirmed’ once the block that includes it has achieved a depth of at least six blocks from the tip of the chain [11].

The time dimension given by the block heights and timestamps enables two lock time rules that are fundamental to payment channels:

**Absolute Lock Time** ensures an entire transaction<sup>1</sup> or a child transaction that is spending an output of a parent transaction<sup>2</sup> cannot be accepted into the Blockchain until a specified absolute block height  $k$  (or time) in the future, **Relative Lock Time** ensures a child transaction that is spending an output of a parent transaction cannot be accepted into the Blockchain until the parent transaction has achieved a relative depth of  $\lambda$  blocks<sup>3</sup>.

## 2.2 Payment Channel Establishment

A payment channel allows two parties to send numerous payments to each other. Instead of settling all transactions directly on the Blockchain, a payment channel only requires two transactions: one to open the channel, and one to close it and settle the final balance. The cornerstone of payment channels is depositing bitcoins into a multi-signature address controlled by both parties and having the guarantee that all bitcoins are eventually refunded at a mutually agreed time if the channel expires.

<sup>1</sup> The `nLockTime` field of the transaction.

<sup>2</sup> The output’s script contains the `OP_CHECKLOCKTIMEVERIFY` opcode.

<sup>3</sup> The output’s script contains the `OP_CHECKSEQUENCEVERIFY` opcode.

Spilman [19] proposed the first payment channel establishment protocol without the need to trust the counterparty. This protocol has a *Funding Transaction* that stores the depositor’s bitcoins and requires the authorisation of both parties to spend, and a *Refund Transaction* that returns the funds to the depositor if no payments have been authorized or the counterparty abandons the protocol. The lock time on the refund determines the lifetime of the payment channel. To establish the channel, the *Funding Transaction* is created by the depositor and remains unpublished until she received valid signatures for the *Refund Transaction* from the counterparty.

Another possibility to realise refunds is through *Non-Interactive Time-Locked Refunds*, that recently became available in Bitcoin with the activation of BIP 65 [20]. *Non-Interactive Time-Locked Refunds* account for the channel timeout directly in the multi-signature output of the *Funding Transaction*, which means that dedicated refund transactions are no longer necessary. Once the *Absolute Lock Time* specified in the output’s script has expired, the refund condition can be redeemed without the cooperation of the counterparty. While the output can only refund a single party, refunding multiple parties is possible with dedicated multi-signature outputs representing each participant’s deposit.

In practice, the use of payment channel protocols is currently hindered by the problem of transaction malleability and the infeasibility to build upon unsigned transactions. Transaction malleability allows a co-signer or an external third party to change the identification hash of a transaction before it is accepted into the Blockchain. This is a problem for contracts that sign child transactions before the parent transaction, whose outputs are being spent, is included in the Blockchain. If a modified parent transaction is accepted into the Blockchain, then all pre-signed child transactions become invalid. Furthermore, it is impossible to build upon unsigned transactions as adding signatures to a transaction changes its identification hash. BIP 66 [22] has been deployed to prevent third-party signature malleability, but co-signer and other types of malleability remain. BIP 141 [10], however, solves these malleability issues and allows to build upon unsigned transactions, and is likely to be deployed soon.

### 2.3 Basic Payment Channels

We introduce unidirectional and bidirectional channels that leverage the *Funding Transaction*’s multi-signature output. The establishment protocol in Sect. 2.2 is used to set up both channels. Bitcoins are sent using subsequent *Payment Transactions* that have two outputs to send each party their respective bitcoins.

**Unidirectional channels** were first implemented by Corallo in Bitcoinj [3] to allow a customer to send incremental payments to a merchant. Each payment has two outputs: the first increases the amount of bitcoins sent to the merchant, and the second returns change to the customer. This introduces the *Replace by Incentive* rule as the merchant only signs and broadcasts the latest *Payment Transaction* that sends them the most bitcoins. Payments can be made until the channel expires or the whole deposit has been transferred to the merchant.

**Bidirectional channels** require the *Payment Transaction* to be associated with an *Absolute Lock Time*. Each incremental payment decrements the lock time by a safety margin  $\Delta$  that represents the expected time for transactions to be accepted into the Blockchain. This introduces the *Replace by Timelock* rule as the latest *Payment Transaction* is guaranteed to be accepted into the Blockchain before any previously authorised transaction. Each payment requires both parties to exchange signatures and reduces the channel's lifetime.

### 3 Proposed Payment Channel Protocols

In this section, we outline two proposals for payment channels. The first one is called Duplex Micropayment Channels, due to Decker and Wattenhofer [6]. It extends the number of transactions that can occur within the lifetime of a bidirectional channel. The second one is called Lightning Channels, proposed by Poon and Dryja [14], and allows the channel to remain open indefinitely.

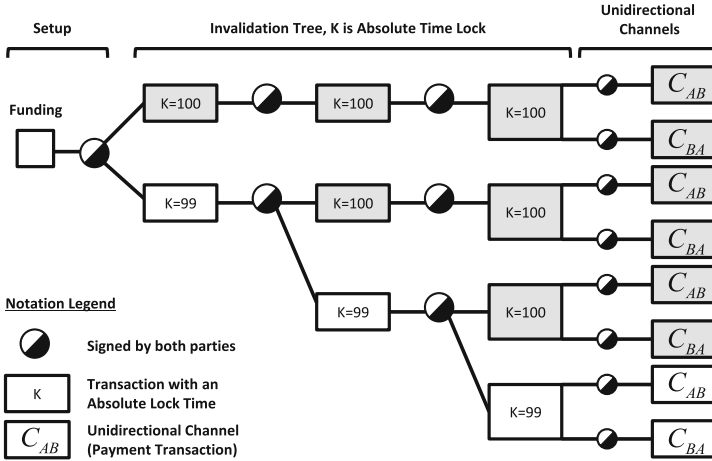
#### 3.1 Duplex Micropayment Channels

Decker and Wattenhofer propose Duplex Micropayment Channels which enable bidirectional payment channels with a finite lifetime [6]. Their scheme builds upon an initial *Funding Transaction* with deposits from two parties  $A$  and  $B$ . It consists of two *Unidirectional channels*  $C_{AB}, C_{BA}$  that together allow bidirectional payments, and an *Invalidation Tree* that sets the minimum lifetime of the channel and is responsible for resetting the bitcoins available to spend in both channels  $C_{AB}, C_{BA}$ . This reset is necessary as Alice may receive 1 BTC from Bob in the unidirectional channel  $C_{BA}$ , but her unidirectional channel to Bob  $C_{AB}$  has exhausted its supply of bitcoins. To continue bidirectional payments it is necessary to refresh  $C_{AB}$  with Alice's 1 BTC.

The core idea of Duplex Micropayment Channels is to apply the *Replace by Timelock* rule (cf. Sect. 2.3) using a tree of timelocked transactions instead of a single transaction. This structure is called an *Invalidation Tree*, exemplarily shown in Fig. 1. The nodes in the tree represent Bitcoin transactions; each edge corresponds to the spending of the previous node's output and is signed by both parties. Each transaction  $T_{d,k}$  has a depth  $d$  in the tree and is associated with an *Absolute Lock Time*  $k$ . Nodes in the active branch  $(T_{1,k}, T_{2,k}, \dots, T_{d,k})$  have *Absolute Lock Times* that are less than previously authorised branches. This guarantees that the active branch is accepted into the Blockchain before previously authorised branches.

The leaf node  $T_{d,k}$  on the active branch has two outputs to represent each unidirectional channel  $C_{AB}$  and  $C_{BA}$ . Each output can be spent if either of the following conditions is satisfied:

1. The first condition requires the signature of both parties to authorise a *Payment Transaction*.



**Fig. 1. Duplex Micropayment Channels.** A *Funding Transaction* is stored in the Blockchain. All payments occur in the unidirectional channels. If either channel becomes exhausted, then a new branch is created in the *Invalidation Tree* with a smaller *Absolute Time Lock*  $k$ . This invalidates all previous branches, and resets the balance of both unidirectional channels. For illustration  $\Delta = 1$ .

2. The second condition requires the signature of the depositor and that the current Blockchain height is greater than an *Absolute Lock Time*  $k_{max}$ . This lock time  $k_{max}$  is the maximum waiting time before the bitcoins can be returned to the depositor using a *Refund Transaction*.

Payments are sent in the unidirectional channels  $C_{AB}, C_{BA}$  as the sender signs subsequent *Payment Transactions* that have two outputs: the first sends bitcoins to the receiver, while the second returns change to the sender. The *Replace by Incentive* rule implies that the receiver will only sign the *Payment Transaction* that sends them the most bitcoins. If the receiver is unresponsive, the depositor is guaranteed to have their bitcoins refunded once the maximum lifetime of the channel  $k_{max}$  expires. It is possible to reset the balance of both unidirectional channels once a channel has exhausted its supply of bitcoins. Resetting the channels requires replacing the current active branch in the *Invalidation Tree* with a new branch whose leaf node  $T_{d,k}$  acts as an anchor for a new set of unidirectional channels.

In the following, we explain how to establish the channel, send bidirectional payments, reset the balance of both *Unidirectional channels* and finally settle the payment channel on the Blockchain.

**Channel Establishment.** First, both parties combine their funds in an unsigned *Funding Transaction*. Then, they build the first branch  $(T_{1,k}, T_{2,k}, \dots, T_{d,k})$  of the *Invalidation Tree* and exchange signatures for the branch and the *Payment Transactions* that represent the unidirectional channels  $C_{AB}, C_{BA}$ . Finally, both parties sign and broadcast the *Funding Transaction*.

The *Absolute Lock Time*  $k_{\max}$  of the *Refund Transaction* should exceed the *Absolute Lock Time* of the leaf node  $T_{d,k}$  by at least a safety margin  $\Delta$ . This ensures that the unidirectional channels have enough time to be accepted into the Blockchain before the *Refund Transactions* become available to spend.

**Send a Payment.** Each payment requires the sender to sign a new *Payment Transaction*. The receiver only signs the transaction that sends them the most bitcoins if they want to settle a dispute on the Blockchain. If either unidirectional channel has exhausted its supply of bitcoins, then both parties must cooperate to reset the balance of both channels before further payments can be made.

**Reset Balance.** Resetting the balance of the unidirectional channels requires both parties to agree a new active branch in the *Invalidation Tree*. Figure 1 demonstrates several branch replacements. An example includes the current active branch  $(T_{1,99}, T_{2,99}, T_{3,99})$  replacing the previous branch  $(T_{1,99}, T_{2,99}, T_{3,100})$ .

First, both parties find the first node  $T_{\alpha,k}$  closest to the leaves that has a greater *Absolute Lock Time* than its parent node by a safety margin of  $\Delta$ , otherwise the root node  $T_{1,k}$  is chosen.

Second, a new branch is created, starting with the chosen node whose *Absolute Lock Time* is decremented by the safety margin  $\Delta$  (i.e.  $T_{\alpha,k-\Delta}$ ). Each child node in this new branch is given an *Absolute Lock Time* greater than  $k - \Delta$  (e.g., the maximum value initially chosen when the tree was established). As lock times are *transitive*, the reduced lock time of the parent transaction automatically invalidates all previously authorised branches. Note, that when the lock time of the root node  $T_{1,k-\Delta}$  is decremented, then the channel's minimum lifetime  $k_{\min}$  is also reduced.

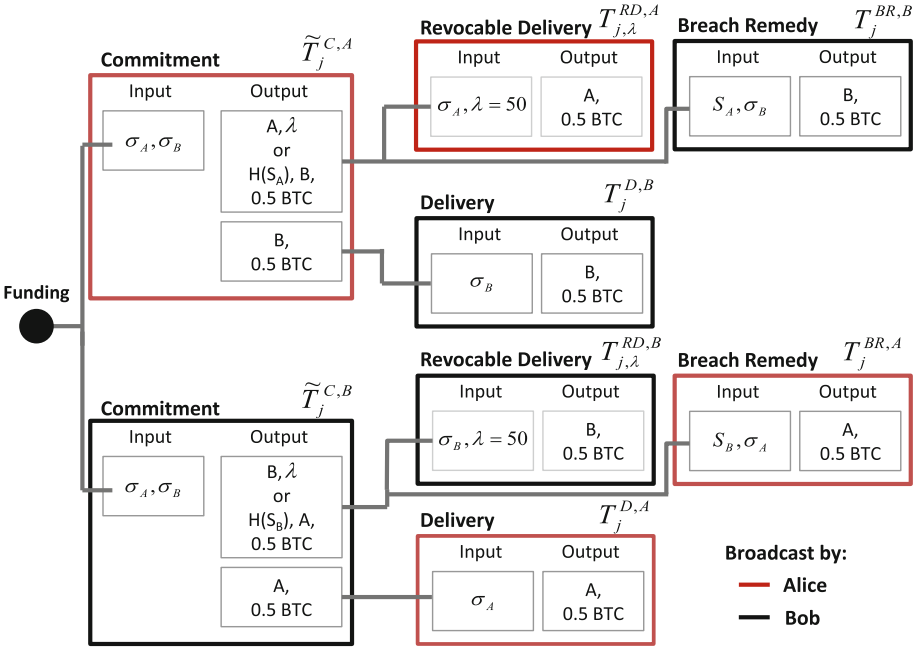
Third, both parties exchange signatures for the new active branch. The signatures for the node  $T_{\alpha,k-\Delta}$  are only exchanged once both parties have successfully exchanged signatures for the *Payment Transactions* that represent the unidirectional channels  $C_{AB}, C_{BA}$  and all of its child nodes  $(T_{\alpha+1,k}, \dots, T_{d,k})$ .

**Settle Channel.** Both parties cooperate to sign and broadcast a transaction that has no *Absolute Lock Time* and has two outputs to send each party their respective final balance. If both parties cannot cooperate, then either party can broadcast the current active branch for inclusion into the Blockchain once the *Absolute Lock Times* expire. If the receiver in a unidirectional channel does not broadcast the *Payment Transaction* that sends them the most bitcoins, then the depositor waits until  $k_{\max}$  to broadcast the *Refund Transaction*.

### 3.2 Lightning Channels

Poon and Dryja propose bidirectional payment channels called Lightning Channels that can remain open indefinitely [14]. Sending a payment requires the cooperation of both parties to authorise a new channel state that represents each party's new balance, before revoking the channel's current state. The revocation mechanism requires both parties to check the Blockchain periodically to detect





**Fig. 2. Lightning Channels.** Each party has a *Commitment Transaction* that only they can broadcast. To settle a dispute on the Blockchain, either party can broadcast their *Commitment Transaction* and *Revocable Delivery Transaction* to claim their share of bitcoins. In response, the counterparty broadcasts their *Delivery Transaction* to receive their share of bitcoins. If a *Commitment Transactions* has previously been revoked, the counterparty can broadcast the *Breach Remedy Transaction* to steal all bitcoins in the channel. (Color figure online)

if a previously revoked channel state has been submitted. If a revoked state is detected, the counterparty that did not broadcast the revoked state can issue a penalty to claim all bitcoins in the channel. Initially, this revocation was performed by exchanging signatures for the penalty transaction. An improvement proposed by Back is outlined in [16] to use the pre-image  $S$  of a revocation hash  $H(S)$ . This revocation hash is used in our description of Lightning Channels.

To describe the protocol, we use the following notation: transactions are denoted as  $T_{j,\lambda}^{\beta,X}$ , where  $\beta$  is an acronym for the transaction’s name,  $X$  is the party that can broadcast the transaction (i.e. it requires party  $X$ ’s signature to complete the transaction),  $j$  is the number of updates that have occurred in the channel and  $\lambda$  is an optional *Relative Lock Time*. Furthermore,  $T$  only requires the broadcaster to sign,  $\tilde{T}$  requires both parties to sign and  $\sigma_{X,j}^\beta$  represents party  $X$ ’s signature for the  $j^{th}$  transaction  $\beta$ .

Figure 2 presents the underlying transaction structure of Lightning Channels. The channel relies upon an initial *Funding Transaction*  $\tilde{T}^F$  that is signed by both parties and deposits bitcoins into a multi-signature address. The symmetric structure of the scheme provides each party with the following transactions:

**Channel Establishment:**

$A \rightarrow B$ : Revocation hash  $h_{A,j}$  and her transaction input  $\pi$

$B \rightarrow A$ : Unsigned  $\tilde{T}^F$ , signature  $\sigma_{B,j}^C$  for  $T_j^{C,A}$ , revocation hash  $h_{B,j}$

$A \rightarrow B$ : Signature  $\sigma_{A,j}^C$  for  $T_j^{C,B}$ , signature  $\sigma_A^F$  for  $T^F$

$B \rightarrow N$ : Broadcast  $\tilde{T}^F$  to the Bitcoin network.

**Send a Payment:**

$A \rightarrow B$ : New revocation hash  $h_{A,j+1}$

$B \rightarrow A$ : Signature  $\sigma_{B,j+1}^C$  for new  $T_{j+1}^{C,A}$ , new revocation hash  $h_{B,j+1}$

$A \rightarrow B$ : Signature  $\sigma_{A,j+1}^C$  for new  $T_{j+1}^{C,B}$ , pre-image  $S_{A,j}$  of previous revocation hash  $h_{A,j}$

$B \rightarrow A$ : Pre-image  $S_{B,j}$  of previous revocation hash  $h_{B,j}$

**Fig. 3.** Sequence of messages exchanged between two parties  $A$  and  $B$  to establish a Lightning Channel and send payments.

The *Commitment Transaction*  $\tilde{T}_j^{C,X}$  spends the multi-signature output and therefore requires signatures from both parties. Each party receives the counterparty's signature in advance. The first output sends bitcoins to the broadcaster, while the second outputs sends bitcoins to the counterparty. Spending the second output only requires the counterparty's signature, but spending the first output is only possible if either of the following two conditions is fulfilled:

1. The first condition requires a signature from the broadcaster and has a *Relative Lock Time* such that the *Commitment Transaction*  $\tilde{T}_j^{C,X}$  needs to achieve a depth of  $\lambda$  before the output is spendable.
2. The second condition requires the pre-image  $S_{X,j}$  of a revocation hash  $h_{X,j} = H(S_{X,j})$  and a signature from the counterparty.

These conditions allow *Commitment Transactions* to be revoked by revealing the pre-image  $S_{X,j}$  to the counterparty. The normal payout to the broadcaster is delayed by a *Relative Lock Time*  $\lambda$  to allow the counterparty to claim the funds if the transaction has previously been revoked. Revoking a *Commitment Transaction* allows both parties to update the channels while ensuring that no revoked transaction will ever be published.

The *Revocable Delivery Transaction*  $T_{j,\lambda}^{RD,X}$  requires the signature of the broadcaster, and sends bitcoins to the broadcaster. It can only be broadcast once the broadcaster's *Commitment Transaction* has achieved a depth of  $\lambda$  in Bitcoin's blockchain.

The *Delivery Transaction*  $T_{j,\lambda}^D,X$  requires the signature of the counterparty that did not broadcast the *Commitment Transaction* and immediately sends the counterparty their share of bitcoins.

The *Breach Remedy Transaction*  $T_j^{BR,X}$  requires the signature of the counterparty that has not broadcast the revoked *Commitment Transaction* and the pre-image  $S_{X,j}$  of the *Commitment Transaction's* revocation hash. It can only be broadcast once a revoked *Commitment Transaction* has been accepted into the Blockchain. No *Relative Lock Time* is associated with this transaction to allow the counterparty to issue the penalty immediately.

In the following and in Fig. 3, we explain how to establish the channel, send bidirectional payments, and finally settle the payment channel on the Blockchain.

**Channel Establishment.** Alice sends Bob her revocation hash  $h_{A,j}$  and a transaction input  $\pi_A$  for the *Funding Transaction*  $\tilde{T}^F$ . Bob responds with an unsigned *Funding Transaction*  $\tilde{T}^F$  that includes inputs of both parties, a signature  $\sigma_{B,j}^C$  for Alice's *Commitment Transaction*  $T_j^{C,A}$  and his revocation hash  $h_{B,j}$ . Then, Alice sends a signature  $\sigma_{A,j}^C$  for Bob's *Commitment Transaction*  $T_j^{C,B}$  and a signature  $\sigma_A^F$  for the *Funding Transaction*  $T^F$ . Finally, Bob signs and broadcasts the *Funding Transaction*  $\tilde{T}^F$  to the network.

**Send a Payment.** Sending a payment requires the cooperation of both parties to authorise a new set of transactions to represent the channel's new balance before invalidating the current set of transactions.

Alice sends Bob a new revocation hash  $h_{A,j+1}$ . Bob responds with a signature  $\sigma_{B,j+1}^C$  for Alice's new *Commitment Transaction*  $T_{j+1}^{C,A}$ , and his new revocation hash  $h_{B,j+1}$ . Alice checks that the bitcoins are correctly distributed to each party in  $T_{j+1}^{C,A}$  before sending her signature  $\sigma_{A,j+1}^C$  for Bob's new *Commitment Transaction*  $T_{j+1}^{C,B}$ . Bob then also checks that the bitcoins are correctly distributed to each party. Once the channel's new state has been authorised, Alice sends the pre-image  $S_{A,j}$  of her revocation hash to Bob, and this pre-image revokes her current *Commitment Transaction*  $T_j^{C,A}$ . Bob checks if the pre-image correctly revokes Alice's current *Commitment Transaction* before sending Alice the pre-image  $S_{B,j}$  for his revocation hash  $h_{B,j}$ . Finally, Alice checks if this pre-image revokes Bob's current *Commitment Transaction*  $T_j^{C,B}$ .

**Settle Channel.** To settle the channel, both parties cooperate to sign and broadcast a transaction that sends each party their share of bitcoins. If either party does not cooperate, then the dispute is settled on the Blockchain. In a dispute, either party broadcasts their most recent *Commitment Transaction*  $\tilde{T}_j^{C,A}$  and *Revocable Delivery Transaction*  $T_{j,\lambda}^{RD,A}$ . The counterparty must then broadcast their *Delivery Transaction*  $T_j^{D,B}$  to receive their share of the coins.

### 3.3 Comparison of Duplex Micropayment and Lightning Channels

This section provides a comparison of Duplex Micropayment Channels and Lightning Channels. We focus on the computation, storage and network access required for both schemes before highlighting if resource-limited participants can outsource responsibility to a trusted third party. Finally, we discuss the total number of transactions that can be facilitated.

**Blockchain Privacy.** A new pair of addresses  $A_1, B_1$  is generated by both parties for the *Funding Transaction*'s multi-signature output. If both channels are closed cooperatively, then  $A_1, B_1$  can be reused in the closing transaction that sends each party their final balance. In terms of privacy, an external Blockchain

**Table 1.** The number of signatures required for each step in Duplex Micropayment Channels and Lightning Channels. Also,  $d$  represents each node in the *Invalidation Tree* and  $\alpha$  is the number of replaced nodes in the *Invalidation Tree*.

	Set up	Payment	Reset	Settle (Co-op)	Settle (Dispute)
Duplex	$(d + 2) \times 2$	1	$(\alpha + 1) \times 2$	$1 \times 2$	$1 \times 2$
Lightning	$2 \times 2$	$1 \times 2$	0	$1 \times 2$	3

observer can see the number of bitcoins each address received, but not identify which receiving address  $A_1, B_1$  corresponds to which depositing address  $A_0, B_0$ .

Throughout the lifetime of both schemes,  $A_1, B_1$  can be reused in each intermediary transaction to minimise computing addresses. If a dispute is settled on the Blockchain, then a Duplex Micropayment Channel achieves the same privacy as using a single transaction to close the channel. While Lightning Channels also provide these privacy guarantees, they allow to identify which address  $A_1, B_1$  raised the dispute by identifying the address that received bitcoins in the *Revocable Delivery Transaction*.

In Duplex Micropayment Channels, both parties can have a different pair of addresses for each unidirectional channel. By inspecting the outputs of both *Payment Transactions* it is not possible to derive which addresses belong to the same owner. It is also not possible to determine which party raised the dispute using the transactions from the Blockchain only. The parties in Lightning Channels can compute 3 addresses to be used for both sets of *Commitment Transactions*. For example, Alice's *Commitment Transaction*'s first output sends bitcoins to either  $A_1$  or  $B_1$ , and the second output sends bitcoins to  $B_2$ . Her *Revocable Delivery Transaction* sends bitcoins to  $A_2$  and Bob's *Delivery Transaction* sends bitcoins to  $B_3$ . Here, it is still possible to determine which addresses raised the dispute and also the final share of bitcoins each set of addresses received.

**The number of signatures** required in each payment channel is shown in Table 1. To establish the channel, both schemes require each party to sign a *Funding Transaction*. In Duplex Micropayment Channels each party must also sign  $d$  nodes in the *Invalidation Tree* and a *Payment Transaction* representing the unidirectional channel, whereas Lightning Channels require each party to sign an additional *Commitment Transaction*.

To send a new payment, the Duplex Micropayment Channels require a single signature from the sender. If either unidirectional channel has exhausted its supply of bitcoins, then both parties cooperate to reset the balance of both channels. This reset requires each party to sign  $\alpha$  replacement transactions in the *Invalidation Tree* and an additional signature for the new *Payment Transactions* that represent the unidirectional channels. Lightning Channels always require a single signature from both parties to authorise a new pair of *Commitment Transactions*. This has implications for popular hubs as Duplex Micropayment Channels do not require the hub's involvement to receive bitcoins (except to perform a reset), while the hub must sign every payment in Lightning Channels.

In both schemes parties can cooperatively close the channel by signing a transaction that settles the final balance. However, they must sign the remaining intermediary transactions if the channel is not closed cooperatively. The unsigned transactions in Duplex Micropayment Channels include the *Payment Transaction*, or a *Refund Transaction* if the counterparty does not sign their *Payment Transaction*. In Lightning Channels, either party can broadcast their *Commitment Transaction* and sign a *Revocable Delivery Transaction* to claim their bitcoins. In response, the counterparty must sign the *Delivery Transaction* to receive their share of bitcoins. If the *Commitment Transaction* was previously revoked, then the counterparty must instead sign a *Breach Remedy Transaction*.

**Local Storage Requirements.** In Duplex Micropayment Channels, both parties store  $d + 1$  transactions which include the current active branch in the *Invalidation Tree* and the most recently received *Payment Transaction* from the counterparty. In Lightning Channels, each party stores the pre-image for every revocation hash used by the counterparty and the current *Commitment Transaction*. To prevent the need to brute-force the revocation hash using all stored pre-images, parties should also store a mapping between each pre-image and corresponding revocation hash.

**Storage in the Blockchain.** When both parties cooperate, both schemes only require 2 transactions, the *Funding Transaction* and a *Settlement Transaction*, to be committed to the Blockchain. In the worst case, when parties do not cooperate, Duplex Micropayment Channels require  $1 + d + 2$  transactions. The  $d$  transactions represent the *Invalidation Tree*'s active branch, plus the *Funding Transaction* and 2 additional transactions representing either *Payment Transactions* for the unidirectional channels or their *Refund Transactions*. Lightning Channels always require 4 transactions. Besides the *Funding Transaction*, this includes a *Commitment Transaction*, the corresponding *Delivery Transaction* and either a *Breach Remedy Transaction* or a *Revocable Delivery Transaction*.

**Frequency of Access to the Blockchain.** Duplex Micropayment Channels rely on the *Replace by Timelock* rule to ensure that only the latest authorised branch in the *Invalidation Tree* is accepted into the Blockchain. Therefore, after establishing the channel, neither party needs to access the Blockchain until the channel has expired. Then, however, it is important to push the  $d$  transaction in the latest branch and the *Payment Transaction* that sends the party their bitcoins as soon as they become valid. Lightning Channels require both parties to periodically scan the Blockchain for previously revoked transactions. The frequency of monitoring is based on the mutually agreed *Relative Lock Time* that delays the *Revocable Delivery Transaction*'s acceptance into the Blockchain.

**Outsourcing Responsibilities** is possible in both schemes to reduce the burden for resource-limited parties. In Duplex Micropayment Channels, a trusted third party can be responsible for broadcasting the active branch of the *Invalidation Tree* and the *Payment Transaction* that sends the party their bitcoins once the channel has expired. The branch must be broadcast within the safety-margin

$\Delta$  time span, otherwise previously authorised branches become spendable. In Lightning Channels, the trusted third party is provided with signed *Breach Remedy Transactions*, pre-images of revocation hashes and the identification hash of previously revoked *Commitment Transactions*. It is then responsible for monitoring the Blockchain for the previously revoked *Commitment Transactions* and broadcasting the corresponding *Breach Remedy Transactions*.

**The total number of transactions** that can occur in Duplex Micropayment Channels is based on the number of bitcoins sent, the frequency of resets to replenish the unidirectional channels, the depth of the tree, and the *Absolute Lock Time* associated with each node. In comparison, the only limitation for Lightning Channels is the bidirectional activity between both parties.

## 4 Routing Payments

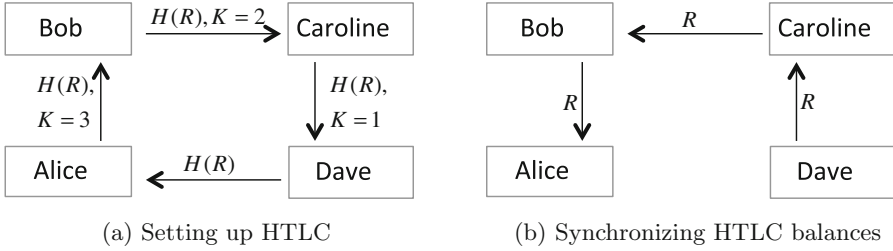
We present how to fairly exchange bitcoins across several payment channels with Hashed Time-Locked Contracts (HTLCs) before discussing how to mitigate routing interruptions and the challenge of route discovery on the payment network.

### 4.1 Hashed Time-Locked Contract (HTLC)

Hashed Time-Locked Contracts fairly exchange bitcoins across several payment channels. They allow two parties to open channels with separate Payment Service Providers (PSP), and then construct a route of payment channels that connects both channels. If we have three channels, Alice  $\rightarrow$  Bob, Bob  $\rightarrow$  Caroline and Caroline  $\rightarrow$  Dave, then Alice can send bitcoins to Dave, via Bob and Caroline. This fair exchange guarantees that intermediary hops receive their bitcoins, once they have sent their bitcoins to the next hop.

Figure 4 outlines the exchange of messages required to fairly exchange bitcoins across all intermediary channels. Once a route has been selected, the final receiver (Dave) provides the initial sender (Alice) with an HTLC hash  $H(R)$ . The initial sender commits bitcoins in a payment channel shared with their PSP under the condition that the pre-image  $R$  is revealed within  $k$  blocks. Each intermediary along the route decrements the lock time  $k$  and repeats this commitment with the next hop. The lock time  $k$  is decremented by  $\omega$  to provide the intermediary a timespan for their bitcoins to be sent to the next hop and to allow them to claim their bitcoins from the previous hop. The last hop is the final receiver, who receives the payment amount contingent upon providing  $R$ .

The final receiver can claim the bitcoins by creating a transaction that reveals  $R$  and spends the HTLC output. Revealing  $R$  then allows the next party to also claim the bitcoins from their previous hop. However, it is not necessary to claim the outputs using Bitcoin's blockchain. Instead, both parties may simply agree to update their shared channel to reflect the new balance. This can be done along the route: every pair of participants updates their channels until the initial sender's bitcoins are taken.



**Fig. 4.** (a) The final receiver provides the initial sender with the HTLC hash  $H(R)$ . This is shared along the HTLC route and the HTLC transfer’s *Absolute Lock Time*  $k$  is decremented for each hop. (b) Outlines how the pre-image  $R$  is revealed in the reverse order for each hop to claim their bitcoins once the final receiver is given  $H(R)$ . For illustration the decrement time is  $\omega = 1$ .

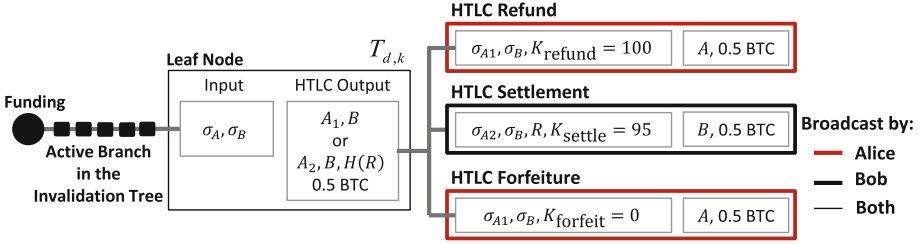
**In Duplex Micropayment Channels** a new HTLC output is either added as part of a new *Payment Transaction* in the unidirectional channel or as an additional output to the leaf node  $T_{d,k}$  in the *Invalidation Tree* (cf. Fig. 5). This HTLC output is claimable if either of the following two conditions is satisfied:

1. The first condition requires a signature from both parties.
2. The second condition requires a signature from both parties<sup>4</sup> and the pre-image  $R$  of the HTLC hash  $H(R)$ .

This HTLC output can be spent using either of the following three transactions. The *HTLC Refund Transaction* guarantees that the bitcoins are returned to the sender by  $k_{\text{refund}}$  and satisfies the first condition of the HTLC output. The *HTLC Settlement Transaction* requires the pre-image  $R$  of the HTLC hash and sends bitcoins to the counterparty. This transaction has an *Absolute Lock Time*  $k_{\text{settle}}$  to delay the receiver claiming the bitcoins to ensure that the HTLC output can later be cancelled if both parties agree. This transaction spends the second condition of the HTLC output. The *HTLC Forfeiture Transaction* has no lock time and is signed by both parties to cancel the HTLC transfer. It spends the first condition of the HTLC output and guarantees that the bitcoins are returned to the sender, even if the pre-image  $R$  of the HTLC hash is revealed.

It is the responsibility of the initial sender to compute the lock time  $k_{\text{refund}}$  for each hop along the route. The sender needs to ensure that each hop’s  $k_{\text{refund}}$  is greater than the hop’s chosen  $k_{\text{settle}}$  (i.e.  $k_{\text{refund}} > k_{\text{settle}}$ ). Both lock times must also be greater than the hop’s leaf node  $T_{d,k}$  lock time  $k_{\text{leaf}}$  (i.e.  $k_{\text{settle}} > k_{\text{leaf}}$ ). This is required as a refund or settlement cannot happen until the leaf node  $T_{d,k}$  is included in Bitcoin’s blockchain. Also, a timespan between the leaf node’s lock time  $k_{\text{leaf}}$ , and the settlement lock time  $k_{\text{settle}}$  is required to allow the hop to cancel the HTLC transfer using a *HTLC Forfeiture Transaction*. This means that the bitcoins are potentially locked for the entire lifetime of the channel.

<sup>4</sup> The sender must have a different Bitcoin address for each condition in the HTLC output, otherwise the receiver can use the signature to satisfy either condition.



**Fig. 5.** The HTLC is attached as an additional output to the *Invalidation Tree*'s leaf node. We have omitted unidirectional channels for space. The *HTLC Refund Transaction* guarantees that the bitcoins are returned to the sender if  $R$  is not revealed by block  $k_{\text{refund}}$ , the *HTLC Settlement Transaction* sends bitcoins to the receiver if the pre-image  $R$  of the HTLC hash is revealed by block  $k_{\text{settle}}$ , and the *HTLC Forfeiture Transaction* can later be signed to cancel the transfer. (Color figure online)

The initial sender's  $k_{\text{refund}}$  is passed to the next hop on the route, who should decrement  $k_{\text{refund}}$  by  $\omega$  for use in their channel. This repeats as each hop passes their  $k_{\text{refund}}$  to the next hop. The initial sender's  $k_{\text{refund}}$  must therefore take into account the largest leaf node's lock time  $k_{\text{leaf}}$  in the route. In the worst case, if the largest  $k_{\text{leaf}}$  is associated with the final receiver's channel, then the lock time  $k_{\text{refund}}$  for all hops along the route must also be larger than the largest  $k_{\text{leaf}}$ .

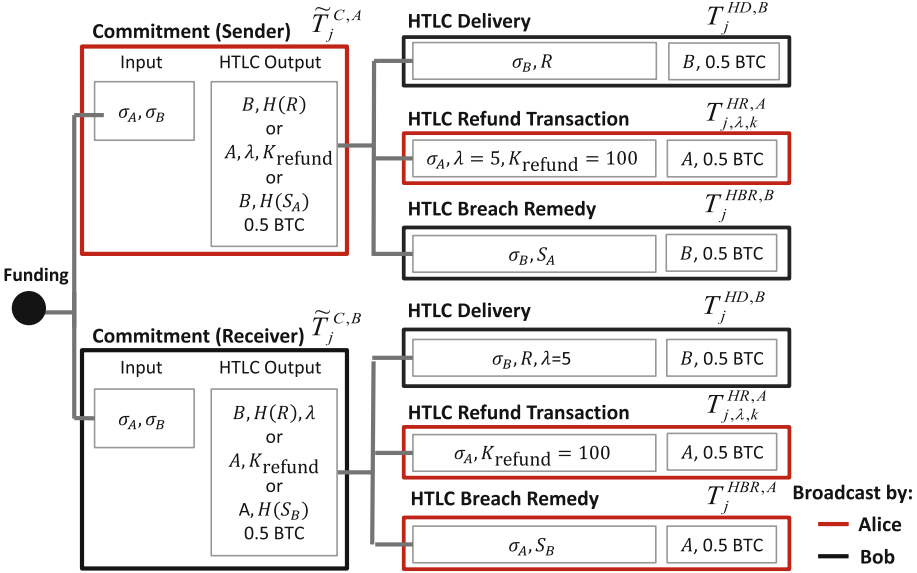
Finally, the hash  $H(R)$  and the pre-image  $R$  can be discarded once the HTLC transfer has been forfeited or a new active branch in the *Invalidation Tree* is mutually agreed to cancel the HTLC transfer.

**In Lightning Channels**, a new HTLC output is associated with both *Commitment Transactions* in a new channel state (cf. Fig. 6). The broadcaster of a *Commitment Transaction* is restrained by a *Relative Lock Time*  $\lambda$  before she can claim the bitcoins in the HTLC output. This provides the counterparty an opportunity to claim the bitcoins. We explain the role of  $\lambda$  once the commonality of the HTLC output's redemption criteria for each *Commitment Transaction* has been presented. The bitcoins can be claimed if either of the following three conditions is satisfied:

1. The first condition requires a signature from the receiver and the pre-image  $R$  of the HTLC hash  $H(R)$ .
2. The second condition requires a signature from the sender once the *Absolute Lock Time*  $k_{\text{refund}}$  has expired.
3. The third condition requires a signature from the counterparty that did not broadcast the revoked *Commitment Transaction*, and the pre-image  $S$  of the broadcaster's revocation hash  $H(S)$ .

The broadcaster of a *Commitment Transaction* must consider both the *Absolute Lock Time*  $k_{\text{refund}}$  that dictates when the HTLC transfer expires, and the *Relative Lock Time*  $\lambda$  which requires the broadcaster's *Commitment Transaction* to achieve a depth of  $\lambda$  blocks in the Blockchain before the broadcaster can claim their bitcoins. To illustrate the additional burden of  $\lambda$  for either party's





**Fig. 6.** The HTLC is an additional output in both *Commitment Transactions* and we have omitted the other outputs for space. The delivery sends bitcoins to the receiver if they know  $R$ , the refund guarantees that the bitcoins are returned to the sender, and the breach remedy sends bitcoins to the counterparty whom did not broadcast a revoked *Commitment Transaction*. (Color figure online)

redemption criteria, we consider the sender’s and receiver’s *Commitment Transaction* in turn.

The sender only broadcasts their *Commitment Transaction* to claim a refund using the *HTLC Refund Transaction* if the HTLC transfer expires, satisfying the second condition. To take the delay caused by  $\lambda$  into account, the sender’s *Commitment Transaction* must be accepted into the Blockchain at block height  $k_{\text{refund}} - \lambda$  for the *HTLC Refund Transaction* to become spendable at the correct time. If the refund is not claimed at the correct time, then  $R$  might be later revealed, and the sender is not guaranteed to receive the bitcoins from the previous hop. This  $\lambda$  provides a timespan for the counterparty to claim the bitcoins using the pre-image  $R$  of the HTLC hash  $H(R)$ , satisfying the first condition, or with the pre-image  $S_A$  of the revocation hash  $H(S_A)$ , satisfying the third condition.

The receiver only broadcasts their *Commitment Transaction* if they know the pre-image  $R$  of the HTLC hash. The bitcoins are redeemed using the *HTLC Delivery Transaction* that satisfies the first condition. To take the delay caused by  $\lambda$  into account, the receiver’s *Commitment Transaction* must be accepted into Bitcoin’s blockchain at block height  $k_{\text{refund}} - \lambda - \Delta$ . This makes certain that the *HTLC Delivery Transaction* has a safety margin  $\Delta$  timespan to be accepted into Bitcoin’s blockchain before the HTLC transfer expires. This  $\lambda$  provides a

timespan for the counterparty to claim the bitcoins if the transfer has expired or if the pre-image  $S_B$  of the receiver's revocation hash  $H(S_B)$  has been revealed.

The initial sender is responsible for computing the lock time  $k_{\text{refund}}$  in advance for each hop. This  $k_{\text{refund}}$  is passed along the route, and each hop decrements  $k_{\text{refund}}$  by  $\omega$  for use in their channel. However, the initial sender's  $k_{\text{refund}}$  must take into account the hop with the largest *Relative Lock Time*  $\lambda$ . In the worst case, if the largest  $\lambda$  is associated with the final receiver's channel, then the lock time  $k_{\text{refund}}$  for all channels must also include the largest  $\lambda$ . This potentially locks the bitcoins for a long period of time as resource-limited parties may require a large  $\lambda$  as it dictates the frequency in which each party must monitor the Blockchain. Also, the pre-image  $R$  can be discarded once both parties have exchanged the pre-images  $S_A, S_B$  of the revocation hashes  $H(S_A), H(S_B)$ .

## 4.2 Routing Interruptions

The routing is interruptible if an intermediary is unresponsive after accepting the HTLC transfer. The initial sender and final receiver should wait until the HTLC transfer expires before reattempting the transfer. To overcome this issue, Poon has proposed a rollback protocol [13] that allows the final receiver to refund the initial sender using a second route. This allows the initial sender to reattempt the HTLC transfer, and if the intermediary returns, the initial sender's bitcoins are refunded using the second route. This rollback is only performed if the final receiver has not revealed the first route's pre-image  $R_1$  of the HTLC hash  $H(R_1)$ .

To perform the rollback, the initial sender provides the final receiver with a new HTLC hash  $H(R_2)$  and an *Absolute Lock Time*  $k_{\text{expire}}$  that represents the expiry time for the initial sender's HTLC transfer with their immediate hop in the first route. The final hop in the second route sends the initial sender the refunded bitcoins and must have an expiry time  $k_{\text{refund}}$  that is greater than  $k_{\text{expire}}$ . This provides a timespan for the initial sender's bitcoins to be claimed in the first route, and for them to receive the refund in the second route.

Finally, the receiver commits bitcoins to the next hop under the condition that the pre-images of the first route's HTLC hash  $H(R_1)$  and the sender's HTLC hash  $H(R_2)$  are revealed. Each hop decrements the lock time  $k_{\text{max}}$  by  $\omega$ , and commits bitcoins to the next hop if  $R_1, R_2$  are revealed. The final hop is the sender who has knowledge of  $R_2$ , but can only claim the bitcoins if  $R_1$  is revealed from the first HTLC transfer.

## 4.3 Challenges for Route Discovery

We now discuss the challenges that payment networks face for route discovery. These challenges include the type of connections available for routing bitcoins, the topology of the network, the difficulty in building reputation systems to help assess the risk of other nodes and the financial incentives for routing bitcoins.

**Node connections** on the network rely on the user's role. End users might exclusively connect to PSPs who are responsible for establishing channels with other PSPs to find routes on the network. How these channels are funded will also

differ as end users might fund the channel with the PSP, while the PSP to PSP channels are mutually funded. Also, the total number of bitcoins in a channel restricts the bitcoins an end user can receive. For example, if the user and PSP share a channel  $U \leftrightarrow P$  in which the PSP has a balance of  $x$  bitcoins, then the user can only receive up to  $x$  bitcoins without establishing a new channel. The receiver must send bitcoins to the PSP to receive further payments.

**Route planning** methods include source-routing in which the sender pre-determines the payment route, and per-hop routing where only the final destination is given to the network and each hop finds the best route.

The former approach simplifies calculating the fee and lock time for the transfer. It is compatible with onion-routing which only reveals the previous and next hop, and allows the sender to enforce the order in which each node is visited. However, this does not prevent a node using intermediaries to route the bitcoins to the next hop. It might also be possible to identify that the next hop is the final recipient using the remaining fee and lock time.

Per-hop routing outsources the route finding to the network and can adapt to changing network conditions. The final destination is revealed to each node to help them find a route, offering less privacy than source-routing. Nodes could potentially offer low (or negative) fees to encourage others to select them for routing and then sell the transaction data to interested third parties. In per-hop routing, estimating the total fee and maximum lock time is difficult as the route is not known in advance, and mechanisms must be put into place that prevent nodes from taking excessive fees.

**The topology** of the network hinges on the ease of becoming a PSP and the potential regulation as money transmitters.

A straightforward approach is a hub-and-spoke model [17] with thousands of well-connected hubs who share route-finding information amongst themselves. In this model, end users open channels with hubs, and it is the hub's responsibility to find a route to the receiver's hub. These hubs are expected to be reliable and fast which potentially results in negligible HTLC transfer halts. This is important for Duplex Micropayment Channels as bitcoins can be locked for the channel's remaining lifetime if a transfer halt occurs. Also, the usage of Duplex Micropayment Channels reduces the computational overhead for hubs as they are only required to sign HTLC transfers while sending bitcoins, and not to receive them. In this model, end-users may not be responsible for computing or revealing the pre-image  $R$  of the HTLC hash as it would allow them to halt transfers.

The fundamental issue with the hub-and-spoke model is that it may lack Bitcoin's open-membership property. Instead, the community's vision for a payment network is to allow anyone to become a PSP which requires a mesh network and gossip protocol to advertise PSP services. Poon has proposed that Bitcoin's blockchain can aid route discovery as the user can identify which PSPs share channels [17]. If a route has been found, there is no guarantee that the PSPs are reliable due to the nature of open-membership. Lightning Channels are more suitable for unreliable transfers as a failure only temporarily locks the bitcoins.

However, the computation overhead increases as PSPs must sign both sending and receiving HTLC transfers.

To prevent routing failures naturally leads to a reputation system to assess the risk of using a PSP. However, in the event of a failure, it is arguably a non-trivial problem to determine the reason behind the halt due to the private nature of routing and the inability to inspect other payment channels. For example, if the hop  $C \rightarrow D$  settles on the Blockchain, it is not possible to determine if Dave had knowledge of the pre-image  $R$ , refused to disclose it, or simply raised a dispute to tarnish the reputation of Caroline [15]. Also, it might not be possible to identify which hop halted the transfer when disputes are not settled on the Blockchain. Furthermore, revealing the internal channel state of intermediaries cannot always reveal the reason behind a halt. For example, it is not possible to identify the duration of time before  $R$  was disclosed.

**Routing fees** need to cover the costs for a PSP to operate a secure node. These costs potentially include a ‘risk’ charge based on the number of bitcoins maintained in their hot wallet<sup>5</sup> to facilitate transfers. The PSP might charge if they are required to deposit bitcoins into the channel for end-users (i.e. merchants) who expect to mostly receive bitcoins. Also, negotiating a fee for routing is not straightforward as some PSPs may fluctuate fees to move funds in a maxed-out channel in a certain direction [4]. How fees are negotiated depends on the topology of the network as PSPs might have direct connections with each hop in the route or rely on fees advertised via a gossip protocol. Finally, the fee structure for end-users can be regular installments or on a per-transfer basis.

## 5 Conclusion

In this paper we presented an overview of Blockchain-based payment networks, a potential scaling solution for Bitcoin. We compared two prominent proposals, Duplex Micropayment Channels and Lightning Channels before discussing how to fairly exchange bitcoins across two or more channels using Hashed Time-Locked Contracts. Finally, we highlighted challenges for route discovery in payment networks. It is our hope that this paper will motivate other researchers to begin tackling the open problems associated with Blockchain-based payment networks.

**Acknowledgements.** We thank Christian Decker, Joseph Poon and Rusty Russell for reviewing this paper. We also thank the participants of the lightning-dev mailing list for their insightful discussions. This work is supported in part by the European Research Council (ERC) Starting Grant (No. 306994) and the German Bundesministerium für Bildung und Forschung (BMBF) under grant agreement No. 13N13505.

## References

1. Andresen, G.: BIP 101: increase maximum block size (2015). <https://github.com/bitcoin/bips/blob/master/bip-0101.mediawiki> Accessed 19 Apr 2016

<sup>5</sup> A wallet is frequently accessed and potentially connected to the internet.

2. Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., Wuille, P.: Enabling blockchain innovations with pegged sidechains (2014)
3. bitcoinj. <https://bitcoinj.github.io/>. Accessed 19 Apr 2016
4. CJP: Routing on the lightning network? (2015). <http://lists.linuxfoundation.org/pipermail/lightning-dev/2015-July/000031.html>. Accessed 19 Apr 2016
5. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Siler, E.G., Song, D., Wattenhofer, R.: On scaling decentralized blockchains. In: 3rd Workshop on Bitcoin and Blockchain Research (2016)
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, Heidelberg (2015)
7. Eyal, I., Gencer, A.E., Siler, E.G., van Renesse, R.: Bitcoin-NG: a scalable blockchain protocol. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI, Santa Clara, CA, USA, 16–18 March 2016
8. Garzik, J.: BIP 100: making decentralized economic policy (2015). <http://gtf.org/garzik/bitcoin/BIP100-blocksizechangeproposal.pdf>. Accessed 19 Apr 2016
9. Lewenberg, Y., Sompolinsky, Y., Zohar, A.: Inclusive block chain protocols. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 528–547. Springer, Heidelberg (2015)
10. Lombrozo, E., Lau, J., Wuille, P.: BIP 141: segregated witness (2015). <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>. Accessed 19 Apr 2016
11. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
12. Pair, S.: A simple, adaptive block size limit (2016). <https://medium.com/@spair/a-simple-adaptive-block-size-limit-748f7cbcfb75>. Accessed 19 Apr 2016
13. Poon, J.: Payment re-routing (2015). <http://lists.linuxfoundation.org/pipermail/lightning-dev/2015-July/000018.html>. Accessed 19 Apr 2016
14. Poon, J., Dryja, T.: The bitcoin lightning network: scalable off-chain instant payments (2016). <https://lightning.network/lightning-network-paper.pdf>. Accessed 19 Apr 2016
15. Russell, R.: Loop attack with onion routing (2015). <http://lists.linuxfoundation.org/pipermail/lightning-dev/2015-August/000153.html>. Accessed 19 Apr 2016
16. Russell, R.: Reaching The Ground With Lightning (draft 0.2) (2015). <https://github.com/ElementsProject/lightning/blob/master/doc/deployable-lightning.pdf>. Accessed 19 Apr 2016
17. Russell, R.: Routing on the lightning network? (2015). <http://lists.linuxfoundation.org/pipermail/lightning-dev/2015-July/000019.html>. Accessed 19 Apr 2016
18. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 507–527. Springer, Heidelberg (2015)
19. Spilman, J.: Anti DoS for tx replacement (2013). <https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html>. Accessed 19 Apr 2016
20. Todd, P.: OP\_CHECKLOCKTIMEVERIFY (2014). <https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki>. Accessed 19 Apr 2016
21. Trillo, M.: Put to the stress test (2014). <http://visatechmatters.tumblr.com/post/96025603185/put-to-the-stress-test-visanet-gets-pushed-to-the>. Accessed 19 Apr 2016
22. Wuille, P.: BIP 66: strict DER signatures (2015). <https://github.com/bitcoin/bips/blob/master/bip-0066.mediawiki>. Accessed 19 Apr 2016

# Statistical Disclosure Control for Data Privacy Using Sequence of Generalised Linear Models

Min Cherng Lee<sup>1</sup>, Robin Mitra<sup>2</sup>, Emmanuel Lazaridis<sup>3</sup>,  
An Chow Lai<sup>1</sup>, Yong Kheng Goh<sup>1</sup>, and Wun-She Yap<sup>1</sup>(✉)

<sup>1</sup> Universiti Tunku Abdul Rahman, Bandar Sungai Long, 43300 Kajang, Malaysia  
{leemc,laiac,gohyk,yapws}@utar.edu.my

<sup>2</sup> Southampton Statistical Sciences Research Institute,  
University of Southampton, Southampton, UK  
R.Mitra@soton.ac.uk

<sup>3</sup> National Institute for Cardiovascular Outcomes Research,  
University College London, London, UK  
emmanuel@lazaridis.eu

**Abstract.** When releasing data for public use, statistical agencies seek to reduce the risk of disclosure, while preserving the utility of the release data. Common approaches such as adding random noises, top coding variables and swapping data values will distort the relationships in the original data. To achieve the aforementioned properties, we consider the synthetic data approach in this paper where we release multiply imputed partially synthetic data sets comprising original data values, and with values at high disclosure risk being replaced by synthetic values. To generate such synthetic data, we introduce a new variant of factored regression model proposed by Lee and Mitra in 2016. In addition, we take a step forward to propose a new algorithm in identifying the original data that need to be replaced with synthetic data. By using our proposed methods, data privacy can be preserved since it is difficult to identify the individual under the scenario that the released synthetic data are not entirely similar with the original data. Besides, valid inference about the data can be made using simple combining rules, which take the uncertainty due to the presence of synthetic values. To evaluate the performance of our proposed methods in term of the risk of disclosure and the utility of the released synthetic data, we conduct an experiment on a dataset taken from 1987 National Indonesia Contraceptive Prevalence.

**Keywords:** Disclosure control · Data privacy · Multiple imputation · Generalised linear models

## 1 Introduction

For the last two decades, open access to data has become more popular among statistical agencies and government organizations around the world. Open data is the concept where data users can access certain data for free, re-use as well

as redistribute the data for any scientific research, social science studies or commercial purposes without restrictions from copyright and infringement. The US and UK governments are among the leaders in promoting the open data concept globally, by launching the open-data government initiatives such as [www.data.gov](http://www.data.gov) and [www.data.gov.uk](http://www.data.gov.uk). More recently, the UK's Open Data Institute (ODI) and Taiwan's Open Data Alliance (ODA) have signed a letter of intent for future cooperation. This collaboration will promote the open data access holds in both countries for the academic, public, and private sectors. With better access to the data, both sides can explore the potentials of research in the area of economy, government policies, climate change, health care issues as well as developing open data technologies.

However, when statistical agencies or the public can access the data freely, it does raise the question of confidentiality of the released data. Most of the data set that have private information on individuals (name or national insurance number) as well as sensitive data (income or personal wealth) are normally collected under confidentiality pledges, which restricts public release of such data. One of the solutions to address this problem is to forbid remote access to the data where data users are only allowed to access the data in designated data centers. This method can limit the chances of disclosure of confidential information from the released data, but occasionally this is inconvenient for the data users as they might have to travel for a long distance to arrive at the data center. Since restricted access to data may not viable, thus the data agencies consider statistical disclosure limitation (SDL) methods for protecting the confidentiality of public used data sets. However, these SDL methods will complicate analyses for users and reduce the utility of the released data.

An alternative approach to SDL methods is to use multiple imputation for statistical disclosure control. The early idea proposed by Rubin [11] is to release multiply imputed fully synthetic data sets which comprised entirely of synthetic data rather than the original data. Since the released data are synthetic values, the identification of unique individuals and their sensitive information are difficult and hence this can protect the confidentiality of the released data sets. Little [7] extended this idea by proposing partially synthetic data approach. In the partially synthetic data approach, only sensitive units or variables need to be synthesized and replaced by synthetic values.

In the partially synthetic data approach, the statistical agency releases multiply imputed, partially synthetic data sets which comprise some original data values with sensitive data replaced by synthetic values. The approach first define an model that describes the complete data, and then select the values that have high disclosure risk to be replaced by synthetic data. In large data sets containing mixed categorical (binary, ordinal and nominal) and continuous variables, the selected values that need to be synthesized normally form a non-monotone pattern. Hence, in this paper, we apply a variant version of the factored regression model proposed by Lee and Mitra [6] to generate the synthetic data. The synthetic values are drawn from a posterior predictive distribution that is the distribution of the synthetic values conditional on the observed values, so that

the statistical properties in the imputed data sets can be preserved. Hence, the users can then obtain valid inference from these synthetic data sets using the combining rules described in [8]. We perform our algorithm on a data set taken from the 1987 National Indonesia Contraceptive Prevalence Survey to ensure that the data privacy can be preserved using our method.

## 2 Literature Review

### 2.1 Statistical Disclosure Limitation (SDL) Methods

Various common SDL methods used to protect the confidentiality of the respondents in public used data sets are described as follows.

**Topcoding/Bottomcoding.** The method of topcoding/bottomcoding [13] is applied on data to prevent disclosure of extreme values in a variable. The method of topcoding replaces the value of the variable  $\mathbf{x}$  by the value  $C$  if  $\mathbf{x} \geq C$ , where  $C$  is some chosen threshold value. For example, the topcoding method replaces all incomes above \$100 000 as “\$100 000 or more”. The same can be done for variables whose values are below a certain threshold  $C$ , i.e.  $\mathbf{x} \leq C$ . Hence, instead of the real and exact individual income, the intruder will know only that the value is greater or smaller than the threshold value  $C$ . However, such method will skew the data distribution and analysts might obtain biased estimates of parameters.

**Recoding.** Statistical agencies might release variables as categories, such as recoding age into 5-year intervals. This method can help to reduce the intruders’ probability of making correct identifications. However, the data will provide less detailed information when the continuous data are replaced with intervals. See [13] for more details on the recoding method.

**Data Swapping.** Statistical agencies can swap data values of sensitive variables with other records’ values in order to reduce the risk of disclosure. This method can preserve univariate distributions. However, one drawback of this method is that it may not maintain multivariate relationships between the variables in the data set. See [2] for more details on the data swapping method.

**Adding Random Noise.** Statistical agencies can protect confidentiality of data by modifying the values of the variables in a stochastic way. This involves adding random noise,  $\epsilon$ , with zero mean and predefined variance to all values of the potentially identifying variables when the variables are continuous. However, this method does not maintain the covariance structure of the original data. See [3] for more details on this method.

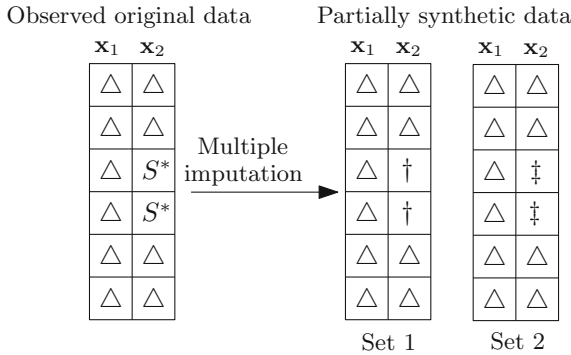
A drawback in general with all these SDL methods are that they can complicate analyses for users and can severely reduce the utility of the released data by distorting the relationships between variables in the data set. Alternative approach such as multiple imputation for partially synthetic data will be discussed in the next section.



### 2.2 Multiple Imputation for Partially Synthetic Data

Suppose we have a fully observed  $n \times p$  data set  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$ , where  $\mathbf{x}_j = (x_{1,j}, \dots, x_{n,j})'$ , for  $j = 1, \dots, p$  is the  $j$ -th variable in  $\mathbf{X}$ . We denote an  $n \times p$  data indicator matrix  $\mathbf{Z} = (z_{1,j}, \dots, z_{n,j})'$ , where  $z_{i,j} = 1$  indicates  $x_{i,j}$  is selected to be replaced by synthetic value and  $z_{i,j} = 0$  indicates  $x_{i,j}$  is unchanged, for  $i = 1, \dots, n$  and  $j = 1, \dots, p$ . We can then denote the original and synthetic values of  $\mathbf{X}$  by  $\mathbf{X}_{ori} = \{x_{i,j} : z_{i,j} = 0\}$  and  $\mathbf{X}_{syn} = \{x_{i,j} : z_{i,j} = 1\}$  respectively. Let  $\mathbf{X}_{syn}$  be the synthetic values generated from the posterior predictive distribution  $p(\mathbf{X}_{syn}|\mathbf{X})$ . We then make  $d$  independent imputations to create  $d$  different synthetic data sets, where each of the synthetic data sets,  $\mathbf{D}_{syn}^{(k)}, k = 1, \dots, d$ , is comprised of  $(\mathbf{X}_{ori}, \mathbf{X}_{syn}^{(k)})$ . These synthetic data sets are then released to the public.

Figure 1 illustrates the procedure of releasing multiply imputed, partially synthetic data. Suppose we have an fully observed data set with variables  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . The sensitive values that need to be replaced by synthetic values in  $\mathbf{x}_2$  are denoted by  $S^*$ . To apply multiple imputation, we generate two sets of independent synthetic values from their posterior predictive distribution, i.e.  $p(\mathbf{x}_2|\mathbf{x}_1)$ . These independent synthetic values are denoted by two different symbols: † and ‡. Here we have imputed the data set two times, to generate two multiply imputed, partially synthetic data sets.



**Fig. 1.** Illustration of releasing multiply imputed, partially synthetic data

Assume that an analyst wishes to use the data to infer about some population quantity  $Q$ , this might be for example the mean of  $j$ -th variable or it could be the coefficient from a regression model for one of the variables on some other subset of variables. To do this they obtain a point estimate,  $q$ , for  $Q$ , and an estimate of its variance  $u$ . For  $k = 1, \dots, d$ , let  $q_k$  and  $u_k$  be the values of  $q$  and

$u$  respectively in the synthetic data set  $\mathbf{D}_{syn}^{(k)}$ . Specifically, the analyst computes the following quantities,

$$\begin{aligned} \bar{q}_d &= \frac{1}{d} \sum_{k=1}^d q_k, & \bar{u}_d &= \frac{1}{d} \sum_{k=1}^d u_k, \\ B_d &= \frac{1}{d-1} \sum_{k=1}^d (q_k - \bar{q}_d)^2, & T_d &= \bar{u}_d + \frac{B_d}{d}. \end{aligned} \quad (1)$$

Analysts can use  $\bar{q}_d$  as a point estimate for  $Q$ , and  $T_d$  as an estimate of the variance for  $Q$ .  $B_d$  is known as the between imputation variance and is the variance between the  $d$  imputed data sets. Confidence intervals can be constructed using a  $t$ -distribution with  $\nu_d$  degrees of freedom [8] given by

$$\nu_d = (d-1) \left( 1 + \frac{d\bar{u}_d}{B_d} \right)^2,$$

See [8] for further details on the justification of the combining rules and confidence interval.

### 3 Data Privacy Preserving Scheme

In this section, we propose a new algorithm and a new model in constructing a data privacy preserving scheme:

- First algorithm involves the identification of high disclosure risk data that requires statistical disclosure control. This algorithm can be applied on other existing statistical disclosure control schemes, not only restricted to synthetic data approach. The algorithm is made available on CRAN as a package named “sdcTarget” (<http://statistics.lazaridis.eu>). Using this sdcTarget, we can determine which data points in the data set are unique and need to be replaced with synthetic values. The advantage of using such method is that we do not need to synthesize the whole data set for preserving data privacy, and hence can preserve the utility of the synthetic data as much as possible.
- The second proposed model is a variant version of factored regression model (FRM) proposed by Lee and Mitra [6]. The original version of FRM was used to multiply impute missing values through a sequence of generalised linear models. Here, we modified the data augmentation method in FRM so that it can be used to generate synthetic data. The major advantage for using such method is we can draw the synthetic data from a proper posterior distribution, which can preserve relationships present in the data and allow valid conclusions to be made from any analysis.

### 3.1 sdcTarget

In this section, we discuss the process of selecting the data points that need to be replaced with synthetic values. We focus on the categorical variables because in most scenarios, the key variables which can lead to identification are assumed to be categorical [4]. Suppose we have categorical variables  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$ , where  $\mathbf{x}_j = (x_{1,j}, x_{2,j}, \dots, x_{n,j})'$  for variables  $j = 1, \dots, p$  and units  $i = 1, \dots, n$ . Each  $x_{i,j}$  for can take values  $1, 2, \dots, w_j$  where  $w_j$  is the number of levels in variable  $\mathbf{x}_j$ . We then order the variables in an anti-lexicographical way, meaning that the first subscript  $\mathbf{x}_1$  will vary the fastest, the second subscript  $\mathbf{x}_2$  will vary the second fastest, and so on (see [12] for more details). An illustration of anti-lexicographical order is shown in Table 1.

**Table 1.** Example of anti-lexicographical ordering

$\mathbf{x}_1$	$\mathbf{x}_2$	$\dots$	$\mathbf{x}_p$
1	1	$\dots$	1
2	1	$\dots$	1
$\vdots$	$\vdots$		$\vdots$
$d_1$	1	$\dots$	1
1	2	$\dots$	1
2	2	$\dots$	1
$\vdots$	$\vdots$		$\vdots$
$d_1$	$d_2$	$\dots$	$d_p$

We cross-classify all the variables and we then denote the maximum number of possible combinations of all the variables by  $D$ , where  $D = \prod_{j=1}^p w_j$ . We can denote a variable  $\mathbf{h} = (h_1, \dots, h_n)'$  for each unit  $i$  where  $h_i \in \{1, \dots, D\}$  and corresponds to the particular combination  $x_{i,1}, x_{i,2}, \dots, x_{i,p}$ . Now let the frequencies for the different values of  $h_i$  be denoted by

$$f_g = \sum_{i=1}^n I(h_i = g), \quad g = 1, \dots, D,$$

where  $I(\cdot)$  is the indicator function:  $I(A) = 1$  if  $A$  is true and  $I(A) = 0$  otherwise. The combination value  $g$  is unique if  $f_g = 1$ . Similarly,  $f_g = 2$  means combination value  $g$  appears twice in the data set and so on. Let the frequencies of frequencies be denoted

$$n_r = \sum_{g=1}^D I(f_g = r), \quad r = 1, 2, \dots$$

where  $n_1$  represents the number of unique combinations in the data set.

Now suppose that subject  $i$  is unique, i.e.  $h_i = g, f_g = 1, g \in \{1, \dots, D\}$ . Instead of synthesizing the whole row of data points  $(x_{i,1}, x_{i,2}, \dots, x_{i,p})$  corresponding to this subject  $i$ , we only synthesize the data points that make this subject  $i$  unique. Table 2 shows a simple illustration of this idea.

**Table 2.** Illustration of choosing one data point that need to be synthesized

subject	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_g$ )
1	2	2	1	2	12	2
2	2	2	1	2	12	2
$i$	2	2	1	①	4	1

Table 2 shows a data set consists of 3 units (denoted as Unit 1, Unit 2 and Unit  $i$ ) and 4 variables,  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ , where  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$  are binary variables (i.e., 1 and 2) and  $\mathbf{x}_4$  is a categorical variable with 3 levels (i.e., 1, 2 and 3). The column  $\mathbf{h}$  represents the combination values for each unit and the frequencies for the combination values are denoted by column  $f_g$ . All the values in columns  $\mathbf{h}$  and  $f_g$  are derived using the method discussed previously. Unit 1 and 2 have the same set of data points (2, 2, 1, 2), which is corresponding to the combination value 12 ( $h_1 = h_2 = 12$ ) and this combination appears twice, hence the frequency  $f_{12} = 2$ . We noticed that unit  $i$  is unique, i.e.  $f_4 = 1$ , with  $g = 4$  being the combination value corresponds to unit  $i$ . We now need to find the data points that make this subject  $i$  unique. It is obvious that the only data point that we need to replace with synthetic data in this example is  $x_{i,4}$  (the circled number). Lets formalise this idea of selecting the data points that need to be replaced with synthetic data. We first denote the function  $h = h(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p)$  where  $h_{i \setminus \{\mathbf{x}_j\}}$  means the element  $\mathbf{x}_j$  is being excluded from the set  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$ , i.e.  $h_{i \setminus \{\mathbf{x}_j\}} = h(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{j-1}, \mathbf{x}_{j+1}, \dots, \mathbf{x}_p)$ . The main idea behind this selecting procedure is that we try to find a data point  $x_{i,j}$  such that when we recompute  $h_{i \setminus \{\mathbf{x}_j\}}$ , for unit  $i = 1, \dots, n$  with a new set of cross-classified values for  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{j-1}, \mathbf{x}_{j+1}, \dots, \mathbf{x}_p$ , i.e. aggregated over variable  $\mathbf{x}_j$  that takes values  $g' \in \{1, \dots, D'\}$ , then for  $g' = h_{i \setminus \{\mathbf{x}_j\}}$ , we would like  $f_{g'} > C_s$ , where  $C_s$  is some threshold value. If such a variable  $\mathbf{x}_j$  can be found, we then replace  $x_{i,j}$  with a synthetic value. So if we now aggregate over variable  $\mathbf{x}_4$  in the data set from Table 2, we obtain Table 3.

From Table 3 we noticed that after aggregating over variable  $\mathbf{x}_4$ , each unit has a new combination value as well as a new frequency value  $f_{g'}$ . Unit  $i$  is no longer unique since  $f_4 = 3$  where  $g' = 4$ . Hence, instead of replacing  $x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}$  with synthetic values, we only need to synthesize  $x_{i,4}$ . In this simple case, we set the threshold value  $C_s = 1$  such that  $f_{g'} > 1$  in order to select the data points that need to be replaced with synthetic data for units which are unique, i.e.  $f_g = 1$ . By setting  $C_s = 2$  such that  $f_{g'} > 2$ , we are selecting the data points that need to be replaced with synthetic data for units which are unique,

**Table 3.** Data set from Table 2 when aggregating over variable  $\mathbf{x}_4$

subject	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_{g'}$ )
1	2	2	1	4	3
2	2	2	1	4	3
$i$	2	2	1	4	3

i.e.  $f_g = 1$  as well as for units which have combination value that repeat twice in the data sets as  $f_g = 2$ . In general, by setting  $C_s = r$ ,  $r = 1, 2, \dots$ , we are selecting the data points for units  $i = 1, \dots, n$  such that  $f_g \leq r$ .

If it is not possible to find a  $x_{i,j}$  such that  $f_{g'} > C_s$  for  $g' = h_{i \setminus \{\mathbf{x}_j\}}$ , we can consider a set of variables  $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$  where  $j' \neq j$  and we try to find data points  $x_{i,j}, x_{i,j'}$  such that for  $g' = h_{i \setminus \{\mathbf{x}_j, \mathbf{x}_{j'}\}}$ , where we aggregate over pairs of variables in the data set, we would like  $f_{g'} > C_s$ , where  $C_s$  is the threshold value. If such variables  $\mathbf{x}_j$  and  $\mathbf{x}_{j'}$  can be found, we then replace  $x_{i,j}, x_{i,j'}$  with synthetic values. Table 4 shows an example where there are 2 data points that we need to replace with synthetic data, which are  $x_{i,2}$  and  $x_{i,4}$ .

**Table 4.** Illustration of choosing two data points that need to be synthesized

subject	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_g$ )
1	2	2	1	2	12	2
2	2	2	1	2	12	2
$i$	2	①	1	③	18	1

Table 4 shows a data set consists of 3 units and 4 variables,  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ , where  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$  are binary variables and  $\mathbf{x}_4$  is a categorical variable with 3 levels. The column  $\mathbf{h}$  represents the combination values for each unit and the frequencies for the combination values are denoted by column  $f_g$ . We noticed that unit  $i$  is unique, i.e.  $f_{18} = 1$ , with  $g = 18$  being the combination value corresponds to unit  $i$ . We now need to find the data points that make this subject  $i$  unique and we first aggregate over variable  $\mathbf{x}_4$  and we obtain Table 5.

**Table 5.** Data set from Table 4 when aggregating over variable  $\mathbf{x}_4$

subject	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_{g'}$ )
1	2	2	1	4	2
2	2	2	1	4	2
$i$	2	①	1	2	1

So similar to before, we notice that after aggregating over variable  $\mathbf{x}_4$ , each unit has a new combination value as well as a new frequency value  $f_{g'}$ . Unit  $i$  is still unique after aggregating over variable  $\mathbf{x}_4$  since  $f_2 = 1$  where  $g' = 2$ . Hence, we need to consider aggregating over pairs of variables so that unit  $i$  will be unique afterwards, in this case, we see that aggregating over variable  $\mathbf{x}_2$  results in Table 6.

**Table 6.** Data set from Table 4 by aggregating over variables  $\mathbf{x}_2$  and  $\mathbf{x}_4$

subject	$\mathbf{x}_1$	$\mathbf{x}_3$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_{g'}$ )
1	2	1	2	3
2	2	1	2	3
$i$	2	1	2	3

Hence unit  $i$  is no longer unique after we aggregate over variable  $\mathbf{x}_2$  and  $\mathbf{x}_4$  since  $f_2 = 3$  where  $g' = 2$ . Hence, we found the data points that need to be replaced with synthetic values, which are  $x_{i,2}$  and  $x_{i,4}$ . In general, if we can't satisfy the criteria  $h_{i \setminus \{\mathbf{x}_j, \mathbf{x}_{j'}\}} > C_s$  by aggregating over variables  $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$ , we will keep increasing the number of variables  $\{\mathbf{x}_j, \mathbf{x}_{j'}, \mathbf{x}_{j''}, \dots\}$ , where  $j \neq j' \neq j''$  until the criteria  $h_{i \setminus \{\mathbf{x}_j, \mathbf{x}_{j'}, \mathbf{x}_{j''}, \dots\}} > C_s$  is satisfied.

If it is possible to find a  $x_{i,j}$  such that  $f_{g'} > C_s$  for  $g' = h_{i \setminus \{\mathbf{x}_j\}}$  and a  $x_{i,\tilde{j}}$  where  $\tilde{j} \neq j$  such that  $f_{\tilde{g}} > C_s$  for  $\tilde{g} = h_{i \setminus \{\mathbf{x}_{\tilde{j}}\}}$ , we will select both the data points  $x_{i,j}$  and  $x_{i,\tilde{j}}$ . Table 7 shows an illustration of this case where we select both  $x_{i,2}$  and  $x_{i,4}$  given we can make unit  $i$  no longer unique by aggregating over either  $\mathbf{x}_2$  or  $\mathbf{x}_4$ .

**Table 7.** Illustration of choosing both data points that need to be synthesized

subject	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_g$ )
1	2	2	1	2	12	2
2	2	2	1	2	12	2
$i$	2	①	2	①	6	1
4	2	1	2	3	22	2
5	2	1	2	3	22	2
6	2	2	2	1	8	3
7	2	2	2	1	8	3
8	2	2	2	1	8	3

From Table 7, we noticed that unit  $i$  is unique, i.e.  $f_6 = 1$ , with  $g = 6$  being the combination value corresponds to unit  $i$ . We now need to find the data points

**Table 8.** Data set from Table 7 by aggregating over variables  $\mathbf{x}_4$ 

subject	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_{g'}$ )
1	2	2	1	4	2
2	2	2	1	4	2
$i$	2	1	2	6	3
4	2	1	2	6	3
5	2	1	2	6	3
6	2	2	2	8	3
7	2	2	2	8	3
8	2	2	2	8	3

**Table 9.** Data set from Table 7 by aggregating over variables  $\mathbf{x}_2$ 

subject	$\mathbf{x}_1$	$\mathbf{x}_3$	$\mathbf{x}_4$	Combination value ( $\mathbf{h}$ )	Frequency ( $f_{g'}$ )
1	2	1	2	6	2
2	2	1	2	6	2
$i$	2	2	1	4	4
4	2	2	3	12	2
5	2	2	3	12	2
6	2	2	1	4	4
7	2	2	1	4	4
8	2	2	1	4	4

that make this subject  $i$  unique. We first aggregate over variable  $\mathbf{x}_4$  and the data set is tabulated in Table 8. We noticed that unit  $i$  is no longer unique after we aggregate over variable  $\mathbf{x}_4$  since  $f_6 = 3$  where  $g' = 6$ . Now suppose instead of aggregating over  $\mathbf{x}_4$ , we aggregate over variable  $\mathbf{x}_2$  and the data set is shown in Table 9.

Hence unit  $i$  is no longer unique after we aggregate over variable  $\mathbf{x}_2$  since  $f_4 = 4$  where  $g' = 4$ . This shows that it is possible to find a  $x_{i,j}$  such that  $f_{g'} > C_s$  for  $g' = h_{i \setminus \{\mathbf{x}_j\}}$  and a  $x_{i,\tilde{j}}$  where  $\tilde{j} \neq j$  such that  $f_{\tilde{g}} > C_s$  for  $\tilde{g} = h_{i \setminus \{\mathbf{x}_{\tilde{j}}\}}$ , in this case, we will select both the data points  $x_{i,2}$  and  $x_{i,4}$ . In the general case of we have two subsets of variables  $\mathbf{A}_i, \mathbf{B}_i \subseteq \{x_{i,1}, x_{i,2}, \dots, x_{i,p}\}$  and  $f_{g'} > C_s$  for  $g' = h_{i \setminus \{\mathbf{A}_i\}}$  and  $f_{\tilde{j}} > C_s$  for  $\tilde{j} = h_{i \setminus \{\mathbf{B}_i\}}$ , then we select the union  $\mathbf{A}_i \cup \mathbf{B}_i$  to synthesize.

### 3.2 Generating Synthetic Data

We now describe our modelling strategy to generate the synthetic data. Following [6], we first model the joint distribution  $p(\mathbf{X}|\theta)$  using a sequence of conditional regression models:

$$\begin{aligned}
 p(\mathbf{X}|\Theta) &= p(\mathbf{x}_1|\boldsymbol{\theta}^{(1)}) \prod_{k=2}^p p(\mathbf{x}_k|\mathbf{x}_1, \dots, \mathbf{x}_{k-1}, \boldsymbol{\theta}^{(k)}) \\
 &= \prod_{i=1}^n \left\{ p(x_{i,1}|\boldsymbol{\theta}^{(1)}) \prod_{k=2}^p p(x_{i,k}|x_{i,1}, \dots, x_{i,k-1}, \boldsymbol{\theta}^{(k)}) \right\}, \quad (2)
 \end{aligned}$$

where  $\boldsymbol{\theta}^{(k)}$  represents the vector of parameters in the regression model for  $\mathbf{x}_k$  and  $\Theta = \{\boldsymbol{\theta}^{(k)}, k = 1, \dots, p\}$ . For each of the models,  $p(\mathbf{x}_k|\mathbf{x}_1, \dots, \mathbf{x}_{k-1}, \boldsymbol{\theta}^{(k)})$ , we will specify the models based on the measurement scale of  $\mathbf{x}_k$ . For various different measurement scales of  $\mathbf{x}_k$ , we use the following regression models:

- When  $\mathbf{x}_k$  is continuous we use a normal regression model.
- When  $\mathbf{x}_k$  is binary we use a probit regression model.
- When  $\mathbf{x}_k$  is ordinal we use an ordered probit regression model.
- When  $\mathbf{x}_k$  is nominal we use a multinomial logistic regression model.

Using Bayesian approach, we specify standard non-informative priors for the model parameters for each of the regression models. Our aim is to generate the synthetic data from the posterior predictive distributions  $p(\mathbf{X}_{syn}|\mathbf{X}_{obs})$ , where the distribution of synthetic data ( $p(\mathbf{X}_{syn})$ ) is conditional on the observed data ( $p(\mathbf{X}_{obs})$ ). Using the data augmentation method approach suggested by Albert and Chib [1] to formulate the probit and ordered probit regression models, we can facilitate the posterior computation in our model.

With this, we have proposed a Metropolis within Gibbs sampler to sample all unknowns from their joint posterior distribution. This proper factorisation of the joint distribution is called the factored regression model (FRM) in [6]. This model was initially used to impute missing values in a given dataset. In our approach, we modified this model to generate synthetic data instead of missing values. In the P-step of FRM, the parameters was updated based on the imputed data set, however in our approach we update the parameters based on the original data. The key idea was to threat the values being selected to be synthesized as missing data, and use the variant version of FRM to release multiply imputed, partially synthetic data sets. Thus, in complete data sets that contain categorical and continuous variables we have proposed a modelling strategy that allows a Metropolis within Gibbs sampler to sample all unknowns from their joint posterior distribution, and hence creates imputed data sets by sampling from the posterior predictive distribution  $p(\mathbf{X}_{syn}|\mathbf{X}_{obs})$ .

### 3.3 Risk of Identification

We now briefly describe how to evaluate the risk of identification in the partially synthetic data sets. We assume the variable that the intruder possesses are categorical variables. Now suppose we have a data set with only key categorical variables  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$ , where  $\mathbf{x}_j = (x_{1,j}, x_{2,j}, \dots, x_{n,j})'$  for  $j = 1, \dots, p$ . Each  $x_{i,j}$  for can take values  $1, 2, \dots, w_j$  where  $w_j$  is the number of levels in variable  $\mathbf{x}_j$ . From Sect. 3.1, each subject  $i$  is assigned a value  $h_i \in \{1, \dots, D\}$  depending



on its value  $x_{i,1}, x_{i,2}, \dots, x_{i,p}$  with  $D$  as defined previously. We denote the  $d$  multiply imputed data sets by  $\mathbf{D}_{syn}^{(l)}$ , for  $l = 1, \dots, d$ . For each of the partially synthetic data  $\mathbf{D}_{syn}^{(l)}$ , we can repeat the process in Sect. 3.1 where each subject  $i$  is assigned a value  $h_i^{(l)} \in \{1, \dots, D\}$  depending on its partially synthetic values  $x_{i,1}, x_{i,2}, \dots, x_{i,p}$ . The risk analysis proceeds as follows:

Suppose the intruder possesses categorical variables  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$ , where  $\mathbf{x}_j = (x_{1,j}, x_{2,j}, \dots, x_{n,j})'$  for  $j = 1, \dots, p$ . From Sect. 3.1, we can express this set of variables as  $n$  records of target information  $t_1, t_2, \dots, t_n$  depending on its value  $x_{i,1}, x_{i,2}, \dots, x_{i,p}$ . We start by matching the first target record  $t_1$  to  $h_i^{(1)}$  from the first partially synthetic data  $\mathbf{D}_{syn}^{(1)}$ . We then create a vector  $\mathbf{m}_1^{(1)} = (m_{1,1}^{(1)}, m_{2,1}^{(1)}, \dots, m_{n,1}^{(1)})'$  where  $m_{i,1}^{(1)} = 1$  for unit  $i = 1, \dots, n$  if the target record  $t_1$  matches record  $h_i^{(1)}$ , and zero otherwise. We can then assign a probability vector  $\mathbf{p}_1^{(1)} = (p_{1,1}^{(1)}, p_{2,1}^{(1)}, \dots, p_{n,1}^{(1)})'$ , where  $p_{i,1}^{(1)} = \frac{m_{i,1}^{(1)}}{\sum_{i=1}^n m_{i,1}^{(1)}}$  for units  $i = 1, \dots, n$ . If  $\sum_{i=1}^n m_{i,1}^{(1)} = 0$ , then  $\mathbf{p}_1^{(1)} = \mathbf{0}$ . For the first partially synthetic data  $\mathbf{D}_{syn}^{(1)}$ , this process is repeated for all target records  $t_k$ , for  $k = 1, \dots, n$ , resulting in  $n$  sets of probability vectors  $\mathbf{p}_k^{(1)}$ , for  $k = 1, \dots, n$ . We then repeat the matching process again for the  $d$  imputed synthetic data sets, where we have  $\mathbf{P}_k^{(l)} = (p_{1,k}^{(l)}, p_{2,k}^{(l)}, \dots, p_{n,k}^{(l)})'$ , for  $k = 1, \dots, n$  and  $l = 1, \dots, d$ . We then average the probability vectors across  $d$  imputed data sets and we have  $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \dots, \bar{\mathbf{p}}_n$  where  $\bar{\mathbf{p}}_k = (\bar{p}_{1,k}, \bar{p}_{2,k}, \dots, \bar{p}_{n,k})'$  for  $k = 1, \dots, n$ , with each  $\bar{p}_{i,k} = \sum_{l=1}^d \frac{p_{i,k}^{(l)}}{d}$  for target records  $k = 1, \dots, n$  and units  $i = 1, \dots, n$ .

We then denote a  $n \times n$  matrix  $\mathbf{P}$  in the following way:

$$\mathbf{P} = \begin{pmatrix} \bar{p}_{1,1} & \bar{p}_{1,2} & \cdots & \bar{p}_{1,n} \\ \bar{p}_{2,1} & \bar{p}_{2,2} & \cdots & \bar{p}_{2,n} \\ \vdots & & \ddots & \vdots \\ \vdots & & & \vdots \\ \bar{p}_{n,1} & \bar{p}_{n,2} & \cdots & \bar{p}_{n,n} \end{pmatrix}$$

with  $\bar{p}_{i,j}$  represent the  $i$ -th row and  $j$ -th column entries in the matrix  $\mathbf{P}$ , which is the probability that the  $j$ -th target record is being matched to the  $i$ -th unit in the data set.

We now describe our risk measures. The first risk measure, which we call perceived match risk, is the number of units with maximum probability of being identified that exceed some threshold value  $c$  that we consider risky, i.e. we compute  $\sum_{j=1}^n I(\max_{1 \leq i \leq n} (\bar{p}_{i,j}) > c)$ . In our case, we set this threshold value to be 0.20. This represents the number of records that have a high chance of being identified, either correctly or incorrectly. The second risk measure, which we call expected match risk, represents the number of records that the intruder would guess correctly from the synthetic data. This can be calculated from the trace of  $\mathbf{P}$ , i.e.  $\sum_{i=1}^n \bar{p}_{i,i}$ . The third risk measure, which we call true match risk, equals

$\sum_{i=1}^n I(\bar{p}_{i,i} = 1)$ , i.e. the number of units the intruder will match correctly with probability 1. For more details on this framework, see [10] and [4].

## 4 Experiment and Results

To ensure that the data privacy can be preserved using our method, we conduct an experiment using a real data set. This data set is drawn from the 1987 National Indonesia Contraceptive Prevalence Survey, which consists of 1473 respondents and 10 attributes. We retrieve this data from the UC Irvine Machine Learning Repository.

### 4.1 Description of Variables

There are 10 attributes in the data set:

- Wife’s age (numerical)
- Wife’s education (categorical: 1=low, 2, 3, 4=high)
- Husband’s education (categorical: 1=low, 2, 3, 4=high)
- Number of children ever born (numerical)
- Wife’s religion (binary: 0=non-Islam, 1=Islam)
- Is wife now working? (binary: 0=yes, 1=no)
- Husband’s occupation (categorical: 1, 2, 3, 4)
- Standard-of-living index (categorical: 1=low, 2, 3, 4=high)
- Media exposure (binary: 0=good, 1=not good)
- Contraceptive method used (class attribute: 1=no-use, 2=long-term, 3=short-term)

The samples are married women who were either not pregnant or do not know if they were at the time of interview. This data set has been used by Lim *et al.* [5] for classifying algorithm.

### 4.2 Calculating the Risks

First we use the selection algorithm described in Sect. 3.1 to select the high risk data and synthesize using our model mentioned in Sect. 3.2. As discussed in [4], we only consider the categorical attributes as these are the key variables that can lead to identification in most scenarios. Using the selection algorithm, we have identified 183 respondents in the data set that are unique, hence we only synthesize the categorical attributes corresponding to these 183 individuals.

We then proceed to generate synthetic data to replace the high risk data in the original data set. We consider 4 cases where for each case, we release 50 multiplied imputed, partially synthetic data sets. With the synthetic data sets, we evaluate the different risk measures mentioned in Sect. 3.3 and the results are tabulated in Table 10.

In all cases, we assume that the attacker uses the original data set as the target set. Case 1 is where the attacker matches the target set to the original

**Table 10.** Different risk measures on the data set drawn from the 1987 National Indonesia Contraceptive Prevalence Survey

	Perceived Risk (0.2)	Expected Risk	True Match
Case 1	443	357.000	183
Case 2	331	183.015	0
Case 3	260	160.549	0
Case 4	0	0.936212	0

data set and evaluate the risk measures. Case 2 is where the attacker matches the target set to the 50 synthetic data, where only the high risk data (categorical) of the 183 unique respondents are being replaced. Similarly, Case 3 is where we synthesize all the categorical attributes of the unique respondents, whether they are at high risk or not. Lastly, Case 4 is where we synthesize all the categorical attributes of all the respondents in the data set. More precisely, the percentages of synthetic data present in the original data are 0%, 3.26%, 9.66%, 77.78% for Cases 1 to 4 respectively.

We notice that in Case 1 all the risk measures are higher compare to other cases. The true match risk is 183 for Case 1, implying that all the 183 unique respondents will be identified if we release the original data. As we increase the number of data points being synthesized, the risk measures drop and hence data privacy can be preserved; however the utility of the synthetic data remains a concern for the data users.

We want to ensure that we preserve as much information as possible in the synthetic data. Suppose now a user is interested to use a machine learning algorithm (e.g., decision tree) to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

We first randomly split the original data into train set (70%) and test set (30%). We then train a model using decision tree on the train set, and test our model using the test set. The sensitivity, specificity and accuracy of our model are then calculated based on the confusion matrix as shown in Fig. 2. In our experiment, there are 3 classes in the predicted outcome: no use, long-term methods, or short-term methods, the confusion can be easily be expanded to a  $3 \times 3$  matrix. The sensitivity, specificity and accuracy are calculated using:

$$\text{Sensitivity} = \frac{TP}{TP + FP}$$

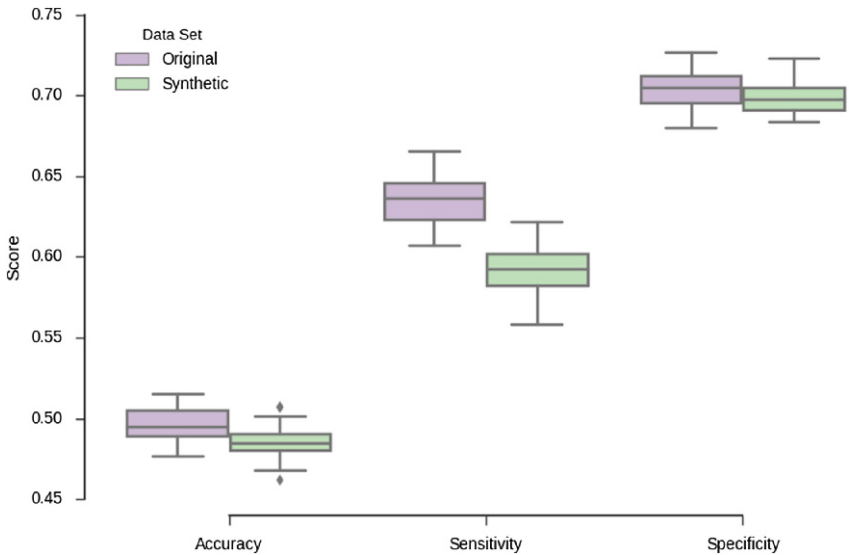
$$\text{Specitivity} = \frac{TN}{TN + FN}$$

$$\text{Accuracy} = \frac{TN + TP}{TN + FN + TP + FP}$$

where  $T$  denotes True,  $F$  denotes False,  $P$  denotes Postive and  $N$  denotes Negative.

		Predicted outcome	
		True	False
Actual outcome	True	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

**Fig. 2.** The confusion matrix



**Fig. 3.** Boxplots for accuracy, sensitivity and specificity (Color figure online)

This process is repeated 50 times and we have 50 sensitivity, specificity and accuracy calculated using the original data set. We repeat the same procedure on our synthetic data from Case 1. We compare the confidence interval of the sensitivity, specificity and accuracy and the results are shown in Fig. 3.

We notice that sensitivity, specificity and accuracy of our synthetic data have dropped (similar results were shown in Cases 2, 3 and 4) due to the presence of synthetic data. The specificity seems to perform better than sensitivity and most importantly, overall, the accuracy remains close the to the original data.

## 5 Conclusion and Future Work

We constructed a data privacy preserving scheme based on our proposed algorithm and model. Such algorithm can determine which data points in the data set are unique and need to be replaced with synthetic values. Meanwhile, the proposed model is a variant version of factored regression model to generate synthetic data. Since only selected unique data are replaced, the privacy of the original data, the utility of the synthetic data and the relationships present in the original data can be preserved. More importantly, valid inference about the data can be made using simple combining rules, which take the uncertainty due to the presence of synthetic values. We evaluated the performance of our proposed methods in terms of the risk of disclosure and the utility of the released synthetic data by using the real data drawn from the 1987 National Indonesia Contraceptive Prevalence. Future work includes comparing the risks and utilities obtained using our parametric modelling method to the classification and regression tree synthesis method [9]. Also, propensity score method [14] can be used to calculate the utility of our synthetic data.

## References

1. Albert, J.H., Chib, S.: Bayesian analysis of binary and polychotomous response data. *J. Am. Stat. Assoc.* **88**(422), 669–679 (1993)
2. Fienberg, S.E., McIntyre, J.: Data swapping: variations on a theme by Dalenius and Reiss. In: Domingo-Ferrer, J., Torra, V. (eds.) PSD 2004. LNCS, vol. 3050, pp. 14–29. Springer, Heidelberg (2004)
3. Fuller, W.A.: Masking procedures for microdata disclosure limitation. *J. Official Stat.* **9**(2), 383–406 (1993)
4. Drechsler, J., Reiter, J.P.: Accounting for intruder uncertainty due to sampling when estimating identification disclosure risks in partially synthetic data. In: Domingo-Ferrer, J., Saygin, Y. (eds.) PSD 2008. LNCS, vol. 5262, pp. 227–238. Springer, Heidelberg (2008)
5. Lim, T.-S., Loh, W.-Y., Shih, Y.-S.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.* **40**(3), 203–228 (2000)
6. Lee, M.C., Mitra, R.: Multiply imputing missing values in data sets with measurement scales using a sequence of generalised linear models. *Comput. Stat. Data Anal.* **95**, 24–38 (2016)
7. Little, R.J.A.: Statistical analysis of masked data. *J. Official Stat.* **9**(2), 407–426 (1993)
8. Reiter, J.P.: Inference for partially synthetic, public use microdata sets. *Surv. Method.* **29**, 181–188 (2003)
9. Reiter, J.P.: Using CART to generate partially synthetic public use microdata. *J. Official Stat.* **21**(3), 441–461 (2005)
10. Reiter, J.P., Mitra, R.: Estimating risks of identification disclosure in partially synthetic data. *J. Priv. Confidentiality* **1**(1), 99–110 (2009)
11. Rubin, D.B.: Statistical disclosure limitation. *J. Official Stat.* **9**(2), 461–468 (1993)
12. Schafer, J.L.: Analysis of incomplete multivariate data. Chapman & Hall/CRC, London (1997)

13. Willenborg, L., de Waal, T.: Elements of Statistical Disclosure Control. Lecture Notes in Statistics. Springer, New York (2001)
14. Woo, M.-J., Reiter, J.P., Oganian, A., Karr, A.F.: Global measures of data utility for microdata masked for disclosure limitation. *J. Priv. Confidentiality* **1**(1), 111–124 (2009)

# Energy-Efficient Elliptic Curve Cryptography for MSP430-Based Wireless Sensor Nodes

Zhe Liu<sup>1</sup>(✉), Johann Großschädl<sup>2</sup>, Lin Li<sup>2,3</sup>, and Qiuliang Xu<sup>3</sup>

<sup>1</sup> University of Waterloo, Waterloo, Canada  
z446liu@uwaterloo.ca

<sup>2</sup> University of Luxembourg, Luxembourg, Luxembourg  
johann.groszschaedl@uni.lu

<sup>3</sup> Shandong University, Jinan, China  
vivililin@gmail.com, xql@sdu.edu.cn

**Abstract.** The Internet is rapidly evolving from a network of personal computers and servers to a network of smart objects (“things”) able to communicate with each other and with central resources. This evolution has created a demand for lightweight implementations of cryptographic algorithms suitable for resource-constrained devices such as RFID tags and wireless sensor nodes. In this paper we describe a highly optimized software implementation of Elliptic Curve Cryptography (ECC) for the MSP430 series of ultra-low-power 16-bit microcontrollers. Our software is scalable in the sense that it supports prime fields and elliptic curves of different order without recompilation, which allows for flexible trade-offs between execution time (i.e. energy consumption) and security. The low-level modular arithmetic is optimized for pseudo-Mersenne primes of the form  $p = 2^n - c$  where  $n$  is a multiple of 16 minus 1 and  $c$  fits in a 16-bit register. All prime-field arithmetic functions are parameterized with respect to the length of operands (i.e. the number of 16-bit words they consist of) and written in Assembly language, whereby we avoided conditional jumps and branches that could leak information about the secret key. Our ECC implementation can perform scalar multiplication on two types of elliptic curves, namely Montgomery curves and twisted Edwards curves. A full scalar multiplication using a Montgomery curve over a 159-bit field requires about  $3.86 \cdot 10^6$  clock cycles when executed on an MSP430F1611 microcontroller.

**Keywords:** Internet of Things · Lightweight cryptography · Montgomery curve · Twisted Edwards curve · Multi-precision arithmetic

## 1 Introduction

More and more non-traditional computing devices, ranging from various kinds of sensors and actuators over consumer electronics and household appliances to smart vehicles and related road-side infrastructure, get equipped with wireless transceivers, which allows them to communicate with each other or connect to

the Internet. According to a whitepaper by Cisco, the number of smart objects (or “things”) connected to the Internet started to exceed the number of people with Internet access at some time between 2008 and 2009 [7]. Consequently, the so-called *Internet of Things (IoT)* has become reality some seven years ago. In the same whitepaper, Cisco estimates that the IoT will encompass no less than 50 billion “things” by the year 2020, which corresponds to some 6.58 connected devices per person. However, since this is an average figure, it can be assumed that, in the developed world, every person will soon be surrounded by dozens of smart devices capable to interact with each other in an entirely autonomous fashion. The IoT is expected to have a profound impact on every sector of the economy, ranging from agriculture to high tech, and touch our daily lives to a much larger extent than the Internet did in the past 20 years [16]. At the same time, it is also clear that 50 billion smart devices connected to the Internet pose unprecedented challenges to the security and privacy of their owners or users.

Similar to the “traditional” Internet (i.e. the Internet connecting commodity computers and servers), public-key cryptography plays a crucial role in the security arena of the IoT. In the recent past, a number of lightweight variants of common security protocols have been proposed (e.g. Datagram TLS [29] and HIP Diet Exchange [25]), taking the very specific requirements and constraints of the IoT into account [13]. These protocols support the use of Elliptic Curve Cryptography (ECC) [12] as a less-costly alternative to RSA [30] for such tasks as authentication and key establishment. However, even though ECC features a much better security-per-bit ratio than RSA, it is still computation intensive and, therefore, poses a massive burden for resource-constrained IoT devices. In fact, a large number of the smart objects that populate the IoT only feature an 8 or 16-bit processor clocked with a frequency of a few MHz [31]. A well-known example for a 16-bit platform targeted towards IoT applications is the MSP430 family of ultra-low power microcontrollers from Texas Instruments [5]. Besides modest processing power, IoT devices often possess only a few kB of RAM and (at most) a few 100 kB of Flash memory. However, for battery-powered devices such as wireless sensor nodes, *energy* is by far the most precious resource. Once deployed in the field, a sensor node is expected to run several months, or even years, without any maintenance and replacement, which means it must survive for long periods of time on a single battery charge.

Past research on improving the energy efficiency of ECC software focussed primarily on reducing the execution time of *scalar multiplication*, which is the main arithmetic operation of virtually all elliptic curve cryptosystems [12]. The energy consumption of (cryptographic) software on an embedded processor is closely tied to its execution time in the sense that a faster execution of a given algorithm normally translates to savings in energy [33, Sect. 6.3]. Possibilities for reducing the execution time of a scalar multiplication exist at both the field and the curve arithmetic layer, respectively. Related to the former are various approaches to speed up multiple-precision arithmetic on the MSP430 platform [9, 22, 27, 32, 37]. Recently, Düll et al. [6] reported impressive new speed records for multiplication (and squaring) modulo a 255-bit prime,



**Table 1.** Classes of constrained devices (source: RFC 7228 [4])

Name	Data size (e.g. RAM)	Code size (e.g. Flash)
Class 0, C0	$\ll$ 10 kB	$\ll$ 100 kB
Class 1, C1	$\sim$ 10 kB	$\sim$ 100 kB
Class 2, C2	50 kB	250 kB

which they achieved through an elaborate implementation of Karatsuba’s technique [17] combined with sophisticated Assembly optimizations. In a second line of research, the impact of alternative curve models, such as Montgomery [24] or twisted Edwards curves [3], has been studied for both high [6] and low to medium security levels [20]. The Montgomery model currently holds the speed record for variable-base scalar multiplication on MSP430 processors, while the twisted Edwards model achieves record-setting performance for fixed-base scalar multiplication. Due to their excellent performance characteristics, Montgomery and twisted Edwards curves also occupy the top spots in terms of energy efficiency.

Besides minimizing execution time, there exists a second avenue to reduce the overall energy consumption of ECC operations performed in an IoT device (e.g. a wireless sensor node), namely to deploy *energy-scalable ECC software*. In cryptographic engineering, the term scalability generally refers to the ability to process operands (including keys) of any length without introducing the need to re-design or re-implement a cryptosystem. More concretely, ECC software is said to be energy-scalable if it can perform scalar multiplication in elliptic-curve groups of varying order (i.e. varying cryptographic strength) without having to re-write and/or re-compile the source code [21]. Energy-scalable ECC software allows for flexible and dynamic (i.e. run-time adaptable) trade-offs between security and energy consumption, whereby the execution time (and thus also the energy cost) of a scalar multiplication increases with the cube of the group size in bits. The real-world benefit of energy-scalable cryptographic software in the context of the IoT is probably best explained by taking a Wireless Sensor Network (WSN) [28] as example. A WSN may utilize ECC for such tasks as access control [38], key exchange [20,21], or broadcast/multicast authentication [8], to list a few. While all these tasks are security critical, their actual requirements with respect to the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP) differ greatly, as we will explain in the following section. A scalable ECC implementation allows one to use smaller groups for less security-critical tasks, thereby saving precious energy. In contrast, ECC software that supports just a single curve falls short in terms of either security or energy efficiency.

Besides energy, also RAM and non-volatile memory (i.e. ROM or Flash) are scarce resources on MSP430-based sensor nodes and other IoT devices. This is little surprising since MSP430 microcontrollers have a common address space of 16 bits for RAM and Flash, which means the addressable memory space is only 64 kB (i.e.  $2^{16}$  bytes) [36, Sect.1.4]. The IETF distinguishes among three

classes of constrained devices on basis of their memory capacity as specified in Table 1. Class 0 devices are so restricted in memory and processing capabilities that, according to [4], “most likely they will not have the resources required to communicate directly with the Internet in a secure manner.” A typical example of a Class 0 device is the WISP 4.1DL, a software-programmable passive RFID tag equipped with an MSP430F2132 microcontroller that features 512 B RAM and 8 kB Flash memory [34, p. 121]. Most wireless sensor nodes fall into Class 1; for example, the TelosB [23] contains an MSP430F1611 microcontroller and provides 10 kB RAM as well as 48 kB Flash. Even though the TelosB is able to store over five times more program code than the WISP tag, it has to be taken into account that a sensor-node operating system alone may, depending on the configuration, use up half of the TelosB’s Flash memory [11]. In addition, since these operating systems do not support any security services other than simple link-layer encryption, application developers have to include auxiliary libraries for security protocols (e.g. Datagram TLS [29]) and the required cryptographic primitives, which significantly increases the total code size. Therefore, it can be expected that, in real-world application scenarios, only a small fraction of the TelosB’s Flash capacity of 48 kB will actually be available for ECC.

In this paper, we introduce an energy-scalable ECC implementation for the 16-bit MSP430 platform that supports both Montgomery and twisted Edwards curves over pseudo-Mersenne prime fields. More precisely, our ECC software is able to carry out variable-base scalar multiplication on Montgomery curves and fixed-base scalar multiplication on twisted Edwards curves. The low-level field arithmetic is optimized for primes of the form  $p = 2^n - c$  where  $n$  is a multiple of 16 minus 1 (e.g.  $n = 159, 191, 223, 255$ ) and  $c$  has a length of at most 15 bits so that it fits into a single register of an MSP430 processor. All functions of the  $\mathbb{F}_p$ -arithmetic library are parameterized by a “length” parameter, which means they take an argument that specifies the number of 16-bit words the operands (resp. result) consist of. In this way, one and the same arithmetic function can be used for pseudo-Mersenne prime fields of different order (e.g. ranging from 159 to 255 bits) without having to re-compile the source code. Contrary to the majority of previous work (e.g. [6, 14, 20]), we aimed for a compromise between performance and code size rather than optimizing solely for speed. In particular, we strived for an implementation that is small enough to fit into the code space of Class 0 and Class 1 devices as listed in Table 1 [4]. Achieving a good trade-off between speed and size is a highly challenging task since the common measures to improve performance (e.g. loop unrolling, storage of pre-computed curve points) have a negative impact on ROM/Flash consumption.

In order to reach high performance, we implemented all low-level operations supported by our  $\mathbb{F}_p$ -arithmetic library in Assembly language, but we refrained from adopting code-size increasing optimizations such as loop unrolling. When classifying ECC software along an axis between speed and size, our implementation is close to the far end towards size because we sacrificed speed for small code size, while most previous implementations sacrificed code size to achieve high speed. The whole  $\mathbb{F}_p$ -arithmetic library we describe in this paper occupies

just some 2.3 kB in Flash memory, which makes our implementation one of the most compact ever reported in the literature. To put this into perspective, the fully-unrolled implementation of multiplication in a 255-bit prime field recently introduced by Düll et al. [6] has a size of roughly 3.6 kB when compiled for an MSP430 processor, i.e. their multiplication function alone consumes 56 % more code space than our complete  $\mathbb{F}_p$ -arithmetic library. We also made an effort to protect the library against timing attacks [18] by implementing all arithmetic operations, except inversion, in a “regular” fashion so that they always execute exactly the same sequence of instructions. Hence, the execution time depends solely on the length of the involved operands (i.e. the number of 16-bit words they consist of), but not on their actual values. We implemented the inversion via the extended Euclidean algorithm, which has operand-dependent execution time. However, as we will show in this paper, timings attacks on the inversion can be effectively thwarted through multiplicative masking. We provide a detailed execution time analysis of various of field-arithmetic operations and scalar multiplication algorithms on an MSP430F1611 processor [36] for Montgomery and twisted Edwards curves over 159, 191, 223 and 255-bit fields.

## 2 WSN Security: Confidentiality Versus Integrity

A WSN can be defined as a wireless network of autonomous sensor nodes (also called *moten* [23]) that are spatially distributed in a certain area of interest to cooperatively monitor a certain physical phenomenon or condition like temperature, humidity, light, smoke, vibrations, etc. [28]. In recent years, WSNs have found widespread adoption in a multitude of application domains ranging from medical monitoring over home automation and traffic control to environmental surveillance. A typical sensor node is an inexpensive, battery-operated device equipped with an 8 or 16-bit microcontroller and has limited memory resources [23]. The way in which the nodes collect, process and transfer sensor readings depends on various factors such as the size, topology, and organization of the WSN. In the simplest case, each node sends its sensor readings directly to the Base Station (BS) or some other data collector for post-processing and decision making. However, this approach wastes precious energy for data transfers since the sensor readings of neighboring nodes are typically very similar and, hence, a lot of redundant data gets sent to the BS. In a large-scale WSN, the nodes are often organized in clusters, in which some pre-processing of sensor data, called data aggregation [28], is performed to filter out redundant or correlated data with the goal of reducing network traffic. Data transmission in a large WSN is usually multi-hop in the sense that the nodes cooperatively receive and forward (i.e. relay) packets from the source towards the destination.

WSNs differ greatly from traditional networks regarding threat models and assumptions about the attacker’s capabilities [28]. Traditional research in network security was always conducted under the assumption that the endpoints of a communication channel are secure. On the other hand, this is not the case in WSN security research since the attacker is assumed to have physical access

to the sensor nodes, which allows him to manipulate or even capture a certain number of nodes. After capturing a node, the attacker will be able to obtain all data stored on it (including cryptographic keys) because wireless sensor nodes are not tamper resistant due to cost reasons. The attacker may even manage to re-program a captured node and integrate it back into the WSN, where it then can conduct all kinds of malicious activities. In contrast to ordinary nodes, the BS possesses plenty of resources and is out of reach from physical attacks.

There exists a massive body of literature with proposals on how ECC could contribute to improve the security of WSNs [1, 21, 38]. The “big picture” of an ECC-enabled WSN can be sketched as follows. Prior to deployment, each node gets equipped with a key pair suitable for ECDH key agreement, whereby the public key along with the node’s network address is signed by a trusted third party, which is called Central Authority (CA) in [1]. In practice, this signature can have the form of a lightweight certificate that contains an expiration date and other relevant information. Besides the key pair and certificate, each node is also loaded with the public key corresponding to the trusted party’s private signing key, which allows the node to check the validity of a certificate. During the initial configuration (i.e. bootstrapping) of the WSN, each node exchanges certificates with its neighbors within communication distance and then verifies the validity of all received certificates with help of the public key of the trusted party. The nodes establish secret keys with all neighbors that are in possession of a valid certificate using static ECDH key exchange (ephemeral ECDH would also be possible, but is slightly more complicated). Now, neighboring nodes can form a cluster, select a cluster head, and carry out various other initialization activities. From time to time, e.g. when a node runs out of battery, a new node needs to be integrated into the WSN. The cluster head verifies the certificate of the new node and sends the node’s network address to the BS so that it can update routing tables and perform other management tasks.

In the ECC-based security architecture for WSNs sketched above, ECDH is used for key agreement between neighboring nodes and ECDSA for signatures (e.g. certificates). These two cryptosystems have greatly different requirements regarding their “strength” (i.e. the hardness of the ECDLP), which also reflects the damage an adversary can cause when breaking them. For example, when an attacker manages to obtain the private ECDH key of a node, he may be able to decrypt the messages sent between this node and its neighbors. For many real-world applications, such a confidentiality breach in a small part of the WSN is a non-critical issue (as mentioned in e.g. [35]) since, despite this data leak, the proper functioning of the WSN is not impacted. On the other hand, when the attacker manages to get hold of the trusted party’s signing key, he is able to forge signatures and certificates, which will allow him to inject malicious nodes into the WSN. Such malicious nodes can, for example, manipulate messages on the way from source to destination or completely drop packets. The consequences of such an integrity breach can be disastrous and may, in the worst case, make the whole WSN useless [28]. Therefore, long-term signature keys used for node authentication deserve a higher level of security than e.g. ECDH keys used to

establish shared secrets between nodes. It is, of course, possible to do both the node authentication and key exchange with a high-security elliptic curve, but in such a setting the key exchange is “over-secured,” which wastes energy.

### 3 Prime-Field Arithmetic

The performance of virtually any ECC implementation depends heavily on the execution time of certain arithmetic operations (in particular multiplication) in the underlying finite field. When implementing ECC in software, it is common practice to use curves defined over some special prime fields that facilitate the modular reduction operation [12]. A well-known example for primes with good arithmetic properties are the so-called *pseudo-Mersenne primes* [12], which are primes of the form of  $p = 2^n \pm c$  where  $c$  is relatively small (to fit into a single register of the target processor) and  $n$  is chosen such that  $p$  meets the desired bitlength. When taking advantage of the congruence relation  $2^n \equiv c \pmod{p}$ , the reduction of a  $2n$ -bit integer modulo  $p = 2^n - c$  has linear complexity, i.e. the execution time of the reduction operation increases linearly with  $n$  [12].

The NIST and numerous other standards bodies recommend elliptic curves over prime fields whose bitlengths are multiples of 32, e.g. 192, 224, or 256 bits [12]. However, as demonstrated by Bernstein in [2], it can be more beneficial to use primes that are slightly shorter than the standard bitlengths (e.g. 255 bits instead of 256), especially if one aims for both high speed and a regular execution profile to counter timing attacks. Having one bit of “slack space” permits some special software optimization techniques that are not applicable when the bitlength is exactly a multiple of the processor’s word size. The elliptic curves we used to benchmark our implementation follow this approach since they are defined over fields given by primes of the form  $p = 2^n - c$  where  $n$  is a multiple of 32 minus 1 (e.g.  $n = 159$  or  $191$ ) and  $c$  is up to eight bits long. However, the parameterized  $\mathbb{F}_p$ -arithmetic library for MSP430 processors we describe in this section is flexible enough to support any  $n$  that is a multiple of 16 minus 1 and any  $c$  that is in the range of  $[1, 2^{15} - 1]$ .

We use the following notation to describe the arithmetic functions:  $n$  is the bitlength of the prime  $p = 2^n - c$ ; this implies that any element  $a \in \mathbb{F}_p$  is also  $n$  bits long. When working on a  $w$ -bit processor, a field element can be stored in an array of  $m = \lceil n/w \rceil$  words, each consisting of  $w$  bits. In our case,  $w = 16$  since the target platform is an MSP430 processor. We use lowercase letters to denote field elements and indexed lowercase letters to refer to individual words of a field element. Consequently,  $a \in \mathbb{F}_p$  can be written as an array of the form  $(a_{m-1}, \dots, a_1, a_0)$  where  $a_{m-1}$  is the Most Significant Word (MSW) and  $a_0$  the Least Significant Word (LSW). All arithmetic functions of our library are able to process incompletely-reduced operands, which means that an operand does not necessarily need to be the least non-negative residue modulo  $p$ , but has to fit into  $m$  words and can, therefore, be up to  $n + 1$  bits long. Also the results produced by our functions for  $\mathbb{F}_p$ -arithmetic may not be fully reduced.

### 3.1 Modular Addition and Subtraction

The straightforward way to perform a modular addition  $r = a + b \bmod p$  is to first calculate the sum  $s = a + b$  and then subtract a multiple of  $p$  from it to obtain a final result that fits into  $m$  words. Since each of the operands can be  $n + 1$  bits long, the subtrahend is either  $0$ ,  $p$ ,  $2p$ , or  $3p$  [20]. However, instead of subtracting a multiple of  $p$ , it is, in general, more efficient to add a multiple of  $c$ , which is possible since  $2^n \equiv c \bmod p$ . An MSP430 implementation of this approach for modular addition with fully-unrolled loops and regular execution profile is described in [20, Sect. 3.1]. The main drawback of such an “add-then-subtract” technique is that it consists of two loops, each introducing overhead if they are not unrolled. Due to the simplicity of these loops, the loop overhead (i.e. updating of a loop counter, checking of a loop-termination condition, and jumping back to the start) can significantly impact the overall performance.

Since we aim for small code size (and, therefore, avoid loop unrolling), it is very important to minimize the loop overhead. An obvious way to achieve this is to employ a modular addition technique that requires only a single loop, like the one described by Düll et al. in [6, Sect. 4.4]. Our implementation is based on their approach and performs an addition in  $\mathbb{F}_p$  as follows. First, we add the MSWs of  $a$  and  $b$ , i.e. we compute the sum  $s = a_{m-1} + b_{m-1}$ , which can be up to 17 bits long when the operands are incompletely reduced. This sum is then split up into a lower part  $s_L$  consisting of the 15 least-significant bits, and an upper part  $s_H$  with the remaining two bits. We temporarily store  $s_L$  in a register and multiply  $s_H$  by  $c$ . Thereafter, the  $m - 1$  remaining words of the two operands are added (with carry propagation), starting with the LSWs  $a_0$  and  $b_0$ . The main difference to a “conventional” multi-precision addition is that the product of  $s_H$  and  $c$  is added to the two LSWs and, therefore, the carry to be propagated to the next-higher word can be 0, 1, or 2. Finally, the carry from the last addition (i.e. the addition of the words  $a_{m-2}$  and  $b_{m-2}$ ) is propagated into  $s_L$ , which is then at most 16 bits long. The final result has a length of no more than  $n + 1$  bits (i.e. fits into  $m$  words), but may be not fully reduced.

The most basic way to perform a modular subtraction  $r = a - b \bmod p$  consists of two simple steps: the computation of the difference  $d = a - b$ , followed by an addition of  $p$  (or a multiple thereof) to get a non-negative result. In the most extreme case, namely when  $a = 0$  and  $b$  has the maximum possible value of  $2^{n+1} - 1$ , it is necessary to add  $3p$ . A constant-time implementation of this modular subtraction technique with unrolled loops is described by Düll et al. in [6, Sect. 4.4]. However, when implemented with “rolled” loops to minimize the code size, this approach suffers from a high overhead since (at least) two loops need to be executed; one for the actual subtraction and the other(s) to obtain a non-negative result. In order to minimize the loop overhead, we perform the modular subtraction by computing  $r = 4p + a - b \bmod p$  since this operation can be implemented with one single loop, similar to the modular addition. The addition of  $4p$ , which ensures that the final result is positive, does not cause much overhead when taking into account that all  $w$ -bit words of  $4p$ , except the two LSWs and the MSW, are identical and can be kept in a register.

---

**Algorithm 1.** Multiple-precision multiplication (product-scanning method)

---

**Input:** Two  $m$ -word operands  $a = (a_{m-1}, \dots, a_1, a_0)$  and  $b = (b_{m-1}, \dots, b_1, b_0)$ **Output:**  $2m$ -word product  $z = a \times b = (z_{2m-1}, \dots, z_1, z_0)$ 

```

1:  $s \leftarrow a_0 \times b_0$ 
2:  $z_0 \leftarrow s \bmod 2^w$ ;  $s \leftarrow s/2^w$ 
3: for  $i$  from 1 by 1 to  $m - 1$  do
4:    $j \leftarrow 0$ ;  $k \leftarrow i$ 
5:   while  $k \geq 0$  do
6:      $s \leftarrow s + a_j \times b_k$ 
7:      $j \leftarrow j + 1$ ;  $k \leftarrow k - 1$ 
8:   end while
9:    $z_i \leftarrow s \bmod 2^w$ ;  $s \leftarrow s/2^w$ 
10: end for
11: for  $i$  from  $m$  by 1 to  $2m - 3$  do
12:    $j \leftarrow i - (m - 1)$ ;  $k \leftarrow m - 1$ 
13:   while  $j \leq m - 1$  do
14:      $s \leftarrow s + a_j \times b_k$ 
15:      $j \leftarrow j + 1$ ;  $k \leftarrow k - 1$ 
16:   end while
17:    $z_i \leftarrow s \bmod 2^w$ ;  $s \leftarrow s/2^w$ 
18: end for
19:  $s \leftarrow s + a_{m-1} \times b_{m-1}$ 
20:  $z_{2m-2} \leftarrow s \bmod 2^w$ 
21:  $z_{2m-1} \leftarrow s/2^w$ 
22: return  $(z_{2m-1}, \dots, z_1, z_0)$ 

```

---

### 3.2 Multiplication and Squaring

Some MSP430 models, including our target processor (the MSP430F1611), are equipped with a hardware multiplier that is capable to perform multiplications and Multiply-ACcumulate (MAC) operations with 16-bit integers. This multiplier is not tightly coupled to the processor core, but attached to it in the form of a memory-mapped peripheral. The MSP430 instruction set does not include dedicated multiply or MAC instructions; instead, the multiplier is accessed via a set of eight peripheral registers that are visible in the address space and can be loaded and read using the `mov` instruction [36]. There are four registers (and associated memory addresses) for the first operand, called `MPY`, `MPYS`, `MAC`, and `MACS`, as well as one register and address for the second operand, referred to as `OP2`. The type of operation (i.e. multiplication or MAC, signed or unsigned) is selected by the address the first operand is written to. For example, to perform an unsigned multiplication, one has to write the first operand to `MPY`. The execution of the selected operation starts immediately and automatically after the second operand is written to `OP2`. There are three result registers: `RESLO` holds the lower 16 bits of the result, `RESHI` the higher 16 bits, while `SUMEXT` contains the carry of the accumulate operation if an unsigned MAC is performed.

From an algorithmic point of view, there are two basic techniques to implement a multiple-precision multiplication, namely the operand-scanning method

and the product-scanning method (see [12, 19] for details). The availability of a hardware-supported MAC operation clearly indicates that the latter technique may reach better performance on an MSP430F1611 device. Algorithm 1 shows our implementation of the product-scanning approach, which is very similar to that in [12, Algorithm 2.10]. It consists of two nested loops; the first computes the lower half of the product  $z$  (i.e. the words  $z_1$  to  $z_{m-1}$ ), whereas the second nested loop contributes the higher words (i.e.  $z_m$  to  $z_{2m-3}$ ). Both inner loops perform MAC operations; in each iteration, two  $w$ -bit words are multiplied and the  $2w$ -bit product is added to a cumulative sum  $s$ . In general, when adding up several such  $2w$ -bit products, the length of the sum  $s$  can exceed  $2w$  bits. The index  $j$  is incremented in the inner loop, while  $k$  is decremented, which means the words of operand  $a$  are loaded in ascending order and those of operand  $b$  in descending order. An operation of the form  $z_i \leftarrow s \bmod 2^w$  as in line 9 assigns the  $w$  least significant bits of  $s$  to the word  $z_i$ , whereas  $s \leftarrow s/2^w$  represents a  $w$ -bit right-shift of  $s$ . Note that the computation of  $a_0 \times b_0$  is “peeled off” from the first nested loop since it is not a MAC operation but just a straightforward multiplication. We also compute the last  $2w$ -bit product,  $a_{m-1} \times b_{m-1}$ , outside the second nested loop because it is not necessary to shift  $s$  after the addition of this product; instead, we can directly write  $s$  to  $z_{2m-2}$  and  $z_{2m-1}$ .

**Listing 1.** First inner loop of the product-scanning method in Assembly language

---

```

1  INNLOOP1:
2      MOV    @APTR+, &MAC
3      MOV    @BPTR, &OP2
4      SUB    #2, BPTR
5      ADD    @EXTPTR, CARRY
6      CMP    INNSTOP, BPTR
7      JGE   INNLOOP1
    
```

---

Listing 1 shows our Assembly implementation of the first inner loop of the product-scanning method (line 5 to 8 of Algorithm 1). APTR and BPTR are two registers that contain pointers (i.e. addresses) through which the 16-bit words of operand  $a$  and  $b$  are accessed. The first MOV instruction writes a word of  $a$  to MAC, thereby configuring the multiplier to execute a MAC operation. Then, the second MOV instruction writes a word of  $b$  to OP2, which immediately starts the execution. A MAC operation consists of the multiplication of the 16-bit words written to MAC and OP2, followed by the accumulation of the 32-bit product to the content of the RESHI|RESLO register pair, whereby the resulting carry bit is placed in SUMEXT. In line 5 of Listing 1, the carry bit gets added to a general-purpose register named CARRY using the indirect addressing mode to read from SUMEXT (see [36, Sect. 7.2.4] for further explanations). MSP430 processors have an autoincrement addressing mode, with which we update APTR, but there is no autodecrement mode. Therefore, we manually decrement BPTR with help of the SUB instruction in line 4. The register INNSTOP holds the address of  $b_0$  (i.e. the LSW of operand  $b$ ) and, consequently, the loop iterates as long as the address



in BPTR is greater or equal to INNSTOP. Each iteration takes 16 clock cycles on an MSP430F1611 processor, to which the loop overhead (i.e. the comparison in line 6 and the jump instruction in line 7) contributes three cycles.

**Optimized Squaring.** Squaring is a special case of multiplication that allows for dedicated optimizations due to the equality of the two operands [12]. When an ordinary multiplication algorithm, such as the product-scanning method, is used for squaring (by setting  $b = a$ ), then all  $2w$ -bit word-products of the form  $a_j \times a_k$  with  $j \neq k$  are computed twice because  $a_j \times a_k = a_k \times a_j$  [19]. Only the  $m$  word-products  $a_i \times a_i$ , which lie in the “main diagonal” and are themselves squares, are generated and processed only once. Dedicated squaring algorithms intend to avoid such overheads by computing each word-product only once and then doubling it (e.g. through a left-shift) if needed. When implemented in this way, the squaring of an integer consisting of  $m$  words requires the computation of  $(m^2 + m)/2$  word-products, which is just about half of the  $m$  word-products that have to be formed in the course of an ordinary multiplication. However, in practice, the saving in execution time is often significantly below 50%.

Our implementation of the squaring function in Assembly language follows closely Algorithm 2 in [19]. This algorithm consists of two nested loops, plus a third one, which is a simple (“un-nested”) loop. The two nested loops compute the word-products to be doubled and are similar to those of the multiplication in Algorithm 1, except that the termination conditions for the inner loops are different since they are iterated fewer times. Both inner loops consist of just six Assembly instructions (similar to Listing 1), and each iteration takes 16 cycles on our MSP430F1611 device. In the third loop, which is iterated  $m$  times, the intermediate result obtained so far is doubled and the  $m$  word-products of the form  $a_i \times a_i$  (which are actually word-squares) are added. Each iteration of the third loop requires 42 clock cycles on an MSP430F1611 processor.

**Modular Reduction.** As mentioned in the beginning of this section, our  $\mathbb{F}_p$ -arithmetic library is scalable and optimized for pseudo-Mersenne primes of the form  $p = 2^n - c$ , where  $n$  is a multiple of 16 minus 1 and  $c$  can be up to 15 bits long. The functions of the arithmetic library can process incompletely-reduced operands, which means the operands can exceed their nominal bitlength by one bit and have a length of  $n + 1$  bits. Consequently, the result of a multiplication or squaring is up to  $2n + 2$  bits long and fits into  $2m$  words. A straightforward approach for reducing a  $2m$ -word product  $z$  modulo  $p = 2^n - c$  is to split  $z$  up into a lower part  $z_L$  and a higher part  $z_H$  such that  $z = z_H \cdot 2^n + z_L$ , followed by a substitution of  $2^n$  by  $c$ , which is possible since  $2^n \equiv c \pmod{p}$ . However, in our case, this method entails some overhead because  $n$  is not a multiple of the word size  $w$  and, thus, shift operations are necessary to extract  $z_H$  from  $z$ .

To avoid bit-level shifts, we perform the reduction operation in two steps as described in [6, 20]. In the first step, the  $2m$ -word product  $z$  is reduced modulo  $2p$  into an intermediate result  $t$  consisting of  $m+1$  words, which is then in the second step further reduced modulo  $p$  to yield a final result with  $m$  words. The first step

requires to partition  $z$  into a lower part  $z_L$  consisting of the  $m$  LSWs (i.e. the  $n + 1$  least significant bits) of  $z$  and a higher part  $z_H$  with the remaining  $m$  words. However, this partitioning does not require any shift operations since it is done at the word-size boundary. Then, we compute the intermediate result  $t = z_H \cdot 2c + z_L$  using the operand-scanning technique [12], whereby the 16-bit quantity  $2c$  needs to be written to MPY only once and can then be used for all  $m$  multiplications [36]. This also explains why the length of  $c$  is limited to at most 15 bits as otherwise  $2c$  would not fit in a register. Note that, when  $z$  is a product of two  $(n + 1)$ -bit integers, i.e. when  $z \leq (2^{n+1} - 1)^2$ , and  $c$  has a length of  $w - 1$  bits, then  $t$  is at most  $n + w + 1$  bits long and can be stored in  $m + 1$  words. In the second step of the reduction operation,  $t$  is split up into a lower part  $t_L$  with exactly  $n$  bits and a higher part  $t_H$  with  $w + 1$  bits. The final result  $r = t_H \cdot c + t_L$  is then obtained by multiplying  $t_H$  by  $c$ , adding the  $2w$ -bit product to the two LSWs of  $t_L$ , and propagating the carry bit up to the MSW. Even though  $r$  may not be fully reduced, it fits into  $m$  words.

### 3.3 Inversion

Since inversion in  $\mathbb{F}_p$  is an extremely costly arithmetic operation, it is common practice in ECC software to use projective coordinates for scalar multiplication so that only a single inversion has to be carried out to convert the result from projective to affine coordinates [12]. However, this final inversion is a potential source of side-channel leakage as it can reveal information about the projective representation of the point obtained as result, which, in turn, may allow an attacker to learn a few bits of the secret scalar [26]. In order to prevent this kind of attack, the inversion has to be implemented in such a way that its execution time is either constant (i.e. operand independent) or appears random [18]. The most common way to achieve the former is to execute the inversion through an exponentiation according to Fermat's little theorem, i.e.  $a^{-1} = a^{p-2} \pmod{p}$ . In most previous papers describing timing-attack-resistant ECC software, such as [6, 20], this exponentiation was implemented using addition chains.

While Fermat-based inversion allows one to achieve constant execution time in a relatively straightforward way, it is significantly slower than the Extended Euclidean Algorithm (EEA) [12]. However, the EEA has an irregular execution profile and, consequently, operand-dependent execution time. Nonetheless, it is possible to thwart timing attacks against EEA-based inversion by employing a simple multiplicative masking method. Let  $Z$  be the  $z$ -coordinate of a point in projective coordinates and let  $u$  be a random element of  $\mathbb{F}_p$  that is unknown to the attacker. Instead of inverting  $Z$  directly, we first multiply  $Z$  by  $u$ , then invert the product  $Zu$  using the EEA to obtain  $(Zu)^{-1}$ , and finally multiply the inversion result  $(Zu)^{-1}$  by  $u$  to get  $Z^{-1}$ . In this way, the execution time of the inversion depends on both  $Z$  and  $u$ , but since the attacker does not know  $u$ , he is not able to draw any conclusions about the actual value of  $Z$ . Note that, in our ECC software,  $u$  is "hard-coded" and can not be changed, which requires us to take care that an attacker can not reveal  $u$  by exploiting variations in the execution time. This is especially important in ECDH key exchange where

an attacker can use a fake public key, e.g. a point of low order, which may enable him to “guess”  $Z$  and use this information to get  $u$ . We thwart such attempts by insisting secret scalars to be a multiple of the curve’s co-factor, as in [2], so that the resulting  $Z$  coordinate is 0 and the inversion is not executed.

## 4 Point Arithmetic and Scalar Multiplication

In this paper, we consider two special families of elliptic curves, namely Montgomery [24] and twisted Edwards curves [3]. The former achieve unprecedented efficiency in variable-base scalar multiplication, such as performed in static and ephemeral ECDH key exchange, whereas the latter excels in the other two use cases, namely fixed-base scalar multiplication (needed in e.g. ECDSA signature generation and ephemeral ECDH key exchange) and double-base scalar multiplication (performed in e.g. the verification of an ECDSA signature).

### 4.1 Montgomery Curve

The Montgomery model of an elliptic curve was originally introduced 20 years ago to speed up algorithms for integer factorization [24]. Formally, a so-called Montgomery curve over  $\mathbb{F}_p$  can be defined by an equation of the form

$$E_M : By^2 = x^3 + Ax^2 + x \quad (1)$$

where  $A, B \in \mathbb{F}_p$  and  $(A^2 - 4)B \neq 0$  [24]. Montgomery curves feature a unique addition law that is “special” in two aspects. First, the addition law describes a so-called differential addition, which means it allows one to compute the sum  $P_1 + P_2$  of two points  $P_1, P_2$  whose difference  $P_1 - P_2$  is known. Second, when using projective coordinates, both a point addition and point doubling can be performed using  $X$  and  $Z$  coordinates only, i.e. the  $Y$  coordinate is not needed for the computation. The usual approach to implement scalar multiplication on a Montgomery curve is to use the Montgomery ladder [24], a simple algorithm that executes both a (differential) addition and a doubling for each bit of the scalar, independent of its actual value. Therefore, the Montgomery ladder has a regular execution profile, which helps to thwart side-channel attacks.

We implemented the differential point addition and point doubling on basis of the formulae given in [24]. Consequently, the point addition consists of three multiplications, four squarings, three additions, and the same number of subtractions in  $\mathbb{F}_p$ . A doubling, on the other hand, takes two multiplications, two squarings, two additions, two subtractions, as well as a multiplication by the constant  $(A + 2)/4$ , which is a relatively cheap operation if the parameter  $A$  is chosen properly [2]. There are two implementation options for the ladder; one is the standard approach with separate addition and doubling functions, while the other combines both into a so-called “ladder step” [6]. The variant with the ladder step requires besides the normal  $\mathbb{F}_p$ -arithmetic operations also a special function for conditional swapping of two field elements, but has the advantage of a highly regular memory access pattern (i.e. the addresses used to load and

store intermediate results do not depend on secret information). We decided to use the standard approach since MSP430 devices have no cache (and, thus, do not leak timing information through secretly-indexed loads) and since we found it a little faster than the ladder-step variant when taking into account that the three least-significant bits of a scalar are 0 and require only doublings.

## 4.2 Twisted Edwards Curve

Twisted Edwards curves were first described in [3] as a generalization of (ordinary) Edwards curves. Some of these curves are equipped with a very fast and complete addition law, whereby the rational point  $\mathcal{O} = (0, 1)$  serves as neutral element. Completeness means that the addition law works for any two points  $P, Q$  that lie on the curve, including corner cases such as  $P = \mathcal{O}$ ,  $Q = \mathcal{O}$ , and  $P = -Q$  [3]. Formally, a twisted Edwards curve over a prime field  $\mathbb{F}_p$  is defined by the equation

$$E_T : ax^2 + y^2 = 1 + dx^2y^2 \quad (2)$$

where  $a, d \in \mathbb{F}_p$  and  $ad(a - d) \neq 0$ . As explained in [3], the completeness of the addition law depends on the two curve parameters. More precisely, when  $a$  is a square and  $d$  is a non-square in  $\mathbb{F}_p$ , then the addition law can be complete and have no exceptions. Completeness is a valuable property if one aims for a side-channel resistant implementation of scalar multiplication since the corner cases mentioned above do not need to be treated separately, which naturally leads to a regular execution profile and constant execution time.

The so-called extended coordinates presented by Hisil et al. [15] allow for a particularly fast addition of points when the curve parameter  $a = -1$ . In this case, a mixed addition (i.e. an addition where one of the two points is given in projective coordinates and the other in affine coordinates) requires only seven multiplications in  $\mathbb{F}_p$ . Extended projective coordinates were originally proposed in [15] as a quadruple of the form  $(X, Y, T, Z)$ , whereby the fourth coordinate  $T = XY/Z$ . In our implementation, we further split  $T$  up into the two factors  $E$  and  $H$ , i.e. we have  $EH = T = XY/Z$ , since this facilitates an optimization of the point doubling formulae. Consequently, we use a quintuple of the form  $(X, Y, E, H, Z)$  with  $EH = T = XY/Z$  to represent a projective point and the triple  $(u, v, w)$  with  $u = (x + y)/2$ ,  $v = (y - x)/2$ , and  $w = dxy$  to represent an affine point. The computational cost for a (complete) mixed addition amounts to seven multiplications, three additions, and three subtractions in  $\mathbb{F}_p$ , while a projective doubling takes three multiplications, four squarings, four additions and two subtractions. We implemented the scalar multiplication according to the highly-regular fixed-base comb technique described in [21], which uses eight pre-computed points and processes four bits of the scalar at a time.

As demonstrated by Bernstein et al. [3], every Montgomery curve over  $\mathbb{F}_p$  is birationally equivalent over  $\mathbb{F}_p$  to a twisted Edwards curve and vice versa. This equivalence is very useful in ephemeral ECDH key exchange since it allows one to perform the fixed-base scalar multiplication (for generating a key pair) on a twisted Edwards curve and the variable-base scalar multiplication (to compute the shared secret) on the birationally-equivalent Montgomery curve [21].

**Table 2.** Execution time and code size of arithmetic operations in 159, 191, 223, and 255-bit pseudo-Mersenne prime fields on a TI MSP430F1611 processor (all execution times include the full function-call overhead and the modular reduction; the values in parentheses indicate the time spent for the reduction operation alone).

Operation	Execution time (in clock cycles)				Code size (in bytes)
	159 bit	191 bit	223 bit	255 bit	
Addition	226	258	290	322	100
Subtraction	244	278	312	346	120
Multiplication	2448 (388)	3304 (452)	4288 (516)	5400 (580)	360 (168)
Squaring	1998 (388)	2578 (452)	3214 (516)	3914 (580)	406 (168)
32-bit Mul.	700 (232)	804 (258)	908 (284)	1012 (310)	282 (164)
Inversion	147440	202358	265318	336270	966

## 5 Results and Comparison

We compiled and assembled the source code of our ECC software for MSP430 microcontrollers using version 6.10 of IAR Embedded Workbench, which comes with a cycle-accurate instruction set simulator. Table 2 specifies the execution time and code size of the major operations of our parameterized  $\mathbb{F}_p$ -arithmetic library for 159, 191, 223 and 255-bit fields. The concrete primes with which we collected the simulation results are those from the four so-called MoTE curves specified in [10], namely  $2^{159} - 91$ ,  $2^{191} - 19$ ,  $2^{223} - 235$ , and  $2^{255} - 19$ . A full multiplication in a 159-bit pseudo-Mersenne prime field takes 2448 clock cycles altogether, to which the reduction contributes 388 cycles, i.e. the multiplication alone (without modular reduction) executes in 2160 cycles. Squaring (including reduction) is roughly 18.4% faster than multiplication. However, the difference between multiplication and squaring increases to some 27.5% in a 255-bit field (5400 versus 3914 cycles). Also provided in Table 2 is the execution time of the multiplication of a field element by a 32-bit integer; this operation can be used in e.g. the point doubling on a Montgomery curve for the multiplication by the constant  $(A + 2)/4$ . Inversion is the by far most expensive arithmetic operation in  $\mathbb{F}_p$ ; a single inversion takes slightly more time than 60 multiplications. Note that all operations listed in Table 2, except inversion, have constant execution time. Since the number of clock cycles for an EAA-based inversion depends on the value of the operand to be inverted, we specify the average execution time in Table 2, which we found through inversion of 100 random field elements.

The last column in Table 2 summarizes the code size of different arithmetic functions of our library. The function for multiplying multiple-precision integers according to Algorithm 1 has a size of 192 bytes, while the modular reduction function occupies only 168 bytes in Flash memory, i.e. both together amounts to 360 bytes. Squaring is, in terms of code size, slightly larger than multiplication (by exactly 46 bytes) since it needs an extra loop. In general, our  $\mathbb{F}_p$ -arithmetic library is very compact because we avoided code-size increasing optimizations

**Table 3.** Execution time (in clock cycles on an MSP430F1611) of variable-base scalar multiplication on a Montgomery curve and fixed-base scalar multiplication on a twisted Edwards curve over 159, 191, 223, and 255-bit fields.

Curve type	159 bit	191 bit	223 bit	255 bit
Montgomery (variable base)	$3.86 \cdot 10^6$	$6.00 \cdot 10^6$	$8.79 \cdot 10^6$	$12.34 \cdot 10^6$
Twisted Edwards (fixed base)	$1.92 \cdot 10^6$	$3.01 \cdot 10^6$	$4.45 \cdot 10^6$	$6.29 \cdot 10^6$

like loop unrolling. The size of the whole library amounts to about 2.3 kB; this includes besides the arithmetic operations also some auxiliary functions (e.g. to copy a field element or to check whether two field elements are equal).

We also evaluated the execution time of variable-base scalar multiplication (using the basic Montgomery ladder) on four different Montgomery curves and fixed-base scalar multiplication (using a comb method with eight pre-computed points) on the four bitrationally equivalent twisted Edwards curve. The curves we used to obtain the simulation results are specified in [10]. As summarized in Table 3, the execution times for variable-base scalar multiplication range from  $3.86 \cdot 10^6$  cycles (Montgomery curve over 159-bit field) up to  $12.34 \cdot 10^6$  cycles (255-bit field). On the other hand, the fixed-base scalar multiplications (on the twisted Edwards curves) take only about one half of the execution time of the variable-base scalar multiplications at the same security level. In terms of code size, the C implementation of the point arithmetic on the Montgomery curve is about 1.7 kB; this includes besides point addition/doubling and scalar multiplication also a few auxiliary functions (e.g. for projective-to-affine conversion of a point). The code size of the point arithmetic on the twisted Edwards curve amounts to roughly 2.1 kB, again including some auxiliary functions. We have eight pre-computed points in extended affine coordinates per curve, which, in total (i.e. for four curves), occupy 2.5 kB in Flash memory.

In recent years, numerous papers on efficient ECC software for MSP430(X) processors have been published, e.g. [6, 9, 14, 20, 22, 27, 32, 37]. However, in the majority of these works, ordinary curves in Weierstraß form were used and the implementations lack protection against timing attacks, which makes it hard to compare the reported results with ours. Only Liu et al. in [20] and Düll et al. in [6] adopted Montgomery curves, but they entirely unrolled the field arithmetic (i.e. these implementations are not scalable). The former authors achieved an execution time of  $3.25 \cdot 10^6$  and  $5.12 \cdot 10^6$  clock cycles for scalar multiplication on a 159-bit and a 191-bit Montgomery curve, respectively, which outperforms our scalable ECC software by less than 20%. Düll et al. reported  $7.93 \cdot 10^6$  cycles for a scalar multiplication on Curve25519 and a code size of 13.1 kB, but these results were obtained using an MSP430FR5969 as evaluation platform. To aid comparison, we simulated our software with the parameters of Curve25519 and obtained an execution time of  $10.85 \cdot 10^6$  clock cycles on the same device. Consequently, our scalable implementation is approximately 1.37 times slower than that of Düll et al., but more than three times smaller.

## 6 Conclusions

We presented the concept of energy scalability as an approach to minimize the total energy consumption of ECC operations in a WSN or, more generally, the IoT. Taking an ECC-based security architecture for a WSN as case study, we argued that node authentication has higher security requirements than e.g. the establishment of a shared secret key between nodes to encrypt short-lived data like sensor readings. By using elliptic-curve groups of smaller order for the less security-critical task(s), it is possible to save precious energy; for example, one could adopt a 191-bit curve for key establishment and a 223-bit curve for node authentication. We introduced a scalable yet efficient software implementation of ECC for 16-bit MSP430 processors that supports Montgomery and twisted Edwards curves. The core component of our ECC software is a parameterized library for arithmetic in pseudo-Mersenne prime fields, which is able to process operands of various lengths and has a binary code size of only 2.3 kB since we refrained from loop unrolling and other size-increasing optimizations. Nonetheless our software is only a factor of 1.37 slower than the high-speed Curve25519 implementation of Düll et al., but three times smaller. In summary, our results show that reaching good performance does not necessarily have to come at the expense of large code size and poor scalability.

**Acknowledgments.** Lin Li and Qiuliang Xu were supported by the National Natural Science Foundation of China under grant No. 61572294. This work was supported by the NSERC CREATE Training Program in Building a Workforce for the Cryptographic Infrastructure of the 21st Century (CryptoWorks21), and Public Works and Government Services Canada.

## References

1. Alfandi, O., Bochem, A., Kellner, A., Göge, C., Hogrefe, D.: Secure and authenticated data communication in wireless sensor networks. *Sensors* **15**(8), 19560–19582 (2015)
2. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
3. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008)
4. Bormann, C., Ersue, M., Keranen, A.: Terminology for Constrained-Node Networks. Internet Engineering Task Force, Light-Weight Implementation Guidance Working Group, RFC 7228, May 2014
5. Dang, D., Plant, M., Poole, M.: Wireless connectivity for the Internet of Things (IoT) with MSP430 microcontrollers (MCUs), March 2014. Texas Instruments white paper, <http://www.ti.com/lit/wp/slay028/slay028.pdf>
6. Düll, M., Haase, B., Hinterwälder, G., Hutter, M., Paar, C., Sánchez, A.H., Schwabe, P.: High-speed Curve25519 on 8-bit, 16-bit and 32-bit microcontrollers. *Des. Codes Crypt.* **77**(2–3), 493–514 (2015)

7. Evans, D.: The Internet of things: how the next evolution of the Internet is changing everything, April 2011. Cisco IBSG white paper, [http://www.cisco.com/web/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf)
8. Fan, X., Gong, G.: Accelerating signature-based broadcast authentication for wireless sensor networks. *Ad Hoc Netw.* **10**(4), 723–736 (2012)
9. Gouvêa, C.P.L., López, J.: Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller. In: Roy, B., Sendrier, N. (eds.) *INDOCRYPT 2009*. LNCS, vol. 5922, pp. 248–262. Springer, Heidelberg (2009)
10. Großschädl, J.: A family of implementation-friendly MoTE elliptic curves. Technical report TR-LACS-2013-01, Laboratory of Algorithmics, Cryptology and Security (LACS), University of Luxembourg, Luxembourg (2013)
11. Gupta, V., Wurm, M., Zhu, Y., Millard, M., Fung, S., Gura, N., Eberle, H., Chang Shantz, S.: Sizzle: a standards-based end-to-end security architecture for the embedded Internet. *Pervasive Mob. Comput.* **1**(4), 425–445 (2005)
12. Hankerson, D.R., Menezes, A.J., Vanstone, S.A.: *Guide to Elliptic Curve Cryptography*. Springer, New York (2004)
13. Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S.L., Kumar, S.S., Wehrle, K.: Security challenges in the IP-based Internet of things. *Wireless Pers. Commun.* **61**(3), 527–542 (2011)
14. Hinterwälder, G., Moradi, A., Hutter, M., Schwabe, P., Paar, C.: Full-size high-security ECC implementation on MSP430 microcontrollers. In: Aranha, D.F., Menezes, A. (eds.) *LATINCRYPT 2014*. LNCS, vol. 8895, pp. 31–47. Springer, Heidelberg (2015)
15. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: Pieprzyk, J. (ed.) *ASIACRYPT 2008*. LNCS, vol. 5350, pp. 326–343. Springer, Heidelberg (2008)
16. Kar, S.: Cisco says Internet of things will have ten times more impact on society than Internet, March 2014. *Cloud Times*, <http://cloudtimes.org/2014/03/07/cisco-says-internet-of-things-will-have-ten-times-more-impact-on-society-than-internet>
17. Karatsuba, A.A., Ofman, Y.P.: Multiplication of multidigit numbers on automata. *Soviet Physics - Doklady* **7**(7), 595–596 (1963)
18. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) *CRYPTO 1996*. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
19. Liu, Z., Seo, H., Großschädl, J., Kim, H.: Reverse product-scanning multiplication and squaring on 8-bit AVR processors. In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) *ICICS 2015*. LNCS, vol. 8958, pp. 158–175. Springer, Heidelberg (2015)
20. Liu, Z., Seo, H., Hu, Z., Huang, X., Großschädl, J.: Efficient implementation of ECDH key exchange for MSP430-based wireless sensor networks. In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2015)*, pp. 145–153. ACM Press (2015)
21. Liu, Z., Wenger, E., Großschädl, J.: MoTE-ECC: energy-scalable elliptic curve cryptography for wireless sensor networks. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) *ACNS 2014*. LNCS, vol. 8479, pp. 361–379. Springer, Heidelberg (2014)
22. Marin, L., Jara, A.J., Skarmeta, A.F.G.: Shifting primes: extension of pseudo-mersenne primes to optimize ECC for MSP430-based future Internet of Things devices. In: Tjoa, A.M., Quirchmayr, G., You, I., Xu, L. (eds.) *ARES 2011*. LNCS, vol. 6908, pp. 205–219. Springer, Heidelberg (2011)



23. Memsic, Inc.: TelosB Mote Platform, March 2007. Data sheet, [http://www.memsic.com/userfiles/files/Datasheets/WSN/6020-0094-02\\_B.TELOS.B.pdf](http://www.memsic.com/userfiles/files/Datasheets/WSN/6020-0094-02_B.TELOS.B.pdf)
24. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. *Math. Comput.* **48**(177), 243–264 (1987)
25. Moskowitz, R.G., Hummen, R.: HIP Diet EXchange (DEX). Internet Engineering Task Force, Network Working Group, Internetdraft draft-moskowitz-hip-dex-04 (work in progress), July 2015
26. Naccache, D., Smart, N.P., Stern, J.: Projective coordinates leak. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 257–267. Springer, Heidelberg (2004)
27. Pendl, C., Pelnar, M., Hutter, M.: Elliptic curve cryptography on the WISP UHF RFID tag. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 32–47. Springer, Heidelberg (2012)
28. Raghavendra, C.S., Sivalingam, K.M., Znati, T.F.: *Wireless Sensor Networks*. Kluwer Academic Publishers, Norwell (2004)
29. Rescorla, E.K., Modadugu, N.G.: Datagram Transport Layer Security Version 1.2. Internet Engineering Task Force, Network Working Group, RFC 6347, January 2012
30. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)
31. Sehgal, A., Perelman, V., Kuryla, S., Schönwälder, J.: Management of resource constrained devices in the Internet of things. *IEEE Commun. Mag.* **50**(12), 144–149 (2012)
32. Seo, H., Shim, K.-A., Kim, H.: Performance enhancement of TinyECC based on multiplication optimizations. *Secur. Commun. Netw.* **6**(2), 151–160 (2013)
33. Sinha, A.: *Energy Efficient Operating Systems and Software*. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (2001)
34. Smith, J.R. (ed.): *Wirelessly Powered Sensor Networks and Computational RFID*. Springer, New York (2013)
35. Stajano, F., Cvrcek, D., Lewis, M.: Steel, cast iron and concrete: security engineering for real world wireless sensor networks. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 460–478. Springer, Heidelberg (2008)
36. Texas Instruments, Inc.: MSP430x1xx Family User’s Guide (Rev. F), February 2006. Manual, <http://www.ti.com/lit/ug/slau049f/slau049f.pdf>
37. Wenger, E., Werner, M.: Evaluating 16-bit processors for elliptic curve cryptography. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 166–181. Springer, Heidelberg (2011)
38. Zhou, Y., Zhang, Y., Fang, Y.: Access control in wireless sensor networks. *Ad Hoc Netw.* **5**(1), 3–13 (2007)

# **National Security Infrastructure**

# A Comparison Study of Wireless Network Security in Several Australasian Cities and Suburbs

Alastair Nisbet<sup>1</sup>(✉) and Andrew Woodward<sup>2</sup>

<sup>1</sup> School of Engineering, Computer and Mathematical Sciences,  
Auckland University of Technology, Auckland, New Zealand  
anisbet@aut.ac.nz

<sup>2</sup> School of Computer and Security Science SRI - Security Research Institute,  
Edith Cowan University Perth, Joondalup, WA, Australia  
a.woodward@ecu.edu.au

**Abstract.** Wireless network technology has been available for public and private use for almost two decades. The casual approach to security with the early standards and channel selection began to cause issues when the initial security standard, WEP, was discovered to have serious flaws. The closer examination of security and efficiency that followed led to better security protocols, easier setup and better guidelines to ensure efficiency of radio communications. A survey of 5 areas throughout New Zealand was conducted and this is compared with a survey of 5 areas around Perth in Western Australia. The results indicate that whilst improvements continue to be made in security implementations, a small percentage of users do not implement their networks with the recommended settings. Whilst Australian users are slightly better at complying with recommendations, it is clear that some work still needs to be done in the areas of security and efficiency.

**Keywords:** Wireless · Network · Security · Privacy · Bandwidth

## 1 Introduction

Wireless networks have been available for public use for almost two decades. Beginning in 1997 with the original IEEE 802.11-1997 standard, the uptake of wireless technology began to see a slow but steady rise. Whilst other wireless technology was available, it tended to be proprietary and enjoyed by relatively few users. The difference with the IEEE standards was that for the first time they could be utilised by all equipment manufacturers and relatively easily work with different computer manufacturers and operating systems. In 1999, the development and ratification of 2 new standards saw wireless acceptance begin to enjoy rapid growth. This was primarily for 3 reasons. Firstly, the new standards, IEEE 802.11a and 802.11b offered much greater bandwidth and the option of increased non-overlapping channels, especially in the case of 802.11a. This allowed for multiple wireless devices to communicate simultaneously without interfering with each other's transmissions, something that greatly improved the use of multiple access points within radio range of each other. The second

improvement was the inclusion of security within the standards, albeit to prove somewhat flawed at a later date. Finally, the Wireless Ethernet Compatibility Alliance (WECA) was established which tested the IEEE wireless devices from various manufacturers for compliance with the standards. If their testing found that devices were suitably designed to comply with the WECA's standards, a stamp of approval was given and users could be assured that these 'Wifi Certified' devices would interoperate with all other similar Wifi Certified devices. In 2003, WECA changed its name to the Wifi Alliance and continues to certify products. By 2015 over 25000 different devices from hundreds of manufacturers have been WiFi Certified (Wifi.org).

Whilst the standards were seeing further developments at regular intervals that generally focussed on improved data transfer rates, the security issues with WEP began to surface in 2000 [1]. This became more serious in 2001, beginning with a theoretical attack against a WEP key published that year [2]. This led to a series of articles highlighting possible weakness in the security of WEP [3] and was followed shortly afterwards by a practical implementation of the FMS attack in 2002 [4]. This led to a period of uncertainty over the security of wireless networks [5] that lasted for several years, even after much improved security protocols were developed [6].

The effect was that wireless device sales slowed with the perception that wireless security was now a problem that was yet to be solved. In 2003, the 802.11G standard was ratified and incorporated into this standard were new security measures intended as an interim measure. WiFi Protected Access (WPA) utilising a pre-shared key between devices and the Temporal Key Integrity Protocol (TKIP) allowed for much greater security by updating keys at regular intervals without user intervention. A further development in security designed as an enterprise solution was IEEE 802.11i [7]. This new standard, also called WPA2 was initially designed to be implemented by a dedicated server attached to a corporate WLAN and utilising the AES encryption algorithm [8]. AES had been adopted by the United States Government as the official security standard because of its extremely high resilience to attack. The new standard would later be modified to utilise a pre-shared key allowing domestic users with a home access point to implement the much greater security offered by AES as an option. This left a range of security choices for users, from no security, often referred to as 'open' security, to the outdated and insecure WEP security, to WPA and the most robust security of WPA2.

Whilst WPA was designed by the WiFi Alliance as an interim standard until a more robust solution could be found, the design proved to provide high security with a relatively simple implementation. This 'temporary' standard is still utilised and for most home users tends to be the choice for security. However, as with WEP, WPA has been shown to suffer from several vulnerabilities. Firstly, during the authentication phase an attack is possible that can discover the PIN code used during the setup [9], and secondly encryption keys fewer than 20 characters are considered insecure. Those keys approaching 20 characters are still possible to crack but it would take a determined attacker many months at least to crack the key [10].

Whilst metropolitan wireless networks and local public areas such as cafes and restaurants may deliberately offer open networks for the public and customers, business organisations and home users should implement high security on their devices. This not only prevents unauthorised users utilising their network connections but ensures

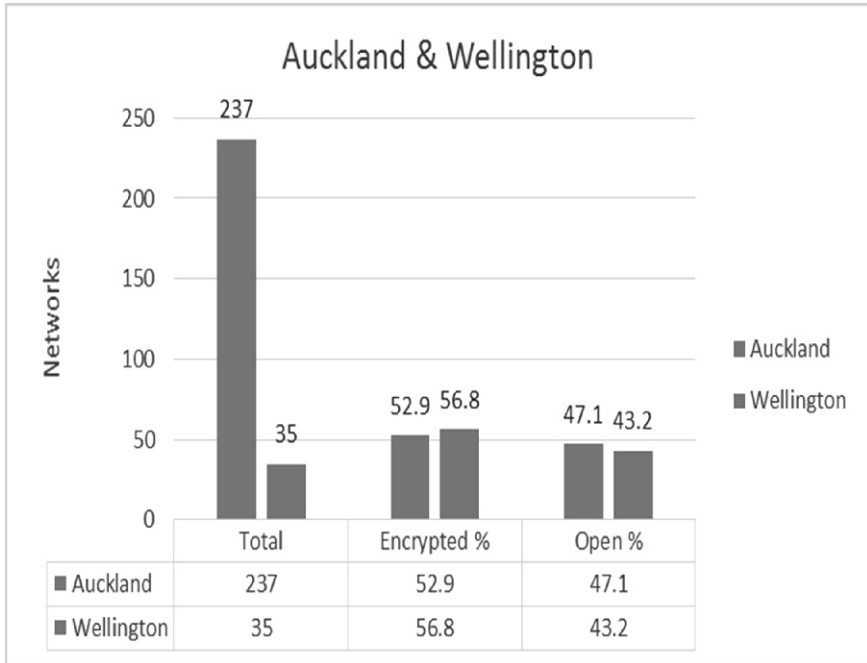
privacy of messages including emails, web browsing and user names and passwords utilised for sites that may have highly confidential information contained within them. The issues with wireless networks and the vulnerabilities have seen much publicity, both from academia and from more mainstream media. This began in 2001 with the FMS attack and has progressed over the years to expose security vulnerabilities in WPA. This should mean that both home and commercial premises are equipped with the highest level of security, with users aware of the dangers of no or low security and a corresponding increase in security of devices.

The following section examines the results of surveys taken in New Zealand cities from 2004 to 2011. This is discussed to show the progression of security over this period leading up to 2012. This is followed by the latest surveys conducted in 2013 in New Zealand's four largest cities and a survey of several urban suburbs of Auckland. The results are used to compare the wireless security to that of several smaller suburbs of Perth as well as the Perth CBD. This comparison will serve to show whether or not security has progressed evenly within the two countries. Conclusions are then drawn examining the state of security and what if anything still needs to be done to increase security of wireless networks.

## 2 New Zealand Wireless Security Surveys

The first survey of wireless networks in New Zealand was conducted in Auckland City CBD. Insider software was utilised with an external aerial to ensure the best possible detection of a wireless network operating. Approximately 12 km was covered within the central city area and the results are shown in Fig. 1.

In 2004, WPA had just began to be implemented. The software utilised at that time did not differentiate between WEP and WPA, so the results simply show whether encryption is utilised or not. Also at that time, free to use wireless networks were fairly uncommon meaning that encryption would be expected to be used on almost all networks. However, results showed that little over half the networks were implemented with security. This appears to be for a number reasons. Firstly, users were far less aware in 2004 of the issues that arose when security was not implemented and secondly the wireless equipment had no security switched on by default. This combination meant that many networks had no security implemented and users were often unaware that this was the case. By 2011 this had changed significantly. In the seven years since 2004, wireless technology had received much more attention from media and manufacturers meaning that when installing their networks, users were more likely to switch security on. The 2011 survey of Auckland and Wellington was conducted and two further cities were added. Christchurch is the third largest city in New Zealand and Dunedin is the fourth largest. These surveys gave not only a good geographical spread of New Zealand but all four cities are diverse in their makeup. Wellington is the capital city and has many government agencies including military agencies and Dunedin has a small centralised CBD with a number of student houses and flats within the central city servicing Otago University and Otago Polytechnic. Christchurch has a much more spread out CBD with older buildings of generally 4 stories at most. By adding the two south island cities, a comparison could be made within New Zealand of how



**Fig. 1.** Auckland and Wellington results 2004

information regarding security settings and technological expertise may have had an impact on wireless network security settings. The results of the 2011 survey are shown in Fig. 2.

The final survey in the series of New Zealand cities was conducted in 2013. This covered all four cities and was conducted on a weekday at the same time of day as the previous surveys. The number of networks detected had increased significantly over this time. A devastating earthquake in 2012 had destroyed many of the inner city buildings of Christchurch and this necessitated the survey taking place around the perimeter of the CBD area. Despite this, a significant increase in the number of networks was detected and a similar drop in the percentage utilising encryption to the other cities was also apparent. For 2013, many of the suburbs surrounding Auckland City were added to the survey. Over 5000 networks were discovered over a distance around the suburban roads of approximately 60 km. This is a 20 % greater distance covered driving around the CBD’s of the four cities.

The number of networks discovered in the 50 km drive through the CBD’s totals 9040, indicating that approximately twice as many networks are present in business areas as in the suburbs. However, encryption is implemented in 97.4 % of suburban networks compared to an average of 88.6 % for inner city networks. Whilst the possible reasons for the lower encryption have been briefly discussed, the very high implementation of encryption in the suburbs is encouraging. It would appear that a combination of publicity and media attention regarding the dangers of unsecured

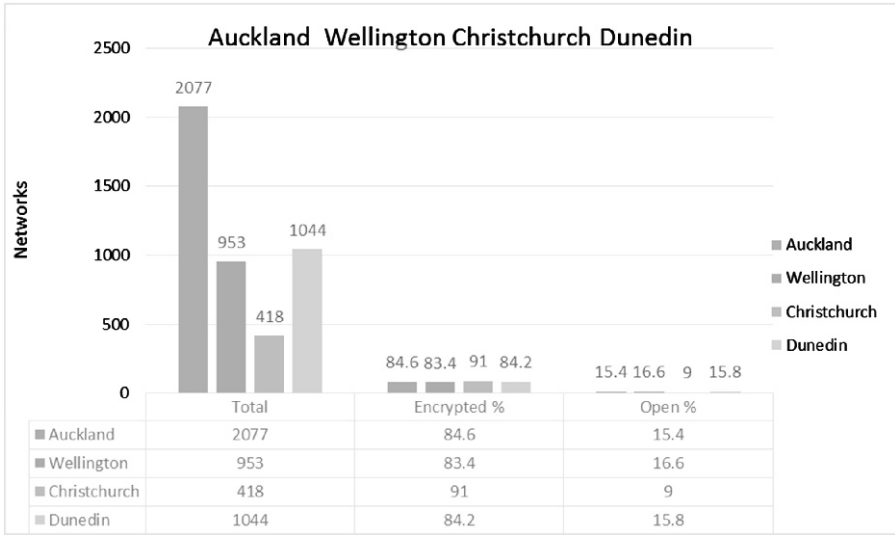


Fig. 2. Four cities throughout New Zealand in 2011

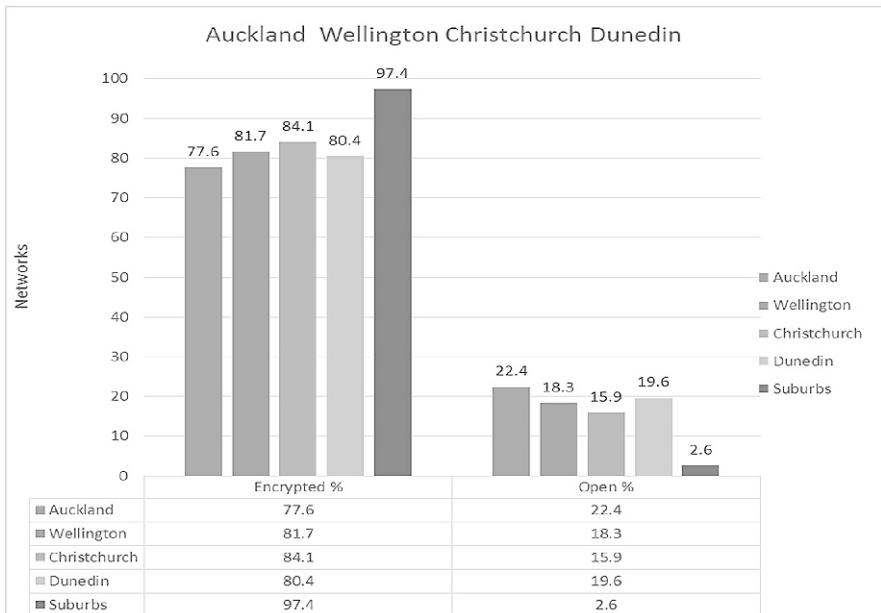
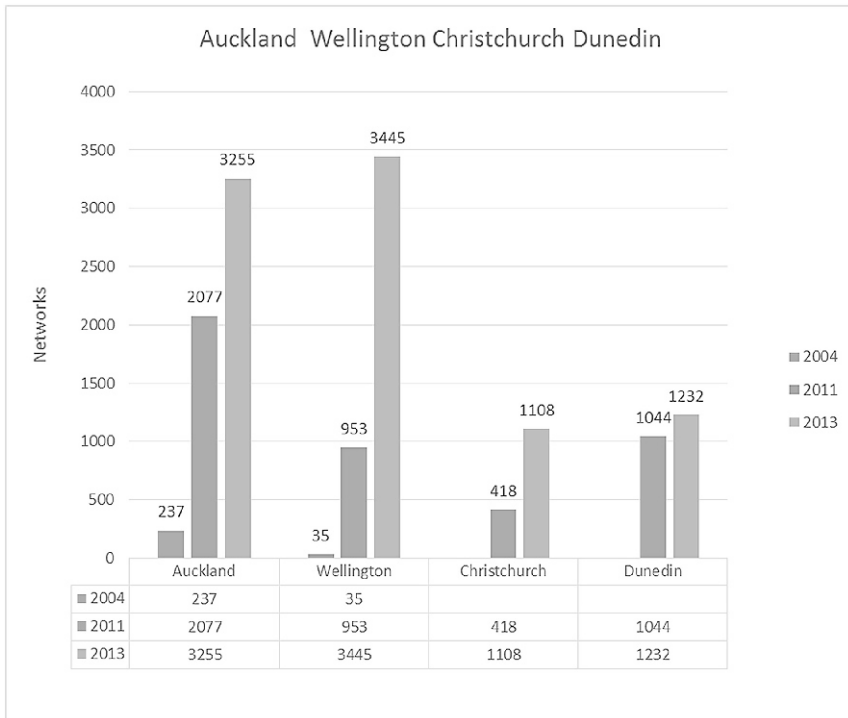


Fig. 3. Encryption percentage in 2013 for four cities throughout New Zealand

networks and the greater ease with which encryption can be set up on home networks, including WiFi Protected Setup (WPS) available at the touch of a button, is having a positive effect in ensuring these networks are secure [9]. The encryption percentage results of this survey are shown in Fig. 3.

The increase in the number of networks detected over the eight years is one interesting factor from the survey. The greatest increase is in the capital city with an almost 10 000 % increase from 35 networks to 3445. Christchurch and Dunedin’s surveys cover a 3 year period only and both show increases but with Dunedin showing a very modest increase. These results are shown in Fig. 4.



**Fig. 4.** Comparison of CBD wireless network numbers over 9 years

Overall, the increase in networks indicates a significant adoption of wireless technology both in business organisations and for home users. The increase in encryption utilised and allowing for the many free and deliberately unsecured networks indicates that confidence in wireless network security has reached a point where it is a trusted technology. Something that may well have occurred much earlier if the original security standard, WEP, had proven to be as secure as promised. The wireless network growth seen up to 2013 may well have occurred much earlier. The following section

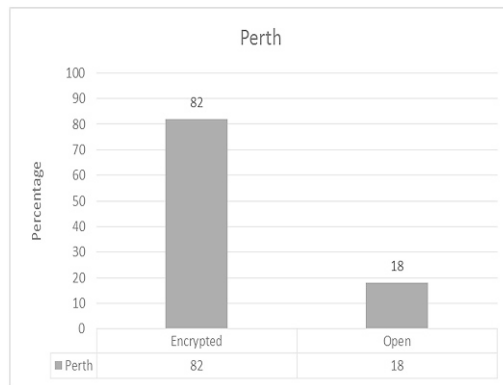


discusses a similar survey of suburban and semi-industrial areas near the city of Perth in Australia.

### 3 Australia Wireless Security Surveys

In 2012 a survey of 4 suburbs near Perth and the Perth CBD was conducted. These areas are a mixture of suburban and light industrial organisations. Perth, like Auckland is a city spread over a large area with multiple outer suburbs. Unlike the New Zealand surveys, the Australian surveys were all conducted within the general area of one, large city. However, the population of Perth at approximately 1.8 million is fairly close to the total population of cities surveyed in New Zealand at 2 million. This and the fact that the cities share many of the same features and qualities of the New Zealand cities make for many comparisons between the two. The research question was whether these similar areas would share similar qualities with their wireless network security. The first of these surveys involved the CBD of Perth where 2142 networks were discovered. The encryption percentages are shown in Fig. 5.

The results show that this compares fairly closely with an average of the results from Auckland for 2011 and 2013, both in numbers of networks and with the encryption utilised. The implication being that similar influences as regards the necessity of encrypting business networks are working in Perth and New Zealand. Next, the outer suburbs of Perth were compared to see how well the results matched with New Zealand's smaller cities. The results of these surveys are shown in Fig. 6.



**Fig. 5.** Encryption percentage in 2013 for Perth CBD

With almost 3000 networks discovered in the outer suburbs and over 2000 within the Perth CBD, a good comparison between the suburbs and the city can be made. As with the New Zealand results, the city shows a reasonably predictable number of wireless networks detected based on the population and as with New Zealand, encryption implementation is more common for the outer suburbs than for the central

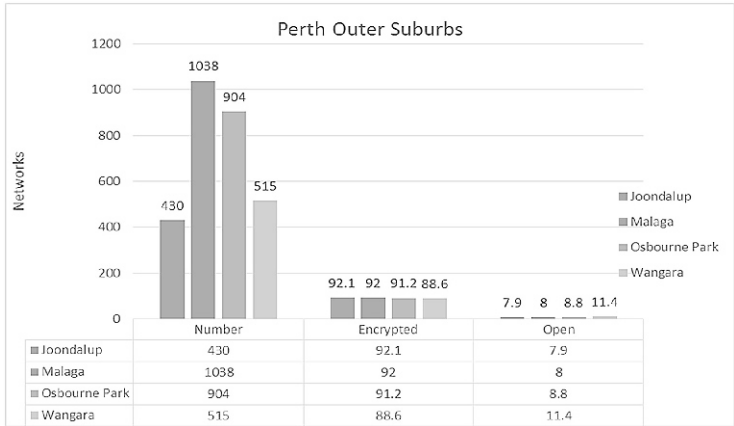


Fig. 6. Encryption percentage in 2012 for Perth’s outer suburbs

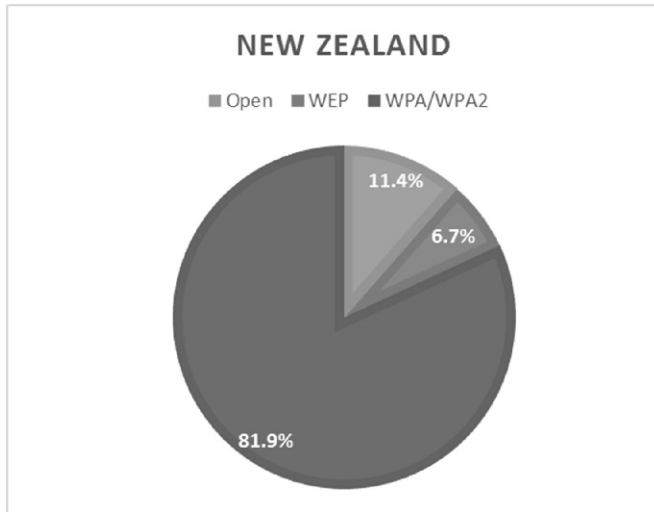
city. This would appear to be for similar reasons as with New Zealand cities where free and deliberately unsecured wireless networks are common in the central CBD. Whilst the message about implementing security and the manufacturers’ assistance with ease of setup for encryption is proving to be successful, an examination of the type of encryption utilised and the channel selections shows a deeper examination of the implications of utilising wireless networks both in a secure and efficient manner.

#### 4 Encryption Protocols and Channel Selections

Whilst a survey such as these gives a good overview of wireless security at a given time, the reasons for the results are often necessarily left largely to speculation. One change that is apparent in the results is that in all four cities in New Zealand, the implementation of encryption has decreased from 2011 to 2013. This appears to signify a problem with security but this would seem to be unlikely. It is far more likely that the growth in free wireless networks within the CBD’s of these cities, which are provided free of charge but with no security, is the reason for the drop in security. The free and unencrypted networks appear to be growing at a faster rate than the networks overall and therefore whilst security in networks in businesses is increasing, the overall percentage is dropping. Therefore, there should be no expectation of a 100 % encryption uptake but rather something less than this should be expected. The problem from a research point of view is that there is simply no way to be sure which networks are deliberately left open, as the Service Set ID of the network does not always permit an assumption of open security. In Auckland, names such as “Auckland Metro WiFi” and in Perth many use the term “Guest” may indicate a deliberately open wireless network, other names are not so clear that the intention is for them to be available to the public unencrypted. However, it would appear that security has reached a point where most business networks in the CBDs are secured with encryption while most of those that are unencrypted are intentionally so. No doubt there are still some networks that are

unintentionally unencrypted but it is simply not possible to be sure which ones or how many there are. What is more of a concern is those networks that are utilising WEP. This type of encryption is insecure and utilising WEP gives a false sense of security, something that is worse than users knowing that there is no security.

An examination in both countries of the type of encryption utilised highlights several issues. Firstly, WEP is still available on most newer models of wireless equipment but is provided to support legacy equipment that does not support later encryption protocols such as WPA and WPA2. Additionally, utilising features of some network equipment may have the undesirable side effect of having to downgrade the encryption implemented to WEP. One example of this is using a wireless access point as a hopping point to extend the range of the main access point. In some equipment that is only a few years old, the encryption utilised for this feature is WEP only, and so the user implementing this feature has no choice but to significantly compromise the security of their network by effectively leaving a side door partially open. Figure 7 shows the security implementations of all New Zealand networks surveyed in 2013.



**Fig. 7.** Encryption protocols for New Zealand in 2013

The breakdown of security protocols implemented shows that WPA and WPA2 account for 81.9 % of the encryption implemented in New Zealand. WPA has had some criticisms regarding its security unless implemented in the recommended way. However, it has proven to be generally secure. WPA2 has not suffered from any criticisms for its security and is therefore generally the recommended best practice. However, WPA and WPA2 are considered to provide sufficient security whereas WEP is considered to give a false sense of security because of the ease with which the encryption key can be recovered by someone with even fairly limited technical skills. From the New Zealand results we can see that 6.7 % of networks are utilising WEP and

therefore the conclusion is that 6.7 % of networks are desired to be secure but are failing in this goal.

For the Australian survey, very similar results were obtained as shown in Fig. 8.

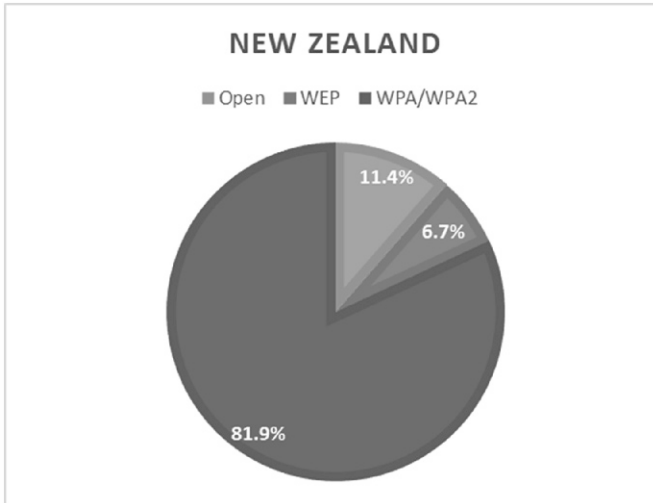
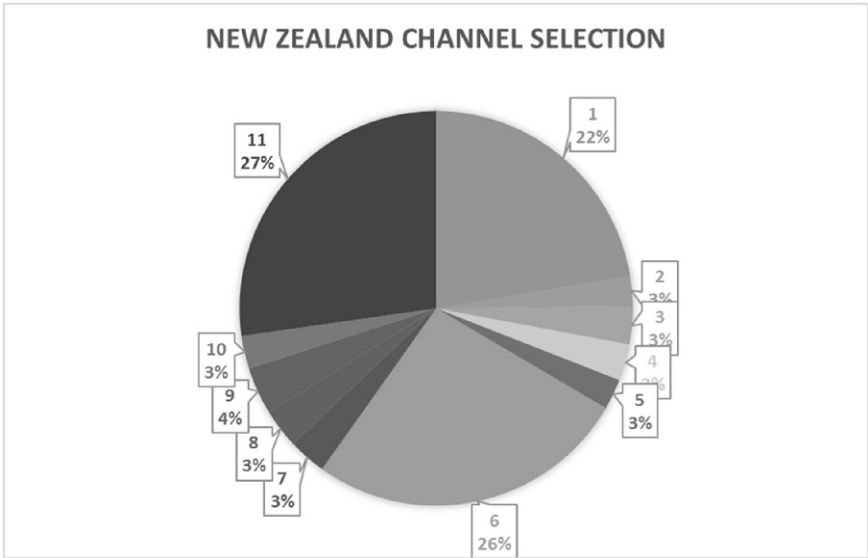


Fig. 8. Encryption protocols for Australia in 2012

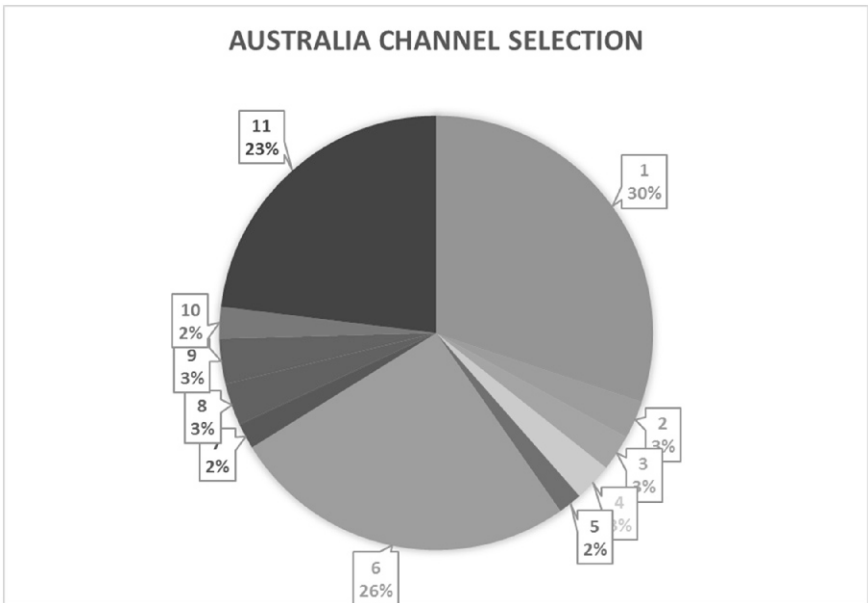
The networks utilising WPA or WPA2 are almost identical in value at 81.5 %. The open networks have a slight difference with more in Perth which may be accounted for with more ‘free to use’ networks being publicly available. There is a slight improvement over the New Zealand results with 1.3 % fewer networks using WEP, something that indicates that the Australians are slightly better at securing their networks than the New Zealanders.

Finally, the selection of channels is compared to indicate how efficiently the networks are being utilised as regards data throughput. Recommendations are that channels 1, 6 and 11 only should be used as the channels ‘bleed’ over into neighbouring channels affecting the throughput of the neighbouring channels. This channel bleed affects the two neighbouring channels so that if only the recommended channels are utilised there will be no interference from the other channels giving 3 non-overlapping channels that can be used at the same time. If a network is using channel 3 for example, it will affect both channels 1 and 6, slowing their throughput. Channel 3’s throughput will be affected by networks using channels 1 and 6, so the effect is much slower throughput for all users. Figure 9 shows the channel selection for the New Zealand survey.

Usage of channels 1, 6 and 11 account for 75 % of the networks surveyed. Whilst this shows that most users are aware that these channels should be used for efficient running of their networks,  $\frac{1}{4}$  of the networks are being implemented inefficiently, and worse affecting those networks that have been installed correctly. Figure 10 shows the results for the Australia survey which compare closely with that of New Zealand.



**Fig. 9.** Channel selection in New Zealand in 2013



**Fig. 10.** Channel selection in Australia in 2012

The total percentage of networks using channels 1,6 and 11 is 79 %, again slightly higher than New Zealand's total. This indicates that users in the Perth area are complying with the recommended channels better than their counterparts in New Zealand. From these results it appears that as the Australian users are using their networks with better encryption and by complying more closely with the recommended channels, they are being better informed of the recommendations and are complying with the information more closely than those in New Zealand. It is interesting to note that in 2012 the Australian Police in Queensland planned to survey wireless networks to identify insecure deployments and inform their owners where possible of their insecurity and how best to deploy them safely [11]. The publicity generated with the best of intentions led to criticism that the Police should not be snooping on networks as it was described as 'none of their business' if users did not secure networks securely. Whilst this odd stance by the media and some of the public led to the information campaign being cancelled, the publicity generated may well have assisted in educating readers at least of the vulnerabilities, perhaps leading to some increase in security of the networks.

## 5 Conclusion

Wireless networks have been available as an IEEE standard since 1997. Security in the form of WEP was provided with the wireless equipment when IEEE 802.11 was extended to the 'a' and 'b' versions in 1999. The initial issues with this security standard were much publicised and in 2003 WPA and later WPA2 became available. Wireless devices very quickly became available at cheaper and cheaper prices that incorporated these later security standards, yet the surveys show that some networks are still being deployed with WEP. This is worrying considering that replacement equipment is cheap and the benefits from secure networks are many, especially with privacy of data and protection of assets. The poor channel selections that account for 25 % of networks in New Zealand and 21 % of networks in Australia indicate that there is still work to be done to educate and assist users in how most effectively to deploy their networks. Whilst these studies are now 3 and 4 years old respectively, later casual surveys indicate that only slight improvements have been made. There still needs to be work from manufacturers, retailers and network administrators to continue improvements to ensure all users are utilising their wireless networks as securely and as efficiently as possible.

## References

1. Walker, J.: Unsafe at any key size, An analysis of the WEP encapsulation (2000). <http://www.dis.org/wl/pdf/unsafe.pdf>. Accessed 19 July 2005
2. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the Key Scheduling Algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 1–24. Springer, Heidelberg (2001)
3. Arbaugh, W.A., et al.: Your 80211 Wireless Network has no Clothes. *IEEE Wirel. Commun.* **9**(6), 44–51 (2002)

4. Stubblefield, A., Loannidis, J., Rubin, A.: Using the Fluhrer, Mantin, and Shamir attack to break WEP. In: Network and Distributed Systems Security Symposium (2002)
5. Cam-Winget, N., et al.: Security flaws in 802.11 data link protocols. *Commun. ACM* **46**(5), 35–39 (2003)
6. Vibhuti, S.: IEEE 802.11 WEP (Wired Equivalent Privacy) Concepts and Vulnerability (2005)
7. Walker, J.: 802.11 Security Series Part III: AES Based Encapsulations of 802.11 Data. [http://jcbserver.uwaterloo.ca/cs436/handouts/miscellaneous/Intel\\_Wireless\\_1.pdf](http://jcbserver.uwaterloo.ca/cs436/handouts/miscellaneous/Intel_Wireless_1.pdf) (2003). Accessed 19 July 2005
8. Nechvatal, J., et al.: Report on the Development of the Advanced Encryption Standard (AES), 2nd edn., October 2000. 19th January 2007
9. WiFi Alliance Introducing WiFi Protected Setup (2007)
10. Lashkari, A.H., Danesh, M.M.S., Samadi, B.: A survey on wireless security protocols (WEP, WPA and WPA2/802.11i). In: 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009 (2009)
11. Kirk, J.: In Australia Secure Your Wi-Fi or Face a Visit From the Police (2012)

# On the Guessability of Resident Registration Numbers in South Korea

Youngbae Song<sup>1</sup>, Hyounghick Kim<sup>1(✉)</sup>, and Jun Ho Huh<sup>2</sup>

<sup>1</sup> Department of Computer Science and Engineering,  
Sungkyunkwan University, Suwon, Republic of Korea  
{youngbae,hyoung}@skku.edu

<sup>2</sup> Honeywell ACS Labs, Golden Valley, USA  
junho.huh@honeywell.com

**Abstract.** This paper studies a potential risk of using real name verification systems that are prevalently used in Korean websites. Upon joining a website, users are required to enter their Resident Registration Number (RRN) to identify themselves. We adapt guessing theory techniques to measure RRN security against a trawling attacker attempting to guess victim's RRN using some personal information (such as name, sex, and location) that are publicly available (e.g., on Facebook). We evaluate the feasibility of performing statistical-guessing attacks using a real-world dataset consisting of 2,326 valid name and RRN pairs collected from several Chinese websites such as Baidu. Our results show that about 4,892.5 trials are needed on average to correctly guess a RRN. Compared to the brute-force attack, our statistical-guessing attack, on average, runs about 6.74 times faster.

**Keywords:** Korean identification system · Resident Registration Number · Brute-force attack · Statistical-guessing attack

## 1 Introduction

Just as Social Security Numbers (SSNs) are used in the United States to identify residents for various governmental purposes (e.g., tax or benefits), all Korean residents get a national identification number called the “Resident Registration Number” (RRN). It is a unique 13-digit number that is issued to every Korean at birth; the “Resident Registration Act,” which came into effect on 1962, prohibits anyone from changing the issued RRN. Government, banking, and medical services all use RRNs to identify and track Korean residents. Hence, RRNs are very sensitive and their confidentiality must be protected. The reality, however, is that there are too many careless uses of RRNs both online and offline. As a result, millions of valid RRNs are actively being traded in the Chinese black-markets. In March 2010, a group of criminals were arrested for attempting to sell about 20 million RRNs [10].

Although online anonymity protects users' privacy, it can also be misused by those who try to spread rumors or defame others. To mitigate such undesirable



online activities, the Korean government passed the “Real Name Verification Law” in July 2007, which regulates the following [4, 8]:

*If a website that has more than 100,000 visitors per day, all users of that website must verify their real name in order to sign up or write posts.*

The most popularly deployed verification system verifies real names by asking users to enter their RRNs upon joining a website or writing posts. Hundreds and thousands of popular Korean websites started collecting users’ RRNs without following any security standards for protecting them. Such trends exposed huge privacy risks, and as one would expect, there were several large-scale database breaches in the last few years that led to users’ RRNs being compromised. For example, the Cyworld and Nate database breach [13] affected about 35 million users. Such incidents forced the government to amend the Real Name Verification Law. The amended law prohibits information and communication service providers from collecting RRNs from their users. That amendment did not discourage many websites though, and many websites still use RRN-based real name verification systems.

In this paper, we discuss a security risk associated with using RRN-based identification systems, demonstrating how easy it is to guess RRNs using commonly used name verification systems. At first glance, the theoretically possible space of 13-digits numbers looks sufficiently large to resist brute-force attacks. This is not true though. The actual RRN space is much smaller, making various types of guessing attacks feasible. Although similar guessing attack has been performed on the U.S. national identification number [1], our work is another valuable case study that provides further insight into the security implications of deploying a national identification number system. Our results further emphasize that deploying a secure and usable national identification number system is challenging because they can easily be misused to impersonate others.

To analyze the security of RRN-based identification systems, we adapt guessing theory techniques to demonstrate how robust existing systems are against a trawling attacker trying to guess correct RRNs using some publicly available personal information such as name, sex or location that can be obtained easily through popular social networks like Facebook or LinkedIn. We used real-world datasets (collected from Chinese websites like Baidu) consisting of 2,326 valid name and RRN pairs to evaluate the feasibility of performing statistical guessing attacks which take into account the probability distribution of real RRNs. Using our statistical-guessing attack, only about 4,892.5 trials are needed on average to correctly guess an RRN, outperforming the pure brute-force attack by about 6.74 times on average. As a result, the actual security of RRN-based identification systems is worse than our hopeful expectation.

To mitigate such statistical-guessing attacks, we recommend using a security policy to limit the number of RRN verification attempts. Our analysis demonstrates that we can effectively prevent about 99.94 % of guessing attack attempts by setting that number to 7.

The rest of the paper is organized as follows. Section 2 explains the structure of RRNs. Section 3 analyzes the guessability of RRNs with the collected RRN datasets. Our suggestions against guessing attacks are discussed in Sect. 4. Next, we explain how ethical issues were considered in Sect. 5. Related work is discussed in Sect. 6, and our conclusions are in Sect. 7.

## 2 Structure of Resident Registration Numbers

RRNs are pervasively used on the Internet, allowing the government, banking, and medical services to identify the Korean individuals. An RRN is validated by comparing the last digit against what it should be based upon the rest of the digits entered. In this section, we describe in detail how RRNs are validated.

### 2.1 Resident Registration Numbers

RRN is a 13-digit number issued to an individual by the Korean government for tracking individuals efficiently. RRNs are much like national identification numbers used in other countries (e.g., Social Security Numbers (SSN) used in the US), and are used by tax, banks, and websites to identify and authenticate the residents in South Korea. However, unlike SSN that is decoupled from individuals' personal data since 2011, RRNs contain residents' personal information such as "date of birth" and "place of birth". That number has the following structure:

$$yymmdd-sccppnv$$

The first six digits, *yymmdd*, represent an individual's date of birth in the order of year, month, and day. For example, an individual born on March 21, 1987 would have an RRN that starts with 870321. The seventh digit, *s*, indicates the sex of an individual. The eighth through eleventh digits, *ccpp*, represent the place of birth. The eight and ninth digits, *cc*, signify an individual's city of birth (e.g., Seoul). The tenth and eleventh digits, *pp*, signify the "dong" of birth, which is the smallest region in a city that has its own government office and staff. The twelfth digit, *n*, is a sequential number used to differentiate those that have the same sex, born on the same day and in the same location (i.e., dong). The thirteenth digit, *v*, is used to verify the digit. It is generated from the rest of the digits using the following equation:

$$v = (11 - (\sum_{i=1}^8 (i+1) \cdot ar[i] + \sum_{i=9}^{12} (i-7) \cdot ar[i] \bmod 11)) \bmod 10$$

In summary, RRNs consist of information about birth (i.e., date of birth, place of birth, and birth registration order). However, since some parts of the birth data and/or their statistical properties are already available to the public, the real RRN space is much smaller than that of the theoretical space—for

example, Gross et al. [5] found that 87.8% of active Facebook users revealed their birth date—this is why guessing attacks can be effective on RRNs. The guessability of RRNs is discussed in the next section.

## 2.2 Example of Resident Registration Numbers Service

A RRN is a government issued 13-digit identification number assigned to South Korean residents, which is used when residents register online or make online transactions. Many South Korean websites require users to enter their full name and a valid RRN to sign up and retrieve forgotten passwords. (see Fig. 1).

성명(Name):	<input type="text"/>
주민등록번호 (RRN)	<input type="text"/> - <input type="text"/>

Fig. 1. Example of Resident Registration Numbers service

## 3 Guessability of RRNs

To evaluate how secure RRN-based identification systems are against guessing attacks, we first collected real-world RRNs and analyzed their statistical characteristics.

Interestingly, large volumes of Korean RRNs are available on Chinese websites such as Baidu (<http://www.baidu.com>), which we suspect are from previous RRN leaks. RRNs are lucrative targets for Chinese hackers too, because RRNs are often needed to impersonate and create user accounts on Korean websites [7, 10]. With increasing Korean pop and drama popularity in China, there is also a rapidly growing interest in Chinese population to access Korean websites, especially Korean online shopping malls, and to purchase trendy Korean items (e.g., clothes and accessories worn by Korean celebrities). Figure 2 shows some examples of RRNs accessed through Chinese websites like Baidu.

During a two-days period, we found several Chinese webpages containing RRNs and collected a total of 3,007 name and RRN pairs. Through a Korean website using RRN-based authentication, we tested the validity of the collected pairs and finally obtained 2,326 valid ones. We examine the statistics of those RRNs as follows.

### 3.1 Occurrence Frequency of Birth Data in RRNs

First, we analyzed the occurrence frequency of the city of birth in the collected RRNs. The frequencies are graphed in a descending order (see Fig. 3(a)). The



(a) Chinese websites (Baidu) (b) Korean residents' RRNs

Fig. 2. Examples of RRNs accessed through Chinese websites like Baidu.

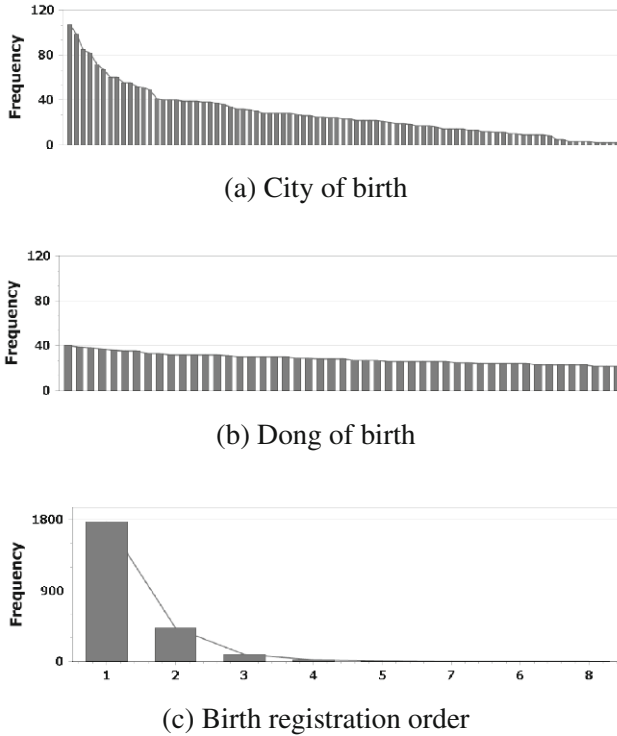
histogram shows the occurrence frequency of the city of birth decreasing dramatically, which indicates that the city of birth distribution is heavily skewed in favor of a small number of common birth places (e.g., Seoul).

Also, the occurrence frequencies of the dong of birth (fine-grained location), sorted in a descending order, are shown in Fig. 3(b). In contrast, the dong of birth was more evenly distributed. It would be relatively easier for an attacker to guess the birth city than the birth dong.

Last, the occurrence frequency of the birth registration order was analyzed. The frequencies are also sorted in a descending order (see Fig. 3(c)). The histogram shows the birth registration order occurrence frequency decreasing dramatically from the 2nd order, which indicates that the birth registration order distribution is heavily skewed. The most popular order, ‘1st’, alone, accounted for 76.3 % (1,775 out of 2,326) of the total number of the collected individuals.

### 3.2 Correlation Between City of Birth and Dong of Birth

We also analyzed the correlation between the birth city and the birth dong. Plotting the relationship between them in a 2-dimensional grid (see Fig. 4; darker the color the higher the number of combinations found) highlighted that there are some combinations of city and dong that appear more frequently in the collected dataset. The most frequently appearing combination had a city code of 93 and a dong code of 21. There were also a few combinations that did not occur at all (see the blank parts in Fig. 4). This trend indicates that the city-dong combination distribution might also be highly skewed in favor of a small number of popularly occupied locations. As a result, we can see that the real RRN space



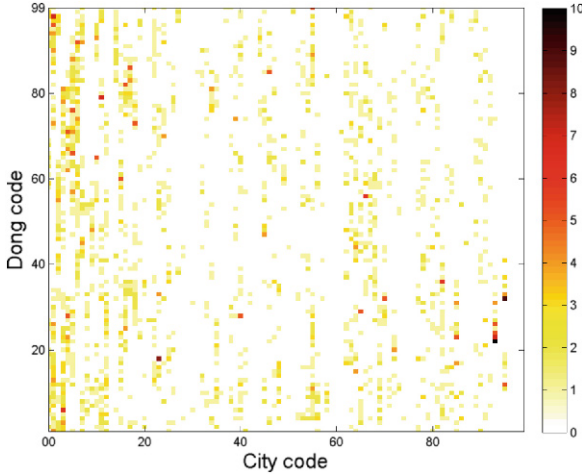
**Fig. 3.** Occurrence frequency of birth data in RRNs.

is much smaller than that of the theoretical space, which would make dictionary attacks more effective.

### 3.3 Effectiveness of Guessing Attacks

Given an individual's name, sex, and birth date, which can be attained through social networks like Facebook, Twitter, or LinkedIn, the goal of the guessing attack is to find his or her corresponding RRN. Stolen RRNs can be used to help criminal activities, e.g., allowing rogue accounts to be created on Korean websites and impersonating innocent people, or accessing Korean residents' personal records or confidential documents maintained by the Korean government. We consider the following, specific adversary.

The adversary already has access to a victim's publicly attainable information (i.e., name, sex, and birth date) and is capable of accessing an RRN validation service, which allows one to submit a name and RRN pair through an online form and validate it. Given that accessibility, the adversary tries to guess the victim's RRN by enumerating through every possible combination of RRN until a valid one is found. In theory, the adversary needs to try  $10^5$  possible RRNs.

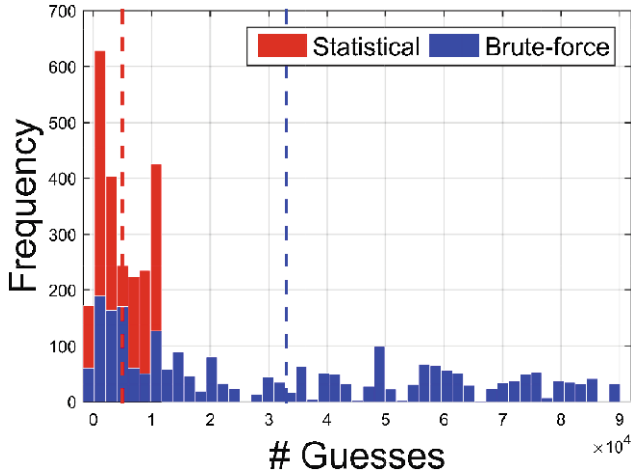


**Fig. 4.** Relationship between city code and dong code. (Color figure online)

Based on such statistical characteristics of the collected set of RRNs, we designed a “statistical-guessing attack” that is highly effective on RRNs. With the two strongly skewed city-dong and birth registration distributions, our RRN guesses were sorted in a descending order by the frequency of city-dong combinations and also by the frequency of birth registration order.

To evaluate the effectiveness of the proposed statistical-guessing attack, the ten-fold cross validation was performed ten times. In each run, one of the folds was used for validation, while the remaining folds were used for obtaining statistical properties of RRNs. We compared the performance of the statistical-guessing attack against a basic brute-force attack that sequentially enumerates every possible RRN combination—this is the lower bound strategy of a guessing attack since it can be performed without any knowledge of RRNs. Those attack results are plotted in Fig. 5.

The graph shows that the number of guessing attempts made in the statistical-guessing attack is significantly lower than the number of guessing attempts made in the brute-force attack ( $p < 0.001$ , two-tailed unpaired t-test). The brute-force attack has a mean number of 32,974.9 guessing attempts with a standard deviation of 27,905.9, while the statistical-guessing attack has a mean number of 4,892.5 guessing attempts with a standard deviation of 3,913.4. The statistical-guessing attack is about 7 times faster. Moreover, unlike the brute-force attack which often failed even with 50,000 guesses, all of the statistical-guessing attack attempts completed successfully (that is a valid RRN was found) within about 10,000 guesses.



**Fig. 5.** Histograms of the numbers for successfully guessing a RRN for statistical-guessing (red) and brute-force (blue) attacks. The x-axis and the y-axis represent the number of guesses to successfully infer a RRN and the number of frequency in each bin, respectively. The dotted lines represent the mean of guess numbers. (Color figure online)

## 4 CounterMeasures

We suggest two defense mechanisms to mitigate the statistical-guessing attack discussed in the previous section.

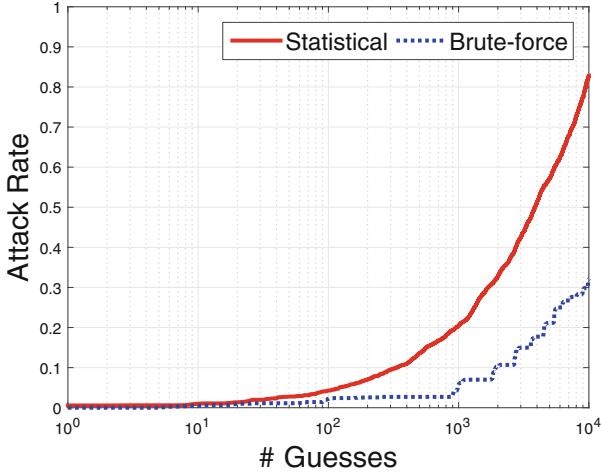
### 4.1 Limiting the Number of RRN Verification Fail Attempts

To mitigate this type of attack, a straightforward solution is to limit the number of RRN verification attempts. The idea of limiting the number of attempts from a particular user (e.g., with an IP address) or imposing a minimum time interval between failed attempts is not new [2]. However, the vast majority of the real-world RRN verification systems that we investigated did not limit the number of attempts, and did not seem to be considering the threats associated with brute-forcing RRNs.

To find an optimal number of RRN verification fail attempts that should be allowed (i.e., a number that would effectively prevent online guessing attacks), we calculated a range of successful rates of guessing attacks by varying the “maximum fail attempts allowed” from 1 to 10,000. Figure 6 shows the results. 7 seems to be a reasonable number to be used as the maximum fail attempts allowed since it can successfully stop 99.94 % of statistical-guessing attacks.

### 4.2 Anomaly Detection

Another promising approach to explore is *anomaly detection*. Guessing attacks that try to brute-force RRNs must inherently generate and send a large volume of



**Fig. 6.** The successful attack rates of statistical-guessing (red) and brute-force (blue) attacks with varying maximum allowed number from 1 to 10000. The x-axis and the y-axis represent the maximum allowed number for guessing attacks and the successful attack rate, respectively. (Color figure online)

query (RRN-based real name verification) messages in a very short time period. Such a spike of query messages will result in unusual traffic patterns and can be treated and flagged as anomalies. For instance, when a service provider receives a series of RRN-based real name verification messages from one IP address, we can classify such messages as a potential guessing attack because that traffic will look significantly different from normal user traffic. Our suggestion is to design and deploy such an anomaly detection system on the RRN verification server. Implementation techniques for anomaly detection are already widely available (e.g., see [3]).

## 5 Ethical Considerations

Our analyses were conducted based on publicly available RRN dataset that we attained through Chinese websites such as Baidu. We clarify that it was not our intention to collect or misuse personal information in any way, or use the collected data for commercial purposes. As can be seen from the previous sections, our goal is to identify security risks associated with current RRN-based identification systems, and recommend practical mitigation solutions to make it difficult for adversaries to perform effective guessing attacks.

## 6 Related Work

Our primary focus is on analyzing security risks associated with Korean RRN-based name verification systems by performing both simple and more advanced



statistical guessing attacks on real-world RRNs. Understanding security issues of RRN-based verification is important because it is still widely used online. In fact, the Korean government passed the *Real Name Verification Law* in July 2007 so that only RRN-verified users (with their real identity) may post comments on websites. Cho [4] discussed both the positive and negative effects of passing the *Real Name Verification Law*. For instance, Cho mentions that identifying posts had a significant positive impact on reducing uninhibited behaviors.

In response to the growing concern about RRN database leaks and severe misuse of the stolen RRNs [10], Pak et al. [11] investigated alternative methods for reliably verifying users' identity and age. In particular, they proposed a new national identification service called "i-PIN," where the Korean government issues and manages an i-PIN identifier and a matching password for Korean residents. The Japanese government also introduced a similar identification service called "My Number" [9].

National identification programs as such, however, have often been exposed to security breaches. The Korean i-PIN system was hacked in 2015, with 750,000 Korean citizens' i-PIN accounts being disclosed [6].

Oh et al. [10] showed that partial RRN information (the first 6 digits representing the date of birth) can be used to design sophisticated phishing attacks. Our work demonstrates how successful statistical-guessing attacks (that use some publicly available personal information) can be on guessing RRNs from real-world name verification services.

Sweeney et al. [12] studied the security of encrypted RRNs, showing that 23,163 encrypted RRNs can be successfully revealed using two de-anonymization methods.

Acquisti et al. [1] analyzed the correlation between the U.S. social security numbers and the individual's birth data, demonstrating the feasibility of statistically inferring social security numbers. We conducted a similar study to show the security risks of another national identification number system that is being used in Korea, showing that current Korean RRNs can be guessed with a high chance based on the personal information collected through public data sources.

## 7 Conclusions

In this paper, we explore an efficient guessing attack that can be performed using publicly available personal information (e.g., obtained through social networks) to reveal national individual identifiers, such as the Resident Registration Numbers (RRNs) used in South Korea. Upon joining websites, Koreans are often required to enter their RRN to prove their identity. Inadequate security protections deployed on such websites expose holes that can be exploited by adversaries to steal RRNs and impersonate innocent users.

We designed a moderately advanced statistical-guessing attack that uses some publicly available personal information such as name, sex, and birth date, to efficiently and accurately guess RRNs. We evaluated the effectiveness of the proposed statistical-guessing attack with a real-world RRN dataset that consists of 2,326 valid name and RRN pairs. Our experimental results showed that

the statistical-guessing attack can guess RRNs with a much smaller number of guessing attempts compared to pure brute-force attacks (that try all the possible combinations without any smart rule). Intriguingly, with the proposed attack, only about 4,892.5 trials were needed on average to successfully guess an RRN.

We suggested two possible mitigation techniques that would work well against statistical-guessing attack. One technique is to limit the number of consecutive RRN verification fail attempts allowed, significantly slowing down online attacks. Our feasibility analysis demonstrated that a carefully set fail attempts allowed number, e.g., 7, would prevent about 99.94 % of statistical-guessing attacks. Most legitimate users should be able to enter their correct RRN within 7 attempts. We strongly recommend that a policy for limiting the number of consecutive fail attempts allowed be enforced on RRN-based name verification systems.

**Acknowledgements.** This work was supported in part by the NRF Korea (No. 2014R1A1A1003707), the ITRC (IITP-2015-H8501-15-1008), and the MSIP/IITP (2014-PK10-28). Authors would like to thank all the anonymous reviewers for their valuable feedback.

## References

1. Acquisti, A., Gross, R.: Predicting social security numbers from public data. *Proc. Natl. Acad. Sci.* **106**(27), 10975–10980 (2009)
2. Alsaleh, M., Mannan, M., Van Oorschot, P.: Revisiting defenses against large-scale online password guessing attacks. *IEEE Trans. Dependable Secure Comput.* **9**(1), 128–141 (2012)
3. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv.* **41**(3), 15 (2009)
4. Cho, D.: Real name verification law on the internet: a poison or cure for privacy? In: *Proceedings of the 10th Workshop on Economics of Information Security* (2011)
5. Gross, R., Acquisti, A.: Information revelation and privacy in online social networks. In: *Proceedings of the ACM Workshop on Privacy in the Electronic Society* (2005)
6. Kovacs, E.: Personal Details of 27 Million South Koreans Stolen by Hacker (2014)
7. Lee, R.: Korean national ID numbers spring up all over Chinese Web (2011)
8. Lee, T.B.: South Korea’s “real names” debacle and the virtues of online anonymity (2011)
9. Miyata, S., Suzuki, K., Morizumi, T., Kinoshita, H.: Access control model for the my number national identification program in Japan. In: *Computer Software and Applications Conference Workshops* (2014)
10. Oh, Y., Obi, T., Lee, J.S., Suzuki, H., Ohyama, N.: Empirical analysis of internet identity misuse: case study of South Korean real name system. In: *Proceedings of the 6th ACM Workshop on Digital Identity Management* (2010)
11. Pak, H., Kim, C., Choi, H.: Preparation a study on the use of the Resident Registration Number and Alternatives for RRN. *World Acad. Sci. Eng. Technol.* **6**(11), 3123–3126 (2012)
12. Sweeney, L., Yoo, J.S.: De-anonymizing South Korean Resident Registration Numbers Shared in Prescription Data. *Technology Science* (2015)
13. Yang, S.: 35m Cyworld, Nate users’ information hacked (2011)

# **Social Network Security**

# Towards Privacy-Preserving Data Mining in Online Social Networks: Distance-Grained and Item-Grained Differential Privacy

Shen Yan<sup>1,2,3</sup>, Shiran Pan<sup>1,2,3</sup>, Yuhang Zhao<sup>1,2</sup>, and Wen-Tao Zhu<sup>1,2</sup>(✉)

<sup>1</sup> State Key Laboratory of Information Security,  
Institute of Information Engineering,

Chinese Academy of Sciences, Beijing 100093, China

{yanshen,panshiran,zhaoyuhang}@iie.ac.cn, wtzhu@ieee.org

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

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

**Abstract.** Online social networks have become increasingly popular, where users are more and more lured to reveal their private information. This brings about convenient personalized services but also incurs privacy concerns. To balance utility and privacy, many privacy-preserving mechanisms such as differential privacy have been proposed. However, most existent solutions set a single privacy protection level across the network, which does not well meet users' personalized requirements. In this paper, we propose a fine-grained differential privacy mechanism for data mining in online social networks. Compared with traditional methods, our scheme provides query responses with respect to different privacy protection levels depending on where the query is from (i.e., is distance-grained), and also supports different protection levels for different data items (i.e., is item-grained). In addition, we take into consideration the collusion attack on differential privacy, and give a countermeasure in privacy-preserving data mining. We evaluate our scheme analytically, and conduct experiments on synthetic and real-world data to demonstrate its utility and privacy protection.

**Keywords:** Differential privacy · Online social networks · Privacy-preserving data mining · Collusion attack

## 1 Introduction

Online social networks (OSNs) are increasingly involved in our daily life. Driven by various OSN applications, large number of profiles are uploaded. Meanwhile, the tendency towards uploading personal data to OSNs has raised privacy concerns. Debates on privacy leakage in OSNs [1–3] have continued for years. Although users have certain degree of privacy concerns, actually such concerns would not be converted into any actions [1]. The contradiction between privacy and utility calls for better privacy mechanisms in OSNs.

Two main aspects of OSNs' privacy researches are privacy-preserving data mining (PPDM) [4] and privacy-preserving data publishing (PPDP) [5], which respectively achieves data mining and data sharing goals without privacy leakage.  $k$ -anonymity [6] is one of the most widely used anonymity models, which divides all records into several equivalence groups and hides individual records in groups. Later models like  $l$ -diversity [7],  $t$ -closeness [8], and  $(a, k)$ -anonymity [9] are proposed to enhance the privacy guarantee in various environments. Nevertheless, it has been shown that former anonymity models are susceptible to several background knowledge attacks [10]. The notion of differential privacy [11] is proposed in the context of statistical databases, which provides a mechanism resistant to background knowledge attacks. Differential privacy provides a quantifiable measurement of privacy via a certain parameter  $\epsilon$ . A query is said to satisfy  $\epsilon$ -differential privacy, if when an individual item in a data set is added or removed, the probability distribution of the query answers varies by a factor of  $\exp(\epsilon)$ . Thus curious users cannot deduce the individual item from the query answer.

Most previous work on differential privacy assigns the same privacy protection levels (i.e.,  $\epsilon$ ) to all users in the network. However, in practice, heterogeneous privacy requirements are needed [12–15]. Users tend to allow people who are more closed to them (e.g., families, close friends) to get more accurate information, whereas those who are just acquaintances or strangers obtain obscure data. In addition, items in each user's data set require various privacy protection levels. For example, individual users may regard their telephone numbers as very private information that should not be leaked to strangers, whereas some business accounts need to publish their telephone numbers so as to get in touch with potential consumers.

Since existing privacy techniques cannot satisfy the increasing and heterogeneous privacy demands, finer-grained privacy-preserving mechanisms are required in OSNs. In this paper, we propose a fine-grained differential privacy mechanism, and the merits of our scheme can be summarized as follows:

- First, each query satisfies  $\epsilon$ -differential privacy, where  $\epsilon$  varies with the distance between users (i.e., distance-grained).
- Second, each user can assign different privacy protection levels for different items in their respective data set (i.e., item-grained).
- Third, our scheme is resistant to the collusion attack in OSNs.

Our work adopts some techniques in [16,17], and combines the merits of them. In addition, our work extends the application scenario in [16] to PPDM, and proposes a new method to resist the collusion attack.

This paper is organized as follows. We present the preliminaries in Sect. 2, and provide the problem formulation in Sect. 3. Section 4 gives an implementation approach to our scheme. The theoretical analysis is proposed in Sect. 5. Section 6 presents experimental results, followed by conclusions in Sect. 7.

## 2 Preliminaries

### 2.1 Differential Privacy

Differential privacy is originally introduced by Dwork [11]. The privacy guarantee provided by differential privacy is implemented via adding noise to the output of the computation, thus curious users cannot infer whether or not the individual data is involved in the computation.

**Definition 1 ( $\epsilon$ -Differential Privacy).** *A random function  $M$  satisfies  $\epsilon$ -differential privacy if for all neighboring data sets  $D$  and  $D'$ , and for all outputs  $t \in \mathbb{R}$  of this randomized function, the following statement holds:*

$$\Pr[M(D) = t] \leq \exp(\epsilon) \Pr[M(D') = t],$$

in which  $\exp$  refers to the exponential function. Two data sets  $D$  and  $D'$  are said to be neighbors if they are different in at most one item.  $\epsilon$  is the privacy protection parameter which controls the amount of distinction induced by two neighboring data sets. A smaller value of  $\epsilon$  ensures a stronger privacy guarantee.

We can achieve  $\epsilon$ -differential privacy by adding random noise whose magnitude is decided by the possible change in the computation output over any two neighboring data sets. We denote this quantity as the global sensitivity.

**Definition 2 (Global Sensitivity).** *The global sensitivity  $S(f)$  of a function  $f$  is the maximum absolute difference obtained on the output over all neighboring data sets:*

$$S(f) = \max_{D \sim D'} |f(D) - f(D')|.$$

To satisfy the differential privacy definition, two main mechanisms are usually utilized: the *Laplace* mechanism and the *Exponential* mechanism. The *Laplace* mechanism is proposed in [18], which achieves  $\epsilon$ -differential privacy by adding noise following *Laplace* distribution. Between the two mechanisms, the *Laplace* mechanism is more suitable for numeric outputs.

**Definition 3 (Laplace Mechanism).** *Given a function  $f: D \rightarrow \mathbb{R}$ , the mechanism  $M$  adds Laplace distributed noise to the output of  $f$ :*

$$M(D) = f(D) + V, \text{ where } V \sim \text{Lap}\left(\frac{S(f)}{\epsilon}\right),$$

where  $\text{Lap}\left(\frac{S(f)}{\epsilon}\right)$  has probability density function  $h_\epsilon(x) = \frac{\epsilon}{2S(f)} \exp\left(\frac{-\epsilon|x|}{S(f)}\right)$ .

### 2.2 Related Work

The notion of differential privacy was introduced by Dwork [11]. After that, differential privacy has been a popular privacy-preserving mechanism in PPDM and PPDP. However, most previous work focuses on theoretical aspects. In this paper, we propose a fine-grained differential privacy mechanism in OSNs scenario.

The majority of previous work on fine-grained differential privacy provides user-grained privacy protection levels [14], where each user can set a personalized protection parameter (i.e.,  $\epsilon$ ). Yuan et al. [19] considered personalized protection for graph data in terms of both semantic and structural information. However, considering the heterogeneous privacy need of different items in users' data sets, finer-grained mechanisms are required. Alaggar et al. introduced a mechanism [17] named *heterogeneous differential privacy* (HDP), HDP allows users set different privacy protection levels for different items in each data set, and Jorgensen et al. [15] extends HDP to *Exponential* mechanism of differential privacy. In our scheme, we adopt some techniques in [17] to realise the item-grained differential privacy.

Different from user-grained differential privacy, distance-grained differential privacy mechanism decides parameter  $\epsilon$  by where the query is from. To the best of our knowledge, Koufogiannis et al. proposed the first distance-grained mechanism in [20], and later introduced a modified distance-grained differential privacy mechanism in PPDP [16].

Furthermore, few works take into consideration the collusion attack on differential privacy. Although the collusion attack does not leak private information directly, it expands the parameter  $\epsilon$ , thus losing the privacy guarantee of the system. Zhang et al. [21] designed a collusion resistant algorithm in PPDM which combines multiparty computation and differential privacy. Koufogiannis et al. [16, 20] proposed mechanisms to address this problem in PPDP.

Inspired by previous work, we design a fine-grained differential privacy mechanism in OSNs. Although some of the tools in [16, 17] are leveraged to provide a solution, we consider a different problem which applies to PPDM rather than PPDP. In PPDM scenario, the users related to a query can be two random users in the network, which makes resisting the collusion attack more difficult.

## 3 Problem Formulation

### 3.1 System Model

Consider an OSN represented as a graph  $G = (N, E)$ , where  $N$  is the set of nodes and  $E$  is the set of edges connecting pair of nodes in  $N$ . Each node in  $N$  represents a user in the network and each edge represents the friendship relationship between users. Each user owns a data set  $D$ , which is denoted by an  $n$  dimensional vector  $D = [b_1, b_2, \dots, b_n]$ .<sup>1</sup>

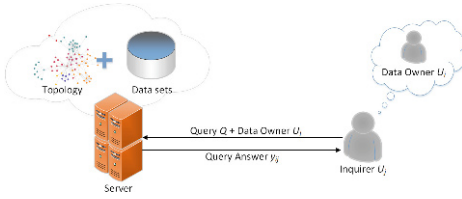
There exist three parties in our model: server, data owner, and inquirer.

- Server usually is a service provider, such as Twitter, Facebook. The server stores the topology of  $G$  and users' data sets. In our scheme, we assume that the server is a trusted third party, which means that it will not leak any information to others, and will always return actual distance between users.
- Data owner is a user in the network, owning a data set  $D$  that to be queried.
- Inquirer is a user in the network, who submits a query  $Q$  to the server.

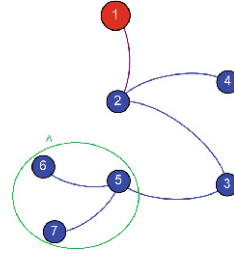
Moreover, each user in the network can be both data owner and inquirer.

---

<sup>1</sup> Without of causing confusing, we interchangeably use node and user in this paper.



**Fig. 1.** System Illustration



**Fig. 2.** A network example with 7 nodes, nodes 5–7 constitute group  $A$ .

In practice, a data mining task can be a numerical computation of  $D$  and  $Q$ , or a searching task on data sets. We define  $f(D)$  as an arbitrary function of  $D$ , where  $f(D)$  is defined by specific data mining tasks, and  $M(D)$  is a modified  $f(D)$  that satisfies differential privacy by adding noise  $V$ . As Fig. 1 shows,  $U_i$  is the data owner,  $U_j$  is the inquirer. When the server receives  $U_j$ 's query  $Q$ , it computes  $M(D)$ , where  $D$  is the data set of  $U_i$ . After the computation, the server returns the query answer  $y_{ij}$  to  $U_j$  in proper methods (e.g., returns the IDs of the users whose computation results satisfy  $Q$ , or returns the computation results directly), where  $y_{ij} = M(D) = f(D) + V$ .

## 3.2 Fine-Grained Differential Privacy

We propose a fine-grained differential privacy mechanism which assigns different protection levels for different inquirers (i.e., distance-grained differential privacy) and simultaneously satisfies different privacy levels for each dimension of  $D$  (i.e., item-grained differential privacy).

### A. Distance-grained Differential Privacy

Practically, under a query, users are willing to give more accurate data to closed people, whereas return obscure result to people with distant relationship. In our scheme, we use shortest path length to measure the relationship between users.

More specifically, inquirer  $U_j$  gets a result that ensures  $\epsilon(d_{ij})$ -differential privacy, where  $d_{ij}$  denotes the shortest path length between the data owner  $U_i$  and inquirer  $U_j$ . If  $d_{ij}$  is smaller,  $\epsilon(d_{ij})$  should be larger, i.e.,  $\epsilon(d_{ij})$  is inversely proportional to  $d_{ij}$ .

### B. Item-grained Differential Privacy

Usually, items in  $D$  have heterogeneous privacy expectations. For instance,  $D$  contains hobbies, locations, jobs, and other personal information of a specific user. The user may consider locations are sensitive information that should not be known by other users, whereas be willing to share his hobbies with others. Thus the user can set higher privacy levels (i.e., smaller  $\epsilon$ ) for the



items of locations, In our scheme, every user can set a privacy weight vector  $W = [w_1, w_2, \dots, w_n]$ , where each element  $w_k$  represents the privacy weight associated to the  $k$ th item  $b_k$  in  $D$  with  $w_k \in [0, 1]$ ; smaller  $w_k$  ensures stronger protection, where 0 corresponds to absolute privacy while 1 refers to standard  $\epsilon(d_{ij})$ -differential privacy.

Consequently, whenever an inquirer  $U_j$  submits a query for data owner  $U_i$  to the server, he will get a perturbed answer satisfies  $w_k \epsilon(d_{ij})$ -differential privacy for each dimension.

### 3.3 Collusion Attack

In practice, users in the network can share information with each other. As Fig. 2 shows, assuming that there is a group  $A$ , each user in  $A$  has a query answer  $y_{ij}$  about data owner  $U_i$ 's data set  $D$ . When users in group  $A$  share their query answers, they may derive a more accurate estimate of  $D$ , even though each user owns a highly noisy answer of the query.

**Theorem 1.** *For a group of users  $A$ , where each user has a query answer  $y_{ij}$  that satisfies  $\epsilon(d_{ij})$ -differential privacy, they can derive an estimator  $y_A$  that ensures  $(\sum_{U_j \in A} \epsilon(d_{ij}))$ -differential privacy under collusion.*

*Proof:.* To simplify the proving process, we assume that group  $A$  has two users:  $U_{j_1}$  and  $U_{j_2}$ . We set  $y_{ij_1} = M_1(D) = f(D) + V_1$ ,  $y_{ij_2} = M_2(D) = f(D) + V_2$ .

$$\begin{aligned}
 & \frac{\Pr[M_1(D) = y_{ij_1}, M_2(D) = y_{ij_2}]}{\Pr[M_1(D') = y_{ij_1}, M_2(D') = y_{ij_2}]} \\
 &= \frac{\Pr[M_1(D) = y_{ij_1}] \Pr[M_2(D) = y_{ij_2}]}{\Pr[M_1(D') = y_{ij_1}] \Pr[M_2(D') = y_{ij_2}]} \\
 &= \frac{h_{\epsilon(d_{ij_1})}(y_{ij_1} - f(D))}{h_{\epsilon(d_{ij_1})}(y_{ij_1} - f(D'))} \frac{h_{\epsilon(d_{ij_2})}(y_{ij_2} - f(D))}{h_{\epsilon(d_{ij_2})}(y_{ij_2} - f(D'))} \\
 &= \frac{\exp\left(\frac{-\epsilon(d_{ij_1})|y_{ij_1} - f(D)|}{S(f)}\right)}{\exp\left(\frac{-\epsilon(d_{ij_1})|y_{ij_1} - f(D')|}{S(f)}\right)} \frac{\exp\left(\frac{-\epsilon(d_{ij_2})|y_{ij_2} - f(D)|}{S(f)}\right)}{\exp\left(\frac{-\epsilon(d_{ij_2})|y_{ij_2} - f(D')|}{S(f)}\right)} \\
 &\leq \exp\left(\frac{\epsilon(d_{ij_1})|f(D') - f(D)|}{S(f)}\right) \exp\left(\frac{\epsilon(d_{ij_2})|f(D') - f(D)|}{S(f)}\right) \\
 &\leq \exp(\epsilon(d_{ij_1})) \exp(\epsilon(d_{ij_2})) = \exp(\epsilon(d_{ij_1}) + \epsilon(d_{ij_2})).
 \end{aligned}$$

Therefore, when the collusion attack happened, the privacy protection parameter will be the sum of colluded users'  $\epsilon(d_{ij})$ .

Although collusion attacks will not leak data owners' information directly, it expands the parameter  $\epsilon$ , thus losing the privacy guarantee of the system.

According to the proof of Theorem 1, we can infer that the collusion attack works when the noise additions are independent of each other. Therefore, we should introduce correlations between the noise additions added to different query answers. In order to meet practical requirements, any group of users cannot

derive a better estimator than the most accurate answer among the group, i.e., the answer of the inquirer who is closest to the data owner in the group.

**Theorem 2.** *For a group of users  $A$ , where each user has a query answer  $y_{ij}$  that satisfies  $\epsilon(d_{ij})$ -differential privacy, and the noise additions of  $y_{ij}$  are correlated to each other, the best estimator  $y_A$  ensures  $(\max_{U_j \in A} \epsilon(d_{ij}))$ -differential privacy.*

*Proof:* To simplify the proving process, we assume that group  $A$  has two users:  $U_{j_1}$  and  $U_{j_2}$ .  $y_{ij_1} = M_1(D) = f(D) + V_1$ ,  $y_{ij_2} = M_2(D) = f(D) + V_2$ , where  $\phi(V_2 - V_1)$  is the conditional probability density function of  $\Pr[V_1|V_2]$ .

$$\begin{aligned}
& \frac{\Pr[M_1(D) = y_{ij_1}, M_2(D) = y_{ij_2}]}{\Pr[M_1(D') = y_{ij_1}, M_2(D') = y_{ij_2}]} \\
&= \frac{\Pr[M_2(D) = y_{ij_2}] \Pr[M_1(D) = y_{ij_1} | M_2(D) = y_{ij_2}]}{\Pr[M_2(D') = y_{ij_2}] \Pr[M_1(D') = y_{ij_1} | M_2(D') = y_{ij_2}]} \\
&= \frac{h_{\epsilon(d_{ij_2})}(y_{ij_2} - f(D)) \phi(y_{ij_1} - f(D) - (y_{ij_2} - f(D)))}{h_{\epsilon(d_{ij_2})}(y_{ij_2} - f(D')) \phi(y_{ij_1} - f(D') - (y_{ij_2} - f(D')))} \\
&= \frac{\exp\left(\frac{-\epsilon(d_{ij_2})|y_{ij_2} - f(D)|}{S(f)}\right) \phi(y_{ij_1} - y_{ij_2})}{\exp\left(\frac{-\epsilon(d_{ij_2})|y_{ij_2} - f(D')|}{S(f)}\right) \phi(y_{ij_1} - y_{ij_2})} \\
&\leq \exp(\epsilon(d_{ij_2})).
\end{aligned}$$

## 4 Scheme Implementation

In this paper, we take user matching as the example of data mining in OSNs, and give a particular implementation approach to our scheme.

User matching is a common application in OSNs, where the server computes the similarity between users' data sets. Therefore,  $D$  is data owner's profile;  $Q$  is another  $n$  dimension vector  $[q_1, q_2, \dots, q_n]$ , which refers to the user attributes that the inquirer searching for, it can be inconsistent with the inquirer's profile;  $f(D)$  is computing the similarity between  $Q$  and  $D$ .

There are many methods to evaluate the similarity between data (e.g., Euclidean distance, Cosine similarity). We choose Euclidean distance  $f_Q(D) = \sqrt{\sum_{i=0}^n (b_i - q_i)^2}$  as the evaluation criterion (i.e.,  $f(D)$ ), where for binary vectors,  $S(f) = \max_{D \sim D'} |f_Q(D) - f_Q(D')| = 1$ .

We define the similarity between user  $U_i$  and  $U_j$  as  $\frac{1}{f_Q(D)}$ . According to former analysis, the value of  $\epsilon$  of each user should be inversely proportional to  $d_{ij}$ . We set  $\epsilon(d_{ij}) = \frac{1}{d_{ij}+1}$ .

The working flow of our scheme consists of three phases: creating correlations, query process, and updating perturbations.

## 4.1 Creating Correlations Phase

Creating correlations is the most crucial and innovative phase in our scheme. The correlation should ensure that each noise addition follows *Laplace* distribution, which makes the perturbations satisfy differential privacy.

**Theorem 3.** Consider two noise additions  $V_1 \sim \text{Lap}(\frac{1}{\epsilon_1})$ ,  $V_2 \sim \text{Lap}(\frac{1}{\epsilon_2})$ , where  $\epsilon_1 < \epsilon_2$ .  $V_1$  and  $V_2$  are correlated with a conditional probability  $\Phi$ , where  $\Phi$  has the density:

$$\phi(x) = (1 - \frac{\epsilon_1^2}{\epsilon_2^2}) \frac{\epsilon_1}{\sqrt{2\pi}} \sqrt{\epsilon_1 x} K_{-\frac{1}{2}}(\epsilon_1 x), \quad (1)$$

where  $K$  is the modified Bessel function of the second kind.

*Proof:* See the Appendix.

We denote (1) as *bessel* distribution, and its input parameters are  $\epsilon_1$  and  $\epsilon_2$ .

Although Koufogiannis et al. [16] introduce similar method to address this problem by building correlations, we design a novel correlation creation mechanism, which is more suitable for PPDM.

We can see edges of the network as the bridges to build up the correlations between nodes. In our scheme, each edge is assigned a perturbation vector, in which each element represents the sampling noise for different distances. In addition, each node is assigned an initial perturbation, which will not change once the node joined in the network.

Algorithm 1 illustrates the method to assign perturbations over the network. In the algorithm, the largest dimension of each edge’s perturbation vector is decided by the diameter of the network. Real OSNs’ diameters are usually small numbers: according to the networks’ data in SNAP<sup>2</sup> database, the Facebook network with 4039 nodes has a diameter of 8, the Gowalla network with 196591 nodes only has a diameter of 14. This property of OSNs guarantees that we only need to save a vector of small size for each edge. In our scheme, *perturbation* is a sample of the *bessel* distribution density, which denotes the amount of noise added to the query answer when passing by the edge.

## 4.2 Query Process Phase

In query process phase, the inquirer submits a query to the server, and tells the server the data owner of this query. The server uses the scalar product of  $W$  and  $D$  to compute  $f_Q(W \cdot D)$ , and adds noise to the computation result, where the noise is the sum of the perturbations on each edge in the shortest path from the inquirer to the data owner (if there exists more than one shortest path between two nodes, the algorithm randomly picks one of the paths). We illustrate the noise computation process in Algorithm 2. The query answer is denoted as  $y_{ij} = M(D) = f_Q(W \cdot D) + V$ .

<sup>2</sup> <https://snap.stanford.edu/data/index.html>.

**Algorithm 1.** Perturbation Assignment

---

```

for node in G.nodes() do
   $R_{node} \sim \text{Laplace}(1)$  // Assign initial perturbations
end for
l = G.diameter()
 $\epsilon[l] = [1, \frac{1}{2}, \dots, \frac{1}{l+1}]$ 
for (j1, j2) in G.edges() do
  i = 0
  while i < l do
     $\text{perturbation}_{j_1, j_2}[i] = \text{bessel}(\epsilon[i], \epsilon[i + 1])$ 
    i = i + 1
  end while
end for

```

---

**Algorithm 2.** Noise Computation

---

```

Input: Data Owner Ui, Inquirer Uj
 $\text{path}[] = \text{shortest\_path}(i, j)$  // path consists of nodes in the shortest path sequentially
 $N = 0$ 
for (t=0; t <  $\text{len}(\text{path}) - 1$ ; t++) do
   $\text{dis} = \text{shortest\_path\_length}(i, \text{path}[t + 1])$ 
   $N = N + \text{perturbation}_{\text{path}[t], \text{path}[t+1]}[\text{dis}]$ 
end for
 $\text{noise} = N + R_i$ 

```

---

### 4.3 Updating Perturbations Phase

In addition to the collusion attack, repeat queries may also cause privacy leakage. If a certain inquirer receives two query answers with the same noise, he can deduce the noise and know the real data sequentially. So perturbation vectors should be updated after each query.

In Algorithm 3, we randomly update half of the perturbations used in the former query, which simultaneously ensures the correlations between noise additions and saves the computation cost.

## 5 Theoretical Analysis

Our scheme provides privacy protection for both the data sets and their corresponding privacy weight vectors. The privacy protections guaranteed by our scheme are:

- Each query answer provides  $\epsilon(d_{ij})$ -differential privacy for data owner's data set  $D$ . More specifically, each dimension satisfies  $w_k \epsilon(d_{ij})$ -differential privacy.
- Each query answer provides  $\epsilon(d_{ij})$ -differential privacy for data owner's privacy weight vector  $W$ .
- Noise additions for different inquirers are correlated.
- Noise additions before and after update are correlated.

**Algorithm 3.** Perturbation Update

---

**Input:** *path* of the former query  
 $m = \lceil \text{len}(\text{path})/2 \rceil$  //Compute the number of updated edges.  
 $K = []$   
**for** ( $t=0$ ;  $t < m$ ;  $t++$ ) **do**  
 $K[t] = \text{random}(0, \text{len}(\text{path}) - 1)$   
**end for**  
**for** ( $t=0$ ;  $t < \text{len}(\text{path})$ ;  $t++$ ) **do**  
 $j_1 = \text{path}[t]$ ,  $j_2 = \text{path}[t + 1]$   
**if**  $t$  in  $K$  **then**  
 $\text{perturbation}_{j_1, j_2}[t + 1] = \text{bessel}(\epsilon[t], \epsilon[t + 1])$   
**end if**  
**end for**

---

**5.1 Privacy Analysis**

**Theorem 4.** *Our scheme ensures that each query answer satisfies  $\epsilon(d_{ij})$ -differential privacy.*

*Proof:* See data owner's data set as a whole, the query answer satisfies:

$$\begin{aligned} \frac{\Pr[M(D) = t]}{\Pr[M(D') = t]} &= \frac{h_{\epsilon(d_{ij})}(t - f(D))}{h_{\epsilon(d_{ij})}(t - f(D'))} = \frac{\frac{\epsilon(d_{ij})}{2S(f)} \exp(\frac{-\epsilon(d_{ij})|t - f(D)|}{S(f)})}{\frac{\epsilon(d_{ij})}{2S(f)} \exp(\frac{-\epsilon(d_{ij})|t - f(D')|}{S(f)})} \\ &= \exp(\frac{\epsilon(d_{ij})(|t - f(D)| - |t - f(D')|)}{S(f)}) \\ &\leq \exp(\frac{\epsilon(d_{ij})|f(D') - f(D)|}{S(f)}) \leq \exp(\epsilon(d_{ij})), \end{aligned}$$

where  $h_{\epsilon}(\cdot)$  is defined in Theorem 1, thus proving the result.

**Theorem 5.** *The random function  $M$  provide  $w_k \epsilon(d_{ij})$ -differential privacy for each dimension of  $D$ .*

*Proof:* First, we use  $S_k(f)$  to denote the local sensitivity of function  $f$ :

$$S_k(f) = \max_{D \sim D^{(k)}} |f(D) - f(D^{(k)})|,$$

where  $D$  and  $D^{(k)}$  are neighboring data sets which are different in the  $k$ th element. In addition,  $S(f) = \max(S_k(f))$ .

In our scheme, the input of function  $f$  is  $W \cdot D$  rather than  $D$ , and Alaggar et al. proves that  $S_k(f) \leq w_k S(f)$  in [17].

So, we have:

$$\begin{aligned}
\frac{\Pr[M(W \cdot D) = t]}{\Pr[M(W \cdot D') = t]} &= \frac{h_{\epsilon(d_{ij})}(t - f(W \cdot D))}{h_{\epsilon(d_{ij})}(t - f(W \cdot D'))} = \frac{\frac{\epsilon(d_{ij})}{2S(f)} \exp\left(\frac{-\epsilon(d_{ij})|t - f(W \cdot D)|}{S(f)}\right)}{\frac{\epsilon(d_{ij})}{2S(f)} \exp\left(\frac{-\epsilon(d_{ij})|t - f(W \cdot D')|}{S(f)}\right)} \\
&\leq \exp\left(\frac{\epsilon(d_{ij})|f(W \cdot D) - f(W \cdot D^{(k)})|}{S(f)}\right) \\
&\leq \exp\left(\frac{\epsilon(d_{ij})S_k(f)}{S(f)}\right) \leq \exp\left(\frac{\epsilon(d_{ij})w_k S(f)}{S(f)}\right) \\
&= \exp(w_k \epsilon(d_{ij})),
\end{aligned}$$

thus concluding the proof.

**Theorem 6.** *The random function  $M$  provide  $\epsilon(d_{ij})$ -differential privacy for each individual privacy weight  $W$ .*

*Proof:*  $W$  and  $W'$  are privacy weight vectors, and they differ in the  $k$ th element in each data set. So  $W \cdot D$  and  $W' \cdot D$  are neighboring data sets.

$$\begin{aligned}
\frac{\Pr[M(W \cdot D) = t]}{\Pr[M(W' \cdot D) = t]} &= \frac{h_{\epsilon(d_{ij})}(t - f(W \cdot D))}{h_{\epsilon(d_{ij})}(t - f(W' \cdot D))} = \frac{\frac{\epsilon(d_{ij})}{2S(f)} \exp\left(\frac{-\epsilon(d_{ij})|t - f(W \cdot D)|}{S(f)}\right)}{\frac{\epsilon(d_{ij})}{2S(f)} \exp\left(\frac{-\epsilon(d_{ij})|t - f(W' \cdot D)|}{S(f)}\right)} \\
&\leq \exp\left(\frac{\epsilon(d_{ij})|f(W \cdot D) - f(W' \cdot D)|}{S(f)}\right) \\
&\leq \exp\left(\frac{\epsilon(d_{ij})S(f)}{S(f)}\right) = \exp(\epsilon(d_{ij})),
\end{aligned}$$

thus concluding the proof.

## 5.2 Correlation Analysis

We use Monte-Carlo method to verify the correlation between noise. For universality, we simulate the process on the network example as shown in Fig. 2, where node 1 is the data owner, nodes 2–7 gets the query answer of the same query, respectively.

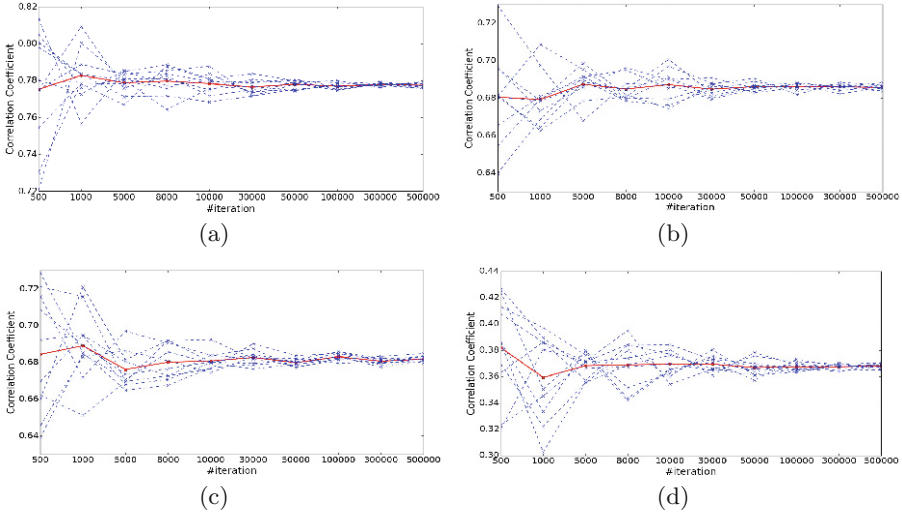
Specifically, we run the perturbation algorithm for over 100000 times, and use (2) to compute the Pearson Correlation Coefficient between results.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (2)$$

In Fig. 3, each blue dash line represents one set of results, the red solid line is the average of all computation results. As shown in Fig. 3(a), after repeated

iterations, the correlation coefficient between two different nodes converges to 0.78, which indicates that noise additions of different nodes are correlated.

Other than the collusion attack from a group of users, a single user can exploit the results he received to deduce more information. Therefore, to avoid this attack, different noise additions of the same query should be correlated, and they should be correlated to other nodes' noise additions as well.



**Fig. 3.** Correlation analysis by Monte-Carlo method: the x-dimension value is the iteration time in Monte-Carlo method, the y-dimension value is the correlation coefficient. (a) shows the correlation coefficient between two nodes' noise additions; (b)(c) show correlation coefficients after 50 and 100 updates, respectively; (d) shows the correlation of different nodes' noise after update. (Color figure online)

In our scheme, we update the assigned perturbations after each query. Figure 3(b) and (c) illustrate that the noise additions after 50 and 100 updates are still correlated with the original noise, and the correlation coefficient does not decrease with the increasing of the number of updates. We can observe that the correlation coefficient eventually converges to 0.68.

Meanwhile, Fig. 3(d) shows that the noise additions of different nodes after updates are still correlated.

## 6 Experiments

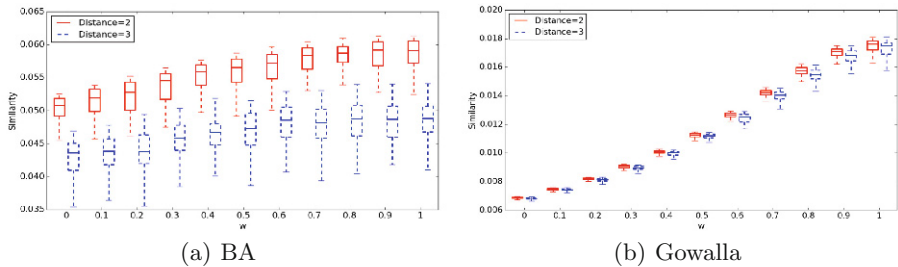
### 6.1 Data Sets

The experiments involve three networks' data: one synthetic network data and two real-world data. Real-world data come from the SNAP database.

- **BA Network:** BA model is a description of scale-free networks, which has the most similar topology to real OSNs. In our experiment, we use a BA network with 100 nodes and 291 edges. In addition, we build the data sets of each nodes by randomly choose 20 integer numbers from 0 to 10.
- **Facebook:** The Facebook network used in this experiment contains 213 nodes and 10305 edges. Each user keeps a data set as a 224 dimensional binary vector which includes user’s name, locations, jobs, etc.
- **Gowalla:** Gowalla is a location-based social networking website where users share their locations by checking in. The network we use in this experiment consists of 34378 nodes and 626658 edges. The data that each user keeps is a two dimensional vector of latitude and longitude.

## 6.2 Privacy Performance

To demonstrate the privacy performance of our scheme, we compare the query results of different users. For better illustration, we modify  $W$  yet the query  $Q$  remains the same. Figure 4(a) and (b) shows the similarity variation in BA network and Gowalla network, respectively. We vary two elements in  $W$  from 0 to 1 in each network, where the step size is 0.1.



**Fig. 4.** The similarity between certain  $Q$  and  $D$  rises with the increase of  $w$ , where  $w$  represents the element we changed in private weight vectors, and  $w \in [0, 1]$ . (Color figure online)

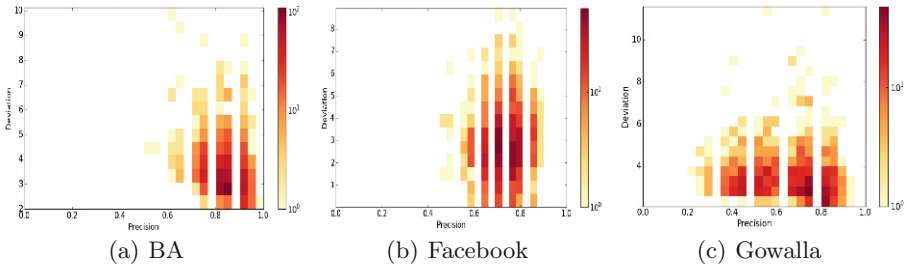
The experiments show that the similarity rises with the increase of the elements in  $W$ , which is consistent with our design goals that small  $w$  means stronger privacy protection. In addition, our results also illustrate that the similarities decrease as the distance increasing, which shows that our scheme realises the distance-grained differential privacy.

## 6.3 Utility Performance

In many practical user matching situations, the server often returns  $k$  IDs of users who have the top  $k$  similarities with  $Q$  instead of one user’s ID. In our experiments, the server returns 20 most similar users’ IDs to the inquirer.



We evaluate the utility of our scheme by comparing the query answers with accurate query results. We define the precision as the ratio of correct IDs (i.e., IDs exist in accurate query result) to the size of query answer (i.e., 20). Figure 5 shows the query precision in BA network, Facebook network, and Gowalla network, respectively. According to Sect. 6.2, privacy weight vectors have a great impact on the query answers. If a particular user wants to hide several elements in his data set (i.e., set quite small values to corresponding locations in the privacy weight vector), the query answers related to that user may significantly differ from the real answer. Therefore, to give a universal evaluation, we assume that all users set their privacy weight vectors as vectors of all ones, which satisfies the standard differential privacy protection.



**Fig. 5.** Precision of the query answers in (a)BA, (b)Facebook, and (C)Gowalla networks. Each point represents a query answer, where the x-dimension value is the precision, while y-dimension is the deviation between received query answers and real query answers. In the thermodynamic diagram, dark points represent the points shows up more frequently, yet the light color ones are those just occur occasionally.

In our scheme, query answers from distant data owners have larger noise, whereas close data owners add less noise to query answers. As a result, the IDs that inquirers received may contain more close data owners rather than real similar users, which may impact the precision of this experiment. However, from Fig. 5, we can observe that the majority of query answers have a precision value larger than 0.7, which proves that our scheme still provides good utility.

## 7 Conclusions

PPDM is an important issue in OSNs. However, most previous work assigns same privacy protection levels to all users in the network, which cannot satisfy users' personalized requirements. In this paper, we propose a scheme that provides fine-grained differential privacy for data mining in OSNs. Our scheme provides both distance-grained and item-grained differential privacy. Specifically, each inquirer in the network receives a query answer that satisfies  $w_k \epsilon(d_{ij})$ -differential privacy for each item in data owner's data set.

In addition, we investigate the collusion attack on differential privacy, and give a solution to this problem in PPDm. Finally, we conduct experiments on both synthetic and real-world data. Experiments show that our scheme guarantees good utility as well as good privacy protection.

**Acknowledgment.** The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China under Grant 61272479, the National 973 Program of China under Grant 2013CB338001, and the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702.

## Appendix

**Proof of Theorem 3:** Assume that  $V_1 \sim Lap(\frac{1}{\epsilon_1})$ ,  $V_2 \sim Lap(\frac{1}{\epsilon_2})$ , where  $\epsilon_1 < \epsilon_2$ . The conditional probability  $\Pr[V_1|V_2]$  has a density function  $\phi(x)$ . Additionally,  $V_1 = h_{\epsilon_1}(x) = \epsilon_1 \exp(-\epsilon_1|x|)$ ,  $V_2 = h_{\epsilon_2}(y) = \epsilon_2 \exp(-\epsilon_2|y|)$ . We use  $g(x, y)$  to denote the joint distribution density of  $V_1$  and  $V_2$ . So  $g(x, y)$  holds:

$$g(x, y) = \phi(y - x)h(x) \quad (3)$$

The density (3) should satisfy the following marginal distributions:

$$\int_{-\infty}^{\infty} g(x, y)dy = h_{\epsilon_1}(x) \quad \int_{-\infty}^{\infty} g(x, y)dx = h_{\epsilon_2}(y) \quad (4)$$

The Eq. (4) could be seen as a convolution operation  $\int_{-\infty}^{\infty} \phi(y-x)h_{\epsilon_1}(x)dx = h_{\epsilon_2}(y)$ . We use Convolution Theorem to solve this equation:

$$\mathcal{F}_\phi(s) = \frac{\mathcal{F}_{h_{\epsilon_2}}(s)}{\mathcal{F}_{h_{\epsilon_1}}(s)}, \quad (5)$$

where  $\mathcal{F}$  denotes Fourier Transform. According to (5), we get:

$$\mathcal{F}_\phi(s) = \frac{\mathcal{F}_{h_{\epsilon_2}}(s)}{\mathcal{F}_{h_{\epsilon_1}}(s)} = \frac{1 - \frac{s^2}{\epsilon_2^2}}{1 - \frac{s^2}{\epsilon_1^2}} = \left(\frac{\epsilon_1}{\epsilon_2}\right)^2 \left(1 + \frac{\epsilon_2^2 - \epsilon_1^2}{\epsilon_1^2 + s^2}\right)$$

We set  $b(x) = |x|^{1-\frac{n}{2}} K_{\frac{n}{2}-1}(|x|)$ , where  $K$  denotes the modified Bessel function of the second kind, and  $\mathcal{F}_b(s) = \frac{(2\pi)^{\frac{n}{2}}}{1+s^2}$ . So,

$$\mathcal{F}_\phi^{-1}(s) = \left(\frac{\epsilon_1}{\epsilon_2}\right)^2 [\delta(x) + \left(\frac{\epsilon_2^2}{\epsilon_1^2} - 1\right) \frac{\epsilon_1}{\sqrt{2\pi}} \sqrt{\epsilon_1 x} K_{-\frac{1}{2}}(\epsilon_1 x)]$$

$$\phi(x) \simeq \left(1 - \frac{\epsilon_1^2}{\epsilon_2^2}\right) \frac{\epsilon_1}{\sqrt{2\pi}} \sqrt{\epsilon_1 x} K_{-\frac{1}{2}}(\epsilon_1 x)$$

More relevant details about the proof are given by Koufogiannis et al. in [16].

## References

1. Dwyer, C., Hiltz, S., Passerini, K.: Trust and privacy concern within social networking sites: a comparison of Facebook and MySpace. In: 13th Americas Conference on Information Systems (AMCIS), pp. 339:1–339:13 (2007)
2. Zhang, C., Sun, J., Zhu, X., Fang, Y.: Privacy and security for online social networks: challenges and opportunities. *Network* **24**(4), 13–18 (2010). IEEE
3. Fogues, R., Such, J.M., Espinosa, A., Garcia-Fornes, A.: Open challenges in relationship-based privacy mechanisms for social network services. *Int. J. Hum.-comput. Interact.* **31**(5), 350–370 (2015)
4. Wu, X., Zhu, X., Wu, G.Q., Ding, W.: Data mining with big data. *IEEE Trans. Knowl. Data Eng.* **26**(1), 97–107 (2014). IEEE
5. Fung, B., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: a survey of recent developments. *ACM Comput. Surv. (CSUR)* **42**(4), 14 (2010). ACM
6. Sweeney, L.:  $k$ -anonymity: a model for protecting privacy. *Int. J. Uncertainty Fuzziness Knowl. Based Syst.* **10**(05), 557–570 (2002)
7. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkatasubramanian, M.:  $l$ -diversity: privacy beyond  $k$ -anonymity. *ACM Trans. Knowl. Discov. Data (TKDD)* **1**(1), 3:1–3:52 (2007)
8. Li, N., Li, T., Venkatasubramanian, S.:  $t$ -closeness: Privacy beyond  $k$ -anonymity and  $l$ -diversity. In: 23rd International Conference on Data Engineering (ICDE 2007), pp. 106–115. IEEE (2007)
9. Wong, R.C.W., Li, J., Fu, A.W.C., Wang, K.:  $(\alpha, k)$ -anonymity: an enhanced  $k$ -anonymity model for privacy preserving data publishing. In: 12th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2006), pp. 754–759. ACM (2006)
10. Dwork, C.: A firm foundation for private data analysis. *Commun. ACM* **54**(1), 86–95 (2011). ACM
11. 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)
12. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. *ACM SIGCOMM Comput. Commun. Rev.* **39**(4), 135–146 (2009). ACM
13. Li, Y., Chen, M., Li, Q., Zhang, W.: Enabling multilevel trust in privacy preserving data mining. *IEEE Trans. Knowl. Data Eng.* **24**(9), 1598–1612 (2012). IEEE
14. Ebadi, H., Sands, D., Schneider, G.: Differential privacy: now it’s getting personal. In: 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2015), pp. 69–81 (2015)
15. Jorgensen, Z., Yu, T., Cormode, G.: Conservative or liberal? Personalized differential privacy. In: 31st International Conference on Data Engineering (ICDE 2015), pp. 13–17. IEEE (2015)
16. Koufogiannis, F., Pappas, G.: Diffusing private data over networks (2015). arXiv preprint [arXiv:1511.06253](https://arxiv.org/abs/1511.06253)
17. Alaggan, M., Gamba, S., Kermarrec, A.M.: Heterogeneous differential privacy. arXiv preprint (2015). [arXiv:1504.06998](https://arxiv.org/abs/1504.06998)
18. 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)

19. Yuan, M., Chen, L., Yu, P.S.: Personalized privacy protection in social networks. *Proc. VLDB Endowment* **4**(2), 141–150 (2010). ACM
20. Koufogiannis, F., Han, S., Pappas, G.J.: Gradual release of sensitive data under differential privacy (2015). arXiv preprint [arXiv:1504.00429](https://arxiv.org/abs/1504.00429)
21. Zhang, N., Li, M., Lou, W.: Distributed data mining with differential privacy. In: *IEEE International Conference on Communications (ICC 2011)*. IEEE (2011)

# **Bitcoin Security**

# Fair Client Puzzles from the Bitcoin Blockchain

Colin Boyd and Christopher Carr<sup>(✉)</sup>

Norwegian University of Science and Technology, NTNU, Trondheim, Norway

{colin.boyd,chris.carr}@item.ntnu.no

**Abstract.** Client puzzles have been proposed as a mechanism for proving legitimate intentions by providing “proofs of work”, which can be applied to discourage malicious usage of resources. A typical problem of puzzle constructions is the difference in expected solving time on different computing platforms. We call puzzles which can be solved independently of client computing resources *fair client puzzles*.

We propose a construction for client puzzles requiring widely distributed computational effort for their solution. These puzzles can be solved using the mining process of Bitcoin, or similar cryptocurrencies. Adapting existing definitions, we show that our puzzle construction satisfies formal requirements of client puzzles under reasonable assumptions. We describe a way of transforming our client puzzles for use in denial of service scenarios and demonstrate a practical construction.

**Keywords:** Bitcoin · Client puzzles · Denial of service resistance · Distributed computation · Proofs of work

## 1 Introduction

Client puzzles, also referred to as *proofs of work* [2, 10, 11], were originally introduced by Dwork and Naor [8] in 1993 and offer a valuable defence against denial of service (DoS). Client puzzles are designed to be used when needed, and can be turned on when a service is receiving an over abundance of requests. Motivation for client puzzles, in the general sense, is fairly intuitive. All service providers would like to ensure that those requesting service legitimately want it, but in an online setting it is difficult to discern a legitimate request for service from a malicious one. So how does one ensure the legitimate intentions of a client machine? This question led to a heuristic that says ‘if a party requesting service is willing to put in some level of effort to connect, then that party is likely to be legitimate’ [18]. Schemes have been proposed that are built around solving computationally expensive puzzles, such as extracting square roots modulo a prime [8], or partially inverting a hash function [3]. Legitimate clients should not find solving any single puzzle to be burdensome, yet, we do not want puzzle solutions to be found trivially either. Thus anyone wishing to exhaust the computational resources of the server by sending multiple connection requests would have to solve one instance of the puzzle per request. It thus becomes difficult for any DoS initiator to attack without access to an enormous amount of computing power.

An issue of primary importance is availability of computational resources on the client’s side, which can vary considerably from device to device. This disparity is a persistent problem with client puzzle applicability. Indeed, another line of research aims to alleviate the problem with computational cost by instead relying on memory bounded cost [1, 16]. By adopting the blockchain for use as the puzzle solution algorithm, we demonstrate an alternative and simpler solution to this problem. We do this by equating monetary cost with computational cost and simultaneously introduce a notion of fairness.

This paper introduces the notion of *fair client puzzles* by exploiting modern developments of distributing computation. Specifically by employing the potential of the Bitcoin blockchain, we create an alternative form of client puzzle which is fair and avoids imposing heavy computations on clients. We go on to demonstrate the theoretical grounding of these fair client puzzles before describing a proof of concept implementation.

**BITCOIN.** The cryptocurrency Bitcoin, in common with its many altcoin variants, requires a computational problem to be solved every time the set of transactions moves forward by formation of a new valid block in the consolidated blockchain. The computational problem is *moderately hard* which means that it is too hard for a single device to solve in a reasonable time, yet easy enough that a dedicated effort by distributed teams of solvers can obtain a solution within a predictable time. The Bitcoin rules ensure that the current difficulty of the computational problem involved is tuned to the available computational effort worldwide so that the expected time to solve a problem remains more or less constant.

The inspiration for the problems used in Bitcoin originally came from the idea of proofs of work. Proofs of work are designed to be solvable by one specific client without regard to its computational power, with solution time varying depending on the local computing power. Whilst in contrast, puzzles in Bitcoin are solvable at a predictable rate.

Based on the above observations, in this paper we propose client puzzles based on the blockchain. We then describe their use as a mechanism for mitigation of denial of service attacks. This is, in a sense, coming full circle by bringing back the *moderately hard* puzzles to a purpose they were originally designed for. At the same time, the return comes with significant benefits due to the distributed way that puzzles are solved in Bitcoin. The puzzles that we define are fairer than conventional client puzzles as they do not affect, or rely upon, the local computation of users. In fact, the fairness we allude to is *order preserving*, which is inherited from the blockchain.

**CONTRIBUTIONS.** We present a novel client puzzle construction that leverages the distributed computing power of the Bitcoin network to solve them. Our central idea is to embed puzzles within the blockchain as part of the puzzle solution mechanism, and requiring clients to prove that their puzzle is contained in it. Instantiating an interactive client puzzle construction, we show that the

puzzles we construct come with considerable practical advantages for both clients and service providers:

1. Any service provider can implement these puzzles, without any requirement for affiliation with the Bitcoin network. Moreover, servers can tune their puzzles to suit their needs, adjusting puzzle difficulty on the fly.
2. The puzzles provide order preserving *fairness*, a previously missing but important goal of client puzzles. In particular, service providers can implement puzzles without concern for hardware availability of their potential clients.
3. The puzzles are adaptable to various Bitcoin-like alternatives (*altcoins*) and other distributed hash chaining schemes.
4. These constructions save expending any individual computational effort of client devices, in contrast with previous client puzzle schemes, by using the work already being produced within the Bitcoin network.

Summarising the DoS resistant attributes, we describe a protocol that operates between a client, or multiple clients, and a server, which may have no association with the Bitcoin network. The protocol runs by the client communication between the blockchain and the server. Referring to Fig. 1, steps 1 and 2 corresponds to a regular request and response round, where the client receives a puzzle in response from the server. Inserting the puzzle in the blockchain makes up step 3, and is performed by the client. Finally, step 4 is the verification step where the client provides proof that the puzzle is contained within the blockchain. Step 0\* is an ongoing process where the server takes information from the blockchain, such as block difficulty. The fifth stage represents general service to the clients, provided that the server accepts at stage 4.

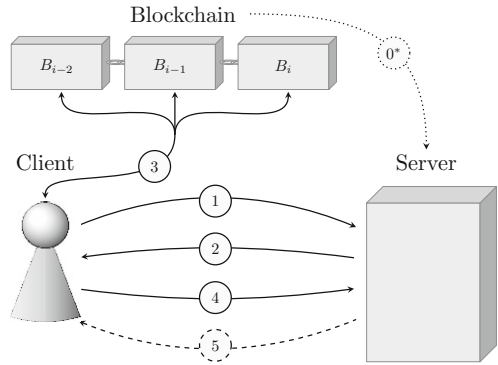


Fig. 1. Client puzzles from Bitcoin

**STRUCTURE OF THE PAPER.** First we introduce client puzzles, with definitions related to the literature [6, 12, 17], discussing and formally defining difficult and costly client puzzles along with their respective security games. We introduce an abstract client puzzle and prove that our construction satisfies these definitions. Next, we move from our theoretical underpinning, to practical composition by means of DoS resistance in Sect. 3, which describes the required properties for DoS resistance, in order to introduce a DoS resistant protocol based on previous work in the literature [17]. The penultimate section demonstrates the proof of concept for DoS mitigation with client puzzles, and discusses how it meets the properties in Sect. 3. The concluding section looks at possibilities for advancing this line of work.



Throughout, we assume some familiarity with the workings of the Bitcoin system. For a broad background understanding we suggest a mix of online resources [20–24] and academic publications [4, 5, 15, 19].

## 2 Client Puzzles

In this section we formally define client puzzles and their security notions, before concluding with a discussion on achieving fair client puzzles. Definition 1 closely follows the one given in the literature [17].

**Definition 1 (Client Puzzle).** *A client puzzle  $CP$  is a tuple of three efficient probabilistic algorithms  $\text{Setup}$ ,  $\text{GenPuz}$ ,  $\text{FindSol}$  and a deterministic algorithm  $\text{VerSol}$ . Let  $\lambda$  be the setup parameter,  $\mathcal{K}$  the key space,  $\mathcal{D}$  the difficulty space,  $\text{Str}$  the string space,  $\mathcal{P}$  the puzzle space and  $\text{Sol}$  be the solution space.*

- $\text{Setup}(1^\lambda)$  : Select  $\mathcal{K}, \mathcal{D}, \text{Str}, \mathcal{P}, \text{Sol}, k \xleftarrow{r} \mathcal{K}, \text{params} \leftarrow (\mathcal{K}, \mathcal{P}, \text{Sol}, \mathcal{D}, \text{Str})$   
Return  $(k, \text{params})$ .
- $\text{GenPuz}(k \in \mathcal{K}, d \in \mathcal{D}, \text{str} \in \text{Str})$  : Return  $p \in \mathcal{P}$ .
- $\text{FindSol}(\text{str} \in \text{Str}, p \in \mathcal{P}, t \in \mathbb{N})$  : Return  $s \in \text{Sol}$  after at most  $t$  clock cycles.
- $\text{VerSol}(k \in \mathcal{K}, \text{str} \in \text{Str}, p \in \mathcal{P}, s \in \text{Sol})$  : Return true or false.

All that is left is to define correctness. Let  $(k, \text{params}) \leftarrow \text{Setup}(1^k)$  and  $p \leftarrow \text{GenPuz}(k, d, \text{str})$ , where  $d \in \mathcal{D}$  and  $\text{str} \in \text{Str}$ , then there exists  $t \in \mathbb{N}$  where

$$\Pr[\text{VerSol}(k, \text{str}, p, s) = \text{true} \mid s \leftarrow \text{FindSol}(\text{str}, p, t)] = 1.$$

Since their inception, client puzzles were studied as a means to mitigate DoS, and reconsidered for use against distributed DoS. Using client puzzles for DoS prevention led to problems with initial designs [7, 13, 17] as the standard security definitions for client puzzles were not robust enough to capture DoS resistance.

### 2.1 Security Notions

We now formalise the security notions for client puzzles in terms of games between an adversary  $\mathcal{A}$  and a server  $S$ . The most obvious way in which an adversary can undermine a client puzzle is to solve a puzzle quicker than expected. Another possibility, as discussed by Chen et al. [7], is a scenario where an adversary could create puzzles independently from the server  $S$ . For an interactive setting we desire that an adversary cannot create a puzzle that the server believes to be valid, referred to as *puzzle unforgeability*. In fact, there are a variety of properties required for DoS mitigation which may or may not be useful as properties for client puzzles. We return to this in Sect. 3.

To define the security games for client puzzles, we first define meaningful oracle calls to help describe them.

- $\text{O.GetPuz}(\text{str})$  : Return  $p \leftarrow \text{GenPuz}(k, d, \text{str})$  and record  $(\text{str}, p)$  in a list.

- $\mathbf{0.GetSol}(str, p)$  : If  $(str, p)$  was not recorded by  $\mathbf{0.GetPuz}$  return  $\perp$ . Else find  $s$  such that  $\mathbf{VerSol}(k, str, p, s) = \mathbf{true}$ . Record  $(str, p, s)$  and return  $s$ .
- $\mathbf{0.VerSol}(str, p, s)$ : Return  $\mathbf{true}$  if  $\mathbf{VerSol}(k, str, p, s) = \mathbf{true}$ ,  $(str, p)$  has been recorded by  $\mathbf{0.GetPuz}$  and  $(str, p, s)$  has not been recorded by  $\mathbf{0.GetSol}$ . Else return  $\mathbf{false}$ .

Let the security game  $\text{EXEC}_{\mathcal{A},d,\text{CP}}^{\text{DIFF}}(\lambda)$  between an adversary  $\mathcal{A}$  and server  $S$ , for a client puzzle CP with setup parameter  $\lambda$  and difficulty  $d \in \mathcal{D}$ , be defined as follows:

1. Server  $S$  performs  $(k, \text{params}) \leftarrow \text{Setup}(1^\lambda)$ , and gives  $\text{params}$  to  $\mathcal{A}$ .
2. Adversary  $\mathcal{A}$  is allowed to make queries to  $\mathbf{0.GetPuz}$  and  $\mathbf{0.GetSol}$ .
3. At any point  $\mathcal{A}$  can run  $\mathbf{0.VerSol}$ , which terminates the game with the result from the algorithm.

**Definition 2 (Difficult Client Puzzle).** *Let CP be a client puzzle for fixed setup parameter  $\lambda$  and  $d \in \mathcal{D}$ . Let  $f_{\lambda,d}(t)$  be a family of monotonically increasing functions on  $t$ . Then CP is said to be  $f_{\lambda,d}(t)$ -difficult if for every  $k \in \mathcal{K}$ ,  $str \in \text{Str}$ ,  $p \in \mathcal{P}$ ,  $s \in \text{Sol}$  and all adversaries  $\mathcal{A}$  running in time at most  $t$ ,*

$$\Pr[\text{EXEC}_{\mathcal{A},d,\text{CP}}^{\text{DIFF}}(\lambda) = \mathbf{true}] \leq f_{\lambda,d}(t).$$

## 2.2 Difficult Client Puzzles from the Blockchain

Now we describe a client puzzle based on hash functions, that are required to produce outputs starting with a certain number of leading zeros. This is similar to a type of puzzle that appears in both the literature and in practice [3, 9]. It is adapted here to describe a high level hash based puzzle, creatable using the blockchain.

Let  $\mathcal{H}_i$ ,  $1 \leq i \leq 4$  be publicly available hash functions running in polynomial time, taking inputs in  $\{0, 1\}^*$  and producing outputs in  $\{0, 1\}^{l_i}$ . Then define publicly available algorithms as follows:

- $\mathbf{BC.VerTx}$  on input  $x, p, m_1$  runs  $\tilde{x} \leftarrow (m_1, \mathcal{H}_1(p), \mathcal{H}_2(m_1, \mathcal{H}_1(p)))$  and returns  $\mathbf{true}$  if  $x = \tilde{x}$ . Else returns  $\mathbf{false}$ .

This process verifies that a transaction  $x$  contains a puzzle  $p$  along with some auxiliary data  $m_1$ , which captures the extra information required to form a transaction.

- $\mathbf{BC.Merk}$  takes up to  $n$  inputs  $\{t_{x_1}, \dots, t_{x_n}\}$ , for some fixed integer  $n$ , and outputs a Merkle root  $r_t$ , by forming a hash tree with hash function  $\mathcal{H}_3$  and returning a single hash output.

The algorithm  $\mathbf{BC.Merk}$  takes  $n$  ordered inputs, under the assumption that the maximum number of inputs to  $\mathbf{BC.Merk}$  is some polynomial on the input parameter, constructing a Merkle tree [14]. These inputs,  $t_{x_i}$  can be thought of as the transactions within Bitcoin that are contained within each block. The first input

$t_{x_1}$  is hashed with the second input  $t_{x_2}$ , using the hash algorithm  $\mathcal{H}_3$ , producing an output in  $\{0,1\}^{l_3}$ . This output is fed back into  $\mathcal{H}_3$  again with the next original input  $t_{x_3}$ , and so on, until all  $n$  inputs have been included:

$$r_t = \mathcal{H}_3(\dots \mathcal{H}_3(\mathcal{H}_3(t_{x_1}, t_{x_2}), t_{x_3}) \dots).$$

- **BC.VerMerk** takes input  $r_t$ , a Merkle root, and up to  $n$  inputs  $t_{x_1}, \dots, t_{x_n}$ , then runs  
 $\tilde{r}_t \leftarrow \text{BC.Merk}\{t_{x_1}, \dots, t_{x_n}\}$ , and returns true if  $r_t = \tilde{r}_t$ . Else returns false.

This verification algorithm checks that the submitted Merkle leaves form into the correct Merkle root.

- **BC.VerBlk** takes inputs  $s', r_t$  and  $m_2$ , returning true if  $\mathcal{H}_4(s', r_t, m_2)$  has  $d$  leading zeros.  
 Else returns false.

This process mimics the Bitcoin block verification algorithm, where  $s'$  is the solution and  $r_t$  is the Merkle tree containing the transactions and  $m_2$  is any auxiliary or extra data to be included.

**ABSTRACT BLOCKCHAIN BASED CLIENT PUZZLE.** Using these algorithms, we can now define an *abstract blockchain-based client puzzle* by instantiating the algorithms in Definition 1 as follows:

- **Setup**( $1^\lambda$ ): Selects  $\mathcal{K} = \emptyset$ ,  $\mathcal{D} \subseteq \{0, 1, \dots, l_4\}$ ,  $\text{Str} \subseteq \{0, 1\}^*$ ,  $\mathcal{P} \subseteq \{0, 1\}^*$ ,  $\text{Sol} \subseteq \{0, 1\}^*$  and returns  $\text{params} = (\mathcal{D}, \text{Str}, \mathcal{P}, \text{Sol})$ .
- **GenPuz**( $d, \text{str}$ ): Returns puzzle  $p = \text{str}$ .
- **FindSol**( $\text{str}, p$ ): Returns  $s$  of the form  $s = (s', r_t, x, t_{x_2}, \dots, t_{x_q}, m_1, m_2)$ .
- **VerSol**( $p, s$ ) where  $s = (s', r_t, x, t_{x_2}, \dots, t_{x_q}, m_1, m_2)$ : Returns true if and only if **BC.VerTx**( $x, p, m_1$ ) returns true.  
**BC.VerMerk**( $r_t, x, t_{x_2}, \dots, t_{x_q}$ ) returns true and **BC.VerBlk**( $s', r_t, m_2$ ) returns true. Else returns false.

Assuming  $\mathcal{H}_4$  is at least surjective, it is easy to find a  $t$  such that correctness holds. Note that we purposely omit describing the inputs to the client puzzle algorithms where they are not used; in this instance the client puzzle is not keyed, so we do not list  $k \in \mathcal{K}$  as an input to either **GenPuz** or **VerSol**.

**Theorem 1.** *Let CP be an abstract blockchain based client puzzle, for security parameter  $\lambda$ ,  $d \in \mathcal{D}$ , and let  $\mathcal{H}_i$ ,  $1 \leq i \leq 4$  be random oracles, with output lengths  $l_1, l_3, l_4 \geq d$ . Let  $f_{\lambda, d}(t+1) = \frac{t}{2^d}$ . Then CP is an  $f_{\lambda, d}(t)$ -difficult client puzzle.*

*Proof.* For Adversary  $\mathcal{A}$  to win the game, it equates to finding an  $s'$  such that  $\mathcal{H}_4(s', r_t, m)$  has at least  $d$  leading zeros.  $\mathcal{A}$  first queries **GenPuz** on some string  $\text{str}$ , returning the puzzle  $p = \text{str}$ , thus  $(\text{str}, \text{str})$  is now recorded. This step must take place, otherwise **0.VerSol** could only output false. Next,  $\mathcal{A}$  generates a valid transaction  $x$ , and Merkle root  $r_t$ , using the public algorithms  $\mathcal{H}_1, \mathcal{H}_2$  and  $\mathcal{H}_3$ .

This first process takes at least one step. All that is left is to find a valid  $s'$  such that  $\mathcal{H}_4(s', r_t, m)$  has  $d$  leading zeros. W.l.o.g. fix  $m$ , then  $\mathcal{A}$  proceeds to run the function  $\mathcal{H}_4(s', r_t, m)$  for different values of  $s'$ . For any one run, the chance of finding a valid  $s'$  satisfying this condition is  $1/2^d$ . If a solution has not been found after the penultimate attempt,  $\mathcal{A}$  may attempt a guess for  $s'$ , along with the call to `0.VerSol`. Hence, the probability of  $\mathcal{A}$  succeeding, and `0.VerSol` returning true, is bounded by  $t/2^d$ .  $\square$

This client puzzle construction can be realised using the blockchain. Finding a solution to a puzzle is simply the process of encapsulating it within a transaction in a subsequent block. Thus, the difficulty must be chosen with respect to the difficulty specified by the blockchain protocol. For the Bitcoin system, taking difficulty  $d$  from the blockchain makes it improbable for any individual to solve a puzzle in a reasonable amount of time. Rather than expect a user to mine a block, we rely on the Bitcoin system to solve the puzzle. Consequently, a client needs only to form a transaction on the blockchain including the puzzle, then wait for the miners to find a solution to the block – hence the puzzle. This client puzzle effectively replaces computational cost by a time delay. Constructing a client puzzle is demonstrated in Sect. 4, with a proof-of-concept implementation.

### 2.3 Stronger Security from Cost-Based Puzzles

Definition 2 only captures the difficulty of solving one puzzle in  $t$  computational steps. This does not fully express the intuition behind client puzzles [9, 17], as it says nothing about the possibility of attempting to solve many puzzles at once. We want client puzzles where solving  $n$  puzzles is approximately  $n$  times as difficult as solving just one. Client puzzles with this property are referred to as *strongly difficult*. We formalise this notion slightly differently, by employing abstract blockchain based client puzzles, and replacing difficulty in terms of computational steps by *cost* in terms of monetary expense.

Despite a time delay feature, the abstract scheme as described is free of any significant cost for the clients and whilst this is the case clients are able to solve as many puzzles as they like provided they do not mind waiting. What is more, any client can solve multiple puzzles in the same time it takes to solve one. Later we will see that both of these problems can be addressed using Bitcoin by imposing a monetary cost to embedding puzzles within a block, and thus finding a solution. Formalising this idea is achieved in a similar manner to the work on *interactive strong puzzle difficulty*, originally given by Stebila et al. [17]. For now, we need to define the security experiment, and make adversarial assumptions.

Define the security game for  $\text{EXEC}_{\text{CP}, \mathcal{A}, d, n, c_t}^{\text{CP}}(\lambda)$  between a p.p.t adversary  $\mathcal{A}$  and server  $S$ , for a client puzzle CP with setup parameter  $\lambda$ ,  $n \geq 1$  and a predetermined cost  $c_t \geq 0$  as follows.

1. Server  $S$  performs  $(k, \text{params}) \leftarrow \text{Setup}(1^\lambda)$ , and gives  $\text{params}$  to  $\mathcal{A}$ .
2. Server  $S$  runs  $p_i \leftarrow \text{0.GetPuz}(str_i)$  for distinct, random  $str_i, i = 1, \dots, n$ . All  $(str_i, p_i)$  are recorded and given to  $\mathcal{A}$ .

3. Adversary  $\mathcal{A}$  has oracle access to `0.GetSol` and a solution verification oracle taking inputs  $(str, p, s)$  and returning `VerSol(k, str, p, s)`. Additionally,  $\mathcal{A}$  may solve any puzzle  $p_i$  at a fixed cost  $c_t$ .
4. Adversary  $\mathcal{A}$  generates  $L = \{(str_i, p_i, s_i) : i \in \{1, \dots, n\}\}$ .
5. At any point  $\mathcal{A}$  can end the game by submitting  $L$  as a solution. The algorithm returns true if for all  $i$  from 1 to  $n$ , `0.VerSol(stri, pi, si) = true`.

With this new version of the puzzle security game we can now define a *costly client puzzle* as an analogue of the strongly difficult client puzzles [17].

**Definition 3 (Costly Client Puzzle).** *Let  $CP$  be a client puzzle, let  $f_{\lambda,d,n}(t)$  be a family of functions monotonically increasing in  $t$  and let  $negl$  be a negligible function where*

$$|f_{\lambda,d,n}(t) - f_{\lambda,d,1}(t/n)| \leq negl(\lambda, d),$$

for all  $t, n$  such that  $f_{\lambda,d,n}(t) \leq 1$ . For a fixed setup parameter  $\lambda$  and difficulty parameter  $d \in \mathcal{D}$ , for  $n \geq 1$ , then  $CP$  is an  $f_{\lambda,d,n}(t)$ -costly client puzzle if for all p.p.t algorithms  $\mathcal{A}$  running in time at most  $t$ , where each puzzle has associated cost  $c_t > 0$ ,

$$\Pr[\text{EXEC}_{CP, \mathcal{A}, d, n, c_t}^{CCP}(\lambda) = \text{true}] \leq f_{\lambda,d,n}(t).$$

*Remark 1.* An adversary attacking the strong security of a blockchain based client puzzle may attempt to embed multiple puzzles in one block. This may be possible, and in fact is a design goal of blockchain based puzzles. However, as the number of embedded puzzles per block increases, the average computational cost expended per puzzle decreases. Therefore the design of the puzzle should ensure that the number of puzzles embedded in a block is limited to some threshold  $n$ . Furthermore, the cost to the adversary of mining a block should remain (much) higher than the cost of solving one puzzle in the intended manner, namely embedding it in a valid Bitcoin transaction.

To achieve strong security of a blockchain-based client puzzle we assume that the adversary is unable to reduce the average cost of finding solutions to puzzles below a certain threshold cost  $c_t$  which we call the *design cost* of the puzzle. The design cost can be assigned in different ways, such as applying transaction fees or “proof-of-burn”[22] (to list just two methods). Note that  $c_t$  is a monetary cost which we freely compare with computational cost. This is important, as it means clients can outsource their puzzles for a cost, which is itself equatable to computation, meaning the outsource is not achieved for free.

Returning to our abstract puzzle definition, the adversary can attempt to solve multiple puzzles at once within a single Merkle root  $r_t$ , and finding one  $s'$  and auxiliary  $m$  such that  $\mathcal{H}_4(s', r_t, m)$  has the requisite number of leading zeros. We need this to cost more than the cost of solving the maximum number of puzzles which can be embedded in one block.

**Assumption 1.** *Let  $CP$  be an abstract blockchain based client puzzle, with security parameter  $\lambda$ . Let  $c_t$  be the design cost of solving one puzzle, and fix a parameter of difficulty  $d$ . Then there exists  $q \in \mathbb{Z}_{>0}$  where, for all  $r_t$  and  $m$ , the average computational cost of finding an  $s'$  such that  $\mathcal{H}_4(s', r_t, m)$  has  $d$  leading zeros, is at least  $q \cdot c_t$ .*

In our Bitcoin embodiment, Assumption 1 expresses the idea that generating a block is so stupendously expensive that it is cheaper to bear the cost of embedding a significant number of puzzles. In practice there is a limit on the number of puzzles that can be embedded in one Bitcoin block due to a fixed limit on the size of blocks. Each block has a capped limit of 1 Mb [22] and a puzzle included within a transaction with one input and one output is approximately 214 bytes, so currently just under 4,900 transactions are embeddable per block. Thus for Bitcoin this can be a realistic choice for  $q$ .

Using Assumption 1 we can prove that the abstract blockchain based puzzle is a costly client puzzle. Thus we can use this puzzle to ensure that an adversary who wishes to solve puzzles must incur a cost that is linear in the number of puzzles they intend to solve, which is a significant disincentive to carrying out DoS attacks by attempting to solve multiple puzzles. This can be an attractive mechanism to use in practice on the assumption that legitimate users are willing to accept a single fee  $c_t$  in order to maintain access to a service during times of attack. We explore the use of such client puzzles in more detail in Sect. 3.

**Theorem 2.** *Let CP be an abstract blockchain based client puzzle, for security parameter  $\lambda$ ,  $d \in \mathcal{D}$ , cost per puzzle  $c_t$ , and let  $\mathcal{H}_i$ ,  $1 \leq i \leq 4$  be random oracles, with output lengths  $l_1, l_3, l_4 \geq d$ . Let  $f_{\lambda, d, n}(t) = (t - c_t(n - 1))(n - 1)/2^d$ . Then CP under Assumption 1 with  $q > n$  is an  $f_{\lambda, d, n}(t)$ -costly client puzzle.*

*Proof.* Following from the assumption, if  $\mathcal{A}$  attempts to attach the received puzzles within a Merkle root  $r_t$ , and then tries to compute an  $s'$ , such that  $\mathcal{H}_4(s', r_t, m)$  has  $d$  leading zeros, it becomes far too costly, i.e. the total cost would be  $q \cdot c_t$ , which is much more expensive than if  $\mathcal{A}$  simply won the game by paying for each solution.

Another strategy for  $\mathcal{A}$  is to ask for the solution to all but one puzzle, each costing  $c_t$  computational steps, leaving  $\mathcal{A}$  with  $t - c_t(n - 1)$  steps remaining. Now,  $\mathcal{A}$  can generate a legitimate transaction  $x \leftarrow (m_1, \mathcal{H}_1(p), \mathcal{H}_2(m_1, \mathcal{H}_1(p)))$ . Note that for each of the paid for  $n - 1$  puzzle and solution pairs,  $(p_j, s_j)$ , the solution contains a Merkle root  $r_{t_j}$ . Thus  $\mathcal{A}$  can attempt to form a Merkle root  $r_t$  by finding some value  $m'$ , such that  $\mathcal{H}_3(x, m')$  gives  $r_t = r_{t_j}$  for any  $j$ . This approach is essentially attempting to find a collision within one of the  $n - 1$  Merkle trees. The output space of  $\mathcal{H}_3$  is  $2^{l_3}$ , so the probability of finding a solution in  $t$  steps is  $(t - c_t(n - 1))(n - 1)/2^{l_3}$ . As  $l_3 > d$  we have  $(t - c_t(n - 1))(n - 1)/2^{l_3} \leq (t - c_t(n - 1))(n - 1)/2^d$  as required.  $\square$

## 2.4 Achieving Fair Client Puzzles

So far we have explored difficult and costly client puzzles, but have not considered fairness. Why do we need fairness in client puzzles? Consider a case that behaves very similarly to a DoS attack, where multiple legitimate clients request service all at once, referred to as a *flash flood* scenario. Here, the natural approach is to serve clients on a first come, first served basis. However, a dated smart phone

would be able to perform significantly less operations per second than a high end desktop computer, and so would be at a disadvantage if it were forced to solve client puzzles to gain access. These are fairness considerations.

Essentially, the fairness required is order preserving. We state this informally as: for any two clients  $C$  and  $C'$ , if  $C$  attempts to solve a puzzle before  $C'$ , it should find a solution no later than  $C'$ . For our abstract blockchain-based client puzzle, instantiated with the Bitcoin blockchain, this becomes achievable. There is an order preserving algorithm, namely the process of adding a transaction to the blockchain. It is also reasonable to assume that any adversary we are concerned with would not generate a block before the worldwide collection of Bitcoin miners. The Bitcoin protocol cannot be fair in an ideal way due to factors such as transaction fees and availability of nearby nodes, which affect the time until a transaction is included in the blockchain. However, in general complete availability is not achievable, and we must assume, along with the built in assumptions on Bitcoin, that provided the correct transaction fees are included nodes do not arbitrarily reject certain transactions.

### 3 Application to Denial of Service Resistance

Denial of service (DoS) attacks, including distributed denial of service (DDoS) attacks, are popular and widely employed techniques of attacking web based resources and services, causing considerable concern for online service providers<sup>1</sup>. DoS is an easily exploitable attack vector, challenging to defend against and relatively simple to invoke. Despite the awareness and wide acknowledgement of the problem, DoS is still a prevalent threat.

A conventional DoS attack, and the one we focus on, aims to deplete the computational resources of the server by requesting and re-requesting connection [17]. If the server demands secure authentication this will require expending computational effort, which the attacker exploits by not completing the authenticated handshake. This type of DoS attack makes the effort expended asymmetric. The attacking clients are free to ignore all replies, thus expending no effort other than a request for service, whilst the server expends considerable effort for each request. Ameliorating this imbalance is the aim of this section, using the abstract client puzzle construction as described so far. The intention is to incorporate these client puzzles within a pre established DoS resistant framework.

#### 3.1 Client Puzzles for Denial of Service Resistance

The criteria provided below are extracted from recent literature on DoS resistance using client puzzles [6, 7, 9, 18], with the exception of Criterion 4, which expresses the notion of fairness developed in Sect. 2.4. The term, *expensive operations* is left ambiguous, since it depends on capabilities of the server.

---

<sup>1</sup> Throughout we use DoS to refer to both DoS and DDoS, when not explicitly stated otherwise.

1. A server  $S$  should not carry out any *expensive operations* unless it believes that the client  $C$  legitimately wants service.
2. Server  $S$  should not perform any *expensive operations* unless it believes that client  $C$  intends to communicate with  $S$ , and not with another server  $\tilde{S}$ .
3. All cost incurred by the client  $C$  in order to solve a client puzzle for server  $S$ , can only be used to prove  $C$ 's intentions to communicate with  $S$ .
4. Any client  $C$ , that attempts to solve a challenge  $p$  from server  $S$ , before any other client  $\tilde{C}$  attempts to solve challenge  $\tilde{p}$ , should find a solution to  $p$  no later than  $\tilde{C}$  finds a solution to  $\tilde{p}$ .
5. The cost of solving  $n$  puzzles is approximately  $n$  times the cost of solving 1 puzzle.
6. No party other than  $S$  should be able to generate valid puzzles for  $S$ , with non-negligible probability.
7. Any client that receives puzzles from  $S$  cannot solve and store the puzzles, then later respond with an overwhelming number of legitimate puzzle solutions at one time.
8. It is not possible to replay previously accepted puzzle solutions.
9. The cost required for solving a puzzle is adjustable.

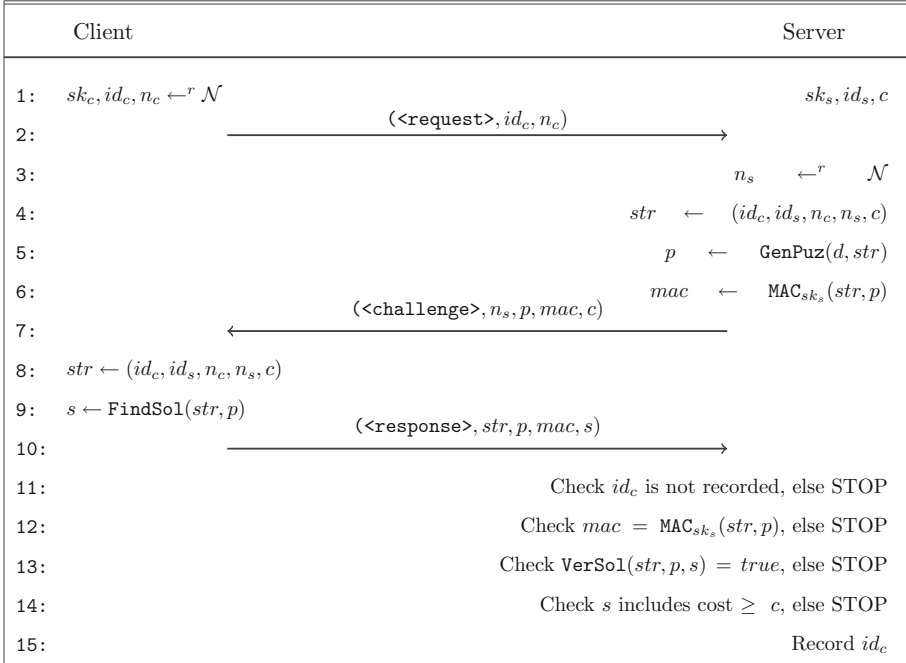
Criterion 2 is designed to prevent a malicious server redirecting requests for service to a legitimate server  $S$ . This exploit is discussed in detail by Mao and Paterson [13], and is crucial to DoS resilience. The concern here is that, with almost no effort expended by the malicious server  $\tilde{S}$ , it is possible to forward all requests on to a legitimate server  $S$ , where the client believes it is communicating with  $\tilde{S}$ , and the server  $S$  believes it is communicating with  $C$ . Another property, Criterion 6, refers to the unforgeability of client puzzles. For interactive client puzzles, this is necessary, however some client puzzles have been designed specifically to be non-interactive, so that a client can generate a valid puzzle. For non-interactive puzzles, unforgeability is meaningless.

### 3.2 Creating a DoS Resistant Protocol

Stebila et al. [17] demonstrate a way of transforming any *interactive strongly difficult* client puzzle into a DoS resistant protocol. This transformation is also achievable for any interactive costly client puzzles as described here. First observe that the only difference between costly client puzzles and strongly difficult ones is the way that cost is measured. We can create an upper bound in terms of cost, which is equatable with computational steps, and thus use the proof from Stebila et al. [17, Eq. 7, p. 28].

Building on this construction, we demonstrate the use of our abstract blockchain-based client puzzle in a DoS resistant scenario, by means of an example – see Fig. 2. This protocol is constructed as detailed by Stebila et al. [17, Sect. 5]. We assume that server and client have public identities  $id_s$  and  $id_c$ . The aim is for the server to accept an identity from the client and store the identity in a list of accepted identities only if a valid run of the protocol for that identity has taken place. The intention here is that the protocol is run prior to any further, perhaps





**Fig. 2.** Abstract Blockchain DoS Resistant Protocol

more expensive, demands on the server, such as key agreement. However, the more general construction [17] allows for any protocol to be built into the first 3 stages, provided that the server does not perform any expensive computation until after the response has been accepted – so after line 15 in Fig. 2.

### 3.3 Meeting the Denial of Service Resistant Criteria

Relating to the DoS resistance criteria specified in Sect. 3.1, notice that for most of the desired properties it is straightforward to see that they are achieved. Here, we expand further on the less obvious criteria.

Point 4 holds with the assumption that once a transaction is created and broadcast, it will be included in the next block, which is a fundamental assumption discussed previously in Sect. 2.4. Thus, under this assumption, we achieve an order preserving fairness where puzzle solutions cannot be found any sooner than solutions to earlier puzzles.

Observe that point 7 is not satisfied in this instance. Originally noted by Groza and Warinschi [9], the proposed method for adapting client puzzles for DoS prevention [17] provides no defence against a next-day attack, as there is no limit on the time allowed to return a solved puzzle. A simple solution involves the server changing keys at regular intervals. To accomplish this, whilst maintaining fairness for clients, the server generates and stores two keys: an old key  $sk_{s_1}$  and

a new key  $sk_{s_2}$ . The server then periodically updates the keys, redeclaring the current newest as the oldest  $sk_{s_1} = sk_{s_2}$ , then generating and storing a new key  $sk_{s_2}$ . The new key,  $sk_{s_2}$ , is then used to generate the MAC on line 6 of Fig. 2, but the verification on line 12 will have to test both keys and continue if the MAC returned by the client is valid under one of the keys.

We have excluded this rekeying process from the protocol for two reasons. Firstly, it further convolutes an otherwise straightforward protocol, and is very easy to solve. Secondly, perhaps most crucially, we expect that a server would only employ this DoS mitigation protocol when receiving high volumes of traffic, and thus generate a fresh key at each instantiation. Unless there is some sustained attack, there may be no need to implement a rekeying procedure. Thus, we leave this process as an optional mechanism.

Point 3 is slightly trickier, as there is nothing stopping a client solving a puzzle on the behalf of someone else. However, if we consider an established beneficiary of  $C$  to be a part of  $C$  itself, it becomes clear that this property is also achieved, by virtue of the string generation on the client side (Step 8 Fig. 2), and the subsequent MAC verification on the server side (Step 12, Fig. 2).

## 4 Proof of Concept

We provide a practical demonstration of the protocol described in Fig. 2, by creating an example puzzle and storing it within a transaction using the Bitcoin Testnet.

Suppose a client  $C$  wishes to register identity  $id_c$  with a server  $S$ , in order to later receive some service. First  $C$  generates a nonce  $n_c$ , and sends it along with  $id_c$  to the server  $S$ . In response, the server generates and returns a puzzle  $p$ , a server nonce  $n_s$ , a cost  $c$  and a message authentication code  $mac$ .

To create the response, the server first forms the string  $str$ , which is made up of the server address  $id_s$  and the client address  $id_c$ , along with their nonces and the server specified cost  $c$ . For our example, we take

```

id_c = mjMnKihRdbr5VVdDV67QGd13EVnUBm6F7k
id_s = mmsU7xHjJLdiqgL3udyJ7oooNXT094M9nE
n_c = 27e7e82f79c5ab86e99fcf7024fd4003b87fc8a7eb99fd00e77d9ba55a02e197
n_s = d053c6e0c1756705b0abfb3ff9374e85a5c1e85d9ed7481f1b61926dcce17f9f
c = 1
str = (id_c||id_s||n_c||n_s||c)

```

Addresses  $id_c$  and  $id_s$  are encoded in the Bitcoin specified base-58, whilst the nonces used are hexadecimal. Cost  $c$  is represented in decimal with cost 1 equating to  $10^{-7}$  bitcoins. In this instance the generate puzzle algorithm, **GenPuz**, simply returns the string  $p = str$ , as the difficulty on the blockchain does not need to be known to create puzzles.

Now the client can solve the puzzle  $p$  by encapsulating it within the next block. One way of inputting this puzzle into the blockchain is to form the puzzle into an address, which requires running the puzzle  $p$  through the Bitcoin public key to address generation algorithm. The client does this by running  $p' \leftarrow \text{RIPEMD}^{160}(\text{SHA-256}(p))$ , then prepends some auxiliary message data  $m_1$  to  $p'$ , then takes  $m_2$ , the first four bytes of  $\text{SHA-256}(\text{SHA-256}(m_1||p'))$ , and appends them to  $m_1||p'$  to form the full output  $m_1||p'||m_2$ . This process is designed to agree with the address generation method of Bitcoin.

Running this algorithm on the string above we get `mwwCiu...p7NS` encoded in base-58. This is now a legitimate Testnet address. The client creates a transaction including this address and cost  $c$ , then sends it to the network.

```

1: "vout" : [
2:     {
3:         "value" : 0.00000010,
4:         "n" : 0,
5:         "scriptPubKey" : {
6:             "asm" : "OP_DUP OP_HASH160 b4180a2
7:                 fdaef5c1afdc3b0e73fe699094e634ff7
8:                 OP_EQUALVERIFY OP_CHECKSIG",
9:             "hex" : "76a914b4180a2fdaef5c1afdc
10:                  3b0e73fe699094e634ff788ac",
11:            "reqSigs" : 1,
12:            "type" : "pubkeyhash",
13:            "addresses" : [
14:                "mwwCiu1hfeJVppaWtfAHCWwKZ2j57Fp7NS"
15:            ]
16:        }
17:    },

```

Above is a portion of a Bitcoin Testnet transaction, created to include the puzzle in the block. Line 14 specifies the receiving address, which we recognise as  $m_1||p'||m_2$ , a function of the puzzle  $p$ . On line 3, the value specifies the number of bitcoins assigned to the address. This is the specified cost.

It is now possible to verify that the transaction is recorded in the block, within transaction  $x = \text{c55} \dots \text{2d5}$ , which is a double  $\text{SHA-256}$  of the complete transaction. This transaction is then encoded within the block `000...aa4`, which can be verified using the block's details.<sup>2</sup> This also allows us to verify the cost associated with the transaction, and note that the address of the puzzle is assigned 0.00000010 bitcoins, as specified by the puzzle – we are using a cost of 1 to be equivalent to  $10^{-7}$  bitcoins. The cost is a protocol level specification, verified at line 14 in the figure. Merkle root creation and block generation are performed by the mining nodes, encompassing the transaction within the blockchain. Once this process is complete, the client can return the valid string  $str, p, s, m$ .

<sup>2</sup> <https://goo.gl/VZTIKe>.

Upon receiving the response, the verification algorithms `BC.VerMerk` and `BC.VerBlk` are run by the server, which also take into account the difficulty of Bitcoin. In Bitcoin, the blockchain difficulty remains static for 2016 blocks at a time. For this example, the difficulty  $d$  was set at 227267.00000000. This verification process is incredibly cheap in terms of computation, and is similar to the process used when participating as a node in the Bitcoin network. In simulations, it was possible to generate one hundred thousand puzzles in 1.225 seconds, with the MAC verification step taking on average 1.061 seconds per hundred thousand puzzle responses.

Provided the straightforward verification stages 11, 12 and 14 from Fig. 2 return true, the only other verification needed is to check that  $x$ , returned by the client as part of the solution  $s$ , exists as a transaction within the block, which includes an encoding of the puzzle  $p$  and supplementary data  $m$ .

## 5 Conclusion

There are various ways to extend and adapt this work. One possibility is to develop a rigorous notion of fairness in terms of client puzzles, and relating that to what is achievable with puzzles constructed from the blockchain, along with analysis that focuses on achieving a fairness property in a practical setting. Extending this work by looking at altcoins could lead to more efficient, and potentially simpler, client puzzles constructions, running quicker than on the Bitcoin network. This approach may increase the scope of potential applications, though we expect there to be some trade off between security and application. From a broader perspective, research on the willingness of clients to spend money over allowing their machine to perform computations is a nice avenue for investigation. Remaining challenges to developing and deploying practical client puzzles based on Bitcoin are discussed in the full version of this paper<sup>3</sup>.

Overall, we have presented a novel approach to client puzzles, providing a fair alternative to previously proposed proof of space and proof of work schemes, which lack this property. Using Bitcoin's proof of work system for client puzzles is in some ways coming full circle, but we have gained considerably along the way. We can now construct client puzzles which are solvable in a fixed time, and independent of client device. Focus can now be directed towards implementation issues, such as speed, reliability and practical application.

## References

1. Atenièse, 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, Heidelberg (2014)

---

<sup>3</sup> The full version of the paper can be found online, on the IACR eprint page: <https://eprint.iacr.org/>.

2. Aura, T., Nikander, P., Leiwo, J.: DOS-resistant authentication with client puzzles. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) *Security Protocols 2000*. LNCS, vol. 2133, pp. 170–177. Springer, Heidelberg (2001)
3. Back, A.: Hashcash-a denial of service counter-measure (2002). <http://www.hashcash.org/papers/hashcash.pdf>
4. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to Better — how to make bitcoin a better currency. In: Keromytis, A.D. (ed.) *FC 2012*. LNCS, vol. 7397, pp. 399–414. Springer, Heidelberg (2012)
5. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: research perspectives and challenges for bitcoin and cryptocurrencies. In: *IEEE Symposium on Security and Privacy, SP 2015*, pp. 104–121. IEEE Computer Society (2015)
6. Boyd, C., et al.: Cryptographic approaches to denial-of-service resistance. In: Raghavan, S.V., Dawson, E. (eds.) *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks*, pp. 183–238. Springer, Heidelberg (2011)
7. Chen, L., Morrissey, P., Smart, N.P., Warinschi, B.: Security notions and generic constructions for client puzzles. In: Matsui, M. (ed.) *ASIACRYPT 2009*. LNCS, vol. 5912, pp. 505–523. Springer, Heidelberg (2009)
8. 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)
9. Groza, B., Warinschi, B.: Revisiting difficulty notions for client puzzles and DoS resilience. In: Gollmann, D., Freiling, F.C. (eds.) *ISC 2012*. LNCS, vol. 7483, pp. 39–54. Springer, Heidelberg (2012)
10. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols. In: *Secure Information Networks: Communications and Multimedia Security, IFIP Conference Proceedings*, vol. 152, pp. 258–272. Kluwer (1999)
11. Juels, A., Brainard, J.G.: Client puzzles: a cryptographic countermeasure against connection depletion attacks. In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 1999*. The Internet Society (1999)
12. Karame, G.O., Čapkun, S.: Low-cost client puzzles based on modular exponentiation. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) *ESORICS 2010*. LNCS, vol. 6345, pp. 679–697. Springer, Heidelberg (2010)
13. Mao, W., Paterson, K.G.: On the plausible deniability feature of Internet protocols (2002). [www.isg.rhul.ac.uk/~kp/IKE.ps](http://www.isg.rhul.ac.uk/~kp/IKE.ps)
14. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) *CRYPTO 1987*. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)
15. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). <https://bitcoin.org/bitcoin.pdf>
16. Percival, C.: Stronger key derivation via sequential memory-hard functions (2009). <http://bitcoin-class.org/0/classes/class16/scrip.pdf>
17. Stebila, D., Kuppusamy, L., Ranganamy, J., Boyd, C., Gonzalez Nieto, J.: Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In: Kiayias, A. (ed.) *CT-RSA 2011*. LNCS, vol. 6558, pp. 284–301. Springer, Heidelberg (2011)
18. Stebila, D., Ustaoglu, B.: Towards denial-of-service-resilient key agreement protocols. In: Boyd, C., González Nieto, J. (eds.) *ACISP 2009*. LNCS, vol. 5594, pp. 389–406. Springer, Heidelberg (2009)
19. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. *IACR ePrint Archive 2015:464* (2015)

20. Web: Ken shirriff [www.righto.com](http://www.righto.com). <http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html>. (Accessed on Oct 15)
21. Web: Bitcoin Block Explorer (2015). <http://blockexplorer.com/>.(Accessed on Nov 15)
22. Web: Bitcoin Wiki (2015). [https://en.bitcoin.it/wiki/Main\\_Page](https://en.bitcoin.it/wiki/Main_Page) (Accessed on Nov 15)
23. Web: CoinDesk (2015). <http://www.coindesk.com/>. (Accessed on Nov 15)
24. Web: [michaelnielsen.org](http://www.michaelnielsen.org) (2015). <http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/>. (Accessed on Nov 15)

# **Statistical Privacy**

# Privacy-Preserving $k$ -Nearest Neighbour Query on Outsourced Database

Rui Xu<sup>1</sup>(✉), Kirill Morozov<sup>2</sup>, Yanjiang Yang<sup>3</sup>,  
Jianying Zhou<sup>4</sup>, and Tsuyoshi Takagi<sup>5,6</sup>

<sup>1</sup> KDDI R&D Laboratories, Inc., Fujimino, Japan  
ru-xu@kddilabs.jp

<sup>2</sup> School of Computing, Tokyo Institute of Technology, Tokyo, Japan

<sup>3</sup> Huawei Singapore Research Center, Singapore, Singapore

<sup>4</sup> Institute for Infocomm Research (I<sup>2</sup>R), Singapore, Singapore

<sup>5</sup> Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan

<sup>6</sup> CREST, Japan Science and Technology Agency, Tokyo, Japan

**Abstract.** Cloud computing brought a shift from the traditional client-server model to DataBase as a Service (DBaaS), where the data owner outsources her database as well as the data management function to the cloud service provider. Although cloud services relieve the clients from the data management burdens, a significant concern about the data privacy remains. In this work, we focus on privacy-preserving  $k$ -nearest neighbour ( $k$ -NN) query, and provide the first sublinear solution (with preprocessing) with computational complexity  $\tilde{O}(k \log^4 n)$  in the honest-but-curious adversarial setting. Our constructions use the data structure called  $kd$ -tree to achieve sublinear query complexity. In order to protect data access patterns, garbled circuits are used to simulate Oblivious RAM (ORAM) for accessing data in the  $kd$ -tree. Compared to the existing solutions, our scheme imposes little overhead on both the data owner and the querying client.

**Keywords:** Privacy-preserving computation ·  $k$  nearest neighbour search · Outsourcing of computation · Encrypted database · Oblivious RAM

## 1 Introduction

The last decade witnessed a rapid development of cloud computing, and the virtue of ubiquitous, convenient, on-demand network access makes it a great success. An interesting application of cloud computing is DataBase as a Service (DBaaS) [7], where the data owner outsources both the database and its management functionality to the cloud service provider to reduce database management cost. Despite all

---

This work was partially done during R.X.'s internship at I<sup>2</sup>R.

This work was partially done when K.M. was with IMI, Kyushu University.

This work was partially done when Y.Y. was with I<sup>2</sup>R.

This research was supported by JST CREST.



the benefits, concerns about data privacy remain in DBaaS, as the cloud cannot be fully trusted and data owners want to protect their data from exposure to the cloud. Encrypting the database before outsourcing it to the cloud is a straightforward method of providing privacy. However, this introduces a challenge of *searching over the encrypted database*, since generally it is not easy to manipulate encrypted data in a meaningful (and yet secure) manner.

In this work, we consider an even more challenging scenario when the data owner and the querying client *distrust* each other. Multi-party computation (MPC) introduced by Yao [28] represents a general tool for dealing with problems of this nature. However, directly applying MPC commonly results in unfavorable performance, since the querying client may use the devices which are restricted in terms of computational and/or communication capacities, such as smart phones. In the DBaaS scenario, the cloud has huge computation resources. Thus, it would be natural to use this advantage.

We focus on privacy preserving  $k$ -nearest neighbour ( $k$ -NN) query, a problem broadly falling in the domain of search over encrypted database. Note that  $k$ -NN serves as a fundamental building block in machine learning and data mining, and it has a wide range of applications. These points motivated many studies of  $k$ -NN in the context of DBaaS, with the objective to protect the privacy of data and queries (see Sect. 1.2 for details). In our solution, we manage to offload the computation and communication of end users to the cloud, employing the technique of combining oblivious RAM and MPC by Gordon et al. [13]. At the same time, the complexity for the cloud is  $\tilde{O}(k \log^4 n)$ , which is relatively low.

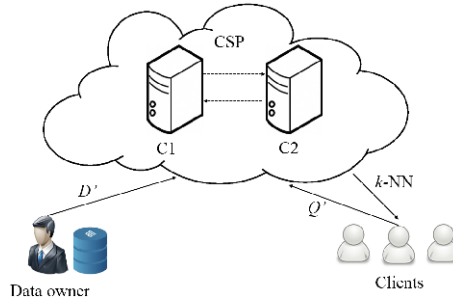
## 1.1 Problem Description

**Privacy-Preserving  $k$ -NN Query:** The data owner has a database  $D$  containing  $n$  points  $d_1, \dots, d_n$  from an  $m$ -dimensional metric space. The client has a query point  $Q$  and wishes to find out the  $k$ -nearest neighbours of  $Q$ . When  $k = 1$  we call it the nearest neighbour (NN) query for short. Throughout this paper, we consider low-dimensional Euclidean space, for example, spatial data points used in location service (usually 2D or 3D points).

Privacy-preserving  $k$ -NN should ensure that the querying client gets no information about the database  $D$  except the result of her query; and the data owner gets no information about the query point and result; and the cloud service provider gets no information about the database, the query point and result.

**Outsourcing Model:** There are conceptually three parties involved: the data owner who possesses the database, the cloud service provider (CSP) who offers storage and computational resources, and the client who launches  $k$ -NN queries. Note that our constructions support multiple clients. Like many other works that considered data outsourcing (see related work), we split the CSP into two non-colluding cloud servers, so as to offload the communication and computation overhead from the client. Refer to Fig. 1 for an illustration of the framework.

We note that the assumption on two non-colluding cloud servers has already been used in a number of previous works on cloud security, e.g., in [4, 5, 9, 25].



**Fig. 1. Privacy-preserving  $k$ -NN query framework.** The data owner uploads her database  $D'$  (some encrypted form of  $D$ ) to the CSP, the query client submits some encrypted form of her query  $Q'$  to CSP and gets the query result, two cloud servers C1 and C2 constitute the CSP.

Moreover, the whole area of information-theoretically secure private information retrieval [6] is based solely on this assumption. Finally, we note that one may allow collusions between the cloud servers for the price of assuming that there are many enough of them, while the size of colluding coalition is small enough. To this end, we point out that our non-colluding model is essentially a special case of the standard threshold assumption widely used in the unconditional MPC and threshold cryptography. However in this work, we focus for simplicity on the case of two cloud servers involved in two-party computation, thus making the threshold equal to 1.

**Security Definition:** We consider the honest-but-curious adversary model, which means that each party follows the protocol but they may attempt to derive extra information of other parties based on the messages gleaned from the protocol execution. We follow the simulation-based security definition in the MPC literature [11].

**Definition 1 (informal).** *We say that a protocol  $\Pi$  securely computes a function if for every party, there exists a polynomial simulator which can simulate the views of that party from her own inputs and outputs.*

## 1.2 Related Work

**Server-aided Secure Computations.** Motivation of our work is similar to the field of server-aided secure computations which are proposed by Kamara et al. [15, 17] to utilize the power of cloud servers to offload huge communication and computation overhead for MPC participants. Follow-up works either construct generic outsourcing MPC protocols [5, 18] or design the protocols for specific applications [1, 16, 24] under the server-aided secure computation model. Our work differs from theirs in two senses: (1) No specific protocols for  $k$ -NN query has been designed under the server-aided secure computation model; (2) All solutions for sever-aided secure computation have only considered offloading the

complexity for the end users, while the burden of cloud servers is still lower bounded by a linear complexity in terms of the input size. However, our protocols also achieve sublinear complexity for the cloud servers.

**Oblivious RAM (ORAM) Based Methods.** Recently Gordon et al. [13] suggested to combine garbled circuits and ORAM to construct amortized sublinear two party computation protocol in the server-client model. Our protocol for privacy-preserving  $k$ -NN query works under this framework. In Gordon et al.'s generic construction they abstract the underlying function as described by an iterative universal next-instruction RAM circuit. However, it turns out to be nontrivial to convert a real-life RAM protocol into the next-instruction function representation. We design *oblivious protocols* for the iterative version of NN query and  $k$ -NN query algorithms based on  $kd$ -tree, which is one of our technical contributions.

Another line of works are oblivious data structures based on ORAM. Wang et al. [26] considered oblivious data structures in the classical ORAM model. They observed that non-recursive ORAM is sufficient to obliviously construct a particular class of data structures which they called bounded-fanout tree. But they leave open the problem of designing the oblivious recursive data structure. We show how to implement this missing piece by turning the recursive data structure *as well as the recursive algorithm* into their iterative counterparts. Our ORAM construction for the  $kd$ -tree is inspired by their observation, but  $kd$ -tree is not exactly a bounded-fanout tree, as in the  $k$ -NN query protocol we need to backtrack from a child node to its ancestor.

**Specific Protocols for Private Preserving  $k$ -NN Query over Cloud.** To securely process  $k$ -NN query on encrypted data, Wong et al. [27] proposed an Asymmetric Scalar-product-Preserving Encryption (ASPE) scheme which preserves the distance between the query point and records in the database. However, the ASPE encryption scheme has been shown to be insecure against chosen plaintext attack by Yao et al. [29]. Zhu et al. [31] updated the construction of Wong et al. to allow privacy-preserving  $k$ -NN query without sharing the key with clients. However, their construction requires the participation of the data owner during the query process. Hu et al. [14] proposed to use privacy homomorphism to address privacy-preserving  $k$ -NN query. But their construction turns out to be insecure for two reasons. Firstly, the privacy homomorphism encryption scheme [8] they chose is insecure [23]. Secondly, even assuming the security of the underlying privacy homomorphism, their construction is still vulnerable to the probing attack [29]. Recently Elmehdwi et al. [9] proposed a protocol for  $k$ -NN query over encrypted database under exactly the same model as ours. They used additive homomorphic encryption to develop a number of two-party computation tools in order to construct a secure  $k$ -NN query protocol. Unfortunately, their construction used the straightforward linear search algorithm so the complexity for cloud server is quasi-linear.

### 1.3 Our Contributions

We propose efficient privacy-preserving protocols for nearest neighbour query and  $k$ -nearest neighbour query on outsourced database in the honest-but-curious adversary model. The proposed protocols have the following advantages: (1) They are designed for more realistic applications where the data owner does not need to trust the querying client; (2) They support multiple querying clients; (3) They provide simulation-based privacy for both data owner and querying clients; (4) They incur a constant overhead for both data owner and querying clients; (5) Computational and communication complexity of the CSP is poly-logarithmic in the size of the database, and in particular our privacy-preserving nearest neighbour query on a size  $n$  low-dimensional database has average complexity  $\tilde{O}(\log^4 n)$ , and our privacy-preserving  $k$ -nearest neighbour query protocol has average complexity  $\tilde{O}(k \log^4 n)$ .

**Table 1.** Comparison of our privacy-preserving  $k$ -NN to existing solutions.

Scheme	C_csp	C_do	C_qc	Honest client	Security		
					A_kpa	A_pa	SB
Wong et al. [27]	$O(n \log k)$	$O(1)$	$O(1)$	yes	no	yes	no
Hu et al. [14]	$O(kn)$	$O(1)$	$O(kn)$	no	no	no	no
Elmehdwi et al. [9]	$\tilde{O}(kn)$	$O(1)$	$O(1)$	no	yes	yes	yes
This work	$\tilde{O}(k \log^4 n)$	$O(1)$	$O(1)$	no	yes	yes	yes

In order to achieve the desired sublinear complexity we construct oblivious  $k$ d-tree and oblivious bounded priority queue using non-recursive Path ORAM with overhead  $\tilde{O}(\log^2 n)$  which may be of independent interest.

For ease of referencing, we compare our privacy-preserving  $k$ -NN query protocol with existing solutions in Table 1. Note that in the table, C\_csp stands for “complexity of CSP”, C\_do for “complexity of data owner”, C\_qc for “complexity of query client”, A\_pka for “against known plaintext attack”, A\_pa for “against probing attack”, and SB for “simulation-based security”. We have simplified the comparison by treating the dimensionality and domain size of the query point as small constants.

**Limitations and Non-Goals:** Similarly to all the previous works employing ORAM to improve efficiency of secure computation, our constructions do not protect against the leakage from the algorithm running time. Also, the reported complexity for our privacy preserving protocols for  $k$ -NN query is the average case complexity. However, to the best of our knowledge, there is no algorithm for  $k$ -NN query with sublinear worst case complexity even in the “plaintext world”.

## 2 Preliminaries

### 2.1 $kd$ -tree

Friedman et al. proposed a data structure called  $kd$ -tree [10] to solve the nearest neighbour problem in the multi-dimensional Euclidean space.  $Kd$ -tree is essentially a hierarchical decomposition of space along different dimensions, and more specially, a  $kd$ -tree is a binary tree where each node is a  $k$ -dimensional point.

We remark that “ $kd$ -tree” stands for  $k$ -dimensional tree, however in this work we use  $k$  as the number of nearest neighbours and our data point has dimensionality  $m$ . The use of  $kd$ -tree is to follow the standard terminology. Readers should not be confused by the different meanings of  $k$ .

We refer the reader to Appendix A for details on the  $kd$ -trees and a toy example. For further reading on  $kd$ -trees, see [20].

### 2.2 ( $k$ )Nearest Neighbour Search in $kd$ -tree

To query the nearest neighbour of a query point  $Q$  from a  $kd$ -tree  $T$ , the algorithm first initiates an estimate point for the nearest neighbour and traverses the tree to update the estimation. A crucial observation is that if there exists a point closer to the query point than the current estimation, it must lie in the hyper-sphere originating from  $Q$  with radius as the distance between  $Q$  and the estimation point. If this hyper-sphere does not intersect with the other half-plane, then the answer can not be in that subtree of the current node (we will call it *intersection rule* for convenience). This observation essentially leads to the reduction of the average complexity from linear to logarithmic.

When it comes to  $k$ -NN query, we need to maintain a bounded priority queue of capacity  $k$  to record the current estimation of  $k$  nearest neighbours instead of a single one. A bounded priority queue (BPQ) is similar to a regular priority queue except that there is a fixed upper bound on the size of the queue. Whenever a new element is to be added to the BPQ, if the queue is full, then the element with the highest priority will be deleted from the BPQ before the insertion occurs.

### 2.3 Path ORAM

Oblivious RAM (ORAM), first proposed by Goldreich and Ostrovsky [12], allows a client to hide her access pattern to the remote server who continuously re-encrypts and shuffles its memory space. Recently lots of constructions improved the efficiency of ORAM algorithms. Among them Path ORAM by Stefanov et al. [22] is the state-of-art implementation with poly-logarithmic overhead.

### 2.4 Garbled Circuits

Due to space limitation, we only present some facts on the known complexity of ORAM and conditional stack simulation using garbled circuits, which will be used to compute the complexity of our oblivious NN query protocol. We refer the

readers to the works by Bellare et al. [3] for a systematic treatment of garbled circuits and [2] for the state-of-the-art implementation.

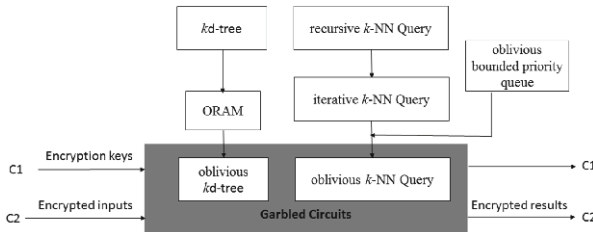
- There are implementations of non-recursive Path ORAM using garbled circuits with complexity  $\tilde{O}(\log^3 N)$  for each ORAM operation, where  $N$  is the size of the data stored on the ORAM [19].
- There are garbled circuits implementations of conditional stack with complexity  $\Theta(\log N)$  for each stack operation, where  $N$  is the size of the stack [30].

### 3 Our Constructions

#### 3.1 Overview

Our constructions use the ORAM based multi-party computation framework [13] to achieve sublinear computational and communication complexity for privacy-preserving  $k$ -NN query using  $kd$ -tree. We construct an oblivious  $kd$ -tree by adapting the oblivious bounded-fanout tree of Wang et al. [26]. However, as explained above,  $kd$ -tree is not an exact bounded-fanout tree so we have to convert the plain *recursive*  $k$ -NN query algorithm into both oblivious and iterative one.

The data owner first pre-processes her database into a  $kd$ -tree  $T$  and uploads  $T$  to an (conceptual) ORAM server. In fact, the ORAM server as well as the ORAM client are simulated by two non-colluding cloud servers C1 and C2. The content of  $T$  is additively shared between C1 and C2. When a client initiates a query with input  $Q$ , she additively shares the data point  $Q$  between C1 and C2. C1 and C2 run the underlying  $k$ -NN query algorithm and access the  $kd$ -tree through an ORAM program. The operation of ORAM access is simulated using secure function evaluation between C1 and C2. The problem of  $k$ -NN query using  $kd$ -tree is a recursive algorithm. We will first turn the recursive algorithm into an iterative one and then make it data oblivious for garbled circuits simulation. After executing garbled circuits simulation, C2 has the encrypted results and C1 holds the decryption keys, they send the encrypted results and the decryption keys to the client respectively. See Fig. 2 for a schematic presentation.



**Fig. 2.** Schematic presentation of our proposal for privacy-preserving  $k$ -NN query.

### 3.2 Iterative NN Query Algorithm

For the ease of presentation, we begin with the NN (Nearest Neighbour) query problem. To turn the recursive NN query algorithm into iterative one, we use a stack  $S$  to store the nodes in the  $kd$ -tree  $T$  and maintain the invariance that the top of  $S$  is the node we are currently visiting. Initially the root of  $T$  is put into  $S$ . In each iteration, we pop out an element from  $S$  and process according to it. If it is the first time we visit this element then we check whether it is a better estimation and push its children to the stack. If it is the second time we visit this element, we call it a return point.<sup>1</sup> For a return point we must decide whether to ignore the other subtree by the *intersection rule*. When  $S$  becomes empty, the algorithm finishes and finds the nearest neighbour of  $Q$ .

### 3.3 Oblivious Protocol for NN Query

We now describe privacy-preserving NN query over outsourced database. The detailed Protocol 1 is deferred to Appendix B.

**Notations and Data Structures.** The data owner will first construct a  $kd$ -tree  $T$  to represent the database  $D$  and to store it on the ORAM server with all necessary pre-processing such as encrypting the data and creating dummy elements. The node of  $T$  is  $node := (id, label, level, data, lid, llabel, rid, rlabel, ln, rn)$ , where “ $id$ ” and “ $label$ ” are the identity of the node and the leaf label in the ORAM tree, “ $level$ ” is the recursive level (mod  $m$ ) of the node in the  $kd$ -tree (i.e., the dimension which it uses to split its children), “ $data$ ” is the actual payload (i.e., the point of  $m$ -dimension), “ $lid$ ”, “ $llabel$ ” are the identity and leaf label of its left child and “ $rid$ ” and “ $rlabel$ ” for the right child respectively. For the sake of convenience, a node also has two additional binary flags “ $ln$ ” and “ $rn$ ”. The flag “ $ln$ ” (“ $rn$ ”) will be set to 1 if the left (right) child is *not* NULL. In the query algorithm, we use a stack  $S$  to remember the algorithm flow. Since all access to  $T$  is done via the ORAM protocol, we will only store the flags and pointers of the node in the stack. An element of the stack  $S$  has data structure as follows:  $e := (value, id, level, label, newlabel, r)$ . The field “ $value$ ” is the  $level$ -th component of “ $node.data$ ” where  $level$  is the dimension used by  $e$  to divide its children, “ $newlabel$ ” is used to assign a fresh uniformly random leaf label when accessing  $e$  using the underlying ORAM protocol. And ‘ $r$ ’ is the binary flag indicating whether  $e$  is a return point as explained in Sect. 3.2. Note that the field “ $value$ ” is only meaningful for a return point when we want to decide whether the potential candidate can be found in the other subtree.

**Oblivious  $kd$ -tree.** We modify the implementation of Wang et al. [26] for an oblivious bounded-fanout tree to construct an oblivious  $kd$ -tree. The method Wang et al. used to construct oblivious bounded-fanout tree structure via non-recursive ORAM is to store the children identities and leaf labels on the parent

---

<sup>1</sup> It means that the algorithm has finished traversing a subtree of that node and returns to the node.

node so that each time the ORAM protocol accesses an element, it gets the identity and label for the next element. However, after each ORAM access of an element, its label changes. To make sure the parent stores the correct leaf labels of its children, Wang et al. uses the pre-selection trick. Specifically, when accessing an element the access protocol chooses a uniformly random label for its child and stores the new child label. This new label will be passed on to the next ORAM access protocol and be used for the child node to update its label. But our application is a little bit different from this assumption. In the iterative algorithm for NN query, we need to deal with the case of a return point, where the access flow is from a node to its ancestor. For example, if  $curr$  is a return point then after accessing  $curr$  through the ORAM protocol, its label changes. Now, the parent of  $curr$  will be storing wrong information about  $curr$ . A possible solution is not to access a return point. But this solution violates the obliviousness property and leaks information about the access pattern. We observe that it is indeed unnecessary to access a return point from the ORAM storage since the return point is only used to check whether we need to search the other subtree and all information needed for this task is available in the stack element. So in the pre-processing phase, data owner also sets up a singular point called *dummy* with identity  $dummyid$  and leaf label  $dummylabel$ . When  $curr$  is a return point the algorithm just accesses the dummy point. Since the dummy element is a singular point, we care neither about the value of its payload nor about its children identities and labels. Thus it is possible to access the dummy element multiple times. Because of the obliviousness of ORAM, it is hard to distinguish between accessing real elements and a dummy element multiple times.

**Oblivious NN Query Protocol.** When a client launches an NN query, she shares the query point  $Q$  to C1 and C2. C1 and C2 then simulate the NN query algorithm over a non recursive Path ORAM using garbled circuits. The iterative NN algorithm iteratively pops an element from the stack and proceeds according to different cases of it. So the algorithm flow depends on the top element of the stack. We use an oblivious stack to make the whole protocol data oblivious. For an oblivious stack  $S$ , there are two operations, **conpush** and **conpop**.

$S.conpush(p, e)$  pushes the element  $e$  to  $S$  iff the predicate  $p$  is true.

$S.conpop(p)$  pops out the top element of  $S$  iff the predicate  $p$  is true.

Protocol 1 describes the oblivious query procedure for NN query using Path ORAM. The ORAM protocol has two operations **Access** and **Eviction**.  $(node, llabel^*, rlabel^*) \leftarrow \mathbf{ORAM.Access}(id, label, label^*)$  returns a node in  $T$  with identity  $id$  on the path from the ORAM tree root to leaf  $label$  and two new labels  $(llabel^*, rlabel^*)$  for its children and updates the leaf to  $label^*$ . Note that the children labels in the returned  $node$  are the old versions  $(llabel, rlabel)$ .

**ORAM.Eviction()** does the cleanup after each  $\mathbf{ORAM.Access}()$ .

Another frequently used operation is  $x \leftarrow \mathbf{MUX}(a, b, c)$ , where  $a$  is a bit value and  $x = b$  if  $a = 1$  and  $x = c$  otherwise.



### 3.4 Garbled Circuits Simulation

An essential complement of Protocol 1 is the simulation of the Path ORAM using garbled circuits. Here, we discuss this point.

Since garbled circuits evaluation is equivalent to oblivious protocol, we do not present circuit description for the basic primitives such as MUX gate, comparison gate etc., in our protocols. Keller and Scholl [19] has implemented Path ORAM for MPC using the SPDZ protocol of Damgard et al. Though different from garbled circuits, the protocol logic is the same. We will substitute the core of their implementation by garbled circuits of Bellare et al. [2] to implement the oblivious NN query protocol. Besides the basic primitives and ORAM simulations we also need to simulate a conditional stack using garbled circuits. We can again use ORAM to construct a conditional stack. However, the implementation of Zahur and Evans [30] using garbled circuits with logarithmic overhead turns out to be more efficient and simple.

## 4 Extension to Privacy-Preserving $k$ -NN Query

In order to implement a  $k$ -NN query, we need to construct a conditional bounded priority queue. With a garbled circuits implementation of conditional bounded priority queue it is straightforward to adapt Protocol 1 into a privacy-preserving  $k$ -NN query protocol.

Wang et al. [26] constructed an oblivious priority queue using non-recursive Path ORAM, but their construction used a client cache and branching at the client side so that it is not easy to translate it into MPC setting. Keller and Scholl constructed an oblivious priority queue for MPC use, but their improved implementation (with overhead  $O(\log^2 N)$ ) uses the recursive SCSL ORAM [21]. In this section will show how to implement an oblivious bounded priority queue using non-recursive Path ORAM with overhead  $O(\log^2 N)$ , where  $N$  is the size of the queue.

In our oblivious bounded priority queue, the bound on the size of the queue is  $k$ . The classical construction of priority queue uses complete binary heap to store the queue, and the heap is represented by an array  $A$  of size  $k$ . For ease of presentation and without loss of generality we assume  $k = 2^h - 1$  for some integer  $h$ . Denote by  $s$  the actual size of the queue. When we dequeue an element from the priority queue, we output the root of the heap, i.e.,  $A[1]$  and put the last element  $A[s]$  to the root and adjust the heap by a BubbleDown operation. We use a simple trick to avoid the dependence of the algorithm flow on the size  $s$ . We will always assume the priority queue is at full capacity  $k$ . Initially, the priority queue is filled with dummy elements with priority  $\infty$ . The bounded priority queue support two operations **Con\_Dequeue**( $p, e$ ) and **Con\_Enqueue**( $p, e$ ).

**Con\_Dequeue**( $p$ ). The conditional dequeue protocol will extract the element with highest priority and remove it if and only if the predicate  $p$  is true.

**Con\_Enqueue**( $p, e$ ). The conditional enqueue protocol will add an element with priority  $e$  to the priority queue if and only if the predicate  $p$  is true.

Since we add dummy elements with priority  $\infty$ , the queue is always at full capacity. We will always perform a conditional dequeue operation followed by a conditional enqueue operation using the same predicate  $p$ . This observation gives us a simple algorithm for performing the conditional dequeue and conditional enqueue together which also avoids the effort of obviously reading the path from root to  $A[s]$ . The idea is that adding an element to a full bounded priority queue needs to first dequeue the element with the highest priority and then to insert the new element to the queue with some adjustment to maintain the heap property. To dequeue the element with the highest priority we simply read the root of the binary heap and remove it. To insert the new element, we replace the root of the binary heap with the new element and use the standard BubbleDown algorithm to maintain the heap property. See Appendix C for the detailed protocol for `Con_Dequeue_And_Enqueue`( $p, e$ ).

## 5 Security and Complexity Analysis

### Security of oblivious NN protocol (i.e., Protocol 1).

**Theorem 1.** *When implemented using garbled circuits providing simulation-based privacy [3], Protocol 1 securely computes the nearest neighbour problem.*

*Proof (sketch).* Due to space limitation, we only give some intuition regarding the security proof. The high-level idea is to use composite theory which states that if each component of a scheme is securely implemented then the whole scheme is also securely implemented. In Protocol 1, all the inputs and intermediate variables are shared between two non-colluding cloud servers C1 and C2, and the only public information is the leaf label of some elements in the ORAM tree. When we access an element in the database, we access it via ORAM. Since we instantiate the ORAM using Path ORAM, which has statistical security, the oblivious protocol achieves simulation-based security if it is implemented using garbled circuits providing simulation-based privacy.

**Complexity of Oblivious NN Protocol.** Recall that the data structure  $kd$ -tree is mainly useful for low dimensional data, and we also assume that each dimension of a point belongs to a small range, say the longitude and latitude of a location coordinates which can be presented using a small number of bits. By the above setting, we can treat the primitive gates concerning the payload data, such as addition gate, comparison gate and even the INDEX() gate as of constant complexity. In Protocol 1, it is easy to see that at any point of the protocol flow the size of our conditional stack is bounded by  $O(\log n)$  ( $n$  is the number of points in the database  $D$ ) and ORAM.Access() and ORAM.Eviction() dominate the complexity. Since ORAM operation can be simulated with overhead  $\tilde{O}(\log^3 n)$  as explained in Sect. 2.4, and average size of the while loop is  $O(\log n)$ , we get a privacy-preserving NN query algorithm with average complexity  $\tilde{O}(\log^4 n)$ .

**Security of Oblivious  $k$ -NN Query Protocol.** The security of oblivious  $k$ -NN query follows easily from that of the oblivious NN query protocol. We just note that our oblivious bounded priority queue in Appendix C is simulation based secure, since we implement it using Path ORAM and simulate the operation using garbled circuits with simulation-based privacy.

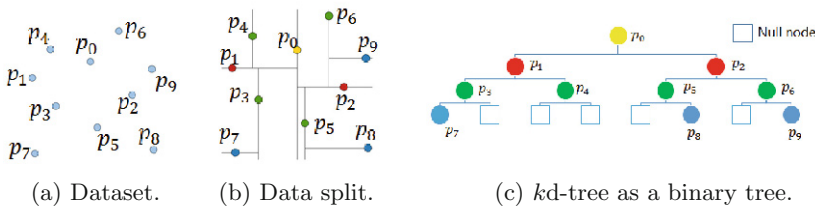
**Complexity of Oblivious  $k$ -NN Query Protocol.** Algorithm based on  $kd$ -tree for computing the  $k$ -NN query of a database with  $n$  records has complexity  $O(k \log n)$  if we assume the dimension of each records is a small constant. By the fact that we can implement an oblivious  $kd$ -tree of size  $n$  using ORAM with overhead  $\tilde{O}(\log^3 n)$  and an oblivious bounded priority queue of size  $k$  using ORAM with overhead  $\tilde{O}(\log^3 k)$ , the complexity of our oblivious  $k$ -NN query is  $\tilde{O}(k \log^4 n)$ .

## Appendix

### A Toy Example of $kd$ -tree Construction and Iterative NN Query Algorithm

We provide a toy example for constructing a  $kd$ -tree and use it to iteratively query the nearest neighbour of a point.

We take 2D points as an example for illustrating how to construct a  $kd$ -tree from a static dataset. The dataset contains 10 2D points  $(p_0, \dots, p_9)$ . See Fig. 3a for the geometric display of the dataset. Refer to Fig. 3b for the illustration. Figure 3c shows the representation of the  $kd$ -tree as a binary tree.



**Fig. 3.** Construction of  $kd$ -tree.

We use the toy example to explain how the iterative NN query algorithm works.

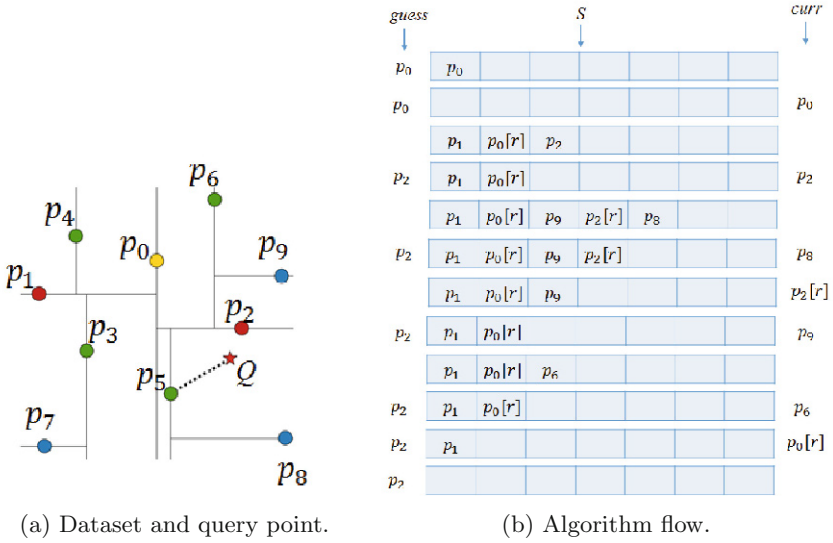


Fig. 4. Iterative NN query algorithm using  $kd$ -tree.

Figure 4a illustrates the dataset and the query point  $Q$ . Let  $guess$  be the estimation for the nearest neighbour,  $S$  be the stack used in the algorithm,  $curr$  is the current point we are visiting. We also use  $p[r]$  to denote that the point  $p$  is a return point. See Fig. 4b for the change of the stack elements during the whole query processing.

## B Oblivious Protocol for NN Query

Protocol 1 (see below) is our main protocol for the oblivious nearest neighbour query problem. This protocol is run between two non-colluding cloud servers. We defer its detailed explanation to the full version of this paper.

## C Oblivious Bounded Priority Queue

The basic unit of storage on the ORAM is called **node**. Each node is constructed as  $node := (id, label, key, lid, llabel, rid, rlabel, )$  where “ $id$ ” and “ $label$ ” are the identity of the node and the leaf label in the ORAM tree, “ $key$ ” is the priority of the node, “ $lid$ ”, “ $llabel$ ” are the identity and the leaf label of its left child, and “ $rid$ ” and “ $rlabel$ ” for the right child respectively. From Protocol 2 (see below) we can observe that for each **Con\_Dequeue\_And\_Enqueue()** operation we have  $3 \log k$  ORAM operation, and each ORAM operation can be simulated using garbled circuits with complexity  $\tilde{O}(\log^3 k)$ . Therefore, one oblivious such operation has complexity  $\tilde{O}(\log^4 k)$ .

---

**Protocol 1.** Oblivious protocol for NN query
 

---

**Require:** The kd-tree  $[T]$ ,  $[Q]$  – the query point,  $[rootid]$ – identity of  $T$ ,  $[rootlabel]$  – leaf label of  $T$  in the ORAM tree.

**Ensure:** Output the nearest neighbour of  $Q$ .

```

1:  $[rootlabel^*] \leftarrow [\text{UniformRandom}]$ 
2:  $([node], [llabel^*], [rlabel^*]) \leftarrow \text{ORAM.Access}([rootid], [rootlabel], [rootlabel^*])$ 
3: ORAM.Eviction()
4:  $[guess] \leftarrow [node]$ 
5:  $[bestdist] \leftarrow \text{dis}([node.data], [Q])$  ▷ initiate guess to the root of  $T$ 
6:  $[e1.value] \leftarrow [0], [e1.id] \leftarrow [node.lid]$ 
7:  $[e1.level] \leftarrow [2], [e1.label] \leftarrow [node.llabel]$ 
8:  $[e1.newlabel] \leftarrow [llabel^*], [e1.r] \leftarrow [0]$  ▷  $e1$  is the left child
9:  $[e2.value] \leftarrow [node.data_1], [e2.id] \leftarrow [node.id]$ 
10:  $[e2.level] \leftarrow [1], [e2.label] \leftarrow [node.label]$ 
11:  $[e2.newlabel] \leftarrow [\text{UniformRandom}], [e2.r] \leftarrow [1]$  ▷  $e2$  is the return point
12:  $[e3.value] \leftarrow [0], [e3.id] \leftarrow [node.rid]$ 
13:  $[e3.level] \leftarrow [2], [e3.label] \leftarrow [node.rlabel]$ 
14:  $[e3.newlabel] \leftarrow [rlabel^*], [e3.r] \leftarrow [0]$  ▷  $e3$  is the right child
15:  $[inleft] \leftarrow ([q_1] \leq [node.data_1])$  ▷  $inleft = 1$  if  $Q$  lies in the left half-plane
16:  $S.\text{conpush}(\text{MUX}([inleft], [node.rn], [node.ln]), \text{MUX}([inleft], [e3], [e1]))$ 
17:  $S.\text{conpush}([node.ln][node.rn], [e2])$ 
18:  $S.\text{conpush}(\text{MUX}([inleft], [node.ln], [node.rn]), \text{MUX}([inleft], [e1], [e3]))$ 
19: while  $S$  is not empty do
20:    $curr \leftarrow S.\text{pop}()$ 
21:    $[thisid] \leftarrow \text{MUX}([curr.r], [dummyid], [curr.id])$ 
22:    $[thislabel] \leftarrow \text{MUX}([curr.r], [dummylabel], [curr.label])$ 
23:    $(node, llabel^*, rlabel^*) \leftarrow \text{ORAM.Access}([thisid], [thislabel], [curr.newlabel])$ 
24:   ORAM.Eviction()
25:    $[dummylabel] \leftarrow \text{MUX}([curr.r], [curr.newlabel], [dummylabel])$ 
26:    $[i] \leftarrow [curr.level]$  ▷ record the level of  $node$  as  $i$ 
27:    $[val] \leftarrow \text{INDEX}([node.data], [i])$  ▷ record the  $i$ -th component of  $node.data$ 
28:    $S.\text{conpop}([curr.r] \cdot (([curr.value] - [q_i]) \geq [bestdist]))$ 
29:    $[currentdist] \leftarrow \text{dis}([node.data], [Q])$ 
30:    $[bestdist] \leftarrow \text{MUX}([currentdist] < [bestdist], [currentdist], [bestdist])$ 
31:    $[e1.value] \leftarrow [0], [e1.id] \leftarrow [node.lid]$ 
32:    $[e1.level] \leftarrow ([i] + 1) \bmod m, [e1.label] \leftarrow [node.llabel]$ 
33:    $[e1.newlabel] \leftarrow [llabel^*], [e1.r] \leftarrow [0]$  ▷  $e1$  is the left child
34:    $[e2.value] \leftarrow [val], [e2.id] \leftarrow [node.id]$ 
35:    $[e2.level] \leftarrow [i], [e2.label] \leftarrow [node.label]$ 
36:    $[e2.newlabel] \leftarrow [\text{UniformRandom}], [e2.r] \leftarrow [1]$  ▷  $e2$  is the return point
37:    $[e3.value] \leftarrow [0], [e3.id] \leftarrow [node.rid]$ 
38:    $[e3.level] \leftarrow ([i] + 1) \bmod m, [e3.label] \leftarrow [node.rlabel]$ 
39:    $[e3.newlabel] \leftarrow [rlabel^*], [e3.r] \leftarrow [0]$  ▷  $e3$  is the right child
40:    $[inleft] \leftarrow ([q_1] \leq [node.data_1])$  ▷  $inleft = 1$  if  $Q$  lies in the left half-plane
41:    $S.\text{conpush}([curr.r] \oplus 1) \cdot \text{MUX}([inleft], [node.rn], [node.ln]), \text{MUX}([inleft], [e3], [e1]))$ 
42:    $S.\text{conpush}([curr.r] \oplus 1) \cdot [node.ln][node.rn], [e2])$ 
43:    $S.\text{conpush}([curr.r] \oplus 1) \cdot \text{MUX}([inleft], [node.ln], [node.rn]), \text{MUX}([inleft], [e1], [e3]))$ 
44: end while

```

---

---

**Protocol 2. Con\_Dequeue\_And\_Enqueue**( $[p], [e], [rootid], rootlabel$ )
 

---

**Require:**  $[p]$  – the predicate,  $[e]$  – the new element to be inserted,  $[rootid]$  – identity of the ORAM tree root,  $rootlabel$  – the leaf label of root.

**Ensure:** Remove the element with highest priority and insert  $e$  if and only if  $p = 1$ .

```

1:  $[node] \leftarrow \text{ORAM.Read\_and\_Remove}([rootid], rootlabel)$ 
2:  $[node.key] \leftarrow \text{MUX}([p], [e], [node.key])$ 
3:  $[lchild] \leftarrow \text{ORAM.Read\_and\_Remove}([node.lid], node.llabel)$ 
4:  $[rchild] \leftarrow \text{ORAM.Read\_and\_Remove}([node.rid], node.rlabel)$ 
5:  $[left\_is\_bigger] \leftarrow ([lchild.key] > [rchild.key])$ 
6:  $[max\_key] \leftarrow \text{MUX}([left\_is\_bigger], [lchild.key], [rchild.key])$ 
7:  $[node.key] \leftarrow \text{MUX}([max\_key] > [node.key], [max\_key], [node.key])$ 
8:  $[max\_key] \leftarrow \text{MUX}([max\_key] > [node.key], [node.key], [max\_key])$ 
9:  $[lchild.key] \leftarrow \text{MUX}([left\_is\_bigger], [max\_key], [lchild.key])$ 
10:  $[lchild.key] \leftarrow \text{MUX}([left\_is\_bigger], [rchild.key], [max\_key])$ 
11:  $[newlabel, node.llabel, node.rlabel] \leftarrow [\text{UniformRandom}]$ 
12:  $[rootlabel] \leftarrow [newlabel]$  ▷ remember the new  $rootlabel$ 
13: ORAM.Add( $[node], [newlabel]$ )
14: ORAM.Add( $[lchild], [node.llabel]$ ), ORAM.Add( $[rchild], [node.rlabel]$ )
15:  $[nextid] \leftarrow \text{MUX}([left\_is\_bigger], [node.lid], [node.rid])$ 
16:  $[nextlabel] \leftarrow \text{MUX}([left\_is\_bigger], [node.llabel], [node.rlabel])$ 
17: while  $i$  from 2 to  $h - 1$  do
18:    $[node] \leftarrow \text{ORAM.Read\_and\_Remove}([nextid], nextlabel)$ 
19:    $[lchild] \leftarrow \text{ORAM.Read\_and\_Remove}([node.lid], node.llabel)$ 
20:    $[rchild] \leftarrow \text{ORAM.Read\_and\_Remove}([node.rid], node.rlabel)$ 
21:    $[left\_is\_bigger] \leftarrow ([lchild.key] > [rchild.key])$ 
22:    $[max\_key] \leftarrow \text{MUX}([left\_is\_bigger], [lchild.key], [rchild.key])$ 
23:    $[node.key] \leftarrow \text{MUX}([max\_key] > [node.key], [max\_key], [node.key])$ 
24:    $[max\_key] \leftarrow \text{MUX}([max\_key] > [node.key], [node.key], [max\_key])$ 
25:    $[lchild.key] \leftarrow \text{MUX}([left\_is\_bigger], [max\_key], [lchild.key])$ 
26:    $[lchild.key] \leftarrow \text{MUX}([left\_is\_bigger], [rchild.key], [max\_key])$ 
27:    $[newlabel, node.llabel, node.rlabel] \leftarrow [\text{UniformRandom}]$ 
28:   ORAM.Add( $[node], [newlabel]$ )
29:   ORAM.Add( $[lchild], [node.llabel]$ )
30:   ORAM.Add( $[rchild], [node.rlabel]$ )
31:    $[nextid] \leftarrow \text{MUX}([left\_is\_bigger], [node.lid], [node.rid])$ 
32:    $[nextlabel] \leftarrow \text{MUX}([left\_is\_bigger], [node.llabel], [node.rlabel])$ 
33: end while

```

---

## References

1. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 48–59. ACM (2010)
2. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. *SP* **2013**, 478–492 (2013)
3. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. *CCS* **2012**, 784–796 (2012)
4. Bugiel, S., Nurnberger, S., Sadeghi, A., Schneider, T.: Twin clouds: an architecture for secure cloud computing. In: Workshop on Cryptography and Security in Clouds (WCSC 2011) (2011)
5. Carter, H., Mood, B., Traynor, P., Butler, K.R.: Secure outsourced garbled circuit evaluation for mobile devices. In: USENIX Security, pp. 289–304 (2013)
6. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* **45**(6), 965–981 (1998). <http://doi.acm.org/10.1145/293347.293350>
7. Curino, C., et al.: Relational cloud: a database-as-a-service for the cloud. In: 5th Biennial Conference on Innovative Data Systems Research, CIDR, pp. 9–12 (2011)
8. Domingo-Ferrer, J.: A provably secure additive and multiplicative privacy homomorphism. In: Chan, A.H., Gligor, V.D. (eds.) *ISC 2002*. LNCS, vol. 2433, pp. 471–483. Springer, Heidelberg (2002)
9. Elmehdwi, Y., Samanthula, B., Jiang, W.: Secure k-nearest neighbor query over encrypted data in outsourced environments. *ICDE* **2014**, 664–675 (2014)
10. Friedman, J.H., Baskett, F., Shustek, L.J.: An algorithm for finding nearest neighbors. *IEEE Trans. Comput.* **24**(10), 1000–1006 (1975)
11. Goldreich, O.: *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge University Press, New York (2009)
12. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. *J. ACM* **43**(3), 431–473 (1996)
13. Gordon, S.D., Katz, J., Kolesnikov, V., Krell, F., Malkin, T., Raykova, M., Vahlis, Y.: Secure two-party computation in sublinear (amortized) time. *CCS* **2012**, 513–524 (2012)
14. Hu, H., Xu, J., Ren, C., Choi, B.: Processing private queries over untrusted data cloud through privacy homomorphism. In: *ICDE 2011*, pp. 601–612. IEEE (2011)
15. Kamara, S., Mohassel, P., Raykova, M.: Outsourcing multi-party computation. In: *IACR Cryptology ePrint Archive 2011/272* (2011)
16. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.: Scaling private set intersection to billion-element sets. In: Christin, N., Safavi-Naini, R. (eds.) *FC 2014*. LNCS, vol. 8437, pp. 193–213. Springer, Heidelberg (2014)
17. Kamara, S., Mohassel, P., Riva, B.: Salus: a system for server-aided secure function evaluation. In: Proceedings of the 2012 ACM conference on Computer and communications security, pp. 797–808. ACM (2012)
18. Kamara, S., Raykova, M.: Secure outsourced computation in a multi-tenant cloud. In: Workshop on Cryptography and Security in Clouds (2011)
19. Keller, M., Scholl, P.: Efficient, oblivious data structures for MPC. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014, Part II*. LNCS, vol. 8874, pp. 506–525. Springer, Heidelberg (2014)
20. Samet, H.: *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Longman Publishing Co., Inc., Boston (1990)

21. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with  $O((\log N)^3)$  worst-case cost. In: Wang, X., Lee, D.H. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011)
22. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path ORAM: an extremely simple Oblivious RAM protocol. CCS **2013**, 299–310 (2013)
23. Wagner, D.: Cryptanalysis of an algebraic privacy homomorphism. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 234–239. Springer, Heidelberg (2003)
24. Wang, C., Ren, K., Wang, J.: Secure and practical outsourcing of linear programming in cloud computing. In: INFOCOM, 2011 Proceedings IEEE, pp. 820–828. IEEE (2011)
25. Wang, J., Ma, H., Tang, Q., Li, J., Zhu, H., Ma, S., Chen, X.: Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. Comput. Sci. Inf. Syst. **10**(2), 667–684 (2013)
26. Wang, X., Nayak, K., Liu, C., Shi, E., Stefanov, E., Huang, Y.: Oblivious data structures. In: CCS 2014 (2014)
27. Wong, W.K., Cheung, D.W.I., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: SIGMOD 2009, pp. 139–152 (2009)
28. Yao, A.C.: Protocols for secure computations. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE (1982)
29. Yao, B., Li, F., Xiao, X.: Secure nearest neighbor revisited. ICDE **2013**, 733–744 (2013)
30. Zahur, S., Evans, D.: Circuit structures for improving efficiency of security and privacy tools. SP **2013**, 493–507 (2013)
31. Zhu, Y., Xu, R., Takagi, T.: Secure k-nn computation on encrypted cloud data without sharing key with query users. In: Proceedings of the 2013 International Workshop on Security in Cloud Computing, pp. 55–60. ACM (2013)



# Reversible Data Hiding for Encrypted Images Based on Statistical Learning

Zhen Li<sup>1</sup> and Wei Wu<sup>2</sup>(✉)

<sup>1</sup> Infocomm Security Department, Institute for Infocomm Research,  
1 Fusionopolis Way, #21-01 Connexis, Singapore 138632, Singapore  
liz@i2r.a-star.edu.sg

<sup>2</sup> Fujian Provincial Key Laboratory of Network Security and Cryptology,  
School of Mathematics and Computer Science, Fujian Normal University,  
Fuzhou 350117, Fujian, China  
weiwu81@gmail.com

**Abstract.** In this paper, we propose a novel reversible data hiding (RDH) approach for encrypted images by using statistical learning. To hide the data, a new random permutation algorithm is proposed using a high-speed stream cipher to secure the data hiding process. A secret message is embedded into the permuted image blocks based on a checkerboard pattern, by modifying the least significant encrypted bits. To detect the hidden data, prior works utilize a single spatial correlation. In contrast, our approach is novel in that it uses a high-dimensional statistical feature vector upon which a new boosting algorithm for high reversibility is proposed. A complete encoding and decoding procedure of RDH for encrypted images is elaborated. The experimental results show that the proposed method can detect secret message bits and restore the original image simultaneously with 100 % reversibility with a higher capacity, significantly outperforming the state-of-the-art RDH methods on encrypted images.

## 1 Introduction

Data hiding is a technique to embed the *payload*, i.e. secret message, into a *cover* media by slightly altering the digital content, to generate the *stego* media. The original digital multimedia as the cover can be different kinds of signals, such as image, audio and text. When data hiding is performed in a reversible manner, the reversible data hiding (RDH) methods [1–5] guarantee that when the payload is extracted from the stego-image, the cover image can be exactly restored. In contrast, in non-reversible data hiding schemes, the distortion caused by payload embedding cannot be completely removed to restore the cover image. Therefore, RDH is required in many applications, e.g. military and medical images where the confidentiality and the integrity of the cover images are highly required after extraction of the hidden data.

Nowadays, digital multimedia data has been largely distributed over the Internet without much protection, causing privacy concerns such as leaking

visual content of photos to unauthorized viewers or surveillance under certain organizations. Accordingly, some applications have been developed to encrypt images before uploading to Internet and, increasingly more image data have been secured by encryption in social media. For example, third-party applications that support file encryption for Google drive [6,7] and Facebook [8], as well as the native application for data protection in Google Drive [9]. In addition, most of military and medical images are encrypted to protect the confidentiality.

By integrating encryption to protect the content, RDH on encrypted images has become a hot topic in the research area [10–14]. RDH in encrypted domain has many applications, for example, covert communication by embedding secret message, image annotation by embedding the description of the visual content, and image content authentication by embedding the fingerprint. When the image owner encrypts the original cover image to generate the encrypted version, the cover image which conveys visual content is converted to a visually meaningless version. Without the encryption key, an adversary is not able to restore the image content. When a secret message is hidden into the encrypted version, resulting in a *stego-encrypted* version, the data hider is not aware of the visual content in the cover image. At the receiver's side, the image owner can decrypt the stego-encrypted version to a stego-image that contains the visual content with the secret message hidden, without knowing the data hiding key. On the other hand, by further using the data hiding key, the cover image can be restored simultaneously when the secret message is detected. Compared to RDH methods on unencrypted images, the difficulty of RDH on encrypted images lies in the fact that the cover image content is unknown to the data hider, so the perfect reversibility cannot be guaranteed.

In order to address such an issue, we propose a new RDH method for encrypted images with high reversibility based on statistical learning. Specifically, secret message is embedded in the encrypted image blocks in a permuted order, which is achieved by designing a new random permutation algorithm based on Black Dragon cipher in the encoding process, and a high reversibility of the cover image is achieved by developing a new classifier based on a high-dimensional feature of the hypothesis image blocks. This is a novel RDH method with perfect reversibility on encrypted images with reasonable capacity based on statistical learning techniques that is deeply applied to this field for the first time.

The rest parts of the paper are organized as follows. Section 2 briefs the related work. Sections 3 and 4 describes the encoding process and decoding process of proposed RDH method for encrypted images, respectively, followed by the experimental results in Sect. 5. Finally, Sect. 6 concludes the paper.

## 2 Related Work

In 2008, Puech *et al.* [10] proposed a pioneer work of RDH method for encrypted images. At first, the original image is divided into disjoint blocks and each block is encrypted by AES in ECB mode. Then a randomly chosen bit of a pixel in

each encrypted block is substituted by one secret message bit, resulting in either an altered block or an unchanged block. The hypothesis with lower deviation of decrypted pixel values is used to restore the original cover image block. Since this method is based on block cipher, a single error in the deviation comparison will generate a whole block of random noise in the restored image, which is not visually tolerable.

In 2011, Zhang *et al.* proposed another RDH method for images encrypted by stream ciphers [11]. By dividing the encrypted image into disjoint blocks, one message bit is embedded in each block by flipping the 3 LSBs of the randomly selected pixels given by the data hiding key. A first-order smoothness function is used to judge the right hypothesis of the embedded bit. However, the lossless recovery of the image and hidden data is not satisfactory. For example, when block size is  $8 \times 8$ , the accuracy of the detected hidden message bits is even lower than 90%. In order to improve the situation, Hong *et al.* [12] use side information to improve the smoothness function. However, the improvement of the detection accuracy of hidden message bits, i.e. the reversibility of the original cover image, is only around 2% for most test images.

The above three methods rely on the spatial correlation, therefore image decryption should be performed ahead of hidden message detection. Some other methods are also proposed to enable hidden message detection before image decryption by reserving space for side information [13, 14]. However, the core technique of secret message detection is still based on the evaluation of deviation or smoothness of the hypothesis image blocks either embedded by 0 or 1.

The main focus of this work is the accuracy of the message bit detection through the core technique of RDH in encrypted messages. A much higher reversibility of the cover image will be achieved by statistically learning a classifier for the hypothesis blocks: original “natural” image blocks against the “unnatural” altered image blocks. Since the message bits embedded in the stego-image blocks are binary, a two-class classifier is required. An effective algorithm of two-class classifier is Adaboost [15]. The main idea is to construct a strong classifier through the combination of weak classifiers, where each weak classifier is derived by each single feature. The adaboosting procedure iteratively trains each weak classifier on weighted training samples with higher weights assigned to the currently misclassified ones. The Adaboost classifier can be developed based on additive logistic regression and empirical loss [16, 17]. In this work, a new Adaboost algorithm will be developed and applied to a high-dimensional feature of hypothesis blocks in order to correctly identify the embedded message bits in the stego-image. Reversible data hiding for encrypted images is crucial for covert communication in military and private data exchange.

### 3 Encoding Process of Data Hiding

The encoding process is composed of two steps which are encryption and data hiding. The overview of the encoding process is illustrated in Fig. 1.

Each non-overlapping block of the cover image is encrypted using a stream cipher called Black Dragon (BD), producing the cipher-blocks that the secret

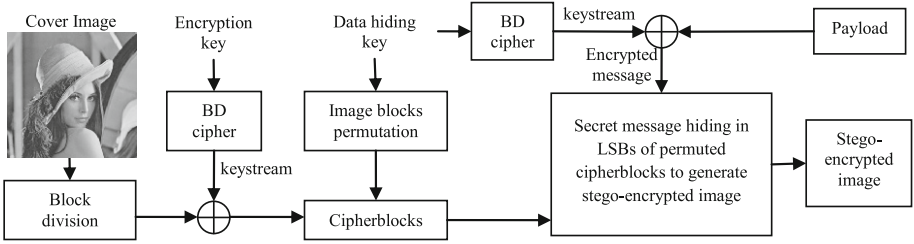


Fig. 1. Overview of the encoding process

message will be embedded in. Anyone without knowing the encryption key, including both the data hider and the malicious attacker, are not able to obtain the original visual content. Before hiding data, the usage of BD cipher is manifold: to randomly permute the ordering of the cipher-blocks in order to secure the subsequent data hiding process, as well as to secure the secret message before embedding into cipher-blocks. In such a way, if an inspector tries to find the message hidden in the stego-encrypted image, it is unlikely to succeed without knowing the data hiding key which generates the embedding order and the encryption key that encrypts the secret message. The encryption key for encrypting the cover image and the data hiding key for re-ordering the cipher-blocks and encrypting secret message are different. Without encryption key, the data hider can embed the payload in the encrypted image but is not able to see the image content. On the other hand, without data hiding key the owner of the image can still decrypt to obtain a stego-image and see the image content but cannot extract the secret message embedded. To hide data, the encrypted message bits are embedded in the least significant bits (LSB) of the cipher-blocks in the permuted order.

### 3.1 Image Encryption Using Black Dragon Cipher

Assume that the cover image is an uncompressed grayscale image with  $M \times N$  8-bit pixels and the pixel values are from set  $P = \{0, 1, \dots, 255\}$ . The image is divided into non-overlapping image blocks of  $n \times n$  pixels  $\mathbf{B}_i = \{p_{i,j}\}$ , where  $(i, j)$  indicates the pixel indexes and  $p_{i,j}$  is the pixel value. The 8 bits of each pixel  $p_{i,j}$  are as follow:  $b_{i,j,r} = (p_{i,j} \gg r) \bmod 2$ ,  $r = 0, 1, \dots, 7$ , where  $\gg$  is the right bit shift operation and  $b_{i,j,r}$  is the  $r$ -th least significant bit of the pixel  $p_{i,j}$ .

In encryption step, the each bit of the cover image pixels are encrypted by the keystream of the stream cipher to generate the ciphertext  $c_{i,j,r} = b_{i,j,r} \oplus z_{i,j,r}$ , where  $c_{i,j,r}$  is the encrypted image pixel bits,  $\oplus$  is the exclusive-or operation, and  $z_{i,j,r}$  is the corresponding bit of the keystream. Any binary additive stream cipher can be used for these tasks. Advanced Encryption Standard (AES) is a common cipher, however, it is advisable to use a cipher that is faster or more secure than AES. We use Black Dragon [18], which is a modern and versatile word-based stream cipher that offers 256 bits of security. It is approximately three times faster than the AES with comparable security.

### 3.2 Random Permutation of Blocks

The indexes of the cover image blocks, i.e.  $1, 2, \dots, n^2$ , are randomly permuted before embedding the payload into the cover image. This is to guarantee the security of the data hiding process. Without knowing the data hiding key  $key_h$ , one can only detect the hidden bits in a permuted order which carries no useful information. In order to make the permutation, a new random permutation algorithm for the re-ordering of image blocks is proposed based on the BD cipher for which no successful attacks are reported in literature.

---

#### Algorithm 1. Image Block Order Permutation using Black Dragon Cipher

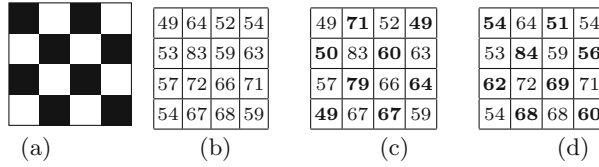
---

1. Input: Set indexes  $1, 2, \dots, n^2$ , where  $n$  is the number of blocks in one dimension of the 2-D image, and hiding key  $key_h$  by data hider;
  2. Divide all indexes into groups of  $\lfloor \frac{256}{\log_2(n^2)} \rfloor$  indexes:  $\mathbf{index} = \{\mathbf{group}_j\}$ , where
 
$$\mathbf{group}_j = \left\{ index_{j,1}, \dots, index_{j, \lfloor \frac{256}{\log_2(n^2)} \rfloor} \right\}, j = 1, 2, \dots, \lceil \frac{n^2}{256} \rceil;$$
  3. Concatenate the indexes in each group to form a 256-bit  $IV_j$ ;
  4. For  $j$  from 1 to  $\lceil \frac{n^2}{256} \rceil$ , compute the  $\left( 64 \times \lfloor \frac{256}{\log_2(n^2)} \rfloor \right)$ -bit keystream of each concatenated 256-bit key  $KS_j = Dragon(key_h, IV_j)$ , and divide each  $KS_j$  into  $\lfloor \frac{256}{\log_2(n^2)} \rfloor$  times of 64-bit keystreams:  $ks_{j,1}, \dots, ks_{j, \lfloor \frac{256}{\log_2(n^2)} \rfloor}$ ;
  5. Sort all the 64-bit keystreams in all groups to obtain the new set of indexes:  $\mathbf{randindex} = Heapsort(\mathbf{ks})$ ;
  6. Output: The randomly permuted indexes:  $\{randindex_i\}, i = 1, 2, \dots, n^2$ .
- 

At first, the bits of the indexes are concatenated to form a 256-bit initialization vector. After applying Black Dragon cipher with the 256-bit key, the generated keystream bits are divided into multiples of 64 bits, each can be regarded as a random mapping from each index. By sorting the 64-bit keystreams, the new order of the indexes are obtained which will be used in data hiding process. The detailed procedure is formalized in Algorithm 1. Since collision of the 64-bit BD keystreams have a extremely low probability with small number of blocks, the case that the randomly permuted indexes having identical values will not affect the data hiding procedure. The heap sort method is used to sort the 64-bit keystreams with the computational complexity  $O(n \cdot \log(n))$ . For a large block size of  $n = 64$  in an image, the whole permutation time is negligible.

### 3.3 Data Hiding Process

For each cipher-block, one message bit is embedded. We use flipping operation to embed the message bits in the encrypted image blocks, which is also used in [11, 12]. Flipping is an invertible operation, that is,  $flip(flip(p)) = p$  for any pixel  $p$ . In this work, before embedding the message bits, the image block is



**Fig. 2.** Example of Data Hiding Process (a) Checkerboard pattern of a 4 × 4 block (b) Cipher-block pixel values (c) bit 0 hidden in 3 LSBs (d) bit 1 hidden in 3 LSBs

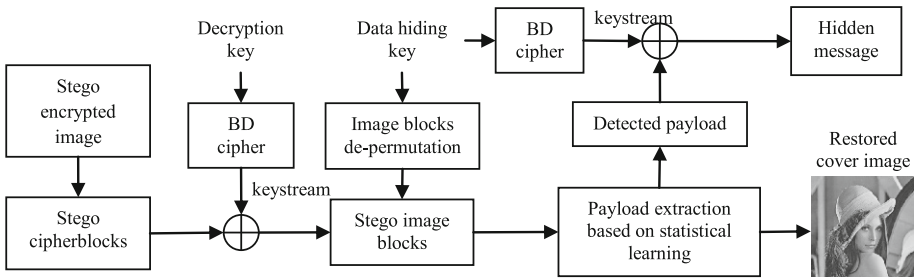
divided into two parts according to the checkerboard pattern, as shown in Fig. 2. In Fig. 2 (a), the neighbors of each black marked pixels are all white marked ones. Figure 2 (b) shows pixels of the cipher-block, and the modified pixels in Fig. 2 (c) and (d) are bold marked.

Based on the checkerboard pattern, if the message bit to be embedded is 0, flip the  $u$  least significant bits (LSB) of the white marked pixels  $c_{i,j,r}^* = flip(c_{i,j,r}), (i + j) mod 2 = 1, r = 1, \dots, u$ . Similarly, if the message bit is 1, flip the  $u$  LSBs of the black marked pixels as follow  $c_{i,j,r}^* = flip(c_{i,j,r}), (i + j) mod 2 = 0, r = 1, \dots, u$ . After embedding the message bits, the altered pixels have a corresponding relationship with the original ones. For example, the 3 LSBs of the corresponding pixel values are exchanged to each other:  $0 \leftrightarrow 7, 1 \leftrightarrow 6, 2 \leftrightarrow 5, 3 \leftrightarrow 4$ .

### 4 Decoding Process of Data Hiding

The decoding process is also composed of two steps which are decryption and payload detection. The overview of the decoding process is illustrated in Fig. 3.

Since Black Dragon cipher is a symmetric stream cipher, the decoding process is similar to the encoding process. At first the stego-image blocks are obtained by decryption using the same key as in the encoding process. Then the order of the stego cipher-blocks can be de-permuted. In the payload detection process, there are two hypotheses from each stego-image block, i.e. the message bit is



**Fig. 3.** Overview of the decoding process

either 0 or 1. To determine the embedded message bit, a high-dimensional statistical feature vector of the stego-image block is extracted, upon which a new Adaboost classifier is proposed which can classify the hypotheses blocks. Finally, the embedded message bits is removed and the original cover image is restored.

### 4.1 Decryption of Stego Encrypted Image

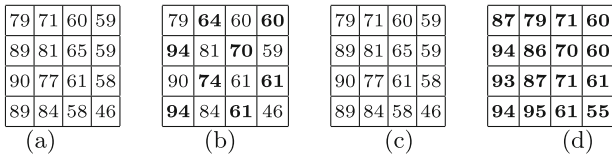
Before the payload is extracted, the encrypted image with embedded payload is first decrypted to obtain the stego-image. For the flipped pixels, we have  $b_{i,j,r}^* = c_{i,j,r}^* \oplus s_{i,j,r}$ , where  $b_{i,j,r}^*$  is a flipped pixel bit of the decrypted image, where  $(i, j)$  is pixel coordinate and  $r = 0, \dots, 7$  is bit index. For the unflipped pixels, we have  $b_{i,j,r} = c_{i,j,r} \oplus s_{i,j,r}$ , where  $b_{i,j,r}$  is an original pixel bit of the decrypted image. Pixels  $b_{i,j}^*$  and  $b_{i,j}$  constitute the decrypted stego-image block  $\mathbf{B}_i^*$ .

Although the distortion is reversible after removing the embedded payload, the amount of the distortion of the directly decrypted image without the payload extraction is still concerned. After the decryption step, half of the pixels of  $\mathbf{B}_i^*$  are flipped from the original pixels while the other half is not affected. The average distortion of flipping  $u$  LSBs is estimated as  $E_u = \left(\frac{1}{2^u} \sum_{i=0}^{2^u-1} (i - (2^u - 1 - i))^2 / 2\right)^{1/2}$ . Therefore,  $u = 3$  is chosen as a tradeoff of distortion and reversibility, and the PSNR of the decrypted stego-image is estimated as  $PSNR = 20 \log_{10} \frac{255}{E_3} = 37.9$  dB, which is validated by preliminary experiments. This PSNR value is high, since it is a common practice that distortion with  $PSNR > 35$  dB is imperceptible to human eyes.

### 4.2 Payload Detection and Cover Image Recovery

We need to judge whether  $\mathbf{B}_i^*$  is embedded with message bit 0 or 1, as well as restore it to the original cover image block  $\mathbf{B}_i$ . For each  $\mathbf{B}_i^*$ , we have two hypotheses blocks  $\mathbf{H}_0$  and  $\mathbf{H}_1$ , where  $\mathbf{H}_0$  is derived by flipping the white marked pixels in  $\mathbf{B}_i^*$  and  $\mathbf{H}_1$  is derived by flipping the black marked pixels. According to the data hiding process, one of  $\mathbf{H}_0$  and  $\mathbf{H}_1$  must be identical to  $\mathbf{B}_i$  while the other one consists of all  $b_{i,j,r}^*$  pixel bits that are flipped from  $\mathbf{B}_i$ , as illustrated in Fig. 4.

The detailed procedure of obtaining hypotheses blocks  $\mathbf{H}_0$  and  $\mathbf{H}_1$  is as follow: In case that  $\mathbf{B}_i^*$  is embedded with bit 0 in the data hiding process, by



**Fig. 4.** Example of Data Hiding Process (a) Cover image block  $\mathbf{B}_i$  (b)  $\mathbf{B}_i^*$  with bit 0 hidden (c) Hypothesis  $\mathbf{H}_0$  of bit 0 detected (d) Hypothesis  $\mathbf{H}_1$  of bit 1 detected

assuming the stego-block contains bit 0 at the receiver's side, the white marked pixels are flipped again, resulting in an identical block as  $\mathbf{B}_i$ . By assuming the stego-image is embedded with bit 1, the black marked pixels are flipped, resulting in a severely interfered block  $\mathbf{U}_i$  compared to  $\mathbf{B}_i$ . In case that  $\mathbf{B}_i^*$  is embedded with bit 1, assumption of bit 0 will produce the severely interfered block while assumption of bit 1 will produce  $\mathbf{B}_i$ .

Therefore, if the cover image block  $\mathbf{B}_i$  can be differentiated from the severely interfered block  $\mathbf{U}_i$ , the hidden bit will be identified according to the flip operation incurred in the receiver's side. In prior works [10–12], single features are used to separate  $\mathbf{B}_i$  from  $\mathbf{U}_i$ . For example, standard deviation of block  $\mathbf{B}_i$  in Fig. 4 (c) is 13.86 which is smaller than 14.45 of  $\mathbf{U}_i$  in Fig. 4 (d). However, a single feature is not very powerful to identify  $\mathbf{B}_i$  correctly.

In a generalized point of view, let  $S$  be the set of all  $256^{n \times n}$  possible states of each block  $\mathbf{B}_i$ , then the flipping operation will separate  $S$  into a pair of distinct sets of states. The payload detection task can be interpreted as classifying  $\mathbf{H}_0$  and  $\mathbf{H}_1$  to identify which hypothesis block is  $\mathbf{B}_i$  (a “natural” image block) and which one is  $\mathbf{U}_i$  (an “unnatural” image block). To achieve this, a feature vector can be extracted from the hypothesis blocks and a classifier can be designed to predict the class label of the two blocks based on the feature vectors.

**Feature Vector of the Hypothesis Blocks.** We start with defining the feature vector  $\mathbf{v}$  of a hypothesis block  $\mathbf{H}$  as  $\mathbf{v}(\mathbf{H}) = \{v_1(\mathbf{H}), \dots, v_D(\mathbf{H})\}$  where  $D$  is the dimensionality of the feature vector and  $v_i(\mathbf{H}) \in R$  is the  $i$ -th feature of the hypothesis block. As we know, the natural images tend to have higher pixel correlation while the flipped images are unnatural which tends to have lower pixel correlation. In order to classify the natural image block against the unnatural image blocks, various statistical features can be extracted from the image blocks  $\mathbf{H}_0$  and  $\mathbf{H}_1$ .

*Spatial Smoothness Features.* Features  $v_1$  =  $\sum_{i=2}^{n-1} \left| c_{i,j} - \frac{(c_{i-1,j} + c_{i+1,j} + c_{i-1,j-1} + c_{i-1,j+1})}{4} \right|$  and  $v_2 = \sum_{i=1}^n \sum_{j=1}^{n-1} |c_{i,j} - c_{i,j+1}| + \sum_{i=1}^{n-1} \sum_{j=1}^n |c_{i,j} - c_{i+1,j}|$  are used in [11] and [12], respectively, to reflect spatial smoothness among the neighborhood pixels, where  $c_{i,j}$  is a pixel value in a  $n \times n$  square image block.

*Proposed Statistical Features.* Statistical feature vector is 10-dimensional, including the average pixel values, mean absolute deviation, etc. The definitions of the features are shown in Appendix A. Each feature can help to differentiate the natural image block from the unnatural one by using a classifier. When  $n = 8$ , the final dimensionality of the feature vector is  $D = 20 + 8 \times 8 = 84$ .



**Statistical Learning of Feature Vectors.** Suppose we have a set of training data with  $N$  samples:  $\mathbf{Z} = \{(\mathbf{X}_i, y_i)\}$ ,  $i = 1, \dots, N$ , where  $\mathbf{X}_i \in \mathfrak{R}^D$  represents a  $D$ -dimensional detection vector for  $\mathbf{B}_i$ , and  $\mathbf{y} = (y_1, y_2)^T$  is the predicted class label with  $y_k = \mathbf{I}(k = c_i) - \frac{1}{2}$ , where  $c_i$  is the ground-truth label (“natural” or “unnatural”) for the  $i$ -th image block and  $\mathbf{I}(\cdot)$  is the identity function. For simplicity,  $\mathbf{X}_i$  is represented by  $\mathbf{X}$  which is defined as follow:

$$\mathbf{X} = \mathbf{v}(\mathbf{H}_0) - \mathbf{v}(\mathbf{H}_1) \quad (1)$$

---

**Algorithm 2.** Proposed two-class Adaboost using Exponential loss function

---

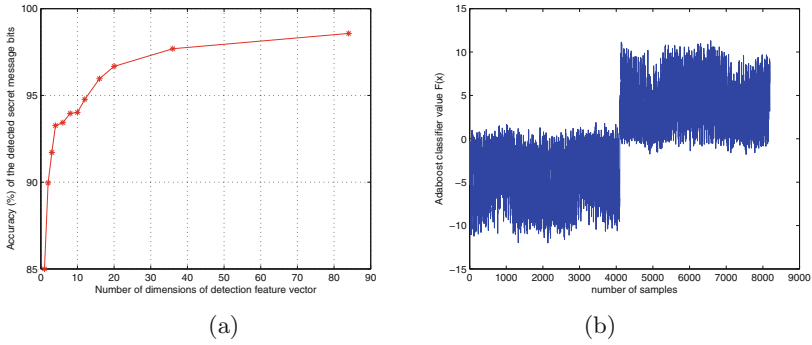
1. Initialize for all the weights :  $w_{i,1} := \frac{1}{N}$  ;
  2. Repeat for  $l = 1$  to  $L$ , compute normalized weights:  $w_{i,l}^* := \frac{w_{i,l}}{\sum_i w_{i,l}}$ ,  $i = 1, \dots, N$  ;  
 calculate optimal  $\mathbf{g}^{(l)}(\mathbf{X})$  and  $\beta^{(l)}$ :  $\arg \min_{\beta, \mathbf{g}} \sum_{i=1}^N w_i \exp(-\frac{1}{2} \mathbf{y}_i^T \beta^{(l)} \mathbf{g}^{(l)}(\mathbf{X}))$ ; and update weights:  $w_{i,l+1} \leftarrow w_{i,l} \cdot \exp\left(-\frac{\beta^{(l)}}{2} \cdot \mathbf{y}_i^T \cdot \mathbf{g}^{(l)}(\mathbf{X}_i)\right)$ ;
  3. Output: final prediction label is  $c^*(\mathbf{X}_i) = \arg \max_k F_k(\mathbf{X}_i)$ , where  $\mathbf{F}(\mathbf{X}_i) = \sum_{i=1}^L \beta^{(l)} \cdot \mathbf{g}_k^{(l)}(\mathbf{X}_i)$ .
- 

The proposed boosting algorithm is summarized in Algorithm 2. Intuitively, if a single feature is used, e.g.  $v_1$  [11],  $\mathbf{X}$  will be a scalar. If  $\mathbf{X} > 0$ ,  $\mathbf{H}_0$  will be classified as “unnatural” so message bit 1 is detected, and *vice versa*. In this work,  $\mathbf{X}$  will be a high-dimensional vector and a classifier is developed. The inference of the proposed boosting algorithm is provided in Appendix B.

## 5 Experimental Results

The training images are obtained from USC-SIPI image database [19] and QMUL-CIP image database [20]. To carry out a statistically significant testing of the proposed RDH method, 1000 test images are randomly selected from UIUC Scene-15 dataset [21]. All images are re-sampled at the resolution of  $512 \times 512$  for experiments. All decrypted stego-images cannot be judged by human eyes whether it has secret message bits embedded in it.

Figure 5 (a) shows the detection accuracy of the secret message bits for the “Baboon” image with different number of dimensions of the 84-D distinguishing feature vector  $\mathbf{X}$  in Eq. 1. It is obvious that when a feature vector of a higher dimension is used, a higher detection accuracy can be achieved. Note that feature  $v_1$  [11] and feature  $v_2$  [12] constitute a subset of the feature vector in this work. When only  $v_1$  is used, i.e. the feature vector is reduced to a scalar, the detection accuracy of the proposed method is exactly the same as in [11]. By adding more features to the feature vector, the detection accuracy gradually increases from

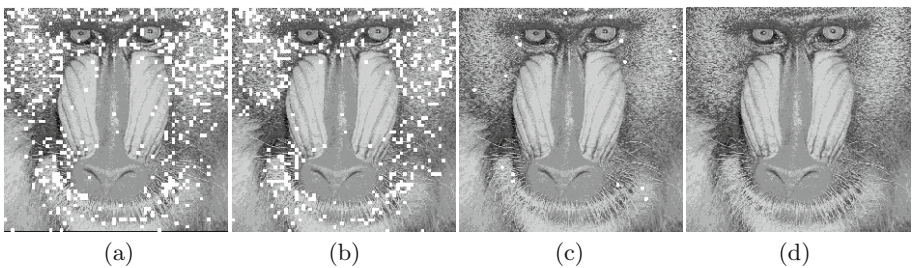


**Fig. 5.** Adaboost classification (a) detection accuracy with different dimensions (b) Adaboost classification value  $F(\mathbf{X})$  of the training samples

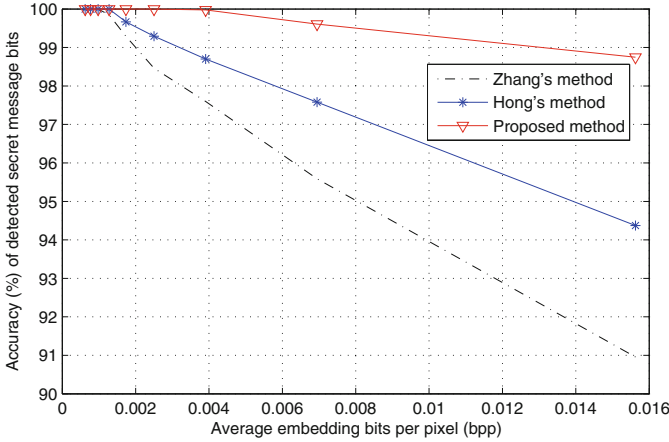
85.0% to 98.6%. However, the accuracy is still unlikely to achieve 100% even if the dimension is very large, and this issue will be solved later in this section. Figure 5 (b) shows the classification values  $F$  in Algorithm 2 of the 8192 training samples. The first half of the samples belong to the “natural” class while the second half belong to the “unnatural” class. It is obvious that most of the samples can be correctly classified by judging the sign of the corresponding classification values.

Figure 6 shows the restored “Baboon” images (additional test image as example) produced by different methods. Incorrect restored image blocks are marked white only for visual purpose. It is obviously that prior methods have much lower accuracy in choosing the original cover image blocks out of the hypothesis blocks. In contrast, only a small fraction of image blocks are incorrectly restored using the proposed method.

Since the incorrect restored blocks occupy less than 2% and they sparsely distribute over the whole image, an Error Code Correction (ECC) scheme can be used to completely erase the errors. ECC adds parity bits to the original secret message bits in order to resist the random burst error in communication. In this



**Fig. 6.** Restored “Baboon” by different methods (a) Zhang’s method [11] (b) Hong’s method [12] (c) Proposed method without ECC (d) Proposed method with ECC



**Fig. 7.** Average rate-distortion performance comparison on 1000 UIUC scene images

work the standard Reed-Solomon error correction scheme  $RS(7, 5)$  [22] is used. With  $RS(7, 5)$  coding of the secret message bits, the restored image is perfect as shown in Fig. 6 (d). However, the actual embedding rate is reduced to  $\frac{5}{7}$  since some redundancy is added to the secret message. In contrast,  $RS(7, 5)$  cannot reduce the error rate of the compared methods since the error rate often exceeds the error correction capacity.

The performance of the data hiding methods can be measured by a tradeoff between embedding rates versus accuracy of the detected secret message bits. Figure 7 lists the curves when adjusting the size of image blocks, i.e. 8 to 40 with step 4. The embedding rate depends on the block size, which is measured as bit per pixel:  $bpp = \frac{1}{blocksize^2}$ . A curve on top of others indicates better performance. For non-smooth images that are challenging to differentiate from unnatural images, the proposed method performs much better than other methods. For smooth images which are more natural, both the proposed method and [12] perform similarly with little error.

Embedding rates of the three RDH methods without ECC are the same. It is evident that the proposed RDH method performs the best. When combining with ECC with block size  $8 \times 8$ , the embedding rate of the proposed RDH is  $0.016 \times \frac{5}{7} \approx 0.011$ , and the accuracy is 100%. In contrast, Hong's method [12] gives less than 97% accuracy at a lower embedding rate 0.01, while the ECC will not increase the accuracy. Note that when ECC is further applied to the proposed method, the accuracy of the detected secret message bits by the proposed RDH method is always 100% for all test images when  $bpp \leq 0.016$ , while Hong's method [12] and Zhang's method [11] cannot guarantee 100% reversibility for  $bpp > 0.004$  and  $bpp > 0.002$ , respectively.

## 6 Conclusions and Future Work

In this work, a novel RDH method for encrypted images is proposed by statistical learning. A high-dimensional feature vector is extracted from image blocks, which is a generalization of the traditional methods based on a single feature. The Adaboost classifier is trained based on the feature vectors, achieving a much higher reversibility of the restored cover image. By further applying error correction code, the proposed method achieves 100 % reversibility, which differs from the traditional semi-reversible data hiding methods for encrypted images in uncompressed domain. Future work includes extending the approach to compressed-domain images such as JPEG.

**Acknowledgment.** We wish to express our sincere thanks to Dr. Matt Henricksen from Institute for Infocomm Research for providing valuable advices and suggestions. Our heartfelt thanks also go to Dr. Jiayuan Fan for her work in the initial phase of the algorithm implementation. Wei Wu is supported by National Natural Science Foundation of China (61472083, 61402110), Program for New Century Excellent Talents in Fujian University (JA14067) and Distinguished Young Scholars Fund of Fujian (2016J06013).

## A Appendix

### *Spatial Smoothness Features*

$$v_3 = \sum_{i=2}^{n-1} \sum_{j=2}^{n-1} (|c_{i,j} - c_{i-1,j}| + |c_{i,j} - c_{i+1,j}| + |c_{i,j} - c_{i,j-1}| + |c_{i,j} - c_{i,j+1}|) \quad (\text{A.1})$$

$$v_4 = \sum_{i=2}^{n-1} \sum_{j=2}^{n-1} (|c_{i,j} - c_{i-1,j-1}| + |c_{i,j} - c_{i+1,j+1}| + |c_{i,j} - c_{i+1,j-1}| + |c_{i,j} - c_{i-1,j+1}|) \quad (\text{A.2})$$

where  $c_{i,j}$  is a pixel value in a  $n \times n$  square image block. Natural images tend to have lower values of these features while unnatural images tends to have higher values.

Average pixel values:

$$v_5 = \bar{c} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n |c_{i,j}| \quad (\text{A.3})$$

Absolute mean value difference between black and white marked pixels:

$$v_6 = \left| \sum \sum_{(i+j) \bmod 2=0} c_{i,j} - \sum \sum_{(i+j) \bmod 2=1} c_{i,j} \right| \quad (\text{A.4})$$

Geometric mean value:

$$v_7 = \left( \prod_{i=1}^n \prod_{j=1}^n c_{i,j} \right)^{\frac{1}{n^2}} \tag{A.5}$$

Mean absolute deviation:

$$v_8 = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n |c_{i,j} - \bar{c}| \tag{A.6}$$

Second to seventh order of the central sample moments:

$$v_{7+k} = M_k = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n |c_{i,j} - \bar{c}|^k, \quad k = 2, \dots, 7 \tag{A.7}$$

where the second order statistic is also used in [10].

*Discrete Cosine Transform.* DCT coefficients constitutes  $n \times n$  features:

$$V_{u,w} = \sum_{i=1}^n \sum_{j=1}^n c_{i,j} \cos \left[ \frac{\pi}{n} \left( i - \frac{1}{2} \right) u \right] \cos \left[ \frac{\pi}{n} \left( j - \frac{1}{2} \right) w \right], \quad 1 \leq u, w \leq n. \tag{A.8}$$

Other features include skewness  $v_{15}$ , kurtosis  $v_{16}$ , median value  $v_{17}$ , trimmed mean value excluding the outliers  $v_{18}$ , range of the values  $v_{19}$  and interquartile range of the values  $v_{20}$ .

## B Appendix

Inspired by [15, 16], a new two-class Adaboost algorithm is developed here based on additive logistic regression and empirical loss:

$$\min_{\mathbf{F}(\mathbf{x})} \sum_{i=1}^N \exp \left( -\frac{1}{2} \mathbf{y}_i^T \mathbf{F}(\mathbf{X}_i) \right) \tag{B.1}$$

with the symmetric constraint  $F_1(\mathbf{X}_i) + F_2(\mathbf{X}_i) = 0$ , where  $\mathbf{F}(\mathbf{X})$  is a continuous-valued Adaboost classifier. We consider  $\mathbf{F}(\mathbf{X})$  that has the following forward stage additive modelling form:

$$\mathbf{F}(\mathbf{X}) = \sum_{l=1}^L \beta^{(l)} g^{(l)}(\mathbf{X}) \tag{B.2}$$

where  $\beta^{(l)} \in \Re$  are weighting coefficients and  $g^{(l)}(\mathbf{X})$  are basis functions satisfying the symmetric constraint  $g_1(\mathbf{X}) + g_2(\mathbf{X}) = 0$ . The additive model is

$\mathbf{F}^{(l)}(\mathbf{X}) = \mathbf{F}^{(l-1)}(\mathbf{X}) + \beta^{(l)}\mathbf{g}^{(l)}(\mathbf{X})$  thus the optimization can be represented as

$$\begin{aligned} (\beta^{(l)}, \mathbf{g}^{(l)}) &= \arg \min_{\beta, \mathbf{g}} \sum_{i=1}^N \exp \left( -\frac{1}{2} \mathbf{y}_i^T \left( \mathbf{F}^{(l-1)}(\mathbf{X}) + \beta^{(l)}\mathbf{g}^{(l)}(\mathbf{X}) \right) \right) \\ &= \arg \min_{\beta, \mathbf{g}} \sum_{i=1}^N w_i \exp \left( -\frac{1}{2} \mathbf{y}_i^T \beta^{(l)}\mathbf{g}^{(l)}(\mathbf{X}) \right) \end{aligned} \quad (\text{B.3})$$

where  $w_i = \exp \left( -\frac{1}{2} \mathbf{y}_i^T \mathbf{F}^{(l-1)}(\mathbf{X}) \right)$  is the current sample weight. In this work, basis function  $\mathbf{g}(\mathbf{X})$  is defined as:

$$g_k(\mathbf{X}) = \frac{1}{A} \cdot \frac{2 \exp(-d_k(\mathbf{X}))}{1 + \exp(-d_k(\mathbf{X}))} - \frac{1}{2} \quad (\text{B.4})$$

where  $A$  is a normalization factor and  $d_k(\mathbf{X})$  is defined as the  $l_1$ -norm distance between the feature vector  $\mathbf{X}$  and the  $k$ -th class of the detection vectors as follow

$$d_k(\mathbf{X}) = \frac{1}{|\mathbf{Y}_k|} \sum_{\mathbf{v} \in \mathbf{Y}_k} \|\mathbf{X} - \mathbf{v}\|, \quad k = 0, 1 \quad (\text{B.5})$$

where  $\mathbf{V}$  is a feature vector,  $\mathbf{Y}_k$  is the set of the feature vectors in the  $k$ -th class, and  $|\cdot|$  denotes the cardinality of a set. A small  $d_k(\mathbf{X})$  means  $\mathbf{X}$  is close to the  $k$ -th class, and  $g_k(\mathbf{X})$  will be large.

## References

1. Fridrich, J., Goljan, M., Du, R.: Invertible authentication. In: Electronic Imaging Photonics West, pp. 197–208. International Society for Optics and Photonics (2001)
2. Celik, M.U., Sharma, G., Tekalp, A.M., Saber, E.: Reversible data hiding. In: IEEE International Conference on Image Processing, vol. 2, pp. II–157 (2002)
3. Shi, Y.Q., Ni, Z., Zou, D., Liang, C., Xuan, G.: Lossless data hiding: fundamentals, algorithms and applications. In: Proceedings of the 2004 International Symposium on Circuits and Systems, vol. 2, pp. II–33. IEEE (2004)
4. Ni, Z., Shi, Y.Q., Ansari, N., Su, W.: Reversible data hiding. IEEE Trans. Circ. Syst. Video Technol. **16**(3), 354–362 (2006)
5. Lu, Z.-M., Li, Z.: High capacity reversible data hiding for 3D meshes in the PVQ domain. In: Shi, Y.Q., Kim, H.-J., Katzenbeisser, S. (eds.) IWDW 2007. LNCS, vol. 5041, pp. 233–243. Springer, Heidelberg (2008)
6. Boxcryptor: May 2011. <https://www.boxcryptor.com/en/google-drive>
7. Dennis, O.: August 2013. <http://www.cnet.com/how-to/two-free-ways-to-encrypt-google-drive-files/>
8. Ra, M.R., Govindan, R., Ortega, A.: P3: toward privacy-preserving photo sharing. In: Presented as Part of the 10th USENIX Symposium on Networked Systems Design and Implementation, pp. 515–528 (2013)
9. CNET: July 2013. [http://news.cnet.com/8301-13578\\_3-57594171-38/google-tests-encryption-to-protect-users-drive-files-against-government-demands/](http://news.cnet.com/8301-13578_3-57594171-38/google-tests-encryption-to-protect-users-drive-files-against-government-demands/)
10. Puech, W., Chaumont, M., Strauss, O.: A reversible data hiding method for encrypted images. In: Electronic Imaging, p. 68191E. International Society for Optics and Photonics (2008)

11. Xinpeng, Z.: Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **18**(4), 255–258 (2011)
12. Hong, W., Chen, T.S., Wu, H.Y.: An improved reversible data hiding in encrypted images using side match. *IEEE Signal Process. Lett.* **19**(4), 199–202 (2012)
13. Xinpeng, Z.: Separable reversible data hiding in encrypted image. *IEEE Trans. Inf. Forensics Secur.* **7**(2), 826–832 (2012)
14. Ma, K., Zhang, W., Zhao, X., Yu, N., Li, F.: Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Trans. Inf. Forensics Secur.* **8**(3), 553–562 (2013)
15. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, Paul M.B. (ed.) *EuroCOLT 1995*. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
16. Buhlmann, P., Yu, B.: Additive logistic regression: A statistical view of boosting-discussion (2000)
17. Zhu, J., Rosset, S., Zou, H., Hastie, T.: Multi-class adaboost. *Ann Arbor* **1001**(48109), 1612 (2006)
18. Henriksen, M.: Two dragons—a family of fast word-based stream ciphers. In: *International Conference on Security and Cryptography, Iceland*, pp. 35–44 (2012)
19. Weber, A.G.: The usc-sipi image database. USC-SIPI Report 315, pp. 1–24 (1997)
20. Hao, P.: (2004). <http://www.eecs.qmul.ac.uk/~phao/cip/images/>
21. UIUC (2006). [http://www-cvr.ai.uiuc.edu/ponce\\_grp/data/](http://www-cvr.ai.uiuc.edu/ponce_grp/data/)
22. Wicker, S.B., Bhargava, V.K.: *Reed-Solomon Codes and Their Applications*. Wiley, New York (1999)

# **Network Security**



# An Ensemble Learning Approach for Addressing the Class Imbalance Problem in Twitter Spam Detection

Shigang Liu, Yu Wang<sup>(✉)</sup>, Chao Chen, and Yang Xiang

School of Information Technology, Deakin University, Geelong, Australia  
{shigang, y. wang, chao. chen, yang}@deakin.edu.au

**Abstract.** Being an important source for real-time information dissemination in recent years, Twitter is inevitably a prime target of spammers. It has been showed that the damage caused by Twitter spam can reach far beyond the social media platform itself. To mitigate the threat, a lot of recent studies use machine learning techniques to classify Twitter spam and report very satisfactory results. However, most of the studies overlook a fundamental issue that is widely seen in real-world Twitter data, i.e., the class imbalance problem. In this paper, we show that the unequal distribution between spam and non-spam classes in the data has a great impact on spam detection rate. To address the problem, we propose an ensemble learning approach, which involves three steps. In the first step, we adjust the class distribution in the imbalanced data set using various strategies, including random oversampling, random undersampling and fuzzy-based oversampling. In the next step, a classification model is built upon each of the redistributed data sets. In the final step, a majority voting scheme is introduced to combine all the classification models. Experimental results obtained using real-world Twitter data indicate that the proposed approach can significantly improve the spam detection rate in data sets with imbalanced class distribution.

**Keywords:** Online social networks · Twitter spam · Machine learning · Class imbalance

## 1 Introduction

Twitter has gained significantly in popularity in recent years and become an important source for real-time information sharing and news dissemination. Inevitably, the growth of Twitter is accompanied by a significant increase of spamming activities targeting on the platform. Twitter spam is usually referred to as the unsolicited tweets that contain malicious links directing victims to external sites with malware downloads, phishing, drug sales, or scams, etc. [1]. A series of incidents showed that Twitter spam not only affects user experience, but also poses significant threats of damages beyond the social networking platform itself. As an example, in September 2014, a nationwide Internet meltdown in New Zealand was caused by a Twitter spam campaign that spread DDoS attack malware in the guise of leaked nude photos of Hollywood celebrities [2].

The traditional approach to detect and filter spam is based on blacklists. For example, Trend Micro develops a blacklisting service called Web Reputation Technology system to filter spam URLs for users [3]. Twitter also implements a blacklist filtering module in their anti-spam system BotMaker [4]. Nonetheless, the blacklist-based schemes fail to protect victims from emerging spam due to the time lag [5]. A previous study shows that more than 90 % of victims may click through a new spam link before it is blocked by blacklists [6].

In order to overcome the limitation of blacklisting, a lot of researchers have proposed machine learning based schemes that detect Twitter spam by mining the spammers' and spam tweets' unique patterns, without checking the embedded URLs [7, 8]. This kind of spam detection schemes usually involve a few steps. First of all, the features that can differentiate spam from non-spam are selected and extracted from the tweets or authors. Example features include account age, number of followers, number of following, and number of characters in a tweet. Second, a small set of training data samples are labelled (as spam or non-spam) based on some ground truth. Finally, various machine learning algorithms can be applied to develop classification models, which can then be deployed to detect spam in a real-time basis. A number of recent studies [1, 9–11] have reported satisfactory results obtained using machine learning based detection schemes.

However, most previous studies overlook the issue of imbalanced class distribution that widely exists in real-world Twitter data. That is, the proportion of the spam tweets is much smaller than that of the non-spam tweets in reality. For example, a study based on a data sample back in 2009 [12] suggested that 3.75 % of tweets are spam. In the meantime, the experimental data sets used in most related works have similar amounts of spam data and non-spam data. Therefore, it is interesting to see the effectiveness of the machine learning based detection schemes on data sets with various distributions of spam and non-spam classes.

In this paper, we investigate the class imbalance problem in machine learning based Twitter spam detection. We begin by examining the detection performance in data sets with varying class imbalance rates from 2 (i.e., the number of non-spam tweets is twice as the number of spam tweets) to 20 with incremental steps of 2. The preliminary results show that the increase of class imbalance rate leads to slightly better precision in spam detection along with a significant decrease (over 30 %) in detection rate. In other words, in the data sets that the non-spam tweets extremely outnumber the spam tweets, a large proportion of spam tweets can be missed by the detection schemes.

To address the problem, we propose an ensemble learning approach that incorporates data preprocessing techniques that mitigate the impact of imbalanced class distribution. Specifically, the proposed approach builds an ensemble classifier in three steps. In the first step, multiple strategies, including random oversampling, random undersampling and fuzzy-based oversampling, are adopted to adjust the class distribution in the data. In the second step, a classification model is trained from each of the redistributed data sets respectively. In the last step, a majority voting scheme is introduced to combine all of the classification models. Experimental results derived from real-world Twitter data shows that the proposed approach can significantly improve the performance of spam detection in data sets with different degrees of class distribution.

The rest of this paper is organized as follows. Section 2 presents a review on Twitter spam detection. In Sect. 3, we demonstrate the class imbalance problem and present the proposed approach. The experimental results obtained from real-world Twitter data is presented and analyzed in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related Work

Spam mitigation is one of the key security challenges in online social networks as well as in the cyber space in general [23–26]. Traditionally, blacklists are used for detecting and filtering unwanted information including spam. For example, Twitter implements a blacklist filtering module in their anti-spam system BotMaker [4]. Trend Micro [3] offers a blacklisting service based on the Web Reputation Technology, which is able to filter harmful spam URLs. Blacklists have a critical disadvantage. That is, it takes some time for the new malicious links to be included in the blacklists. A lot of damages can have already been caused during the time lag [5, 6].

Heuristic rule based methods for filtering Twitter spam have been developed in some earlier attempts to overcome the limitations of blacklisting. Yardi et al. [13] proposes to detect spam in #robotpickupline (hashtag created by themselves) through three rules, which are suspicious URL search, username pattern matching and keyword detection. Kwak et al. [14] proposes to remove all the tweets that contain more than three hashtags so as to eliminate the impact of spam for their research.

A lot of recent studies propose to apply machine learning techniques for identifying Twitter spam based on a range of features, including tweet-based, author-based, and social graph based attributes. For instance, Wang [16] present an approach based on Bayesian models to detect spammers on Twitter, and Benevenuto et al. [1] propose to detect both spammers and spam using the Support Vector Machine algorithm. In [9], Stringhini et al. train a classifier using the Random Forest algorithm, which is then used to detect spam in three social networks, including Twitter, Facebook and MySpace. Lee et al. [15] deploy some honeypots to derive the spammers' profiles, and they extract the statistical features for spam detection using several machine learning algorithms, such as Decorate, RandomSubSpace and J48.

It has been showed that some basic features used in the above studies can be easily fabricated by purchasing followers, posting more tweets, or mixing spam with normal tweets. Accordingly, researchers propose some robust features that rely on the social graph to avoid feature fabrication. For example, Song et al. [17] propose to extract the distance and connectivity between a tweet author and its audience to determine whether it is a spam tweet or not. After merging the sophisticated features with the basic feature set, they show that the performance of several classifiers is improved to nearly 99 % True Positive and less than 1 % False Positive. Yang et al. [10] also propose some robust spam features, which include Local Clustering Coefficient, Betweenness Centrality and Bidirectional Links Ratio. They show that their feature set can outperform the features used in the previous works [1, 9, 15, 16]. In this work, we use the simple tweet-based and author-based features to study the impacts of class imbalance. We do not adopt the advanced features, because they require comprehensive knowledge of user connections.

Besides, some researchers resort to analyzing the embedded URLs in tweets to detect spam. Thomas et al. [18] make use of several URL based features, such as the domain tokens, path tokens and query parameters, along with some features of the landing page, such as DNS information and domain information. Lee and Kim [19] investigate into the characteristics of Correlated URL Redirect Chains, and some relevant features, such as URL redirect chain length, relative number of different initial URLs.

### 3 Learning from Imbalanced Twitter Spam Data

#### 3.1 Class Imbalance Problem in Twitter Spam Detection

In order to apply machine learning techniques to Twitter spam detection, each tweet in the data set is presented as a feature vector. The feature vector consists of the observed values on a set of predetermined features of a tweet. In this work, we define the feature set using both tweet-based and author-based attributes. Specifically, the tweet-based features are number of retweets, number of hashtags, number of user mentions, number of URLs, number of characters, and number of digits in the tweet. Besides, the author-based features include number of followers, number of followings, number of published tweets, number of user favorites, and number of lists. The complete list of features are given in Table 1. We note that all the selected features can be extracted out of the tweets with little computational overhead, such that they are suitable for real-time detection.

**Table 1.** Twitter Feature Set

No.	Feature	Description
1	account_age	The number of days since user account creation
2	no_follower	The number of followers of this twitter user
3	no_following	The number of followings/friends of this twitter user
4	no_user_favorite	The number of favorites this twitter user received
5	no_list	The number of lists this twitter user added
6	no_tweet	The number of tweets this twitter user sent
7	no_retweet	The number of retweets this tweet
8	no_hashtag	The number of hashtags included in this tweet
9	no_user_mention	The number of user mentions included in this tweet
10	no_URL	The number of URLs included in this tweet
11	no_char	The number of characters in this tweet
12	no_digit	The number of digits in this tweet

Now suppose we are given a set of labeled data consisting of  $N_+$  spam tweets and  $N_-$  non-spam tweets:  $D = \{(\mathbf{x}_1, \omega_+), \dots, (\mathbf{x}_{N_+}, \omega_+), (\mathbf{x}_{N_++1}, \omega_-), \dots, (\mathbf{x}_{N_++N_-}, \omega_-)\}$ , in which each  $\mathbf{x}_n \in \mathbb{R}(n = 1, \dots, N_+ + N_-)$  is the feature vector of  $i$ th tweet, while  $\omega_+$  and  $\omega_-$  are the corresponding class label for spam and non-spam respectively (here we

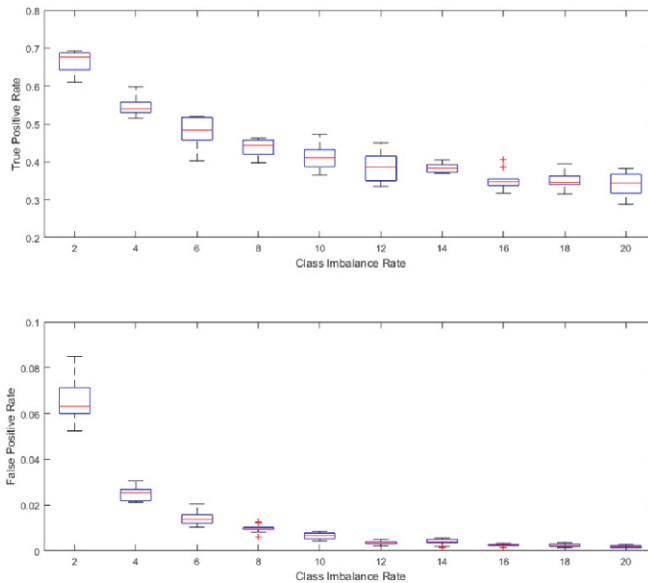
consider spam to be the positive class). Based on  $D$ , we can build a classification model using supervised learning algorithms, which can predict whether any given test tweets belongs to  $\omega_+$  or  $\omega_-$ , i.e.,  $F(\mathbf{x}) : \mathbb{R} \rightarrow \{\omega_+, \omega_-\}$ .

As in lots of data mining application domains, Twitter spam detection faces the class imbalance problem. That is, in a data set randomly collected from the Twitter platform, the number of spam tweets is usually much less than the number of non-spam tweets (i.e.,  $N_+ \ll N_-$ ). In this work, we define the class imbalance rate in data set  $D$  as:

$$\gamma = \frac{N_-}{N_+}.$$

In practice, the class imbalance rate can be varying depending on the activities of spam campaigns during the observation or data collection period. For example, a study based on a data set of 2000 tweets [12] showed that 3.75 % of the collected tweets are spam. This yields a class imbalance rate over 25. However, most previous studies carry out experiments without the class imbalance problem in mind, so that relatively “balanced” data sets are used in the experiments, which are formed by including roughly the same amounts of spam and non-spam samples.

To demonstrate the problem, we conduct a series of preliminary experiments using data sets with varying class imbalance rates from 2 to 20 with incremental steps of 2, i.e.,  $\gamma = 2, 4, 6, \dots, 20$ . Figure 1 shows the detection performance, which are obtained using Random Forest [20] as the base classification algorithm. We can find that as the class imbalance rate rises from 2 to 20, the true positive rate of the spam class (i.e., the spam detection rate) witnesses a significant decrease of 33 % in average (among 10 tests with



**Fig. 1.** Performance degradation caused by class imbalance

independent data sets). In particular, when  $\gamma = 20$ , the averaged detection rate is down to 34 %, which means over 66 % of spam are missed by the detection. In the meantime, the false positive rate drops from around 6 % to less than 1 % when the class imbalance rate increases. This is because the classifier is significantly bias to the negative class.

### 3.2 Proposed Approach

In order to address the class imbalance problem in Twitter spam detection, we propose an ensemble learning approach that incorporates a majority voting scheme to combine multiple classification models. In particular, each model is independently built upon a data set that is re-balanced from the given imbalance data set using one of the following data sampling methods: random oversampling, random undersampling and fuzzy-based oversampling.

The proposed ensemble learning approach consists of three steps. In the first step, we adopt three data sampling methods to pre-process the imbalanced training data set.

*Random oversampling (ROS)*: This method takes as input the data of the spam class  $D_+$  and an oversampling ratio  $\alpha$ . It then randomly selects a number of  $\alpha \times N_+$  samples from  $D_+$  with replacement. The generated data samples are combined with  $D_+$  to form the new set of training samples  $D'_+$  for the spam class.

*Random undersampling (RUS)*: This method takes as input the data of the non-spam class  $D_-$  and an undersampling ratio  $\beta$ . It then randomly selects a number of  $(1 - \beta) \times N_-$  samples from  $D_-$  and discards the rest to form a new non-spam set  $D'_-$ .

*Fuzzy-based Information Decomposition oversampling (FID)*: This method takes as input the data of the spam class  $D_+$  as well as an oversampling ratio  $\alpha$ . It then generates a number of  $\alpha \times N_+$  synthetic samples using a fuzzy-based information decomposition algorithm [21]. In general terms, the algorithm generates the synthetic samples based on the probability distribution of original minority class data samples. To do this, FID divides the feature space into  $t = \alpha \times N_+$  intervals in each dimension, and it employs a fuzzy membership function to estimate the weights of the available samples to each interval. These weights are then used for generating a number of  $t$  synthetic data values in each dimension.

In particular, FID creates a mapping from the feature space to a discrete universe set:

$$\begin{aligned} \mu : \mathbf{y}_i \times \mathbf{u}_i &\rightarrow [0, 1], \\ (y_{ji}, u_{si}) &\rightarrow \mu(y_{ji}, u_{si}), \end{aligned}$$

where  $\mathbf{y}_i$  is a column feature vector (i.e., the original data values in the  $i$ th dimension) and  $\mathbf{u}_i$  is a discrete universe set of  $\mathbf{y}_i$ ;  $y_{ji}$  is the  $j$ th value in  $\mathbf{y}_i$ , and then  $u_{si}, s = 1, \dots, t$ , represents the center of each interval. Here we use the following membership function:

$$\mu(y_{ji}, u_{si}) = \begin{cases} 1 - \frac{\|y_{ji} - u_{si}\|}{h_i}, & \text{if } \|y_{ji} - u_{si}\| \leq h_i \\ 0 & , \text{otherwise} \end{cases}$$

**Table 2.** The proposed ensemble learning algorithm

<b>Ensemble Learning Algorithm</b>
<p><b>TRAINING</b>  <b>INPUT:</b> Training data set <math>D</math>; Oversampling ratio <math>\alpha</math>; Undersampling ratio <math>\beta</math>;  <b>OUTPUT:</b> Multiple base classifiers <math>F_1(\mathbf{x})</math>, <math>F_2(\mathbf{x})</math>, and <math>F_3(\mathbf{x})</math>  <b>1:</b> Generate <math>D_1 = D'_+ \cup D_-</math> with random oversampling method <math>D'_+ = ROS(D_+, \alpha)</math>;  <b>2:</b> Generate <math>D_2 = D_+ \cup D'_-</math> with random undersampling method <math>D'_- = RUS(D_-, \beta)</math>;  <b>3:</b> Generate <math>D_3 = D''_+ \cup D_-</math> with FID oversample method <math>D''_+ = FID(D_+, \alpha)</math>;  <b>4: for</b> <math>i</math> in 1, 2, and 3; <b>do</b>  <b>5:</b> Train classifier <math>F_i</math> using base classification algorithm: <math>F_i(\mathbf{x}) = BASE(D_i)</math>;  <b>6: end for</b>  <b>7: return</b> <math>F_1(\mathbf{x})</math>, <math>F_2(\mathbf{x})</math>, and <math>F_3(\mathbf{x})</math></p>
<p><b>TESTING</b>  <b>INPUT:</b> Test data point <math>\mathbf{z}</math>  <b>OUTPUT:</b> Class prediction <math>\omega</math> for <math>\mathbf{z}</math>  <b>1:</b> <math>v_{i+} = 0</math>, <math>v_{i-} = 0</math>;  <b>2: for</b> <math>i</math> in 1, 2, and 3 <b>do</b>  <b>3: if</b> <math>F_i(\mathbf{z}) = \omega_+</math> <b>do</b> <math>v_{i+} = v_{i+} + 1</math>;  <b>4: else do</b> <math>v_{i-} = v_{i-} + 1</math>;  <b>5: end if</b>  <b>6: end for</b>  <b>7: if</b> <math>v_{i+} &gt; v_{i-}</math> <b>do</b> return <math>\omega_+</math>;  <b>8: else do</b> return <math>\omega_-</math>;  <b>9: end if</b></p>

where  $h_i$  is the step length of the divided intervals. Then if  $\sum_j \mu(y_{ji}, u_{si}) = 0$ , we set the mean of all observed values as the  $s$ th generated feature value  $\tilde{m}_{si}$ . Otherwise, the weighted mean is set as the generated data value. That is,

$$\tilde{m}_{si} = \begin{cases} \bar{y}_i & , \text{ if } \sum_j \mu(y_{ji}, u_{si}) = 0 \\ \frac{\sum_{j=1}^m m_{jsi}}{\sum_{j=1}^m \mu(y_{ji}, u_{si})} & , \text{ otherwise} \end{cases}$$

where  $m_{jsi} = \mu(y_{ji}, u_{si}) \times y_{ji}$ . Finally, similar to *ROS*, the generated data samples are combined with  $D_+$  to form a new set of training samples  $D''_+$  for the spam class.

In the second step of our approach, a classification model is trained from each of the generated data sets, i.e.,  $D'_+ \cup D_-$ ,  $D_+ \cup D'_-$ , and  $D''_+ \cup D_-$ , respectively. In this work,

we adopt a number of widely used supervised learning algorithms for the purpose of evaluation, including Naïve Bayes, SVM (Support Vector Machines), C4.5 Decision Trees and Random Forest [22].

In the last step, a majority voting scheme is introduced to combine the classification models to predict the class of any given testing data point. The algorithm is summarized in Table 2.

## 4 Evaluation

### 4.1 Evaluation Methods

To evaluate the proposed approach, we use a real-world Twitter spam data set that we published in an earlier study [27]. The data set consists of more than 600 million tweets with URLs. The ground truth of the data set is set up using the Web Reputation Service provided by Trend Micro. In particular, the service is able to identify whether any given URL is malicious as well as to which category the URL belongs. We define those tweets that contain malicious URLs as Twitter spam. The data set has been made publicly available to fellow researchers for the purpose of validation and extension.

From the raw data, we construct a series of experimental data sets with ten different class imbalance rates, i.e.,  $\gamma = 2, 4, 6, \dots, 20$ . For example, given a class imbalance rate of 10, we randomly select one thousand spam tweets and ten thousand non-spam tweets to form the spam class and the non-spam class respectively. Moreover, for each of the class imbalance rates, we repeat the process for 10 times to derive 10 independent data sets. In other words, the results given in the following are obtained from 10 independent tests. In each derived data set, 60 % of tweets are randomly selected to be training data and the rest are used for testing.

We use a number of metrics to measure the spam detection performance, including true positive rate (which is also called recall, or spam detection rate in our application domain), false positive rate, precision, and f-measure. Given the true positive, false negative, true negative and false positive as illustrated in Table 3, the above metrics can be calculated as follows.

$$\text{True Positive Rate} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$\text{False Positive Rate} = \frac{\text{false positive}}{\text{false positive} + \text{true negative}}$$

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$F - \text{measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$



**Table 3.** Confusion matrix

	Classified as spam	Classified as non-spam
Spam	true positive	false negative
Non-spam	false positive	true negative

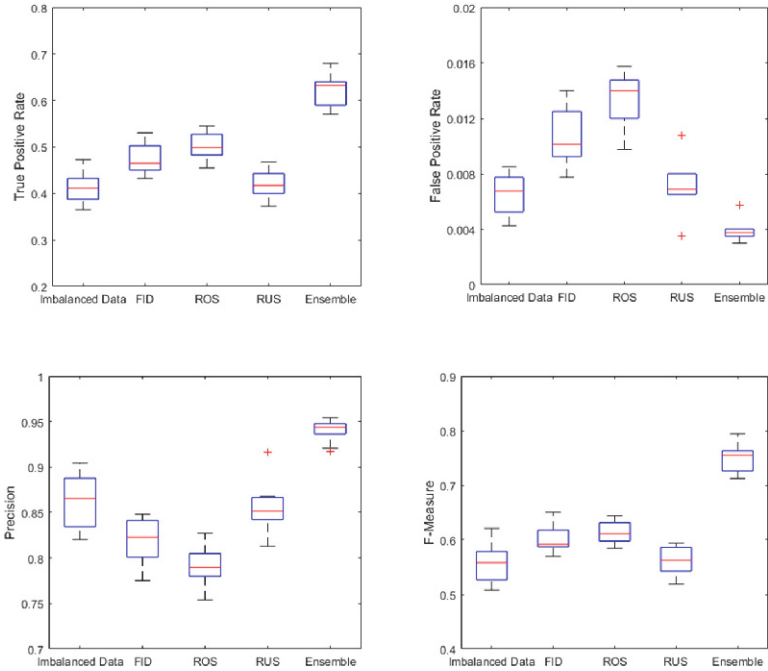
In the evaluation we use 4 popular supervised learning algorithms that are commonly used in the related studies, including Naïve Bayes, SVM (Support Vector Machines), C4.5 Decision Trees and Random Forest [22]. In particular, we find that random forest achieves the best detection performance across all experiments, thus we only report the results for random forest in the following due to space limits. We also note that although the spam detection performance is weaker for the other three classifiers, the proposed ensemble approach manages to significantly boost their performance in general.

## 4.2 Results of Learning from Imbalanced Data

Figure 2 presents the spam detection results derived by random forest classifiers based on data sets in which the number of non-spam tweets are 10 times as the number of spam tweets (i.e.,  $\gamma = 10$ ). For the purpose of comparison, the performance of learning from the imbalanced data directly are given as a baseline. Besides, we compare the proposed ensemble learning approach with the fundamental methods for processing imbalanced data, i.e., random oversampling (ROS), random undersampling (RUS), and the fuzzy-based oversampling (FID). For each of the oversampling methods, we test a number of oversampling rate (i.e.,  $\alpha$ ) ranging from 20 % to 2000 %, plus the equal class distribution setting (oversampling until the two classes have the same number of data points). For the undersampling method, we test a number of undersampling rate (i.e.,  $\beta$ ) ranging from 20 % to 90 %, plus the equal distribution setting. Through extensive experiments, we select the parameter for each method that yield best results in average. The selected parameters are 200 % for oversampling, 90 % for undersampling.

In terms of true positive rate, directly learning from data sets with an imbalance rate of 10 yields an average result of 41 %. While the oversampling methods ROS and FID improve the performance to 50 % and 47 % in average, the undersampling method RUS does not help much in this case (42 %). Nonetheless, the ensemble approach manages to significantly raise the true positive rate to 63 %.

The ensemble approach also achieves the lowest averaged false positive rate (0.4 %) among the others. We can see that the oversampling methods ROS and FID raise the false positive rate to around 1 %, which is much higher than directly learning from the imbalanced data and RUS (which are 0.6 % and 0.7 %). This indicates that oversampling boosts the true positive rate in the cost of raising false positive rate at the same time. In contrast, the proposed ensemble approach is able to overcome this tradeoff, such that it improves the detection rate as well as the precision (which is showed in Fig. 1 as well).



**Fig. 2.** Spam detection results in data sets with a class imbalance rate of 10

The graph of f-measure shows the general pattern in the experimental results. First, directly learning from imbalanced data yields poor performance. Second, ROS, RUS and FID can somehow increase the performance a bit but the ensemble approach is able to significantly boost the performance of spam detection to another level.

### 4.3 Impact of Class Imbalance Rate

In the last section, we have seen that when the non-spam tweets significantly outnumber the spam tweets in the data, the performance of machine learning based spam detection is degraded. In the following, we continue to investigate how varying class imbalance rates affect the spam detection performance.

Figure 3 compares the averaged true positive rate of the five considered approaches in regards to the class imbalance rate  $\gamma$ . When  $\gamma = 2$ , random forest classifiers are able to identify 66 % of the spam tweets by directly learning from the imbalanced data sets. By preprocessing the imbalanced data sets using FID, ROS, and RUS, the detection rates are improved to 74 %, 74 % and 79 % respectively. Moreover, the proposed ensemble approach successfully identifies 84 % of the spam tweets in average. As  $\gamma$  increases, the true positive rate of all approaches exhibits a gradual decrease. After  $\gamma > 16$ , the true positive rates of the ensemble approach remain stable at around 65 %.

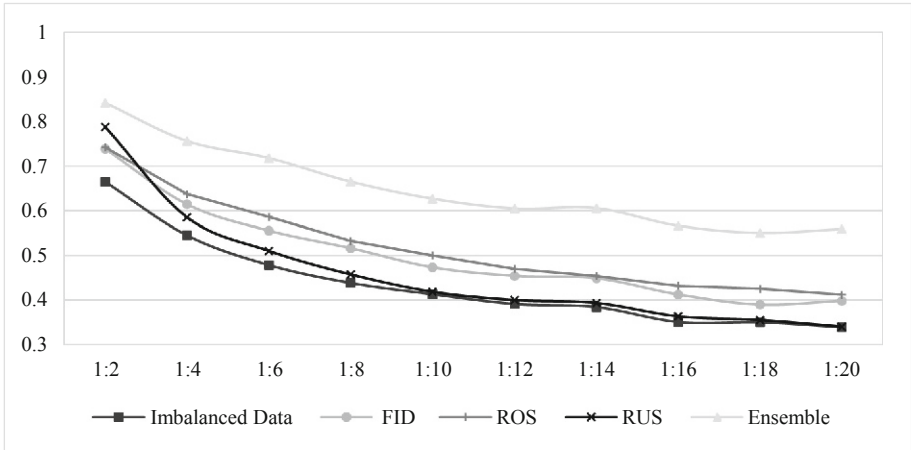


Fig. 3. True positive rate versus varying class imbalance rate

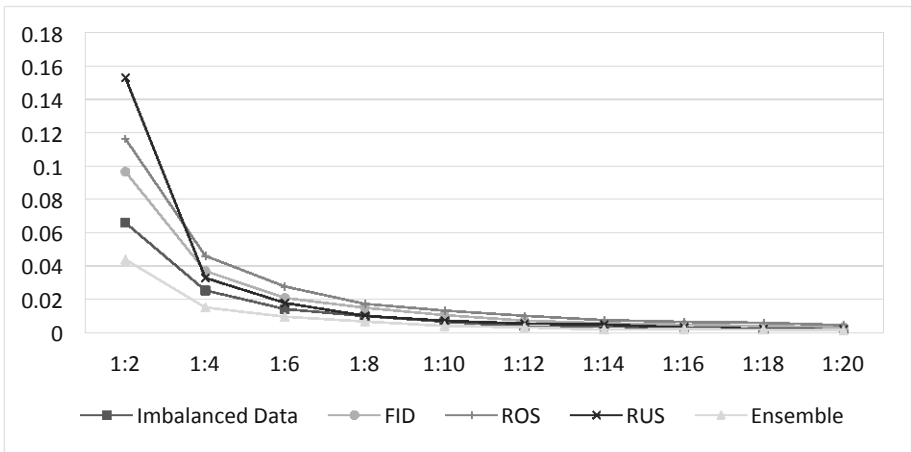


Fig. 4. False positive rate versus varying class imbalance rate

In comparison, the two oversampling methods (i.e., ROS and FID) maintain true positive rates between 40 % and 45 %, while the other two methods (i.e., RUS and learning from imbalanced data) only identify around 35 % of the spam. In general, the ensemble approach boosts the spam detection rate up more than 20 % (compared to direct learning) given any class imbalance rates.

Figure 4 depicts the averaged false positive rate of the five considered approaches with respect to the class imbalance rate. When the imbalance among classes is not significant (i.e.,  $\gamma = 2$ ), the ensemble approach yields a lowest false positive rate of 4 % in average, while the classifiers trained from imbalanced data directly obtain the second lowest at around 6.5 %. The three data preprocessing methods FID, ROS, and

RUS raise the false positive rate to 9.6 %, 11.6 % and 15 % respectively. As the class imbalance rate rises to 4, the false positive rates of all approaches decrease to below 5 %, in which the ensemble approach and the classifiers trained from imbalanced data obtain the lowest at 1.5 % and 2.5 % respectively. If  $\gamma$ . rises over 16, the false positive rates of all approaches fall below 1 %. This is expected as the classifiers are bias to the negative class, so that most of the errors are type II errors (false negatives) and type I errors (false positives) are sparse.

The averaged spam detection precision performance is illustrated in Fig. 5. Learning from imbalanced data directly can achieve 84 % to 90 % precision across all of the class imbalance rates. The ensemble approach shows an improvement between 4 % and 9 %, while the other methods lower the precision to different extents.

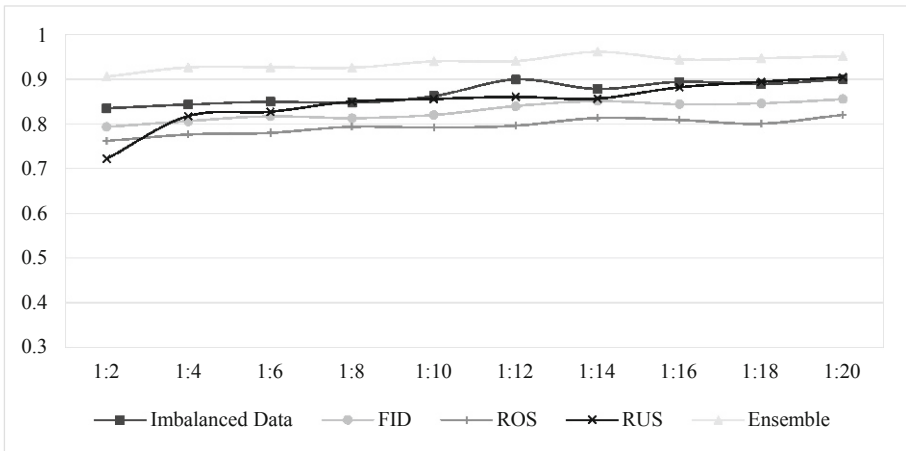


Fig. 5. Spam detection precision versus varying class imbalance rate

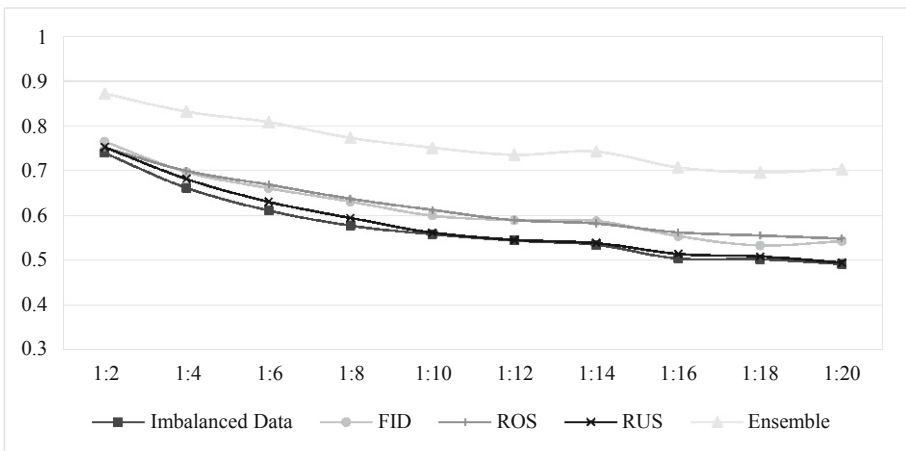


Fig. 6. Spam detection f-measure versus varying class imbalance rate

From the above discussion, we know that the proposed ensemble learning approach not only identifies more spam tweets (higher true positive rate) but also raises less false alarms (lower false positive rate and higher precision) in the data sets with various rates of class imbalance. This makes the f-measure results given in Fig. 6 predictable, as the f-measure is a harmonic mean of precision and recall (true positive rate). In particular, the ensemble learning approach generates an f-measure of 88 % when  $\gamma = 2$ , and then the value gradually decreases to around 70 % as  $\gamma$  goes up to 20. In general, this shows an improvement up to 20 % compared with learning from the imbalanced data directly, and also a lead by at least 10 % to the other data preprocessing methods.

## 5 Conclusion

This paper investigates the class imbalance problem in machine learning based Twitter spam detection. It has been showed that the effectiveness of detection can be severely affected by the imbalanced distribution of spam tweets and non-spam tweets, which is widely seen in real-world Twitter data sets. An ensemble approach has been proposed to mitigate the impact of class imbalance. Extensive experiments have been conducted using real-world Twitter data. The results show that the proposed approach significantly improves the spam detection performance on imbalanced Twitter data sets with a range of imbalance degrees.

## References

1. Benevenuto, F., Magno, G., Rodrigues, T., Almeida, V.: Detecting spammer on twitter. In: Seventh Annual Collaboration, Electronic messaging, Anti-abuse and Spam Conference, July 2010
2. Pash, C.: The lure of naked hollywood star photos sent the internet into meltdown in New Zealand. *Business Insider*, September 2014
3. Oliver, J., Pajares, P., Ke, C., Chen, C., Xiang, Y.: An in-depth analysis of abuse on twitter. Technical report, Trend Micro, 225 E. John Carpenter Freeway, Suite 1500 Irving, Texas 75062 USA, September 2014
4. Jeyaraman, R.: Fighting spam with botmaker. *Twitter Engineering Blog*, August 2014
5. Grier, C., Thomas, K., Paxson, V., Zhang, M.: @spam: the under- ground on 140 characters or less. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010*, pp. 27–37. ACM, New York (2010)
6. Thomas, K., Grier, C., Song, D., Paxson, V.: Suspended accounts in retrospect: an analysis of twitter spam. In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011*, pp. 243–258, ACM, New York (2011)
7. Gao, H., Chen, Y., Lee, K., Palsetia, D., Choudhary, A.: Towards online spam filtering in social networks. In: *NDSS* (2012)
8. Yang, C., Harkreader, R., Zhang, J., Shin, S., Gu, G.: Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In: *Proceedings of the 21st International Conference on World Wide Web, WWW 2012*, pp. 71–80, USA (2012)

9. Stringhini, G., Kruegel, C., Vigna, G.: Detecting spammers on social networks. In: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC 2010, pp. 1–9. ACM, New York (2010)
10. Yang, C., Harkreader, R., Gu, G.: Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Trans. Inf. Forensics Secur.* **8**(8), 1280–1293 (2013)
11. Zhang, X., Zhu, S., Liang, W.: Detecting spam and promoting campaigns in the twitter social network. In: Data Mining. *IEEE ICDM 2012*, pp. 1194–1199 (2012)
12. Pear Analytics: Twitter Study, August 2009
13. Yardi, S., Romero, D., Schoenebeck, G., Boyd, D.: Detecting spam in a twitter network. *First Monday* **15**(1–4) (2010). <http://dx.doi.org/10.5210/fm.v15i1.2793>
14. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 591–600. ACM, New York (2010)
15. Lee, K., Caverlee, J., Webb, S.: Uncovering social spammers: social honeypots + machine learning. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 435–442. ACM, New York (2010)
16. Wang, A.H.: Don't follow me: spam detection in twitter. In: Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT), pp. 1–10 (2010)
17. Song, J., Lee, S., Kim, J.: Spam filtering in twitter using sender-receiver relationship. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 301–317. Springer, Heidelberg (2011)
18. Thomas, K., Grier, C., Ma, J., Paxson, V., Song, D.: Design and evaluation of a real-time url spam filtering service. In: Proceedings of the 2011 IEEE Symposium on Security and Privacy, SP 2011, pp. 447–462. IEEE Computer Society, Washington, DC (2011)
19. Lee, S., Kim, J.: Warningbird: a near real-time detection system for suspicious urls in twitter stream. *IEEE Trans. Dependable Secur. Comput.* **10**(3), 183–195 (2013)
20. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
21. Liu, S., Zhang, J., Wang, Y., Xiang, Y.: Fuzzy-Based feature and instance recover. In: Nguyen, T.N., et al. (eds.) ACIIDS 2016. LNCS, vol. 9621, pp. 605–615. Springer, Heidelberg (2016)
22. Weka 3: Data Mining Software in Java. <http://www.cs.waikato.ac.nz/ml/weka/>
23. Choo, K.-K.R.: The cyber threat landscape: challenges and future research directions. *Comput. Secur.* **30**(8), 719–731 (2011)
24. Lai, S., Liu, J.K., Choo, K.-K.R., Liang, K.: Secret picture: an efficient tool for mitigating deletion delay on OSN. In: Qing, S., et al. (eds.) ICICS 2015. LNCS, vol. 9543, pp. 467–477. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29814-6\_40
25. Norouzi, F., Dehghantanha, A., Eterovic-Soric, B., Choo, K.-K.R.: Investigating social networking applications on smartphones: detecting Facebook, Twitter, LinkedIn, and Google+ artifacts on android and iOS platforms. *Aust. J. Forensic Sci.* 1–20 (2015). doi:10.1080/00450618.2015.1066854
26. Quick, D., Martini, B., Choo, K.-K.R.: *Cloud Storage Forensics*. Syngress Publishing/Elsevier, Boston (2013)
27. Chen, C., Zhang, J., Chen, X., Xiang, Y., Zhou, W.: 6 million spam tweets: a large ground truth for timely twitter spam detection. In: *IEEE International Conference on Communications (ICC 2015)* (2015)

# **Smart City Security**

# Putting the User in Control of the Intelligent Transportation System

Catalin Gosman<sup>1</sup>, Tudor Cornea<sup>1</sup>, Ciprian Dobre<sup>1</sup>, Florin Pop<sup>1(✉)</sup>,  
and Aniello Castiglione<sup>2</sup>

<sup>1</sup> Computer Science Department, University Politehnica of Bucharest,  
313, Splaiul Independentei, 060042 Bucharest, Romania  
{catalin.gosman,tudor.cornea}@cti.pub.ro,  
{ciprian.dobre,florin.pop}@cs.pub.ro

<sup>2</sup> Department of Computer Science, University of Salerno,  
Via Giovanni Paolo II, 132, 84084 Fisciano, Salerno, Italy  
castiglione@ieee.org

**Abstract.** Intelligent Transportation Systems (ITS) demonstrate innovative services for different modes of transport and traffic management. They enable users to be better informed and make safer, coordinated, and smarter use of transport networks. However, for such systems to be effective, security and privacy considerations have to be enforced. GPS data can lead to accurate traffic models, but such data can also be used by malicious advisories to gain knowledge about the whereabouts of users. In this paper, we propose a model and related constructs for users to define sharing policies for the data they contribute. The model allows entities to define data sharing policies associated with personal information captured by an ITS application. While ITS applications need to use specific data under various quality and context-based constraints, the user-oriented security policies could specify additional usage constraints, and our solution is an approach to mediate between these two sides. Secondly, we propose an algorithm to manage the trust level in the data contributed by users. The solution is based on the quality parameters of the information disseminated in the system, e.g. spatial whereabouts and timestamp for the transmitted data regarding various events happening in the system, combined with the reputation specific to the sources that transmit data. We evaluate a pilot security implementation of the model under real-world assumptions.

**Keywords:** Security · Traffic · Sharing · Policies · Trust in information

## 1 Introduction

Intelligent transportation systems (ITS) rely on data collected from various sensors, available within the road infrastructure or contributed by volunteer users, to derive models of traffic, automate recognition of transport-related safety events, with the end objective of reducing congestions, making traffic safer, monitor and



optimize transportation for car fleets, etc. However, in order to be effective, such systems need large amounts of data related to traffic (i.e., they rely on accurate traffic models). Most current ITS today are being developed as proprietary, closed-source solutions. Besides the fact that this leads to problems related to a major lack of interconnectivity between such systems, this also means that users have no guarantees about the security mechanisms being used, and no control over the data sharing process. For example, a user contributing to a navigation application (e.g., Waze, Google Traffic) has little knowledge about what entities (and under what terms) could potentially have access to his private contributed data. At a first glance, this would not be an important problem. However, sharing location data could potentially lead to the user disclosing sensitive information about driving routes or driving preferences (i.e., even if data is anonymized at individual level, an attacker could still gain important knowledge about the traffic as a flow, an information that could be used further for own benefit). Sharing data in ITS may raise questions regarding users' privacy needs and requirements: Who owns the data, and under which conditions is the data shared with other entities in ITS, who benefits from the data collection process? In many of the current approaches, users have little control over the data collection process, and cannot influence the information sharing process. *Users willingly contributing in such ITS lose control and access to their personal shared data.*

Our contribution in this paper can be summarized as follows: First, *we propose a new model for users to define data sharing policies for their personal information that is captured by an ITS.* Let us consider context data collection. Information such as timestamp, location, possible speed, hashed identity of collecting device, etc., are used as parameters of the security policy, and are combined using logical operators. As a pilot implementation, we have developed an ITS navigation system, where a security manager imposes the security policies defined by users in their mobile clients. Second, *we present a solution to quantify the level of trust for the information that users get back from the ITS for personal use.* In many cases, the ITS can hardly guarantee quality for the disseminated information. According to the authors in [8], contextual information is either incorrect (if it does not reveal the true state of the world it is trying to reflect), inconsistent (if it contains contradictory information), or incomplete (if it contains unknown data so that it can be used). Without being able to determine the level of trust for the information disseminated in the ITS it is very difficult to choose whether or not to use the data when building an application.

Some authors state that trust in information is closely related to the entities in the system. However, ITS have some particularities: the context is changing frequently and dynamically. Moreover, the system is described by short term interactions between entities. So, trust in information must be established by taking into consideration the information disseminated in the system and not rely only the reputation of the sources in the system. For example, in ITS, a node's identity is irrelevant and insufficient in order to establish the level of trust regarding system's events. Our approach for determining the level of trust is based on the disseminated data, particularly its quality parameters: spatial

accuracy regarding the event, the temporal closeness when transmitting data about an event, combined with ITS source's reputation that transmit data.

The rest of the paper is structured as follows: In Sect. 2 we analyse security in various ITS applications, and show how they compare, if any, to our proposal. Next, in Sect. 3 we present our security policies approach. In Sect. 4 we present our mechanism for establishing the level of trust in information. Our pilot implementation and experimental results are in Sect. 5. Section 6 presents conclusions and future work.

## 2 Related Work

Today, one of the biggest challenges for ITS remains security. We stopped below at some of the most important approaches in this direction.

PEPSI [4] developed a centralized system with anonymity requirements for ITS applications needing to share data and discover relevant contextual information out of it. To deal with user anonymity, PEPSI uses the Registration Authority, which collects contextual demands from users, and offers in return cryptographic material. The disadvantage is that the problem is now with the Registration Authority, that knows in advance all demands regarding contextual information, and also the identities of all users that want to access contextual information. Authors assume in PEPSI that the Registration Authority is always the reliable component – which is not always true (in reality, any component is vulnerable to an attacker).

In the PRISM project [9], participating nodes (mobile devices) are registering to a server that keeps track of the participants, and offers back to users information such as location of other participants. In PRISM, security mainly ensures that unregistered users cannot access the gathered data. The drawback of PRISM is that it does not ensure anonymity, neither for participants that ask data about others, neither for participants that share data.

For the development of a secure system, authors in [6] proposed a modular concept, divided into several layers: the node registration layer, the test and certification layer, the pseudonym layer, the revocation layer and the data assessment and intrusion handling layer. The advantage of the solution consists in the modular concept, the drawback is the components' interconnections so that security requirements are met.

Having a way in which users share information in order to develop ITS applications, academic research has focused on determining the level of trust for the information used in order to develop an ITS application. Various authors proposed different techniques to compute the degree of trust in information. Using the majority principle, the trust level is decided by the majority of nodes. Using the most trustworthy source principle, the trust level is equal to the most “trustworthy” source. This technique can isolate and identify entities in the system that have a high degree of “trustworthiness” compared with other sources. Using Bayesian inference, the trust level is calculated as posterior probability of the event taking into consideration new reports about the event. In Dempster-Shafer

theory, in case of two contrary events, the lack of knowledge about an event can be considered solid proof for the opposite event.

In addition to these, we acknowledge the work of other authors in securing ITS platforms. Security requirements for ITSs are analyzed by Stampoulis and Chai [17], Parno and Perrig [13], or Raya and Hubaux [15]. Raya and Hubaux [15], in particular, have analyzed the use of symmetric cryptography. However, their solution cannot ensure the non-repudiation Wang et al. [18] have extended further this symmetric cryptography approach to provide non-repudiation and confidentiality services. Several approaches based on group communication have been presented by Raya and Hubaux [15], Caballero-Gil et al. [2]. For communication inside a group, nodes use group signatures and hence, node identities are not disclosed outside the group. The main problems are group formation and leader election which become complicated due to the dynamic topology of the network. More recent work in pervasive systems study the possibility to ensure privacy by hiding the true owner (source generator) of information “into the crowd”. This means that the MAC address, for example, is preserved by using from pseudonyms to linking it to only a group of devices [10, 19, 21]. While these are interesting results, we assume the collecting mobile device to not always be trustful. In respect to this, we assume only the existence of at least one trusted entity: the collecting platform.

Our work is developed within the context of the nationally-funded MobiWay and DataWay projects. The MobiWay project [11] aims to construct a platform for connecting and sharing data between ITS actors (systems, applications, users), supporting information management on a large scale. A special component is dedicated to connecting various ITS sources together, facilitating the introduction and maintenance of ITS services, achieved through mediation, service specific selection and composition. After data is gathered, we store it in Data Vaults (DV), which are logically distinct data stores (i.e., per user or group of users) that declare and impose a set of policies or rules for each individual’s data [12]. Before the data can be extracted and processed in MobiWay, a Security Engine is first queried, and a decision is made about the availability of data in regards to that particular processing step. The Security Engine enforces the requirements of the DVs. To achieve this, every processing request made by an ITS service triggers a policy check. Policies are further extracted on request from a Policy Store. The Security Engine is also responsible for preserving the privacy of all participants. The mechanism used in order to determine the level of trust in information facilitates the way in which our security model filters the information shared by the users. Our proposed approach for determining the level of trust is based on the information disseminated in the system, particularly its quality parameters, like spatial accuracy regarding the event, the temporal closeness when transmitting data about an event, combined with the reputation specific for ITS sources that transmit data. Furthermore, the data needs to be securely processed, objective for the DataWay project [3].

### 3 Security Policies in ITS

We propose a security and privacy-aware model designed for ITS applications where users need to cooperate and exchange personally-sensed data that is further aggregated into models of traffic conditions<sup>1</sup>. Examples of such use cases include car sharing, carpooling, taxi management applications, or navigators like Waze and Google Traffic.

We assume the existence of several sources ( $S_i$ , for  $i$  in the set of all sources) sensing and sending traffic data to a collecting platform  $P$ . There, the data is potentially aggregated into traffic models, or used directly by various ITS applications and services ( $A_i$ , for  $i$  in the set of all applications needing the collected data). In this case, a security policy has to specify permissions oriented towards the data source. As specification, the security policy has to specify (1) sensitivity of the data and (2) conditions that must hold for the data to be accessible. Condition (1) means that data is classified according to the sensing context. Condition (2) means that data can be accessed differently by different actors, or under different usage patterns. Using security policies, privacy considerations can be expressed at both group level (e.g., a taxi company could decide that all data belonging to its fleet of cars is to be shared under one unique sharing policy), but also on individual level (a user having installed an ITS application on his mobile device (source  $S_i$ ) is able to change preferences regarding sharing policy with an ITS collecting platform (fine granularity)).

A common problem perceived by users contributing with data in ITS is the lack of access control options for their stored information. Most often, a source  $S_i$  can collect sensitive information from which a third party could benefit: location could disclose route patterns or potential traffic events [16]; GSM and/or Wireless data could disclose crowd behavior [1]. A user cannot influence how and by what application/service  $A_i$  the personally collected data will be used.  $A_i$  should guarantee confidentiality and data protection for all information stored by any source  $S_i$ . Otherwise, sources can become reluctant in contributing with information. Thus, if privacy requirements are not treated properly, ITS providers may be confronted with having only small volumes of data, but also with legal constraints regarding user data privacy violations.

In order to focus on each individual privacy demands, our proposed solution uses *security policies*. Security policies are applied at the Data Vault  $DV_i$  level – the Data Vault is a logical data storage structure used to separate the data being collected by each particular source. The  $DV_i$  acts as a proxy for source  $S_i$ , allowing it to control access to its data in ITS. In this manner,  $DV_i$  has multiple functions: it ensures data protection and access control to stored information, and it maintains flexibility regarding the security policies accepted, forcing data

<sup>1</sup> The proposed security model can be applied to various other ITS applications. For example, data sensed by privately-owned devices could further be exchanged by users interested in social-based applications, such as distributed social networks of traffic participants. Another applicability is the participatory sensing paradigm, where citizen campaign can collect traffic-related data, such as noise or pollution levels, to derive accurate models of potential problems within the city.

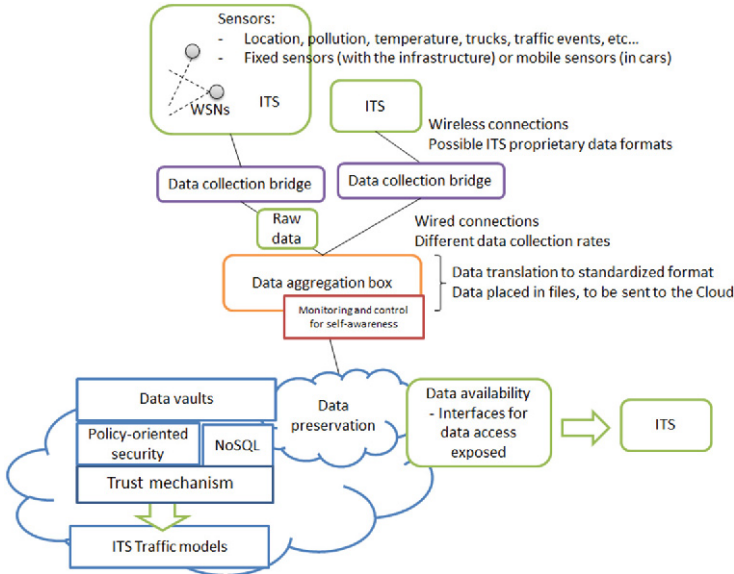
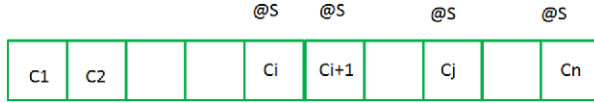


Fig. 1. An overview of the ITS eco-system, as defined in MobiWay.

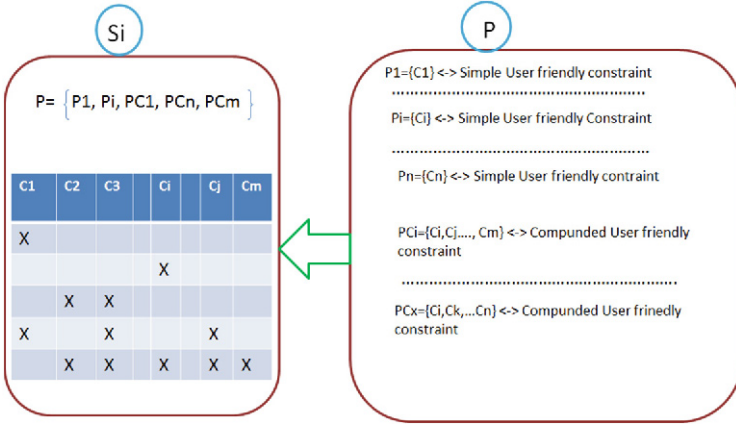
to be shared only according to the security policies accepted at the  $DV_i$  level. The mechanism defined for specifying what and who can access (the ‘policy’) should be flexible enough – for various situations (defined as the context flowing in time [20]), context could be ‘secured’ differently (e.g., location could be hidden for particular areas in town, or should be shared differently considering various time moments). The policy needs to be easily adapted to such changes regarding how  $S_i$  wants its data to be shared. Such a change should be initiated by the user, or should automatically be made possible by the system.

Therefore, the ITS becomes a combination of the following actors (see Fig. 1). We have several sources  $S_i$  that can collect data on-the-ground. This data and related security sharing policies are further aggregated by the collecting platform  $P$ , where information belonging to each source is logically separated in Data Vaults ( $DV_i$ ). Finally, on top, we have applications and services  $A_i$  using the data (either individual data, or data aggregated from several sources).

At the source level, a security policy can be defined using annotations. In this way, a user can specify what data is sensitive according to the accepted security policy. Below is a generic example for using annotation at field level. Fields  $C_i, C_{i+1}, C_j, C_n$  are defined as being sensitive by the source  $S_i$ . In accordance with policy  $P_i$  accepted at  $DV_i$  level, these fields will be shared according to a predicate defined by annotation  $S$  (see Fig. 2). For example, annotation here could dictate that fields are either shared or not (boolean), it could state that two particular fields will never be shared simultaneously with the same application, or it could state conditions regarding the applications – the data could be shared differently, depending on the application requesting it.



**Fig. 2.** Representation of a basic security policy applied at the field level.



**Fig. 3.** Application of how different security policies at the Data Vault level.

Security policies are defined at the platform level  $P$ , a central entity part of ITS, as shown in Fig. 1. When a source  $S_i$  accepts a security policy defined on platform  $P$ , the source should understand the technical implications even in case of a user having little understanding of security policies interactions. Accordingly, we define at the platform level a set of default simple security policies that users can understand: *share or not* location, *share or not* WiFi data, etc. This simple textual representation of policies can be aggregated into complex sharing filters on the platform. For advanced users, the security policy can be defined on a fine-grained level: for these fields, apply these particular sharing operators (predicates). Finally, the user can apply several such privacy policies simultaneously for the data being collected (and the aggregation/filtering is done at the  $DV_i$  level). For example, one rule could state that WiFi is not to be shared. Another one could state that location should be shared, only under particular circumstances. This situation is represented in Fig. 3.

For lack of space limits, we will not include here the formal demonstration and experimental results of the approach.

## 4 Establishing the Level of Trust in Information

Having a way in which users consensually share information in order to develop ITS applications under our proposed security policies, we next focus on establishing the level of trust for the information disseminated in order to develop

an ITS application. Our proposed security model for determining the level of trust in information is designed for ITS applications which assume that users cooperate and work together to have a common sense of traffic conditions. On platform  $P$  (see Fig. 1), the trust level mechanism is run so that invalid information disseminated in the system is filtered.

ITS are dynamic and frequently changing systems, with short interactions between sources  $S_i$ . Considering these particular features, the security mechanism needs to be based on the shared data, and in particular on parameters like location and timestamp, combined with the reputation of the sources that transmit data in the system. Information is gathered from the variety of sources  $S_i$ . Sources  $S_i$  differ one from another due to the quality of information shared, but also due to sensor's imperfections and failures. In dynamic systems such as ITS, the quality of information provided is influenced by the data sources diversity, but also by the particular specifications of the devices that collect data. In order to describe information quality, we consider parameters such as: sensor technical specifications, timestamp when the information was gathered, geo-location related to the observation, etc. All these parameters are specific to sources  $S_i$  that share information. So, source  $S_i$  can be considered to be the main provider for the parameters describing quality of the information disseminated in the system. Usually, there are multiple sources of information  $S_i$  that disclose information about different events in the system. Therefore, in order to trust and process that specific event, the ITS needs a mechanism that analyses data originating from different sources, and determine a level of trust for the shared information, regarding a specific (traffic) event. Previous approaches for dealing with information trust consider that trust is closely related to the reputation of the entities existing in the system. In such approaches, the level of trust in information is exclusively determined by the relationships established between entities within the system. For example, in systems that rely on sources' reputation, trust in information is derived using a chain of interactions between nodes. In such cases, the logic used to establish the level of trust in the disseminated information is reduced to the sources' reputations. The drawback of such an approach is represented by the tedious process of changing the source  $S_i$  reputation. For example, the source benefits from a particular reputation level visible to other entities in the system. Altering this reputation level is a slow process, that occurs only after a number of interactions with other entities in the system.

These observations indicate that the level of trust in information based only on the reputation of sources  $S_i$  is inadequate in dynamically and frequently changing systems like ITS. We argue that in ITS the level of trust in information can be established considering also the actual shared information (and not only the source's reputation). In ITS, a node's identity is irrelevant and insufficient for determining the trust in information – a “reliable” source could mislead and declare imperfect information about a particular road event for different causes. For example, in Waze users are asked to declare whether a particular event (such as ice on road) is still occurring, and they can falsely indicate the event is over simply because they do not notice it occurring (but this does not automatically

mean the user is not to be trusted). Our solution for determining the level of trust is based on the information shared in the system. A source can observe and report different events, and each such observation is done with a different trust/sensitivity level. The quality parameters specific for the shared information vary in regards to the different events observed. We are providing the following simple taxonomy for events that can occur in ITS: simple and complex events. We consider that set  $\{E_1, \dots, E_n\}$  consists in primary events specific for ITS, which are mutually exclusive events. E.g: ice on the road, traffic accident etc. Complex events are unions of primary events. E.g: ice on the road together with a traffic accident may lead to a traffic jam, which is a complex event. The degree of trust in information is specific for both primary and complex events.

A source  $S_i$  can share information about event  $E_i$  (e.g., car accident), but also about event  $E_j$  (jammed intersection). The two system events are each unique, even though they are advertised by the same source  $S_i$ . However, for an unique event in the system we can establish just one unique level of trust, obtained as an aggregation function of the shared information from all sources  $S_i$  that observe the particular event. Therefore, the level of trust in the shared information is determined as an aggregation function for all the information received from sources  $S_i$  regarding a specific event in the system.

Therefore, in order to determine the level of trust associated with event  $E_i$  occurring in ITS, our solution relies on the data collected from multiple sources  $S_i$ . In contrast with the reputation system approach, our solution considers that more than one source reports about a specific event (otherwise, the event is not disseminated to other nodes). The quality parameters used for establishing the level of trust are: timestamp when a source  $S_i$  reports data about event  $E_i$ , location from where a source  $S_i$  reports data about event  $E_i$ , source reputation in ITS, but also the fact that a source  $S_i$  can confirm or infirm the existence of event  $E_i$ . Therefore, on the data collecting platform  $P$ , a specific level of trust is calculated for each source  $S_i$  that shares data about event  $E_i$ . The final level of trust in regards to event  $E_i$  is an aggregation function for all trust levels calculated for each source  $S_i$  that shares data about event  $E_i$ . The level of trust in the shared information is determined as an aggregation function considering all the information received from sources  $S_i$  regarding the specific event  $E_i$  in the system. When source  $S_i$  reports data about a specific event, the source is described by its position and the timestamp when the observation takes places.

Let us consider the following notations:  $d(E_i)$  is the distance to observed event  $E_i$ ,  $t(E_i)$  is the timestamp when event  $E_i$  occurred and  $lp(E_i)$  is the life period of event  $E_i$ . The spatial accuracy for source  $S_i$  in regards to the observed event  $E_i$  can be described as following. If the distance of source  $S_i$  in regards to the observed event is less than the maximum distance from where the event can be observed, then:  $A_d(S_i) = 1 - \frac{\text{distance}(S_i)}{\text{maximum } d(E_i)}$ . Otherwise,  $A_d(S_i) = 0$ .

The temporal accuracy for source  $S_i$  in regards to the observed event  $E_i$  can be described as following. If the timestamp when source  $S_i$  reports data about the observed event is included in the interval [*timestamp when event  $E_i$  occurred*,



timestamp when event  $E_i$  occurred + life period of event  $E_i$ ], then:  $A_t(S_i) = 1 - \frac{\text{timestamp}(S_i)}{t(E_i)+lp(E_i)}$ . Otherwise,  $A_t(S_i) = 0$ .

The reputation for source  $S_i$  is adjusted dynamically, according to the accuracy of data shared in the system. The cases for the reputation are:  $r_i = 1$ , when source is valid in the system, or it has just joined ITS. It is considered as a trustworthy source when transmitting information about different events in the system;  $r_i = 0$ , when source has been revoked from the system. Intermediary values, in interval  $[0, 1]$ , are used in order to denote how source reputation is dynamically adjusting in regards to the information shared in ITS.

We must also consider that source  $S_i$  can confirm or infirm event  $E_i$ :

$$E(S_i) = \begin{cases} 1, & \text{source } S_i \text{ confirms event } E_i \\ -1, & \text{source } S_i \text{ infirms event } E_i \end{cases}$$

If the timestamp of the observation is after the event expired (the time when the event occurred, plus the life time of the event is smaller than the observation timestamp), the information provided by source  $S_i$  should not be taken into consideration when calculating the level of trust in information regarding event  $E_i$ . Therefore, the level of trust in information calculated for source  $S_i$  that provides information about event  $E_i$  is:  $Tr(S_i) = (A_d(S_i) + r_i) * E(S_i)$ .

Let us consider  $N$  the number of sources that provide information about event  $E_i$ . Therefore, the trust level in information calculated for event  $E_i$ , considering all  $N$  sources  $S_i$  that provide information about event  $E_i$  is:  $Tr(E_i) = \sum \frac{Tr(S_i)}{N}$ .

We can observe that the level of trust computed for source  $S_i$  is different than the aggregated level of trust computed for event  $E_i$ . If a source  $S_j$  in the system does not confirm the existence of event  $E_i$ , the aggregated trust level in information decreases. Once the sources percentage that infirm an event is higher than the sources percentage that confirm an event, it means a new majority has formed. Therefore, the old majority, that in the initial phase confirmed the event, will be adjusted with a smaller level of reputation. This is because their shared information is contrary with the majority of the information disseminated about the specific event  $E_i$ .

## 5 Experimental Validation and Results

We evaluated the proposed solutions in the context of the MobiWay<sup>2</sup> framework (see Fig. 4). MobiWay targets the development of a robust, open-service and standards-based collaboration platform that will ensure interoperability between various sensing mobile devices and a wide-range of mobility applications.

Data Forwarding Plane (DaFP) is a grouping and forwarding plane, responsible for interlinking and routing the proper urban mobility sources to subscribed services. The new service demands can be satisfied by using dynamic information gathering mechanisms on modern smartphones. The Service Mediating and

<sup>2</sup> <http://mobiway.hpc.pub.ro>.

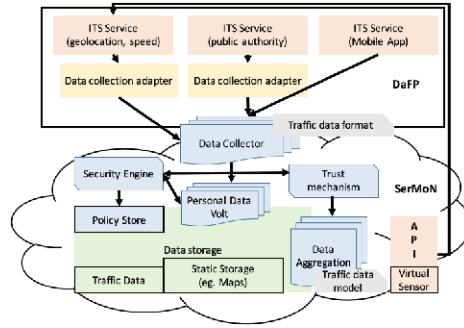


Fig. 4. MobiWay concept and architecture.

Monitoring System's role is to facilitate the introduction and maintenance of services, achieved through mediation, service specific selection and composition, by using the semantic knowledge collected from the Virtual Sensors (ViSe) capabilities, from DaFP control information and the registered services. After data is gathered, it is located in users' Personal Data Vaults (PDVs). Before the data can be extracted and processed in MobiWay, the Security Engine (see Fig. 4) is queried, and a decision is made about the availability of data in regards to that particular processing step. The Security Engine enforces the requirements of the PDVs. To achieve this, every processing request made by an ITS service triggers a policy check. Even more, the Security Engine is responsible for determining the level of trust in the information disseminated in the system. The mechanism used to determine the level of trust in information facilitates the manner in which our model filters the information shared by the users.

Our proposed approach for determining the level of trust in the shared data is based on the information disseminated in the system, particularly its quality parameters, like spatial accuracy regarding the event, the temporal closeness when transmitting data about an event, combined with the reputation specific for ITS sources that transmit data. A pilot implementation for the proposed security model was developed by extending the SerMoN layer with components for routing over OpenStreetMap data [7]. We used open-source solutions like pgRouting [14], extended with own routing schemes that use data coming from traffic, and OpenStreetMap (for the street-level data). Real-time data is provided by users that run the MobiWay app (that collects timestamp, GPS location, and speed). SerMoN is developed as a scalable platform that is capable to store and process a large number of user supplied data per second [5]. We employ the use of private Data Vaults to restrict the publication of potentially sensible data. Each user has the ability to filter the amount of information he wants to share. Moreover, data shared by users is filtered even more using the mechanism that determines the level of trust in information. An event noticed by a user must be confirmed by the majority of the sources that have knowledge about it.

To validate the mechanism used in order to determine the level of trust in the information disseminated in the system, we have designed several experiments. We have considered that event “road accident” or “road block” occurs on a specific street. The event has a particular life period, afterwards it is not valid anymore. Users can share information about the event if they are in the maximum distance range from where they can observe the event. Also, they can share information about the event as long as the event is still occurring (the time when the user’s observation takes places is included in the life period of the event). Otherwise, if one the two conditions above is not met, the trust mechanism disregards the information shared by that source.

In the first scenario, the number of users that affirm the existence of the event is less than half of the total users that report about the existence of the event. In this case, we notice that the routing decision is not influenced by the small percentage of malicious users that disseminate false information in the system. Due to the fact that valid users constitute the majority, in Fig. 5, we can see that the routing decision is changed accordingly, users are avoiding the traffic area affected.

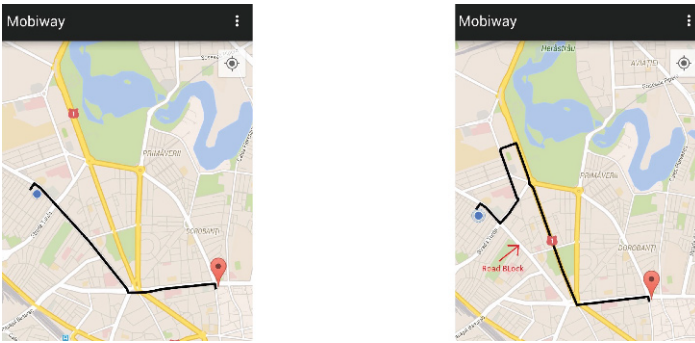


Fig. 5. Routing before and after a road block for a majority of valid users.

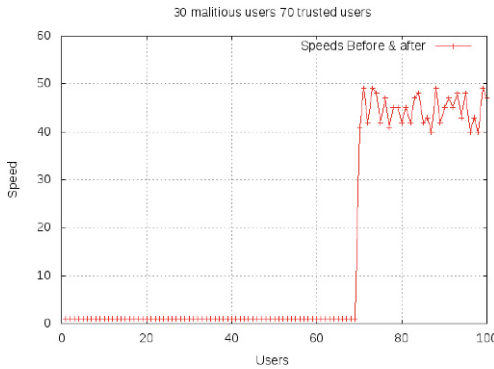
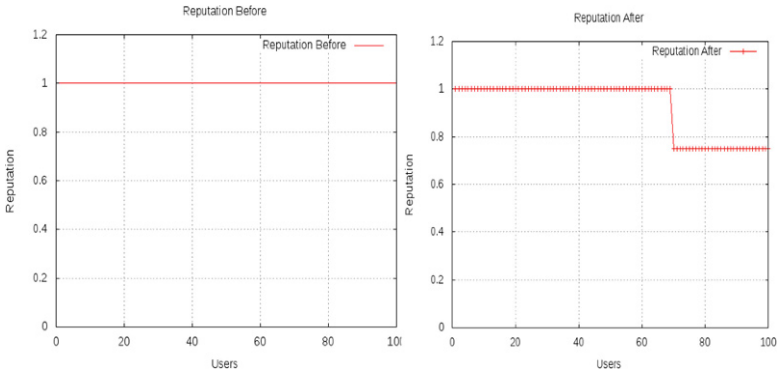


Fig. 6. Users’ speed before and after applying the level of trust mechanism.



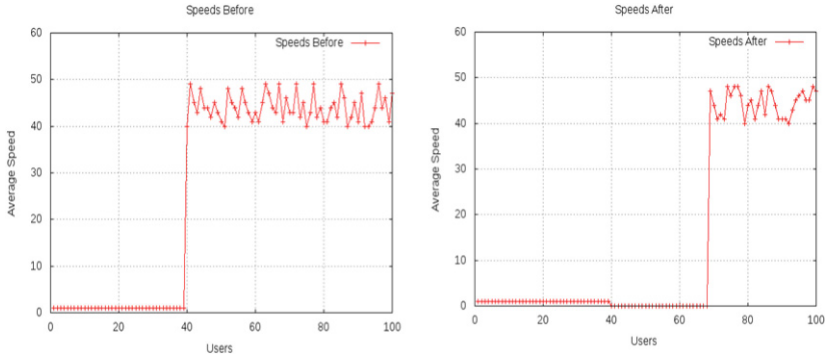
**Fig. 7.** Users’ reputation before and after applying the level of trust mechanism.

Figure 6 shows how the mechanism used to determine the level of trust in information influences the malicious users’ data maintained in their data vaults. If we compare the malicious users’ speed before and after applying the level of trust mechanism, we observe that the data provided by malicious attackers is not taken into consideration. For example, users 71, 78, 85, 92, 99 in Fig. 6 are malicious users (they have high speeds which are not in accordance with the road block in traffic). Because their shared data is not in accordance with the majority, their disseminated information is not taken into account when determining the trust level for the event. Figure 7 shows how the mechanism used to determine the level of trust in information influences user’s reputation in the system. Because their shared data is not in accordance with the majority, their reputation is decreased. This can be observed for users in range [70,100].

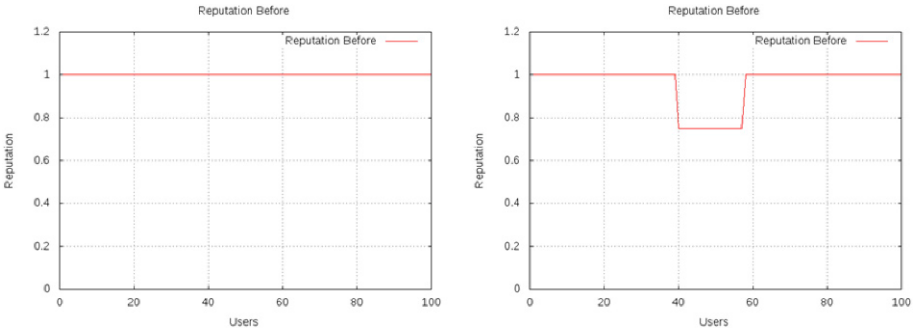
In our second scenario, the number of malicious users that disseminate false reports in the system is more than half of the total users that share information about a particular event, in our case a road block. In this situation, we notice that the routing decision is influenced in a negative manner by the malicious attackers in the system, as the event is not advertised properly.

The mechanism used to determine the level of trust in information has a negative impact over the data maintained in data vaults by valid users. Data specific to valid users will not be processed for determining the level of trust for that particular event, because they do not constitute the majority that shares information about that event (see Fig. 8: the missing data regarding the speed represents nullified information. After applying our filtering algorithm, we discard this data). As being opposed to the majority, their reputation level will be decreased (see Fig. 9).

In order to verify how the proposed approach scales, we have computed various routes, and compared our solution with OSRM. We have considered the case of relying solely on maximum legal speed on the road segments. As weights for pgRouting, we have considered the computed time to travel a road segment (Table 1).



**Fig. 8.** Users’ speed before and after applying the level of trust mechanism in case we have a majority of malicious users.



**Fig. 9.** Users’ reputation before and after applying the level of trust mechanism in case we have a majority of malicious users.

**Table 1.** Comparison - our solution vs OSRM

	OSRM	pgRouting
Route 1: Time to Compute	469 ms	603 ms
Route 1: Length	7.23 km	7.39 km
Route 1: Avg. Speed	48 km/h	49 km/h
Route 2: Time to Compute	32 ms	1165 ms
Route 2: Length	9.91 km	9.50 km
Route 2: Avg. Speed	51 km/h	52 km/h
Route 3: Time to Compute	635 ms	687 ms
Route 3: Length	16.87 km	22.31 km
Route 3: Avg. Speed	51 km/h	52 km/h
Route 4: Time to Compute	240 ms	670 ms
Route 4: Length	6.48 km	6.36 km
Route 4: Avg. Speed	51 km/h	51 km/h

The results show a tendency in our pgRouting solution to always pick roads which are faster, even when a traffic congestion occurs. The drawback is that our solution is slower in terms of time to compute the route on average. This is because OSRM does not use a database to store its information, it preprocesses the information and tries to have the data already in memory for quick access. Also, OSRM uses the Contraction Hierarchies approach, while we have used a simple Dijkstra solution.

## 6 Conclusions

Intelligent Transportation Systems (ITS) attempt to relieve congestion and decrease travel time by assisting drivers in making informed decisions, by selecting better routes, departure times, and even various modes of travel. Most current ITS platforms, unfortunately, fail to include adequate support for dealing with the privacy and trust requirements coming from users voluntarily participating. The data being collected, if aggregated over periods of time, could disclose sensitive information to third-party applications about preferences in routes, or to malicious users about critical traffic events. Moreover, some of the data disseminated by different sources about a specific event occurring in the system may contain contrary information in comparison to the majority of nodes that share information about the same event. Here we propose a new model for the definition of security policies targeted specifically for the ITS. Our approach offers control access to sensitive data before being shared with various ITS applications and services. On the same time, we want users to be able to easily adapt and modify their security policies at any given time. We also propose a solution for dealing with trust in the information users contribute within the ITS. Our proposed approach for determining the level of trust in information is based on quality parameters such as spatial accuracy regarding the event, the temporal closeness when transmitting data about an event, combined with ITS sources' reputation that transmit data. Using the mechanism that determines the level of trust in information, data shared by users in accordance with the accepted security policies is filtered even more. sources that have knowledge about it in order to be advertised in the system.

**Acknowledgment.** The research in this paper is supported by national projects MobiWay - Mobility beyond Individualism (PN-II-PT-PCCA-2013-4-0321), and DataWay - Real-time Data Processing Platform for Smart Cities: Making sense of Big Data (PN-II-RU-TE-2014-4-2731).

## References

1. Bonne, B., Barzan, A., Quax, P., Lamotte, W., Wifipi: involuntary tracking of visitors at mass events. In: 2013 IEEE 14th International Symposium and Workshops on World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–6. IEEE (2013)

2. Caballero-Gil, P., Caballero-Gil, C., Molina-Gil, J., Hernández-Goya, C.: Flexible authentication in vehicular ad hoc networks. In: 15th Asia-Pacific Conference on Communications, APCC 2009, pp. 576–879. IEEE (2009)
3. DataWay: Dataway (2015). <http://dataway.hpc.pub.ro/>
4. De Cristofaro, E., Soriente, C.: Extended capabilities for a privacy-enhanced participatory sensing infrastructure (pepsi). *IEEE Trans. Inf. Forensics Secur.* **8**(12), 2021–2033 (2013)
5. Fratila, C., Dobre, C., Pop, F., Cristea, V.: A transportation control system for urban environments. In: 2012 Third International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), pp. 117–124. IEEE (2012)
6. Gerlach, M., Festag, A., Leinmüller, T., Goldacker, G., Harsch, C.: Security architecture for vehicular communication. In: Workshop on Intelligent Transportation (2007)
7. Haklay, M., Weber, P.: Openstreetmap: user-generated street maps. *IEEE Pervasive Comput.* **7**(4), 12–18 (2008)
8. Krause, M., Hochstatter, I.: Challenges in modelling and using quality of context (QoC). In: Magedanz, T., Karmouch, A., Pierre, S., Venieris, I.S. (eds.) MATA 2005. LNCS, vol. 3744, pp. 324–333. Springer, Heidelberg (2005)
9. Krontiris, I., Dimitriou, T.: A platform for privacy protection of data requesters and data providers in mobile sensing. *Computer Communications* (2015)
10. Li, Q., Cao, G.: Providing privacy-aware incentives for mobile sensing. In: 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 76–84. IEEE (2013)
11. MobiWay. Mobiway (2014). <http://mobiway.hpc.pub.ro/>
12. Mun, M., Hao, S., Mishra, N., Shilton, K., Burke, J., Estrin, D., Hansen, M., Govindan, R.: Personal data vaults: a locus of control for personal data streams. In: Proceedings of the 6th International Conference, p. 17. ACM (2010)
13. Parno, B., Perrig, A.: Challenges in securing vehicular networks. In: Workshop on Hot Topics in Networks (HotNets-IV), pp. 1–6 (2005)
14. Patrushev, A.: Shortest path search for real road networks and dynamic costs with pgrouting. In: Proceedings of the 2008 Free and Open Source Software for Geospatial Conference (2008)
15. Raya, M., Hubaux, J.-P.: Securing vehicular ad hoc networks. *J. Comput. Secur.* **15**(1), 39–68 (2007)
16. Siebel, N.T., Maybank, S.: The advisor visual surveillance system. In: ECCV 2004 Workshop Applications of Computer Vision (ACV), vol. 1. Citeseer (2004)
17. Stampoulis, A., Chai, Z.: A survey of security in vehicular networks. *Project CPSC*, 534 (2007)
18. Wang, N.-W., Huang, Y.-M., Chen, W.-M.: A novel secure communication scheme in vehicular ad hoc networks. *Comput. Commun.* **31**(12), 2827–2837 (2008)
19. Wang, X., Cheng, W., Mohapatra, P., Abdelzaher, T., Artsense: anonymous reputation and trust in participatory sensing. In: 2013 Proceedings IEEE INFOCOM, pp. 2517–2525. IEEE (2013)
20. Wang, Y.-K.: Context awareness and adaptation in mobile learning. In: Proceedings of the 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education, pp. 154–158. IEEE (2004)
21. Yang, D., Fang, X., Xue, G.: Truthful incentive mechanisms for k-anonymity location privacy. In: 2013 Proceedings IEEE INFOCOM, pp. 2994–3002. IEEE (2013)

# **Digital Forensics**



# Exploring the Space of Digital Evidence – Position Paper

Carsten Rudolph<sup>(✉)</sup>

Faculty of Information Technology, Monash University, Clayton, Australia  
carsten.rudolph@monash.edu

**Abstract.** Digital evidence is much more than what is acquired during forensic investigations. In particular when building systems that are supposed to provide secure digital evidence it is necessary to clearly define requirements. Various work on forensic evidence provides different sets of such requirements. Also ISO standardization work is concerned with forensic evidence. However, currently there is no full overview of the different relevant areas for digital evidence that can be used for guidance in the requirement phase of system engineering. Furthermore, a rigorous specification of requirements for digital evidence is missing. Formal methods have been applied to security protocols and other types of requirements, but not to describe the various requirements of digital evidence.

One approach towards defining the available space for digital evidence suggests three dimensions. First, and most obviously, is the time when data is collected, processed, retained and correlated for potential forensic use. This dimension includes data collected at run-time, data collected for particular transactions, in case of deviations, for incidents, “post-mortem” forensic investigations, and the digitization of evidence for court procedures. The second dimension describes the goal for which digital evidence is produced. This can be either for showing compliance, i.e. for proving that somebody was not responsible for some incident or for showing malicious events that happened and to find who did what. Finally, the third dimension consists of the actual information to be documented. Examples are the documentation of the normal system behaviour, compliance information, accidents, safety issues, malicious behaviour, identity information and various relevant parameters. A formal framework for security requirements that was developed for security requirements engineering is one promising candidate to derive a precise characterization of requirements for digital evidence in the different areas of the available evidence space.

This paper is a position paper to drive the discussion and development in forensic readiness and security of digital evidence.

**Keywords:** Forensic readiness · Secure digital evidence · Security engineering · Formal methods

## 1 Introduction

Digital evidence is in many cases considered to be the data collected in the context of forensic evaluations and then used by law enforcement, prosecution,

defense attorneys and others to get additional information and to support their case. In principle, all available digital information can potentially be regarded as digital evidence as long as it provides any information on what actually happened. However, very few systems are directly targeted at producing suitable digital evidence. Examples include secure logging mechanisms or digital signatures on transaction data. Other data, such as complete hard-disk images, often are created in a *post-mortem* forensic evaluation. Therefore, many people relate the term *digital evidence* to forensic evaluations. While at the end the actual forensic use of data can turn any data into digital evidence the reverse view can also be useful. The question is here, which data can potentially be useful as digital evidence and what are the goals for which this data can give evidence. The range is quite large, somebody might collect data to have an advantage in a dispute, police might find remainders of erased files on a hard-disk, paper records can be scanned and digitized to create a large repository of information that becomes digital evidence just by turning it from physical paper records to digital entries in a digital archive. The format can greatly differ. Evidence can be a file, a data record, a complete snapshot of a data-base, a movie, a large set of surveillance videos or a number of virtual machines sitting in a cloud server.

Obviously, as the examples show, the available range of digital data with the potential to become digital evidence is very big. Furthermore, depending on the goal and type of data, many different approaches need to be taken for building a system that collects a particular type in way that is suitable for using it as digital evidence. One important term in this area is *forensic readiness*. It is used for the ability of organisations to provide forensic information, and also on a more technical level for the development of systems to collect, store and provide digital evidence. Forensic readiness on organisational level has been intensively studied. Also, theoretical frameworks have been developed, but there are still many critical issues as shown by Elyas et al. [7]. Further, there is a variety of approaches to provide forensic readiness to particular environments with a strong focus on the challenging area of cloud computing [14] or some particular applications, such as cloud storage [5].

A clear understanding of all different parameters for digital evidence becomes particularly important if a system shall be designed and engineered in a way that it provides secure digital evidence. Building such a system requires a *forensic readiness by design* approach. Recent work by Rahman et al. [15] has proposed a framework for *forensic-by-design* in the context of cyber-physical cloud systems.

Definitions for the term *secure digital evidence* exist. One possible definition was proposed by Kuntze et al. [13]. It states that a data record can be considered secure if it was created authentically by a device for which the following holds:

- The device is physically protected to ensure at least tamper-evidence.
- The data record is securely bound to the identity and status of the device (including running software and configuration) and to all other relevant parameters (such as time, temperature, location, users involved, etc.)
- The data record has not been changed after creation.

Digital Evidence according to this definition comprises the measured value (e.g. a photo and speed measurement) and additional information on the state of the measurement device. This additional information on the state of the measurement device aims to document the operation environment providing evidence that can help lay the foundation for admissibility. This definition assumes that a particular device can be related to the data record. In general, the situation is more complex and the term “device” probably needs to be replaced with a more complex system. Nevertheless, the requirements for secure digital evidence remain and become more complex.

This paper proposes a *digital evidence space* that distinguishes various different types of digital evidence. The goal of this approach is to provide a structured approach that can guide to developing systems providing secure digital evidence for the different areas in this three-dimensional space. The choice of dimensions is not supposed to provide a complete view, but to create a clear distinction with regards to most relevant parameters. The main goal is to clarify the differences and to guide engineering for forensic readiness.

Further, the paper explores the application of a formal security modelling framework to the domain of forensic readiness and digital evidence. The application of formal methods to security has mostly been focused on specific solutions, such as security protocols, access control or firewall configuration. Also generic notions of security have been developed. A more generic formal security modelling framework [12] seems to be suitable to express the various (sometimes contradicting) requirements for secure digital evidence from the different views of stakeholders involved.

The following section introduces the digital evidence space in more detail and then discusses two examples that motivate the choice of the three dimensions and clearly show the differences with regards to engineering for digital forensic readiness. Finally, the applicability of a formal security modelling framework is explored.

This work does not provide a fully developed method. It should be seen as a position paper showing the need for additional research in the areas of forensic readiness and secure digital evidence.

## 2 The Digital Evidence Space

*Forensic readiness* is a term frequently used to describe properties of a digital system that enable the sound forensic use of data produced by the system. Many technical solutions have been proposed to achieve forensic readiness in particular scenarios (e.g. in [8, 16, 18]) and it is also in the process of being standardized [1]. Defining clear criteria for the development of systems that are supposed to be used for forensic evaluation has shown to be very useful for the analysis of the suitability of technology for the generation and collection of data. One example are devices to collect forensic data from logging information in IT networks, such as enterprise networks. Going through examples of digital data with potential forensic quickly shows that some requirements are applicable for many scenarios and can be considered more or less generic, while others are very different.

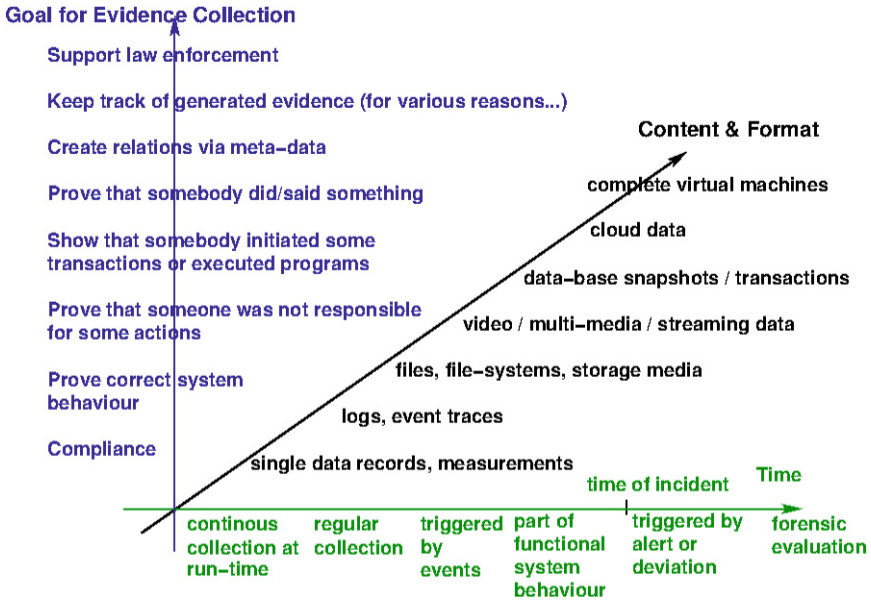


Fig. 1. Graphical view of the digital evidence space with examples for the three dimensions

The idea of the digital evidence space is to identify characteristics that can influence the requirements for forensic readiness in different application scenarios. Figure 1 shows a first attempt that uses three dimensions to highlight differences between various scenarios where data is produced that can be forensically relevant. The three dimensions are *time of digital evidence collection*, *the goal for evidence collection*, and *the content and format of digital evidence*. The following paragraphs explain the choice of these dimensions in more detail.

### 2.1 Time of Digital Evidence Collection

The time that information is created and then contained in digital form is very crucial for the actual technical process to generate secure digital evidence. The range goes from continuous collection at run-time, via event-triggered collection before or after incidents have occurred, collecting digital data as part of forensic investigations, to finally the digitization of other evidence for use in digitized court processes.

In addition to the time of the creation it is also relevant to decide on the time data is stored. This topic includes issues such as secure archiving but also the question of data deletion if, for example, regulations permit to keep c data only for a particular time.

## 2.2 Goals for Digital Evidence Collection

In the view of forensic investigations, the goal of forensic evidence is to provide information on what happened in relevance to a particular case or incident. However, when looking from the view of system engineering and the targeted generation of digital evidence, the situation is much more complex. Different goals require different technologies for realization. One strong motivation, in particular for enterprises, is to collect evidence to be always able to show compliance to various regulations. In the context of insurances it might be an advantage to be able to show that a system was always working correctly as specified. Then, the goal is to prove that the owner of the system or some operator followed all the rules.

Administrators in IT systems often have access to a lot of information and can change relevant configurations. In this context digital evidence can have two goals. First, one might want to control and prove that an administrator has maliciously (or accidentally) accessed some information or created damage in the system. Part of this can also be evidence that shows that the administrator would have been able to do so. Second, the reverse, namely showing that an administrator was not responsible and has not done any malicious action is also useful information.

The range of possible goals is very big. One might want to prove that some actions have been initiated/completed at some particular time. Digital data can show who that somebody did something, was in a particular place at a particular time or said something.

The task of engineering a forensically ready system gets more difficult if the goals for collecting digital evidence become more complex. Digital evidence can in principle be used to provide enough information to evaluate complex incidents. Management solutions are required if one needs to keep track of all generated evidence in a system (e.g. in order to check compliance with privacy regulation, or continuously monitor completeness and correctness of evidence for future use). Another goal is to support law enforcement and forensic investigations. Technical solutions in this field include access interfaces for lawful interception, collecting massive surveillance data and others.

## 2.3 Content and Format of Digital Evidence

The actual content and format of digital data makes a difference for implementing forensically ready solutions. Obviously, video streams need a different treatment as hard-disk images or single data records produced by calibrated devices, such as metering instruments or speed cameras. However, the range is much bigger. Even cases that seem to be very clear, such as hard disks, require careful consideration. Physical images need to be distinguished from data constructed from a RAID array or from encrypted hard-disks. Virtual disks in the cloud increase the complexity of the situation. Further, evidence can occur in the form of complete data-bases, as complete virtual machines somewhere in the cloud, or might be spread out as distributed information over large technical infrastructures.

### 3 Examples in the Digital Evidence Space

The size and variance of the digital evidence space shows that many different requirements can be identified for forensic readiness. The following sections briefly discuss two examples to highlight the relevance of the correct identification of requirements during the engineering process of forensically ready systems.

#### 3.1 Lawful Interception of Voice over IP Communication

In many countries, law enforcement can apply for the right to intercept communication data [9, 17]. For good reasons, the process of lawful interception usually is highly regulated [3]; however, the admissibility of data records achieved through lawful interception leaves various open questions.

*Goals.* The goal of the collection of data is rather clear. One would like to get information about what people say and how they are related to a particular incident or criminal activity.

*Time.* Lawful interception usually should occur before an actual incident happens or after it has happened to support the investigations. In most jurisdictions there needs to be a clearly defined trigger to start the process and also storage and evaluation of the collected data is regulated.

*Content and format.* In principle, the content and format of data in lawful interception is clearly defined. Streams of voice over IP traffic are collected. However, keeping this data as secure digital evidence is not straightforward. To prevent manipulations, data need to be digitally signed. Further, parts of the communication might need to be deleted for privacy reasons without invalidating the signatures, but still clearly showing where data was deleted. There are possible realizations for these requirements available, but they are not used in practice.

#### 3.2 Calibrated Devices for Documentation in Precise Farming

The evolution of technology in agricultural processes has changed the way that different steps in farming are executed [4]. Examples include the calculation and planning for fertilizers or creating the correct mixture of chemicals for plant protection [19]. These technologies provide the means for a very precise use of seeding material, fertilizer, etc. Large farms can be managed and controlled based on recorded data. Especially in the growing market for sustainable agriculture and eco-farming, there is a need for a qualitative value monitoring of the processes and the materials used such as seeds, fertilizers etc. Further, farming subsidiary schemes for funding a farmer to grow particular crops are automatically controlled using data records produced by the machines used in these processes. Parameters include GPS positions to calculate the location and size of the area and the types of crop [21].

*Goals.* The goals in this scenario are much more complex. A typical scenario consists of devices installed in the different types of farm machinery (tractors, harvesters, operating devices like fertilizer distribution equipment) and a central computer that collects and evaluates all the different data records. Depending on the particular use of the data records, different types of requirements exist. If genetically manipulated crops are used, it is very important to be able to reliably document where exactly the crops have been planted. In the case of fertilizers and pesticides or fungicides a wrong calculation of the amount of chemicals used in a particular area can create extensive damage. In the case of a question as to the predicted origin of farm produce or proper verification of the innocuousness of pesticides or fungicides, etc., this evidence is relevant. This becomes even more important as consumer concern increases regarding eco-farming products and the evidence of eco farming. Further, when the data is used as proof for subsidiaries as to which crop was grown in which area, manipulating these data records can be used to calculate wrongful claims.

*Time.* For precise farming, the collection of digital evidence is a continuous process. Farming machinery continuously creates and stores logs with relevant information. This information needs to be aggregated in regular intervals to produce the actual evidence records.

*Content and format.* The content and format is prescribed by the parameters and data available in the different machinery. However, the evaluation and interpretation of this data requires suitable tools and support. Furthermore, the security of this data is critical. Manipulations can result in large financial losses or in the fraudulent acquisition of subsidies. Current precise farming solutions do not satisfy requirements of forensic readiness and secure digital evidence.

Assuming correctness of the sensor measurements, generic concepts of a device for generating secure evidence can be applied directly; however, devices for secure documentation in precise farming technology need to consider that various sensors contribute to the data records and that most of them potentially can be manipulated. Therefore, a solution must be designed that combines the attestation of the platform with some kind of run-time validation for the correctness of the sensor information. Additionally, as physical manipulations to the sensor or to the environment of the sensor are possible, a straightforward technical solution is difficult. Nevertheless, using technologies such as Trusted Computing as well as protection mechanisms for sensors, or plausibility calculations for a collection of sensors, these devices can be protected.

Additionally data transfer between the devices and the central storage unit must be secured and enabled for overall verification of the data, as well as of the condition of the sensors. A process must be defined for creation of a complete chain of evidence.

## 4 Formal Specifications of Requirements for Secure Digital Evidence

Laws and court procedures deliberately are not formally specified and leave room for interpretation and case-by-case decisions by humans looking at the context of a particular case and the individual situation. Therefore, a formal representation for laws and regulations will not be suitable. However, for digital evidence the situation is a bit different. Before evidence is presented at court, it is generated and processed by digital devices. On the level of these devices, it is therefore possible to clearly show what set of different behaviours might have led to this particular digital evidence being generated and what the assumptions for these behaviours are. Unfortunately, this information is often not fully available to people having to judge about the value of a particular piece of digital evidence. Thus, on the level of digital evidence, it can be useful to explore how formal methods can provide a basis for a more rigorous treatment of digital evidence and might lead to improvements in best practice rules on how to deal with this type of evidence. Further, developing forensically ready systems also needs precise requirements specifications. Recent work has proposed to use formal models to represent adversary models similar to the so-called Dolev-Yao model for cryptographic protocols. One such proposal was developed for the area of mobile devices [6]. Such models can complement this generic framework by providing a foundation for suitable system assumptions representing possible attacks, i.e. properties that can be assumed to be satisfied even in the presence of a powerful attacker.

### 4.1 A Generic Framework for Security Requirements Specifications

The digital evidence space shows that there is a large variety of types of digital evidence and goals for this evidence to be collected. Therefore, a formal framework to describe requirements within this space also needs to support a large variety of security properties and cannot be restricted to particular attack scenarios and attacker types. One possible candidate is the security modelling framework introduced by Gürgens et al. [12].

### 4.2 Introduction of a Formal Security Modelling Framework

The *behaviour*  $S$  of a discrete system can be formally described by the set of its possible sequences of actions (traces). Therefore  $S \subseteq \Sigma^*$  holds where  $\Sigma$  is the set of all actions of the system, and  $\Sigma^*$  is the set of all finite sequences of elements of  $\Sigma$ , including the empty sequence denoted by  $\varepsilon$ . This terminology originates from the theory of formal languages, where  $\Sigma$  is called the alphabet, the elements of  $\Sigma$  are called letters, the elements of  $\Sigma^*$  are referred to as words and the subsets of  $\Sigma^*$  as formal languages. Words can be composed: if  $u$  and  $v$  are words, then  $uv$  is also a word. This operation is called the *concatenation*; especially  $\varepsilon u = u\varepsilon = u$ . A word  $u$  is called a *prefix* of a word  $v$  if there is a word



$x$  such that  $v = ux$ . The set of all prefixes of a word  $u$  is denoted by  $\text{pre}(u)$ ;  $\varepsilon \in \text{pre}(u)$  holds for every word  $u$ . We denote the set of letters in a word  $u$  by  $\text{alph}(u)$ .

Formal languages which describe system behaviour have the characteristic that  $\text{pre}(u) \subseteq S$  holds for every word  $u \in S$ . Such languages are called *prefix closed*. System behaviour is thus described by prefix closed formal languages.

The set of all possible continuations of a word  $u \in S$  is formally expressed by the *left quotient*  $u^{-1}(S) = \{y \in \Sigma^* \mid uy \in S\}$ .

In the digital evidence space it is obvious, that different stakeholders have a different view of the system and can therefore rely on different assumptions on the system. Therefore, also what can be deduced from particular instances of digital evidence also depends on who is looking at it. Ultimately, when used at court, a commonly acceptable set of assumptions must be used which can also take into account inter-dependencies between different pieces of digital evidence and the particular context. However, in order to correctly specify requirements on digital evidence, one needs to distinguish different views by stakeholders (or participants) in the system.

For any participant  $P$  its knowledge about the global system behaviour  $W_P \subseteq \Sigma^*$  is considered to be part of the system specification. It expresses system assumptions by  $P$ . For example, if  $P$  trusts that a particular cryptographic algorithm cannot be broken with a reasonably large probability, all behaviours that require this algorithm to be broken will be excluded from  $W_P$ . Also knowledge about physical security or other constrained can be modelled in  $W_P$ . Note that  $W_P$  does not contain information on specified system behaviour that is not enforced. It is necessary that malicious actions can alter the system in any possible ways that do not contradict the assumptions taken for  $W_P$ . Every behaviour which is not explicitly excluded by some  $W_P$  can happen in the view of  $P$ . System assumptions are really crucial. A system  $S$  may satisfy a security property with respect to  $W_P$ , but may fail to satisfy the same property with respect to a different  $\tilde{W}_P$ , if  $P$  considers more actions to be possible on account of weaker system assumptions and security mechanisms.

Another important information is which parts of the system are actually visible to  $P$ . No participant can see all parts of a system. Thus, what  $P$  can see is just a projection of the system. Therefore, a *local view* is defined, that expresses what parts of the system behaviour are visible. Note, that for some actions  $P$  might see the action itself and some parameters, but not all parameters. One example is an encrypted message where  $P$  might have seen that the message was send and might also be able to look into meta-data contained in the header, while the actual payload is encrypted and therefore no in  $P$ 's local view. The local view of an agent  $P$  on  $\Sigma$  is denoted by  $\lambda_P : \Sigma^* \rightarrow \Sigma_P^*$ . Note that in contrast to the original modelling framework that was targeted at security protocols, in the digital evidence space one cannot assume that the union of all local views of all participants will cover the complete system behaviour. For a sequence of actions  $\omega \in \Sigma^*$  and agent  $P \in \mathbb{P}$ ,  $\lambda_P^{-1}(\lambda_P(\omega)) \subseteq \Sigma^*$  is the set of all sequences that look exactly the same from  $P$ 's local view after  $\omega$  has happened.

Depending on its knowledge about the system  $S$ , underlying security mechanisms and system assumptions,  $P$  does not consider all sequences in  $\lambda_P^{-1}(\lambda_P(\omega))$  possible. Thus it can use its knowledge to reduce this set:  $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$  describes all sequences of actions  $P$  considers to be possible when  $\omega$  has happened. The set  $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$  is similar to the possible worlds semantics that have been defined for authentication logics in the context of cryptographic protocols [2, 20], but this notion is more general because for authentication logics  $\lambda_P$  and  $W_P$  are fixed for all systems (with fixed attacker models).

A variety of notions of generic security properties have been defined within this formal security modelling framework. They include authenticity and non-repudiation [10], and parameter-confidentiality [11]. In addition, the framework has the flexibility to define particular processes or behaviours as security requirements to be secured against malicious interference. Thus, security can also be defined with respect to a *mission* a system should fulfill.

### 4.3 Applying This Framework in the Digital Evidence Space

Before actually defining security requirements for digital evidence, the formal security modelling framework needs a specification of the system and assumptions by different participants or stakeholders. This specification is not necessarily very technical. Indeed, it can be done on the level of business processes or other high-level processes. This specification can then be refined towards a more technical level for the actual digital evidence to be considered. In fact, often the technical evaluation of digital evidence is used to derive knowledge of higher-level behaviour and processes. Therefore, it would not be useful to restrict modelling to technical level of protocols and computing systems. The models also need to include human interaction with the systems and can include assumptions about human behaviour. Note that the framework requires all assumptions that restrict possible system behaviour to be made in an explicit way. This is essential for the evaluation of digital evidence.

The formal representation of security properties can be used for three different tasks. First, it can be used to describe assumptions about evidence itself, e.g. which parameters cannot be manipulated, whether it happened at a particular time or in particular order of actions, in whose local view the actual collection of digital evidence was present, etc. Second, the properties that the piece of digital evidence can provide information about and finally, requirements on generation, storage, handling, and evaluation need to be expressed.

The model needs to reflect all three dimensions of the digital evidence space, namely the goals for collection, the time when it was collected and the content and format. These dimensions have to be considered in the system model, details for action, in security properties defined for digital evidence, in local views defined for different participants and finally also assumptions on attacks to be excluded can depend on the goals for evidence collection.

In order to be able to reason about time, there needs to be a way to reflect time. All requirements, properties and also different system views are relative to actions defined in the model. Obviously, there already is a notion of relative time

due to behaviours being seen as sequences of actions. However, this type of *has happened before* or *has happened after* is not sufficient. Therefore, parameters of actions can be used to model various types of absolute and relative time.

Different content and format is usually not explicitly included in the model, but the model needs to provide sufficient information to syntactically and semantically distinguish different formats and contents. generation, storage, handling and evaluation events need to behave in accordance with the different types of data. Furthermore, different sets of assumptions should be defined as a kind of *best practice* rule-set for particular types of content and representations.

Finally, the actual goals of collecting digital evidence need to be modelled in terms of precisely specified security properties. The framework provides a wide range of options that should enable a formal (or semi-formal, but still exact) way of describing properties that digital evidence shall provide reliable information on.

A large part of digital evidence will require authenticity and non-repudiation properties. Non-repudiation means that some participant cannot deny that some particular action has happened. Thus, non-repudiation needs to involve a proof that can later be checked and demonstrated to a third party. Note that a witness cannot provide a proof in any technical sense. If a person  $P$  considers non-repudiation to be satisfied, it means that  $P$  can provide a proof that can be verified by a third party. Secure digital evidence often has the goal to provide non-repudiation where the set of assumptions should be as small and as precise as possible.

Authenticity is a slightly weaker requirement. Authenticity means that  $P$  can know that a particular action has happened, but might not be able to generate a proof for this. Thus, a witness might know that some action was authentic, but as there is no proof in a technical sense, it is just the report of the witness that can be used. However, digital evidence, such as log files or mail records might show that  $P$  indeed was in the position to check the authenticity of some action, just without actually producing secure digital evidence for non-repudiation. One example can be e-mail sent from  $Q$  to  $P$ . E-mail records might not be considered secure digital evidence (depending on the context) as e-mail traffic can be forged. However,  $P$  might have called  $Q$  on the phone (shown by phone records metadata). Thus,  $P$  was in the position to verify that  $Q$  has sent this mail. However, as the content of the phone call was not recorded, there is no evidence for non-repudiation.

Confidentiality is an obvious requirement for the collection and storage of digital evidence. But these are processes for managing digital evidence and not concerned with the security of digital evidence as such. Confidentiality is tricky and rather unusual as a requirement of what can be shown by digital evidence. However, there are some goals for digital evidence where confidentiality can play a strong role. One example is the goal to *prove that somebody was not responsible for some actions*. If digital evidence can show that some person cannot have had knowledge about some particular information (i.e. this information was confidential) one could deduce that this person cannot have executed some actions for which this information would have been necessary.

Digital evidence can also be used to prove that some particular processes or sequences of actions (possibly parameterised) have happened. This requirements can be seen as a generalization of non-repudiation or authenticity requirements. However, it has a number of additional parameters, such as the order of actions that might have happened.

The security modelling framework provides formal notions for all these concepts. Thus, formal requirements specifications can be developed for various kinds of digital evidence. These formal specifications can have different uses. First, they clearly show under which assumptions some digital evidence can be considered *secure* digital evidence for a set of precise actions. Second, formal specifications can provide the link between rather technical digital evidence to processes and higher-level behaviours. Finally, in the area of forensic readiness, formal representations can be used to guide the development of products and to develop precise standards for forensic readiness.

## 5 Conclusions

The two small examples show that the variance of possible digital evidence is very big. Thus, we cannot expect simple solutions for forensic readiness and the creation of secure digital evidence, that work in all different areas. The complexity of the digital evidence space gives some insight into the task of building forensically ready systems. Of course, good and reliable solutions exist for some areas in the digital evidence space, in particular in the field of forensic investigations. Nevertheless, actual processes for concepts such as *forensic readiness by design* do not exist for large parts of the digital evidence space. Thus, the space can provide some guidance for future developments in the area of forensic readiness by design.

The current version of the digital evidence space as shown in this paper was developed during discussions with many international scientists and practitioners in the field of digital forensics, digital archiving and law. It should be considered to be a working document that will grow and be populated in the future. Also, the three dimensions probably provide a rather simplified view. However, these dimensions have shown to be useful to pinpoint some important characteristics of digital evidence and therefore, seems to provide a useful view on different scenarios. Nevertheless, future developments might require an extended multi-dimensional digital evidence space to cover all relevant characteristics.

The evaluation of using a formal security modelling framework shows that formal mechanisms do exist that can be used to precisely define which properties are required for secure digital evidence. Further, this formal framework provides a more abstract view on security properties that clearly shows which information is required for precise specifications of the different properties while providing formal (operational) semantics for the properties that can again be directly mapped to particular behaviours in the system. Finally, the framework also provides security-preserving abstractions through language homomorphisms that can be used to also formally link rather technical secure digital evidence to higher-level representations of processes.

Obviously, digital forensics is more than *post mortem* analyses of systems or collection of log files. Proper notions for secure digital evidence can help to develop systems that provide reliable evidence where it is necessary and other systems that deliberately are not suitable to provide evidence (e.g. for privacy reasons).

**Acknowledgements.** The authors thank all participants of the Dagstuhl Seminar *Digital Evidence and Forensic Readiness* 2014 for useful feedback to an early version of the digital evidence space developed on a black board at Schloss Dagstuhl, and for intensive and fruitful discussions on the topic.

## References

1. ISO/IEC DIS 27043. Information technology - security techniques - incident investigation principles and processes. Under development
2. Abadi, M., Tuttle, M.R.: A semantics for a logic of authentication. In: Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Canada, pp. 201–216, August 1991
3. Akdeniz, Y., Taylor, N., Walker, C.: Regulation of Investigatory Powers Act 2000 (1): Bigbrother. gov. uk: State surveillance in the age of information and rights [2001]. *Criminal Law Review*, pp. 73–90 (2001)
4. Auernhammer, H.: Precision farming the environmental challenge. *Comput. Electron. Agric.* **30**(1), 31–43 (2001)
5. Quick, D., Martini, B., Choo, R.: *Cloud Storage Forensics*. Syngress, Waltham (2013)
6. Do, Q., Martini, B., Choo, K.R.: A forensically sound adversary model for mobile devices. *PLoS ONE* **10**(9), e0138449 (2015)
7. Elyas, M., Ahmad, A., Maynard, S.B., Lonie, A.: Digital forensic readiness: expert perspectives on a theoretical framework. *Comput. Secur.* **52**, 70–89 (2015)
8. Endicott-Popovsky, B., Frincke, D., Taylor, C.: A theoretical framework for organizational network forensic readiness. *J. Comput.* **2**(3), 1–11 (2007)
9. Gleave, S.: The mechanics of lawful interception. *Netw. Secur.* **2007**(5), 8–11 (2007)
10. Gürgens, S., Ochsenschläger, P., Rudolph, C.: Authenticity and provability - a formal framework. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) *InfraSec 2002*. LNCS, vol. 2437, pp. 227–245. Springer, Heidelberg (2002)
11. Gürgens, S., Ochsenschläger, P., Rudolph, C.: Abstractions preserving parameter confidentiality. In: di Vimercati, S.C., Syverson, P.F., Gollmann, D. (eds.) *ESORICS 2005*. LNCS, vol. 3679, pp. 418–437. Springer, Heidelberg (2005)
12. Gürgens, S., Ochsenschläger, P., Rudolph, C.: On a formal framework for security properties. *Int. Comput. Stan. Interface J. (CSI)* **27**(5), 457–466 (2005). Special issue on formal methods, techniques, tools for secure, reliable applications
13. Kuntze, N., Rudolph, C., Alva, A., Endicott-Popovsky, B., Christiansen, J., Kemmerich, T.: On the creation of reliable digital evidence. In: Peterson, G., Sheno, S. (eds.) *Advances in Digital Forensics VIII*. IFIP Advances in Information and Communication Technology, vol. 383, pp. 3–17. Springer, Heidelberg (2012)
14. Kebande, V.R., Venter, H.S.: Adding event reconstruction to a cloud forensic readiness model. In: *Information Security for South Africa (ISSA) 2015*, pp. 1–9, August 2015

15. Rahman, N.H., Glisson, W.B., Yang, Y., Choo, K.R.: Forensic-by-design framework for cyber-physical cloud systems. *IEEE Cloud Comput.* **3**(1), 50–59 (2016)
16. Reddy, K., Venter, H.S., Olivier, M.S.: Using time-driven activity-based costing to manage digital forensic readiness in large organisations. *Inf. Syst. Front.* **14**(5), 1061–1077 (2012)
17. ETC-STAG. Security techniques advisory group (stag); definition of user requirements for lawful interception of telecommunications: requirements of the law enforcement agencies (1996)
18. Van Staden, R.F., Venter, H.S.: Using performance monitoring software to implement digital forensics readiness. In: 8th Annual IFIP WG 11.9 International Conference on Digital Forensics (2011)
19. Wang, N., Zhang, N., Wang, M.: Wireless sensors in agriculture and food industry: recent development and future perspective. *Comput. Electron. Agric.* **50**(1), 1–14 (2006)
20. Wedel, G., Kessler, V.: Formal semantics for authentication logics. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) *ESORICS 1996*. LNCS, vol. 1146, pp. 219–241. Springer, Heidelberg (1996)
21. Wolf, S.A., Wood, S.D.: Precision farming: environmental legitimization, commodification of information, and industrial coordination. *Rural Sociol.* **62**(2), 180–206 (1997)

# **Lightweight Security**

# Towards Lightweight Anonymous Entity Authentication for IoT Applications

Yanjiang Yang<sup>1(✉)</sup>, Haibin Cai<sup>2(✉)</sup>, Zhuo Wei<sup>1</sup>, Haibing Lu<sup>3</sup>,  
and Kim-Kwang Raymond Choo<sup>4</sup>

<sup>1</sup> Huawei Singapore Research Center, Singapore, Singapore  
{yang.yanjiang,wei.zhuo}@huawei.com

<sup>2</sup> East China Normal University, Shanghai, China  
hbcai@sei.ecnu.edu.cn

<sup>3</sup> Santa Clara University, Santa Clara, USA  
hlu@scu.edu

<sup>4</sup> University of South Australia, Adelaide, Australia  
raymond.choo@fulbrightmail.org

**Abstract.** Preservation of individual privacy is an important issue in future IoT applications, which calls for lightweight anonymous entity authentication solutions that can be executed efficiently upon a wide range of resource-constrained IoT devices and gadgets. Existing anonymous credential techniques are not well fitted to the setting of IoT, and it is especially so when credential revocation support is considered. In this paper, leveraging on dynamic accumulator we propose a lightweight anonymous entity authentication scheme with outsource-able witness update, solving the main bottleneck of anonymous credentials. We further improve the performance of the scheme with the idea of self-blinding, in such a way that the computation by the prover works entirely in the compact bilinear group of bilinear map. Our performance evaluation shows that the proposed schemes are good for resource-constrained devices.

**Keywords:** Anonymous entity authentication · Lightweight · IoT · Anonymous credential · Group signature · Dynamic accumulator · Credential revocation

## 1 Introduction

We are entering the era of Internet of Things (IoT), where a wide range of smart devices embedded with electronics, software, sensors, and network connectivity are inter-connected, collecting and exchanging data. Individual privacy is a particularly important issue in IoT, as users' information can be easily gathered and dossier-ed in an inter-connected environment of a large scale. It is well accepted

---

H. Cai—Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China 200062.



that anonymous entity authentication whereby IoT devices can be authenticated in an anonymous manner is a fundamental step towards the protection of individual privacy.

A particular constraint for anonymous entity authentication techniques in IoT is that many IoT devices are limited in resources such as computation, bandwidth, storage and power supply. This calls for *lightweight* anonymous entity authentication schemes that can be executed efficiently upon the resource-constrained IoT devices. Existing anonymous entity authentication techniques have not been well optimized for IoT devices.

**State-of-the-art Anonymous Entity Authentication.** Anonymous credential, e.g., [6, 15, 17, 20], is a major anonymous entity authentication technique. Specifically, an anonymous credential enables its holder (say Alice) to convince a verifying entity (e.g., a service provider) of a certain property (e.g. her qualification to access the provider's service), while withholding any identifying information from the verifier. This feature is more precisely specified as *unlinkability* – different uses of the same credential to the verifier cannot be linked or recognized as different instances under the same credential.

Group signature, e.g., [1, 3, 8, 16, 22, 23], can also be used for anonymous entity authentication. Group signature attains unlinkability of signatures generated by the same signer. That is, a signature can only be recognized as valid, but cannot be attributed to any individual signer. In line with its intended applications for unlinkable signing of documents such as e-contracts, group signature provides a tracing mechanism, whereby a designated trusted authority can “open” signatures to reveal the actual signer in case of disputes.

From a constructional point of view, anonymous credential and group signature are the same, implemented by means of zero-knowledge proof of knowledge proving the possession of certain secrets in an anonymous manner. Hence, we do not specifically differentiate them hereafter. State-of-the-art anonymous credential schemes such as [4, 7, 8, 17, 19, 20] often invoke costly zero-knowledge proof of knowledge, and as such, they are not satisfactory for weak IoT devices such as sensors, smart cards, e-passports, and vehicular devices. Specifically, the RSA-based schemes (e.g., [20, 21]) entail expensive zero-knowledge range proofs, which generate tens of kilobits in communication and tens of modular exponentiation operations. Although the IDEMIX anonymous credential and the Direct Anonymous Attestation (both are RSA-based) have been implemented on JAVA card, e.g., [5, 9, 17, 34], the implementations either are not practically efficient enough or do not well address credential revocation. The state-of-the-art bilinear map-based schemes such as [7, 8, 19] rely on bilinear pairing operations or computations over large bilinear groups. The resulting heavy computation/communication overheads are unfavorable for IoT devices.

**Credential Revocation in Anonymous Credential.** Credential revocation is another thorny issue in anonymous credential. There are usually two credential revocation approaches. The first one is verifier-local revocation [13, 17], whereby revoked credentials are collected in a list managed by the verifier; during credential showing (for anonymous entity authentication), the verifier needs to check

the credential in use against all the revoked credentials to determine whether the credential has been revoked or not. A main issue with verifier-local revocation is that the computation overhead of the verifier is linear with the total number of revoked credentials, which may grow quite large over time.

The second approach is to use *dynamic accumulator*, e.g., [19,20,33]. A dynamic accumulator allows a set of values to be accumulated into a single value called the *accumulator*, and for each accumulated value, there exists a *witness* evidencing that the accumulated value is indeed contained in the accumulator. Prominently, credential revocation based on dynamic accumulator solves the linear computation problem in the verifier-local approach, but it has its own issue: every time a credential is revoked, all the remaining legitimate users need to update their witnesses based on the new accumulator. Whilst a user can choose to perform witness update in a batch mode and thus needs not be online around the clock, the computation overhead is accumulated accordingly. The problem becomes particularly acute for IoT devices with limited resources.

**Our Contributions.** We are thus motivated to develop lightweight anonymous credential intended for IoT devices. Our rationale has several ingredients: (1) we stick to the use of dynamic accumulator for credential revocation, due to the avoidance of linear computation by the verifier; (2) to address the witness update issue, we adopt the strategy of outsourcing the task of witness update to, e.g., a cloud server, so that users need to neither get connected all the time, nor do batch update; (3) we observe that a dynamic accumulator itself can be used as an anonymous credential scheme, but it does not directly work under the strategy of witness update outsourcing. In particular, our contributions are as follows:

- Under the rationale of witness update outsourcing, we present a lightweight anonymous credential scheme with outsource-able witness update, by extending dynamic accumulator in [33].
- We further improve the performance of the scheme with the idea of self-blinding in [36,38], such that computation by the prover works entirely in  $G_1$  of an asymmetric bilinear map  $\hat{e} : G_1 \times G_2 \rightarrow G_T$ , and the resulting authentication data only consist of group elements of  $G_1$ . As  $G_1$  is much more compact than  $G_2$  and  $G_T$ , our scheme attains much better performance than existing anonymous credentials. Our empirical experimental results corroborate the analytical evaluation.

**Organization.** In Sect. 2, we provide a review of related work. Preliminary knowledge is given in Sect. 3. An anonymous credential construction with outsource-able witness update is proposed in Sect. 4, followed by an improved scheme under the self-blinding strategy in Sect. 5. Performance evaluation is presented in Sects. 6 and 7 contains the concluding remarks.

## 2 Related Work

Privacy-preserving entity authentication is an extensively studied area, with many different sorts of cryptographic primitives having been proposed. Below we restrain ourselves to anonymous credential and related techniques.

As discussed above, anonymous credential [6, 15, 17, 19, 20] and group signature [1, 3, 8, 13, 16, 22] are major cryptographic primitives allowing for entity authentication to proceed in an anonymous manner where unlinkability of authenticating entities is achieved. For some of the schemes in [6, 20], a credential can only be used once, and any reuse would be linkable. These one-use anonymous credentials have good performance, but the credential holders have to acquire new credentials from the credential issuer repetitively.  $k$ -TAA ( $k$ -Times Anonymous Authentication) [4, 35] improves one-use anonymous credential such that a credential can be used unlinkably for up to  $k$  times. [29] aims to mitigating the concern of single credential issuer by proposing decentralized anonymous credential issuing.

The self-blindable certificate scheme [36] is essentially lightweight anonymous credential, and the light-weighted-ness comes from its distinctive self-blinding mechanism. Despite its efficiency advantage, it lacks satisfactory credential revocation support. Two revocation solutions are proposed in [36]. One is to enforce short-term certificates by embedding with each certificate an expiry date. However, there are several drawbacks with this approach: (1) it downgrades privacy protection, e.g., a unique expiry date identifies its bearer. In the general case, an expiry date links all certificates with the same expiry date, thus narrowing the space for linking a certificate; (2) it does not allow for immediate revocation, and a certificate can only be invalidated upon the expiry date, even if it needs to be revoked instantly. In addition, it has the problem of determining a proper life span for a certificate. The other revocation approach resembles On-line Certificate Status Protocol (OCSP), where the certificate holder has to encrypt the blinding factors used in generating a blinded certificate to a trusted third party (TTP). The TTP helps the verifier determine the revocation status of the blinded certificate by de-blinding the blinded certificate. However, it is widely known that establishing an online trusted server implies a myriad of difficulties and drawbacks.

Structure preserving anonymous credentials (e.g., [2, 10]) have been proposed, making use of the non-interactive zero-knowledge proof systems [28]. An anonymous credential scheme is *structure preserving* if all of its public keys, messages, credentials, and authentication data (generated when showing a credential) are group elements of  $G_1$  and  $G_2$ . Structure preserving anonymous credentials are intended to show credentials non-interactively while avoid the usual practice of the Fiat-Shamir heuristics [27]. In comparison, the authentication data generated from showing a credential in our improved scheme (Sect. 5) include only group elements of  $G_1$ , more promising from an efficiency perspective. All in all, the objective of our proposal is a lightweight anonymous authentication primitive suitable for resource-constrained IoT devices, rather than strengthened security. Thus we still recur to the Fiat-Shamir heuristics to achieve no-interactive

credential showing, and we believe that this is not an issue in practice if care is taken in implementing the corresponding cryptographic hash function.

Following the trend of using lattices for constructing cryptographic primitives, many anonymous credential and group signature schemes based on lattices have been proposed recently, e.g., [24, 26, 30, 31]. However, due to the large key size (at least), lattice-based anonymous credentials (and group signature) are far from practicality upon resource-constraint weak IoT devices.

### 3 Preliminaries

Throughout the paper, we use  $s \in_R S$  to denote that an element  $s$  is randomly chosen from a set  $S$ .

**Zero-Knowledge Proof of Knowledge.** A usual Zero-Knowledge Proof of Knowledge (ZKPoK) protocol is a two-party three-round protocol, whereby a prover proves to a verifier about the knowledge of a secret without disclosing any information on the secret. The three-round is “commit-challenge-response”, which has been well studied in the literature. For simplicity, we use  $ZKPoK\{(x) : y = g^x\}$  to denote the zero-knowledge proof of knowledge protocol that proves the knowledge of  $x$  satisfying  $y = g^x$ . In fact, more complex relations can be proved, e.g.,  $PoK\{(x_1, x_2, \dots, x_\ell) : y = g_1^{x_1} g_2^{x_2} \dots g_\ell^{x_\ell}\}$ , and  $ZKPoK\{(x) : y_1 = g_1^x \wedge y_2 = g_2^x\}$ . A zero-knowledge proof of knowledge protocol can be made non-interactive by means of the Fiat-Shamir Heuristics [27], whereby the prover herself generates the challenge by applying the commitment to a collision-free hash function.

**Nguyen’s Accumulator.** Our proposal is based on Nguyen’s dynamic accumulator [4, 33], which works over an asymmetric bilinear map  $\hat{e} : G_1 \times G_2 \rightarrow G_T$ , with  $G_1, G_2, G_T$  being cyclic groups of a prime order  $q$ . Let  $h$  be a generator of  $G_2$ . The public parameters include  $(Z = h^z, h)$ , and the private key is  $z \in Z_q^*$ . The accumulator, denoted as  $\Lambda$ , for a set of values  $(k_1, k_2, \dots, k_\ell)$  is calculated as  $\Lambda = \bar{g}^{\prod_{j=1}^{\ell} (k_j + z)}$ , where  $\bar{g} \in G_1$  is a public parameter. The witness for a value  $k$  accumulated in  $\Lambda$  is  $W = \Lambda^{\frac{1}{k+z}}$ . As such,  $(W, k)$  can be verified by  $\hat{e}(W, Z \cdot h^k) = \hat{e}(\Lambda, h)$ . Interestingly, the proof showing that  $k$  is accumulated in  $\Lambda$  can be performed using zero-knowledge proof of knowledge without revealing any information on  $k$  and  $W$ , which is denoted as  $ZKPoK\{(W, k) : \hat{e}(W, Z \cdot h^k) = \hat{e}(\Lambda, h)\}$ . More details on the proof can be found in [4, 33].

Nguyen’s accumulator is dynamic in the sense that it supports value addition and removal, both requiring witness update. Here we make a small modification to avoid witness update in case of value addition: initially, the accumulator  $\Lambda$  is assigned a random value in  $G_1$ , representing all values having been accumulated already; computation of the witness for an accumulated value remains the same. With this modification, only value removal requires witness update. Specifically, the removal/revocation of value  $k_j$  in the present accumulator  $\Lambda_{pre}$  is performed

as follows. The authority managing the accumulator computes the new accumulator as  $\Lambda_{new} = \Lambda_{pre}^{\frac{1}{k_j+z}}$ , and publishes a new entry  $\langle \Lambda_{new}, k_j \rangle$  in a public board. In response, every witness holder needs to update their witness in order to keep consistent with the new accumulator. In particular, for a holder with witness  $W_i$  (corresponding to  $k_i$ ), the witness can be updated by computing  $W_i^{new} = W_i^{\frac{1}{k_j+z}} = (\Lambda_{pre}^{\frac{1}{k_i+z}})^{\frac{1}{k_j+z}} = \Lambda_{pre}^{(\frac{1}{k_i+z} - \frac{1}{k_j+z}) \cdot \frac{1}{k_j-k_i}} = (\frac{W_i}{\Lambda_{new}})^{\frac{1}{k_j-k_i}}$ . It is clear that each witness holder can update their witness without the knowledge of the system authority's private key  $z$ .

## 4 Lightweight Anonymous Credential with Outsource-Able Witness Update

We are now ready to present a lightweight anonymous credential construction in this section. We start by elaborating on the basic idea behind the construction.

### 4.1 Basic Idea

Our main observation is that dynamic accumulator itself suffices to be anonymous credential, although it was originally designed to be a mechanism providing credential revocation support to anonymous credential. However, as mentioned earlier we are faced with the problem of witness update, due to the fact that legitimate users have to update their witnesses in accordance with the new accumulator resulting from every revocation of a credential. This problem is particularly challenging for weak IoT devices as they are not capable of connecting to the revocation server around-the-clock and responding to each credential revocation real time; in addition, batch witness update is too costly and time-consuming for these devices as well.

Our solution is to leverage on the ‘computation outsourcing’ paradigm – to outsource the task of witness update to an untrusted third party server such as a dedicated cloud service. Nevertheless, a difficulty arises with this solution: in Nguyen’s dynamic accumulator [33], witness update requires the knowledge of both the original witness and the accumulated value, which make up an entire anonymous credential as envisioned in our solution. This clearly contradicts our assumption of an untrusted server. We solve this problem with an idea of extending the accumulator so as to generate an extra secret for a credential, apart from the accumulated value and its witness – this extra secret will always be kept secret by the credential holder.

We remark that many *revocable* anonymous credentials that have incorporated within dynamic accumulator already fit the ‘witness update outsourcing’ paradigm, because in these schemes only a certain element of the credential (but not the entire credential) is accumulated into the accumulator [38]. But compared to these schemes, our anonymous credential proposed below is bound to be more compact and efficient.

## 4.2 The Construction

We base our construction on Nguyen’s dynamic accumulator, with the idea of generating an extra secret for each credential, in addition to the secrets associated with the accumulator itself. The following processes describe the construction.

**System Setup.** The credential issuance authority sets up the system parameters as follows: determine a bilinear map  $\hat{e} : G_1 \times G_2 \rightarrow G_T$ , where  $G_1, G_2, G_T$  are cyclic groups of a prime order  $q$ ; select  $a, d, \Lambda \in_R G_1$ ; select  $h \in_R G_2$  and  $z \in_R Z_q^*$ , and compute  $Z = h^z$ ; set the public system parameters  $params = (\hat{e}, a, d, h, Z, \Lambda, CRL = \emptyset)$  and the master secret key  $msk = (z)$ , where  $\Lambda$  acts as the initial accumulator and  $CRL$  is the credential revocation list.

**Credential Issuance.** Each user registers to the credential issuance authority and is issued a credential in this algorithm. The authority needs to check the validity of the user, but this goes beyond the scope of this work. Let the current accumulator be  $\Lambda$ . For a registering user  $i$ , the authority selects two random numbers  $k_i, s_i \in_R Z_q^*$ , and computes  $V_i = (a^{s_i} d \Lambda)^{\frac{1}{z+k_i}}$  and  $W_i = \Lambda^{\frac{1}{z+k_i}}$ . The credential for user  $i$  is thus set to be  $(V_i, W_i, k_i, s_i)$ .

We remark that the credential is essentially a combination of Nguyen’s accumulator and a simplified version of the BBS+ signature [4]; herein  $s_i$  is the extra secret we have desired, while  $(W_i, k_i)$  is for witness update and this tuple will be given to the third party witness update server.

**Credential Revocation.** To revoke a credential  $c_j = (V_j, W_j, k_j, s_j)$ , the credential issuance authority updates  $CRL$  with  $k_j$  such that  $CRL = CRL \cup \{k_j\}$ ; also updates the accumulator as  $\Lambda = \Lambda^{\frac{1}{z+k_j}}$ . Both  $CRL$  and the new accumulator are signed (to maintain authenticity) and published by the credential issuance authority.

In response to the credential revocation incident, the witness update server for user  $i$  performs witness update as reviewed in Sect. 3.

**Credential Showing.** Suppose a user has a credential  $(V_{pre}, W_{pre}, k, s)$  at the time of authentication. The user first retrieves the latest  $W$  from their credential update server, and updates  $V_{pre}$  to be  $V = \frac{V_{pre} \cdot W}{W_{pre}}$ . The correctness of the update is easily checked and we do not elaborate.

The credential showing procedure is interactive between the user and a verifier. In particular, the user computes the following zero-knowledge proof of knowledge  $ZKPoK\{(V, k, s) : \hat{e}(V, Z \cdot h^k) = \hat{e}(a, h)^s \cdot \hat{e}(d, h) \cdot \hat{e}(\Lambda, h)\}$  based on the challenge message from the verifier. This zero-knowledge proof is what the user needs to pass to the verifier for verification.

If we commit  $V$  as  $A = V^r$ , where  $r \in_R Z_q^*$ , then the zero-knowledge proof of knowledge  $ZKPoK$  is to prove the knowledge of  $(k, r, s)$  satisfying  $\hat{e}(A, Z) = \hat{e}(A, h)^{-k} \cdot \hat{e}(a, h)^{r \cdot s} \cdot \hat{e}(d \cdot \Lambda, h)^r$ . The details of the proof are as follows.

1. The user selects  $r, \alpha_k, \alpha_r, \alpha_s, \beta_1, \beta_2, \beta_3 \in_R Z_q$ , and computes  $A = V^r, C_1 = \hat{e}(A, h)^{-\alpha_k} \cdot \hat{e}(a, h)^{\alpha_r \cdot \alpha_s} \cdot \hat{e}(d \cdot \Lambda, h)^{\alpha_r}, C_2 = e(a, h)^{-(\beta_1 + \beta_2 + \beta_3)}$ . The user sends  $(A, C_1, C_2)$  to the verifier.

2. The verifier responds with a challenge message  $c$ .
3. The user computes and sends the following responses to the verifier (all arithmetic operations are modulo  $q$ ):

$$\begin{aligned}
 t_k &= \alpha_k + c \cdot k \\
 t_r &= \alpha_r + c \cdot r \\
 t_s &= \alpha_s + c \cdot s \\
 t_1 &= \beta_1 + (1 - c) \cdot \alpha_r \cdot \alpha_s \\
 t_2 &= \beta_2 + c \cdot \alpha_r \cdot s \\
 t_3 &= \beta_3 + c \cdot \alpha_s \cdot r
 \end{aligned}$$

4. The verifier checks and accepts if the following holds:

$$\left\{ \begin{array}{l} A \neq 1 \in G_1; \\ \text{check ZKPoK} : \hat{e}(A, h)^{-c \cdot t_k} \cdot \hat{e}(a, h)^{t_r \cdot t_s - t_1 - t_2 - t_3} \cdot \hat{e}(d \cdot \Lambda, h)^{c \cdot t_r} \\ \stackrel{?}{=} \hat{e}(A, Z)^{c^2} \cdot (C_1)^c \cdot C_2; \end{array} \right.$$

**Remark.** Note that  $A \neq 1 \in G_1$  is to guarantee  $r \neq 0 \pmod{q}$ ; otherwise, any value of  $k$  would trivially satisfy the zero-knowledge proof.

### 4.3 Security Analysis

Unforgeability and anonymity/unlinkability are the two main security requirements upon anonymous credential, and they have been well understood and formalized in the literature, e.g., [2, 8, 12, 19, 26]. To avoid repetition, readers are referred to the relevant literature for the formalization of these security requirements. Below is the analysis showing that the above construction satisfies unforgeability and anonymity.

*Unforgeability.* Primarily we need to show that it is infeasible for an adversary to forge  $(V_i, W_i, k_i, s_i)$  such that  $V_i = (a^{s_i} d \Lambda)^{\frac{1}{z+k_i}}$  and  $W_i = \Lambda^{\frac{1}{z+k_i}}$ . First of all, it is clear that computing  $(W_i, k_i)$  by the adversary means the compromise of Nguyen’s dynamic accumulator, so we are only concerned with  $(V_i, k_i, s_i)$ .

We actually relate unforgeability of the credentials to the  $\ell$ -Strong Bilinear Collusion Attack Assumption ( $\ell$ -SBCAA), which is implied in [7, 8]. Let  $G_1, G_2$  be as in the above bilinear map, and  $g \in G_1, h \in G_2$ .  $\ell$ -SBCAA asserts that for  $z \in Z_q^*$ , it is hard to compute  $(k_0 \in Z_q^*, g^{\frac{1}{k_0+z}})$ , given a  $(\ell + 3)$ -tuple  $(g, h, h^z, (k_1, g^{\frac{1}{k_1+z}}), (k_2, g^{\frac{1}{k_2+z}}), \dots, (k_\ell, g^{\frac{1}{k_\ell+z}}))$ , for  $\forall k_i \in Z_q^*$ .

In particular, we have the following theorem asserting the unforgeability of our construction.

**Theorem 1.** *The above construction achieves unforgeability, as long as the  $\ell$ -SBCAA holds.*

*Proof.* We should first show that the zero-knowledge proof of knowledge extracts  $(V, k, s)$ . This step is standard and we omit the details. Then we show that it is infeasible to forge new  $(V, k, s)$ , given a set of valid credentials. To this end, we construct a game where an adversary  $\mathcal{A}$  for  $\ell$ -SBCCA simulates to an adversary  $\mathcal{A}'$  that tries to forge  $(V, k, s)$ . The main idea is outlined below: given a  $(\ell + 3)$ -tuple  $(g, h, h^z, (k_1, g^{\frac{1}{k_1+z}}), \dots, (k_\ell, g^{\frac{1}{k_\ell+z}}))$ ,  $\mathcal{A}$  sets  $d = g, a = g^{r_a}, \Lambda = g^{r_\Lambda}$ , where  $r_a, r_\Lambda \in_R Z_q$ . Now  $\mathcal{A}$  can easily transform the  $\ell$ -tuple  $(k_1, g^{\frac{1}{k_1+z}}, \dots, (k_\ell, g^{\frac{1}{k_\ell+z}})$  into  $\ell$  credentials  $(V_i = a^{s_i} d \Lambda = (g^{(r_a \cdot s_i + r_\Lambda + 1) \cdot \frac{1}{k_i+z}}, k_i, s_i \in_R Z_q), i \in [1.. \ell])$ . With the  $\ell$  credentials in possession,  $\mathcal{A}$  can definitely answer the requests for credentials from  $\mathcal{A}'$ .  $\square$

*Anonymity.* It is easier to argue for anonymity of credential showing: intuitively,  $A, C_1$  unconditionally hide  $V, k, s$ ; the zero-knowledge proof of knowledge does not reveal information on  $k, r, s$ , as per the zero-knowledge property of the proof. As a result, anonymity is achieved.

Formally, we need to show that  $(A = V^r, \hat{e}(A, Z) = \hat{e}(A, h)^{-k} \cdot \hat{e}(a, h)^{r \cdot s} \cdot \hat{e}(d \cdot \Lambda, h)^r)$  is indistinguishable from  $(V^r, R \in_R G_T)$ , with the knowledge of  $(V, k, s)$ , and the zero-knowledge proof is indeed zero-knowledge. The latter is standard. For the former, it is actually implied by the so called XDH assumption. Let  $g_1, g_2 \in G_1$ , where  $G_1$  is the compact bilinear group of a bilinear map. The XDH assumption states that it is computationally infeasible to distinguish between  $(g_1, g_2, g_1^x, g_2^x)$  and  $(g_1, g_2, g_1^x, R)$ , where  $x \in_R Z_q$  and  $R \in_R G_1$ .

## 5 Self-Blindable Credentials: Further Improvement

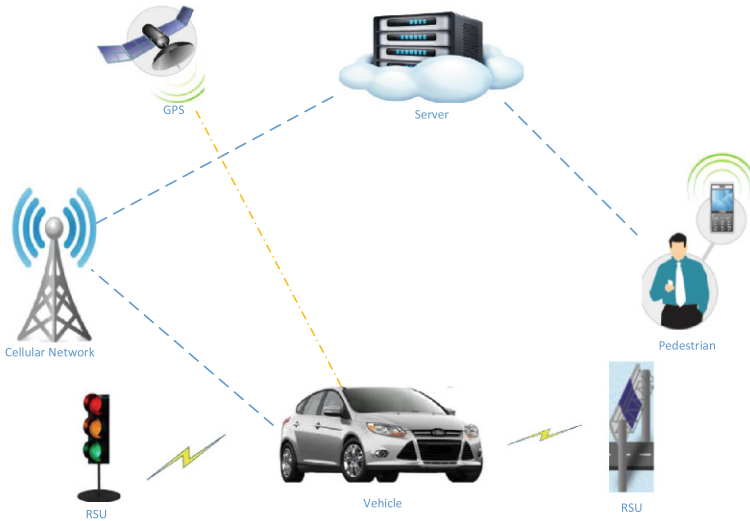
Although the above construction is compact and relatively efficient, it still requires the prover to compute bilinear pairing operations and exponentiations in  $G_T$ . These operations are much more costly than computations in  $G_1$ , which is much more compact than  $G_2$  and  $G_T$  – it is especially so when achieving higher level of security, e.g., 192-bit or 256-bit security. Following the idea of self-blinding in [36, 38], we further improve the performance of the above construction, in such a way that computations at the prover side are entirely in  $G_1$ . Considering the fact that IoT devices are often required to authenticate to a (powerful) authentication server, this improved scheme is especially useful in such scenarios where asymmetry in resources between the prover and verifier exists, as shown in Fig. 1.

### 5.1 Basic Idea

Let us continue to hide  $V$  as  $A = V^r$  as in the above. That is,  $A = V^r = (a^s \cdot d \cdot \Lambda)^{\frac{r}{k+z}}$ . Thus,  $A$  and  $A' = (a^s \cdot d \cdot \Lambda)^r$  can be verified by

$$\hat{e}(A, Z \cdot h^k) = \hat{e}(A', h) \quad (1)$$





**Fig. 1.** Application scenarios where prover and verifier have differences in resources

To fully conceal the credential, we need to hide  $k$  as well. From Eq. 1, we get

$$\begin{aligned}
 \hat{e}(A, Z \cdot h^k) &= \hat{e}(A', h) \\
 \Leftrightarrow \hat{e}(A, Z) \cdot \hat{e}(A, h^k) &= \hat{e}(A', h) \\
 \Leftrightarrow \hat{e}(A, Z) \cdot \hat{e}(A^k, h) &= \hat{e}(A', h)
 \end{aligned}
 \tag{2}$$

We need to further blind  $k$  in  $A^k$ . One possible way is  $A^k \cdot t^{r'}$ , together with  $h^{r'} \in G_2$  (which is used to cancel out the randomness  $t^{r'}$  in the credential verification step), where  $t \in_R G_1, r' \in_R Z_q^*$ . However, this requires  $h^r$  to be an element in  $G_2$  which imposes heavy computation on the prover end, thus contradicting our design objective. The challenge is to make the prover work entirely in  $G_1$ .

We get over this difficulty by introducing an extra pair of public parameters  $(t, T = t^z, \cdot)$ , where  $t \in G_1$ . Specifically, the credential  $(V, k, s)$  is blinded as follows, where  $r, r_1, r_2 \in_R Z_q^*$ :

$$\begin{aligned}
 A &= V^r \cdot t^{r_1} \in G_1 \\
 B &= V^{r \cdot k} \cdot T^{r_2} \in G_1 \\
 A' &= (a^s \cdot d \cdot \Lambda)^r \in G_1 \\
 T_1 &= T^{r_1} \in G_1 \\
 T_2 &= t^{r_2} \in G_1.
 \end{aligned}
 \tag{3}$$

As such, verification of these elements can be done as  $\hat{e}(A, Z) \cdot \hat{e}(B, h) \stackrel{?}{=} \hat{e}(A', h)\hat{e}(T_1, h)\hat{e}(T_2, Z)$ . It goes without saying that zero-knowledge proof of knowledge must be put in place to ensure that these elements are well formed.

## 5.2 Construction Details

To avoid repetition, we only highlight the differences from the construction in above Sect. 4. The public system parameters  $params = (\hat{e}, a, d, t \in_R G_1, T = t^z, h, Z, \Lambda, CRL = \emptyset)$ , while master secret key remains the same  $msk = (z)$ . The credential issuance process and the credential revocation process are the same. We next focus on the credential showing process.

**Credential Showing.** Suppose that the user's latest credential is  $(V, k, s)$ , and the latest accumulator is  $\Lambda$ . The credential showing process between the user and the verifier is below:

1. The user selects  $r, r_1, r_2 \in_R Z_q$ , and computes

$$\begin{aligned} A &= V^r \cdot t^{r_1} \in G_1 \\ B &= V^{r \cdot k} \cdot T^{r_2} \in G_1 \\ A' &= (a^s \cdot d \cdot \Lambda)^r \in G_1 \\ T_1 &= T^{r_1} \in G_1 \\ T_2 &= t^{r_2} \in G_1. \end{aligned}$$

In addition, the user needs to compute zero-knowledge proof  $ZKPoK\{(k, r, \varsigma, \gamma, r_1, r_2) : B = A^k t_1^{-\gamma} T_2^{r_2} \wedge A' = a^\varsigma (d \cdot \Lambda)^r \wedge T_1 = T^{r_1} \wedge T_2 = t^{r_2}\}$ , where  $\gamma = k \cdot r_1, \varsigma = s \cdot r$  to ensure the well-formedness of these elements. To do this, the user computes the following commitments, where  $\alpha_k, \alpha_\gamma, \alpha_\varsigma, \alpha_r, \beta_1, \beta_2 \in_R Z_q^*$ :

$$\begin{aligned} C_1 &= A^{\alpha_k} t_1^{-\alpha_\gamma} T^{\beta_2} \in G_1 \\ C_2 &= a^{\alpha_\varsigma} (d \cdot \Lambda)^{\alpha_r} \in G_1 \\ C_3 &= T^{\beta_1} \in G_1 \\ C_4 &= t^{\beta_2} \in G_1 \end{aligned}$$

The user sends  $(A, B, A', T_1, T_2, C_1, C_2, C_3, C_4)$  to the verifier.

2. The verifier sends back a challenge message  $c$ .
3. Based on the challenge, the user computes and sends back the following response to the verifier:

$$\begin{aligned} t_k &= \alpha_k + c \cdot k \\ t_\gamma &= \alpha_\gamma + c \cdot \gamma \\ t_r &= \alpha_r + c \cdot r \\ t_\varsigma &= \alpha_\varsigma + c \cdot \alpha_\Lambda \\ t_{\beta_1} &= \alpha_{\beta_1} + c \cdot \beta_1 \\ t_{\beta_2} &= \alpha_{\beta_2} + c \cdot \beta_2 \end{aligned}$$

4. The verifier accepts if all of the following checks pass:

$$\left\{ \begin{array}{l} A' \neq 1 \in G_1; \\ \text{check ZKPoK} : B^{t_k} t_1^{-t_\gamma} T^{t_{\beta_2}} = A^c C_1 \wedge a^{t_\zeta} (dA)^{t_r} = A'^c C_2 \wedge T^{t_{\beta_1}} = T_1^c C_3 \\ \wedge t^{\beta_2} = T_2^c C_4; \\ \hat{e}(A, Z) \cdot \hat{e}(B, h) \stackrel{?}{=} \hat{e}(A', h) \cdot \hat{e}(T_1, h) \cdot \hat{e}(T_2, Z); \end{array} \right.$$

### 5.3 Security Analysis

Unforgeability largely remains unchanged from the construction in Sect. 4, and we focus on anonymity. We need to show that with the knowledge of  $(V, k, s)$ , it is infeasible to relate  $(A, B, A', T_1, T_2)$  to  $(V, k, s)$ . Note that the zero-knowledge proof of knowledge does not reveal more information on the elements that are proved. Intuitively,  $s$  in  $A'$  is blinded by  $r$ ;  $V$  in  $A$  is blinded by  $r_1$ , and  $k$  in  $B$  is blinded by  $r_2$ ; but what compounds the situation is that  $r_1, r_2$  are further revealed in  $T_1$  and  $T_2$ . Formally, we have Theorem 2 as follows.

**Theorem 2.** *The above scheme achieves anonymity in the generic group model.*

*Proof.* The adversary is supposed to have the knowledge of  $(V, k, s)$ . Note that  $s$  in  $A'$  is blinded by  $r$ , and  $A'$  only relates to  $A, B, T_1, T_2$  through the credential verification equation. Clearly this relationship does not help in credential linking. So we only consider  $A, B, T_1, T_2$ , in which case  $k$  could be used to relate  $A$  and  $B$ . Intuitively, the only way to use  $k$  for credential linking is by applying bilinear map, i.e., to distinguish between  $\frac{\hat{e}(A^k, Z)}{\hat{e}(T_1, h)}$  and  $\frac{\hat{e}(B, h)}{\hat{e}(T_2, Z)}$ . However,  $\frac{\hat{e}(A^k, Z)}{\hat{e}(T_1, h)} = \hat{e}(V, Z)^{r \cdot k}$  and  $\frac{\hat{e}(B, h)}{\hat{e}(T_2, Z)} = \hat{e}(V, h)^{r \cdot k}$ , so without knowing  $z$ , it is infeasible to distinguish  $\hat{e}(V, Z)^{r \cdot k}$  and  $\hat{e}(V, h)^{r \cdot k}$  under the DDH assumption in  $G_T$ . This intuition can be readily argued in the generic group model [37], to distinguish between  $(V^r \cdot t^{r_1}, V^{r \cdot k} \cdot T^{r_2}, T^{r_1}, t^{r_2})$  and  $(V^r \cdot t^{r_1}, V^{r'} \cdot T^{r_2}, T^{r_1}, t^{r_2})$ , where  $r, r', r_1, r_2 \in_R Z_q^*$ . The actual proof can follow the proving method in [18] and will be included in the full version.

The above analysis actually indicates that this scheme achieves anonymity against adversaries who cannot be the credential issuance authority who knows  $z$ . This security guarantee is weaker than regular anonymous credential schemes (excluding those enforcing verifier-local revocation) where unlinkability even goes against the credential issuance authority.  $\square$

## 6 Performance Evaluation

In this section, we evaluate the performance of our constructions.

**Analytical Evaluation.** We first provide an analytical evaluation on the performance of our constructions. Let  $|G|$  denote the bit length of an element in group  $G$ , and  $\text{EXP}(G)$  denote an exponentiation operation in  $G$ ,  $\text{Pair}$  denote a bilinear pairing operation. We list in Table 1 the prover-side computation and

**Table 1.** Performance Analysis

	Computation	Communication
Construction in Sect. 4	$1 \cdot \text{EXP}(G_1) + 3 \cdot \text{EXP}(G_T) + 1 \cdot \text{Pair}$	$1 G_1  + 2 G_T  + 6 Z_q $
Construction in Sect. 5	$9 \cdot \text{EXP}(G_1)$	$9 G_1  + 7 Z_q $

communication costs of our two schemes. Note that by applying optimized algorithms for multi-exponentiations, e.g., [25, 32], the computation overhead for a multi-exponentiation is just slightly more than a single-exponentiation. We thus do not distinguish multi-exponentiation and single-exponentiation by  $\text{EXP}(G)$  in the calculations.

When instantiating the bilinear map over MNT curves with embedding degree 6, it is estimated in [14] for 80-bit security that  $|G_T| = 1026$ ,  $|G_1| = 171$ , and  $|Z_q| = 160$ , and  $1 \cdot \text{Pair} \approx 1 \cdot \text{EXP}(G_T) = 10 \cdot \text{EXP}(G_1)$ . In such a setting, the computation overhead of our second construction is about 20% of the first scheme; for communication, the first construction has overhead of about 3Kb and the second construction has about 2.7 Kb. It can be seen that the second construction indeed excels in both metrics. As a matter of fact, the performance advantage of the second construction turns even more evident when the curve of the bilinear map has larger embedding degrees.

**Experimental Evaluation.** We also empirically evaluate the performance of the second construction on a vehicular embedded device - NXP ATOP (Automotive Telematics On-board unit Platform), which is an automotive qualified device for vehicle telematics, equipped with an ARM9 processor up to 266 MHz. The implementation on the NXP ATOP uses JAVA and the Bouncy Castle Java Crypto Library<sup>1</sup>. Since all exponentiation computations in our construction are over the compact group  $G_1$ , the implementation does not require the user platform to offer algebraic support for bilinear pairing operation, although the construction itself is based on bilinear map.

*Algorithmic Optimization.* Since for a particular user with a credential  $(V, k, s)$ , the value of  $a^s d$  does not change, we treat it as a pre-computed constant and its related exponentiation operation becomes fixed-base single-exponentiation.

To accelerate the computation of the fixed-base exponentiation operations involved in the credential showing algorithm, we construct a lookup table for each of the bases,  $a^s d, a, d, t, T$ . Each entry corresponds to one bit of  $|q|$ , and contains one  $G_1$  element. Taking parameter  $T$  for example, the  $i^{\text{th}}$  entry of its table is  $T^{2^i}$ . Therefore, for any  $x \in Z_q^*$  represented as  $b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + b_3 \cdot 2^3 + \dots$ ,  $\forall b_i \in \{0, 1\}$ , the computation of  $T^x$  is simply a multiplication of the elements corresponding to  $b_i = 1$ , which are pre-computed in the table.

Furthermore, to avoid repeated lookup table scans, exponentiations over the same base are computed together. For example, the computation of  $T_1 = T^{r_1}$

<sup>1</sup> <http://www.bouncycastle.org/java.html>.

and the commitment for  $ZKPoK\{(r_1) : C_3 = T^{\beta_1}\}$  are executed at the same time, such that the lookup table for  $T$  is only scanned once.

*Experimental Results.* We run the experiments upon the testing platform for 50 times, and the averaged result is that it take about 2.51 second for the prover to complete the proving process. This turns out to be reasonable considering the underlying NXP ATOP is rather weak, and our implementation does not exploit any specialized hardware for cryptographic operations. An IoT device with built-in cryptographic circuits (as on JAVA cards) would tremendously enhance the performance.

## 7 Conclusion

Towards designing practical anonymous entity authentication techniques suitable for resource-constrained IoT devices, we first propose a lightweight anonymous credential construction with Nguyen's dynamic accumulator under the "witness update outsourcing" paradigm. Further performance improvement was made to make the prover supposedly using a weak IoT device work entirely on  $G_1$  of an asymmetric bilinear map, which is an order of magnitude faster than computing pairing operations. Although our performance analysis and evaluation in corroborates the promising efficiency of our constructions, future work is needed to further optimize the constructions or to explore new constructions with even higher performance.

## References

1. Ateniese, G., Camenisch, J.L., 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)
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
3. Ateniese, G., de Medeiros, B.: Efficient group signatures without trapdoors. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 246–268. Springer, Heidelberg (2003)
4. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic  $k$ -TAA. In: Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
5. Balasch, J.: Smart Card Implementation of Anonymous Credentials. Master thesis, K. U. Leuven (2008)
6. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
7. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

9. Bichsel, P., Camenisch, J., Groth, T., Shoup, V.: Anonymous credentials on a standard java card. In: ACM Conference on Computer and Communication Security, CCS 2009, pp. 600–610. ACM (2009)
10. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
11. Batina, L., Hoepman, J.-H., Jacobs, B., Mostowski, W., Vullers, P.: Developing efficient blinded attribute certificates on smart cards via pairings. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 209–222. Springer, Heidelberg (2010)
12. 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.) Advances in Cryptology — EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
13. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security, CCS 2004, pp. 168–177. ACM (2004)
14. Boyen, X.: A tapestry of identity-based encryption: practical frameworks compared. *J. Applied Crypt.* **1**(1), 3–19 (2008)
15. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. *Commun. ACM* **28**(10), 1030–1044 (1985)
16. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) Advances in Cryptology — EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
17. Camenisch, J., Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: ACM Conference on Computer and Communication Security, CCS 2002. ACM (2002)
18. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: ACM Conference on Computer and Communications Security, CCS 2006, pp. 201–210. ACM (2006)
19. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
20. Camenisch, J.L., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
21. Camenisch, J.L., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
22. Camenisch, J.L., Michels, M.: A group signature scheme with improved efficiency. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 160–174. Springer, Heidelberg (1998)
23. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CAIP 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
24. Camenisch, J., Neven, G., Rückert, M.: Fully anonymous attribute tokens from lattices. In: Visconti, I., Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 57–75. Springer, Heidelberg (2012)
25. Dimitrov, V., Jullien, G., Miller, W.: Complexity and fast algorithms for multi-exponentiations. *IEEE Trans. Comput.* **49**(2), 141–147 (2000)

26. Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: Abe, M. (ed.) *Advances in Cryptology — ASIACRYPT 2010*. LNCS, vol. 6477, pp. 395–412. Springer, Heidelberg (2010)
27. 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)
28. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
29. Garman, C., Green, M., Miers, I.: Decentralized Anonymous Credentials. In: *NDSS Symposium* (2014)
30. Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: Sako, K., Sarkar, P. (eds.) *ASIACRYPT 2013, Part II*. LNCS, vol. 8270, pp. 41–61. Springer, Heidelberg (2013)
31. 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)
32. Moeller, B., Möller, B.: Algorithm for multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) *Selected Areas in Cryptography*. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
33. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) *CT-RSA 2005*. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
34. Sterckx, M., Gierlich, B., Preneel, B., Verbauwhede, T.: Efficient implementation of anonymous credentials on java card smart cards. In: *Information Forensics and Security, WIFS 2009*, pp. 106–110. IEEE (2009)
35. Teranishi, I., Furukawa, J., Sako, K.:  $k$ -times anonymous authentication (Extended Abstract). In: Lee, P.J. (ed.) *ASIACRYPT 2004*. LNCS, vol. 3329, pp. 308–322. Springer, Heidelberg (2004)
36. Verheul, E.R.: Self-blindable credential certificates from the weil pairing. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, p. 533. Springer, Heidelberg (2001)
37. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
38. Yang, Y., Ding, X., Lu, H., Weng, J., Zhou, J.: Self-blindable credential: towards anonymous entity authentication upon resource constrained devices. In: Desmedt, Y. (ed.) *Information Security*. LNCS, vol. 7807, pp. 238–247. Springer, Switzerland (2015)

# Hybrid MQ Signature for Embedded Device

Shaohua Tang<sup>(✉)</sup>, Bo Lv, and Wuqiang Shen

School of Computer Science and Engineering,  
South China University of Technology, Guangzhou, China  
shtang@IEEE.org, csshtang@scut.edu.cn

**Abstract.** The embedded security system is a special embedded system to perform secure authentication and communication with cryptographic technique, and it is one of the effective approaches to solve security problems of the embedded system at present. Multivariate public key cryptosystem has the potential to resist the attacks of quantum computers. And its operating efficiency is generally better than that of number theoretic-based system, so it is very suitable for the resource-limited device like embedded device. In view of this, this paper provides a secure multivariate signature scheme RGB respectively effectively implemented on S5PV210 and MT6582 microprocessors. RGB signature scheme of security levels at  $2^{64}$ ,  $2^{80}$ ,  $2^{96}$ ,  $2^{118}$  and  $2^{128}$  have been respectively performed for convenient in practical use. Experimental data show that our implementations are highly efficient, meeting requirements of security level and real-time operation for high-end users and low-end users, and play a referential role in implementing other (multivariate) public key cryptographies on resource-limited devices.

**Keywords:** Embedded security system · Multivariate signature · RGB signature scheme

## 1 Introduction

The embedded system is a special system to perform specific requirements or functions [8]. It brings people's lives profound influences and is widely used in industrial, commercial, military, education, medical and scientific research fields, etc. The products can be seen everywhere, such as mobile phone, PM3/4, vehicle systems, automated banking service terminal, digital home, wireless sensor and so on. However, an obvious question raises: how to ensure the embedded system or device to be used securely? Based on this, two problems should be considered [12], that is, kernel security of the embedded operating system, and security authentication and communication between a embedded middleware server and a target embedded system, for example, communication security of mobile banking, communication security of an automated banking service terminal, and communication security between a wireless sensor node and a base station. For the former, more attention is paid to security details in terms of design. For the latter, technologies related to cryptography, such as encryption, signature and



authentication technologies, can be used to effectively solve problem. This paper only focuses on the security of the latter, that is, so-called security authentication and communication for embedded system.

The public key cryptography is a mainstream cryptographic technique, which can solve the problem of the security authentication and communication for the user effectively. Currently, most public key cryptosystems are mainly based on integer factorization problem and discrete logarithm problem. However, in 1997, Shor [23] presented an effective algorithm in a quantum computer to solve the integer factorization problem and discrete logarithm problem. This means the traditional public key cryptography based on the integer factorization problem and discrete logarithm problem will face a huge threat with the arrival of the quantum computer. Therefore, people should consider cryptography that can resist the attacks of quantum computers. At present, the cryptography against quantum computer attacks mainly includes [6]: lattice-based cryptography, error-correcting code-based cryptography and multivariate public key cryptography. In these, the multivariate public key cryptography is a promising cryptographic technique, for example, UOV signature scheme [13], Rainbow signature scheme [7], HFE [17], enTTS, SflashV2 [1], IBUOV signature scheme [22], ZHFE signature scheme [19], RGB hybrid signature scheme [21], QUAD stream cipher [3] and Quartz signature scheme [18]. Each of the above cryptographic techniques has a public key made up of a group of multivariate polynomials over a finite field, and the security relies on a difficult problem of solving the group of multivariate polynomials over the finite field. All the multivariate public key cryptosystems (MPKC) schemes above are faster in running speed than that of number theoretic-based cryptosystems, and very applicable to devices of limited resource, such as smart card, embedded device, RFID, and vehicle-mounted system. Based on this, it is provided in this paper a secure multivariate signature scheme RGB respectively effectively implemented on S5PV210 and MT6582 microprocessors. RGB signature scheme of security level at  $2^{64}$ ,  $2^{80}$ ,  $2^{96}$ ,  $2^{118}$  or  $2^{128}$  have been respectively performed for convenient in practical use. Experimental data show that our implementations are highly efficient, and play a referential role in implementing other (multivariate) public key cryptography on resource-limited devices.

## 2 Related Work

The embedded security system can provide function of confidentiality, integrity and authentication in communication of an embedded device, so it is highly welcomed by people and widely studied. In 2011, Oliveira et al. [16] implemented a pairing technology based on identity authentication called TinyPBC, which has a running time of 1.90 s, 1.27 s and 0.14 s on ATmega128L (MICA2 and MICAZ nodes), MSP430 (TelosB and Tmote Sky) and PXA27x (Imote2) respectively. Fan et al. [9] designed and implemented an embedded-Ethernet security access system on a S3C6410 ARM11 development board and a DM9000 chip based on embedded Linux operating system. Similarly, Li et al. [14] designed an Embedded Multi-biometric Recognition Platform on a Samsung S3C6410 development

board. Thomas and Erich [25] implemented a pairing calculation of 254-bit BN curve in a time of less than 0.1 *s* based on ARM Cortex-M0+, the reusability of which provides a solution based on microprocessor embedded application to other elliptic curve cryptosystems. Hao [10] studied the lossless image compression on the embedded Linux operating system. Kim [11] implemented a cost-effective home lighting control system on the embedded Linux operating system, which has a remote control function. In order to verify content stored in an embedded device, Seshadri et al. [20] presented a software-based attestation technique called SWATT. Muthukumaran et al. [15] focused on data integrity of a mobile phone system. Courtright et al. [4] further studied the security of the embedded system, and presented a simple instruction encryption mechanism based on XOM model.

For MPKC, Balasubramanian et al. [2] asserted a Rainbow signature can sign a message in 0.012 *ms* using AMI 0.35 *um* CMOS technology. Tang et al. [24] also asserted a secure Rainbow signature can be generated on a FPGA in 3960 ns. Yang et al. [26] implemented enTTS (20,28) on an ASIC in a signature time of 0.044 *s*. For embedded software implementation, Czypek et al. [5] implemented UOV(28,37), 0/1 UOV(28,37), Rainbow(18,13,14) and enTTS(9,36,52) on ATxMega128al in signature time of 113.66 *ms*, 110.20 *ms*, 54.38 *ms* and 19.03 *ms* respectively.

### 3 Multivariate Signature Scheme RGB

Shen et al. [21] presented an uncommon hybrid multivariable system, that is, a secure multivariate public key signature scheme RGB, of which a center map has a multivariable polynomial including a lot of quadratic cross terms. The following sections will describe the scheme and give a series of secure and practical parameters of RGB for users to use on the embedded device, based on the analysis of security in [21].

#### 3.1 Solution Overview

Assuming  $F$  is a finite field of cardinality  $q$ , while  $r$ ,  $g$  and  $b$  are positive numbers, and  $r + g + b = n$ . Assuming the red variate is  $Y = (y_1, \dots, y_r)$ , the green variate is  $Z = (z_1, \dots, z_g)$  and the blue variate is  $T = (t_1, \dots, t_b)$ .

Defining a three-colour polynomial  $w$  by

$$w = \sum_{i=1}^r \sum_{i'=1}^r A_{ii'} y_i y_{i'} + \sum_{i=1}^r \sum_{j=1}^g B_{ij} y_i z_j + \sum_{i=1}^r \sum_{k=1}^b C_{ik} y_i t_k + \sum_{j=1}^g \sum_{k=1}^b D_{jk} z_j t_k \\ + \sum_{k=1}^b \sum_{k'=1}^b E_{kk'} t_k t_{k'} + \sum_{i=1}^r G_i y_i + \sum_{j=1}^g H_j z_j + \sum_{k=1}^b L_k t_k + M, \\ A_{ii'}, B_{ij}, C_{ik}, D_{jk}, E_{kk'}, G_i, H_j, L_k, M \in F.$$

Defining a three-colour map  $W$  from  $F^n$  to  $F^g$ :  $W(y_1, \dots, y_r, z_1, \dots, z_g, t_1, \dots, t_b) = (w_1, \dots, w_g)$ , Wherein each  $w_i (1 \leq i \leq g)$  is a three-colour polynomial. Defining an invertible affine transformation  $S_1$  over  $F^r$ , an invertible

affine transformation  $S_2$  over  $F^{g+b}$ , and an invertible affine transformation  $S_3$  over  $F^g$ , wherein the  $S_3$  must be linear.

Defining a map  $\bar{W}$  from  $F^n$  to  $F^g$ :  $\bar{W}(x_1, \dots, x_n) = (\bar{w}_1, \dots, \bar{w}_g) = S_3 \circ W(S_1 \times S_2)$ , and setting a public key  $pk = \bar{W}$  and a private key  $sk = (S_1, S_2, S_3, W)$  for the system.

For a message to be signed  $Y' = (y_1', \dots, y_r') \in F^r$ , the signer should firstly calculate:  $\tilde{Y} = (\tilde{y}_1, \dots, \tilde{y}_r) = S_1(Y)$ . Then, randomly choosing some values  $\tilde{T} = \tilde{t}_1, \dots, \tilde{t}_b \in F^b$ , which may be plugged into the three-colour map  $W$  together with  $\tilde{Y} = (\tilde{y}_1, \dots, \tilde{y}_r)$ , to form the following equation system:

$$\begin{cases} w_1(\tilde{y}_1, \dots, \tilde{y}_r, z_1, \dots, z_g, \tilde{t}_1, \dots, \tilde{t}_b) = 0 \\ \vdots \\ w_g(\tilde{y}_1, \dots, \tilde{y}_r, z_1, \dots, z_g, \tilde{t}_1, \dots, \tilde{t}_b) = 0 \end{cases}$$

Solving the above system and denoting one of the solved solutions by  $\tilde{Z} = (\tilde{z}_1, \dots, \tilde{z}_g)$ . It is noted that if the system does not have any solution, another random value  $\tilde{T}$  should be chosen once again, and the new random value  $\tilde{T}$  may be plugged into the system until the system has at least one solution. Then calculating  $X' = (x_1', \dots, x_{g+b}') = S_2^{-1}(\tilde{z}_1, \dots, \tilde{z}_g, \tilde{t}_1, \dots, \tilde{t}_b) = S_2^{-1}(\tilde{Z} || \tilde{T})$ , wherein  $||$  is a symbol for connecting characters.  $X' = (x_1', \dots, x_{g+b}') \in F^{g+b}$  is a signature of  $Y'$ .

The verification process is very simple, it only needs to plug the message  $Y' = (y_1', \dots, y_r') \in F^r$  and the signature  $X' = (x_1', \dots, x_{g+b}') \in F^{g+b}$  into the public key for verification.

### 3.2 Practical Parameters of RGB

In order to better meet different secure requirements for daily usage of the embedded security system, we will give a series of practical parameters of RGB for users to use in the implementation, based on the analysis of security in [21]. The security levels of these parameters reach  $2^{64}$ ,  $2^{80}$ ,  $2^{96}$ ,  $2^{118}$  and  $2^{128}$  respectively. Table 1 lists their message lengths, signature lengths, public key sizes, private key sizes and security levels, wherein the parameters are provided in the form of  $RGB(q, r, g, b)$ .

**Table 1.** Some Practical Parameters of the RGB Signature System

Practical Parameters	Message size [bit]	Signature size [bit]	Public key size [KB]	Private key size [KB]	Security [bit]
$RGB(2^8, 16, 18, 10)$	128	224	18.19	16.56	$2^{64}$
$RGB(2^8, 24, 24, 12)$	192	288	44.32	39.74	$2^{80}$
$RGB(2^8, 30, 32, 16)$	240	384	98.75	86.46	$2^{96}$
$RGB(2^8, 32, 34, 18)$	256	416	121.36	106.45	$2^{118}$
$RGB(2^8, 40, 35, 20)$	320	440	159.14	143.41	$2^{128}$

## 4 Implementation on S5PV210 and MT6582 Microprocessors

At present, the embedded Linux system and the Android system are two kinds of very important embedded operating systems. Therefore, in order to reflect the effectiveness and applicability of our embedded security system, we will implement the RGB signature scheme on the two main platforms. Meanwhile, in order to meet requirements of security levels for different users, we design and implement the RGB signature scheme of different security levels according to the security parameters listed in Table 1.

### 4.1 Target Platform and Tool

The RGB signature scheme will be implemented on the embedded Linux operating system and the Android system. The selected target platforms includes: (1) an embedded development board Tiny210, wherein the microprocessor is Samsung S5PV210 with an operating frequency of 1 GHz and a memory of 512MB, which works at a embedded Linux operating system; and (2) a common quad-core mobile phone, wherein the microprocessor is MT6582 with an operating frequency of 1.3GHz and a memory of 1GB, which works at a Android 4.2 operating system.

### 4.2 Field and Its Operation

It is known from the nature of the RGB system that all operations in the system is built on the finite field  $F$  of cardinality  $q$ , and the operation of affine transformation and the solution of the system of linear equations should be considered when the signature algorithm is performed; and when the verification is performed, only the implementation of the addition and multiplication of the finite field should be considered. In addition, particular attention should be paid to the choice of the finite field, for example, the common  $GF(2)$ ,  $GF(2^7)$  and  $GF(2^8)$  are generally used. Here,  $GF(2^8)$  is chosen as a practical parameter for the RGB signature scheme.

### 4.3 Storage of Key

As  $GF(2^8)$  being chosen as the finite field while implementing the embedded security system, one byte storage space is used to store an expressed number. In the view of the system structure, it doesn't involve linear map  $S_3$  in the process of signature and verification, so only the storage of the public key  $\overline{W}$  and the private key  $(S_1, S_2, W)$  should be considered in the storage of the key of the embedded security system, wherein: for the public key  $\overline{W}$ , the  $n$ -multivariate public key polynomials in a number of  $g$  should be stored, and at this time, we use a method of coefficient-subscript to store the coefficients of monomial expressions in order, which can benefit calculation; for the private key  $S_1$ , only

related 8-bit value are store; for the private key  $S_2$ , only its inverse is stored, instead of itself, that is,  $S_2^{-1}$  is stored, which is generated by an external program of the RGB when the system is initialized; and for the center map  $W$ , it is also stored by the method of coefficient-subscript.

#### 4.4 Consumption of RAM

From the above, it is known that the implementation of the embedded security system needs to calculate  $S_1(\cdot)$ , solve  $W(\cdot) = 0$  and calculate  $S_2^{-1}(\cdot)$  during signature process, and calculate  $\overline{W}(\cdot) = 0$  during verification process. Obviously, operations in other steps are relative simple and need fewer RAM except for solving  $W(\cdot) = 0$ . A Gaussian elimination method is required to solve  $W(\cdot) = 0$ . During Gaussian elimination process, a certain amount of RAM is required. In general, modest RAM is required for the implementation of the whole embedded security system.

#### 4.5 Optimization of Signature

In order to optimize the implementation of the signature and reduce the time for signature, a look-up table method is directly used to calculate  $S_1(\cdot)$  and  $S_2^{-1}(\cdot)$ . It is convenient to implement the operation of affine transformation with the look-up table method, and greatly reduce computational overhead, which does not require much memory, since the finite field  $GF(2^8)$  of the selected embedded security system RGB has a small cardinality. In addition, when solving  $W(\cdot) = 0$ , an optimized Gaussian elimination method is used to obtain the solution of  $W(\cdot) = 0$ .

## 5 Experiment Result

According to the design and implementation of Sect. 4, we have implemented the hybrid multivariate signature scheme RGB on the Tiny210 and the mobile phone respectively, wherein the security parameters of the RGB involve some different security levels including  $2^{64}$ ,  $2^{80}$ ,  $2^{96}$ ,  $2^{118}$  and  $2^{128}$ . We have obtained signature times and verification times of RGB at these security levels, as shown in Table 2.

These experiments show that: the signature time is 4.8033 *ms*, 11.5127 *ms*, 24.9703 *ms*, 30.9148 *ms* and 41.3265 *ms* respectively, and the verification time is 5.8528 *ms*, 14.2684 *ms*, 31.7285 *ms*, 38.9493 *ms* and 51.1344 *ms* respectively, on the first platform which is S5PV210 microprocessor; the signature time is 10 *ms*, 17 *ms*, 32 *ms*, 45 *ms* and 64 *ms* respectively, and the verification time is 4 *ms*, 11 *ms*, 24 *ms*, 31 *ms* and 47 *ms* respectively, on the second platform which is MT6582 microprocessor. Thus it can be seen that our implementations herein have a good performance. In addition, our experiment results also show that the implementation on embedded Linux is more effective than the implementation on Android at a same security level, which also reflect the actual situation, because

**Table 2.** Implementations of RGB on Different Platforms

Parameters	Platform 1 Tiny210, S5PV210 The embedded linux system		Platform 2 mobile phone, MT6582 the android system		Security [bit]
	Signature time [ms]	Verification time [ms]	Signature time [ms]	Verification time [ms]	
$RGB(2^8, 16, 18, 10)$	4.8033	5.8528	10	4	$2^{64}$
$RGB(2^8, 24, 24, 12)$	11.5127	14.2684	17	11	$2^{80}$
$RGB(2^8, 30, 32, 16)$	24.9703	31.7285	32	24	$2^{96}$
$RGB(2^8, 32, 34, 18)$	30.9148	38.9493	45	31	$2^{118}$
$RGB(2^8, 40, 35, 20)$	41.3265	51.1344	64	47	$2^{128}$

the performance of Tiny210 is better than the performance of MT6582 mobile phone. As we all know, embedded Linux based equipment is mainly for real-time application scenarios, and it's strict with the time consumed for signature and verification. While Android based equipment is mainly for middle and low-end users and it's not that strict with time consuming, which is reflected in our experiment results.

## 6 Performance Comparison

In order to better explain our implementation, we compare our work with implementations of other multivariate signature scheme. It is a pity that there are few implementations of MPKC on embedded Linux and Android system. Instead, there are many implementations on other embedded systems, such as [2, 5, 24, 26]. Therefore, we only compare our work with implementations of MPKC schemes on some other different platforms. Table 3 shows the comparative results.

The experiments show that the signature time is 4.80 *ms*, 11.51 *ms*, and 41.33 *ms* respectively, and the verification time is 5.85 *ms*, 14.27 *ms*, and 51.1344 *ms* respectively, at security levels of  $2^{64}$ ,  $2^{80}$ , and  $2^{128}$ , for RGB implemented on embeded Linux operating system. The signature time and verification time of RGB are less than other systems, such as UOV, 0/1 UOV, Rainbow and enTTS, at any security level. Specifically, when the security level is for example  $2^{80}$ , the signature time of Rainbow(18,13,14) is 9.9 times as that of RGB, the signature time of 0/1 UOV(28,37) is 9.6 times as that of RGB, the signature time of Rainbow(18,13,14) is 4.7 times as that of RGB, and the signature time of enTTS(9,39,52) is 1.7 times as that of RGB. In addition, the verification times of these other system are 8.6, 7.0, 4.8 and 14.6 times respectively as that of RGB. Thus it can be seen that our implementation can work perfectly in the embedded environment, and it can meet the security requirement of the service embedded system from the speed.

**Table 3.** Comparison with Other Implementations

Security [bit]	Scheme	Public key size [Byte]	Private key size [Byte]	Signature time [ms]	Verification time [ms]	Signature code size [Byte]	Verification code size [Byte]
$2^{64}$	UOV(21,28)[5]	25725	21462	50.49	52.83	2188	466
	0/1 UOV(21,28)[5]	4851	12936	49.29	43.60	2258	578
	Rainbow(15,10,10)[5]	12600	9250	26.51	31.58	4162	466
	enTTS(7,28,40)[5]	22960	2731	10.37	79.95	24898	827
	RGB( $2^8$ , 16, 18, 10)	18630	16960	4.80	5.85	29116	29116
$2^{80}$	UOV(28,37)[5]	60060	49728	113.66	122.23	2188	466
	0/1 UOV(28,37)[5]	11368	30044	110.20	100.37	2258	578
	Rainbow(18,13,14)[5]	27945	19682	54.38	69.19	4162	466
	enTTS(9,39,52)[5]	49608	4591	19.03	208.07	41232	827
	RGB( $2^8$ , 24, 24, 12)	45384	40692	11.51	14.27	29116	29116
$2^{128}$	UOV(44,59)[5]	235664	194700	416.07	441.70	2188	466
	0/1 UOV(44,59)[5]	43560	116820	399.43	424.04	2258	578
	Rainbow(36,21,22)[5]	135880	97675	257.11	288.01	4162	466
	enTTS(15,60,88)[5]	234960	13051	66.94	962.17	116698	827
	RGB( $2^8$ , 40, 35, 20)	162960	146855	41.33	51.13	29116	29116

## 7 Conclusion

Embedded Linux system and Android system are two major mainstream operating systems at present, and embedded security systems running on the two kinds of systems have attracted much attention. It is provided in this paper a secure multivariate signature scheme RGB respectively effectively implemented on S5PV210 and MT6582 microprocessors. In order to better meet the demands of daily users for security level, parameter selection of RGB involves security levels of  $2^{64}$ ,  $2^{80}$ ,  $2^{96}$ ,  $2^{118}$  and  $2^{128}$ . Experimental data show that signature time for RGB is 4.8033 ms, 11.5127 ms, 24.9703 ms, 30.9148 ms and 41.3265 ms respectively, and verification times is 5.8528 ms, 14.2684 ms, 31.7285 ms, 38.9493 ms and 51.1344 ms respectively, running on the S5PV210 microprocessor. The signature time for RGB is 10 ms, 17 ms, 32 ms, 45 ms and 64 ms respectively, and the verification time is 4 ms, 11 ms, 24 ms, 31 ms and 47 ms respectively, running on the MT6582 microprocessor. Through above comparison, it is found that our implementations are highly efficient, and can play a referential role in implementing other (multivariate) public key cryptography on resource-limited devices.

**Acknowledgments.** This work was supported by 973 Program (No. 2014CB360501), the National Natural Science Foundation of China (Nos. U1135004 and 61170080), Guangdong Provincial Natural Science Foundation (No. 2014A030308006), and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011).

## References

1. Akkar, M.L., Courtois, N.T., Duteuil, R., Goubin, L.: A fast and secure implementation of sflash. In: Desmedt, Y.G. (ed.) *Public Key Cryptography — PKC 2003*. LNCS, vol. 2567, pp. 267–278. Springer, Heidelberg (2003)
2. Balasubramanian, S., Bogdanov, A., Rupp, A., Ding, J., Carter, H.W.: Fast multivariate signature generation in hardware: the case of rainbow. In: *IEEE Symposium on Field-programmable Custom Computing Machines*, pp. 25–30 (2008)
3. Berbain, C., Gilbert, H., Patarin, J.: QUAD: a practical stream cipher with provable security. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 109–128. Springer, Heidelberg (2006)
4. Courtright, K., Husain, M.I., Sridhar, R.: LASE: Latency aware simple encryption for embedded systems security. *Int. J. Comput. Sci. Netw. Secur.* **9**(10), 1 (2009)
5. Czyppek, P., Heyse, S., Thomae, E.: Efficient implementations of MQPKS on constrained devices. In: Prouff, E., Schaumont, P. (eds.) *CHES 2012*. LNCS, vol. 7428, pp. 374–389. Springer, Heidelberg (2012)
6. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate public key cryptosystems. *Advances in Information Security* **25** (2006)
7. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) *ACNS 2005*. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
8. Di Emilio, M.P.: Design techniques of embedded system. *Embedded Systems Design for High-Speed Data Acquisition and Control*, pp. 49–92. Springer, Switzerland (2015)
9. Fan, C., Li, Z., Ding, Q., Liu, S.: Design of embedded ethernet interface based on arm11 and implementation of data encryption. *Intell. Data Anal. Appl.* **I**, 431–439 (2014)
10. Hao, L.: Research on the lossless image compression algorithm based on linux embedded system. In: *Proceedings of the 2nd International Conference on Green Communications and Networks 2012 (GCN 2012)*, vol. 1, pp. 711–717 (2013)
11. Kim, C.G., Kim, K.J.: Implementation of a cost-effective home lighting control system on embedded linux with OpenWrt. *Pers. Ubiquit. Comput.* **18**(3), 535–542 (2014)
12. Kim, S.S., Lee, D.G., Park, J.H.: Efficient scheme of verifying integrity of application binaries in embedded operating systems. *J. Supercomputing* **59**(2), 676–692 (2012)
13. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) *EUROCRYPT 1999*. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
14. Li, J., He, Y., Zou, Z., Huang, K.: Design of an embedded multi-biometric recognition platform based on DSP and ARM. In: Sun, Z., Shan, S., Sang, H., Zhou, J., Wang, Y., Yuan, W. (eds.) *CCBR 2014*. LNCS, vol. 8833, pp. 426–433. Springer, Heidelberg (2014)
15. Muthukumar, D., Sawani, A., Schiffman, J., Jung, B.M., Jaeger, T.: Measuring Integrity on Mobile Phone Systems. In: *ACM Symposium on Access Control Models and Technologies*, pp. 155–164 (2008)
16. Oliveira, L.B., Aranha, D.F., Gouva, C.P.L., Scott, M., Cmara, D.F., Lpez, J., Dahab, R.: TinyPBC: pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Comput. Commun.* **34**(3), 485–493 (2011)



17. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
18. Patarin, J., Courtois, N.T., Goubin, L.: QUARTZ, 128-bit long digital signatures. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 282–297. Springer, Heidelberg (2001)
19. Porras, J., Baena, J., Ding, J.: ZHFE, a new multivariate public key encryption scheme. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 229–245. Springer, Heidelberg (2014)
20. Seshadri, A., Perrig, A., Van Doorn, L., Khosla, P.: SWATT: SoftWare-based ATTestation for embedded devices. In: 2012 IEEE Symposium on Security and Privacy, pp. 272–272 (2004)
21. Shen, W., Tang, S.: RGB, a Mixed Multivariate Signature Scheme. *Comput. J.* (2015). doi:[10.1093/comjnl/bxv056](https://doi.org/10.1093/comjnl/bxv056)
22. Shen, W., Tang, S., Xu, L.: IBUOV, A provably secure identity-based UOV signature scheme. In: 2013 IEEE 16th International Conference on Computational Science and Engineering (CSE), pp. 388–395 (2013)
23. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41**(2), 1484–1509 (1997)
24. Tang, S., Yi, H., Ding, J., Chen, H., Chen, G.: High-speed hardware implementation of rainbow signature on FPGAs. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 228–243. Springer, Heidelberg (2011)
25. Unterluggauer, T., Wenger, E.: Efficient pairings and ECC for embedded systems. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 298–315. Springer, Heidelberg (2014)
26. Yang, Bo-Yin, Cheng, Chen-Mou, Chen, Bor-Rong, Chen, Jiun-Ming: Implementing minimized multivariate PKC on low-resource embedded systems. In: Clark, John A., Paige, Richard F., Polack, Fiona A.C., Brooke, Phillip J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 73–88. Springer, Heidelberg (2006)

# **Secure Batch Processing**

# Batch Verifiable Computation with Public Verifiability for Outsourcing Polynomials and Matrix Computations

Yujuan Sun<sup>1,2</sup>, Yu Yu<sup>3,4</sup>(✉), Xiangxue Li<sup>1,2,5</sup>(✉), Kai Zhang<sup>1</sup>,  
Haifeng Qian<sup>1</sup>, and Yuan Zhou<sup>6</sup>

<sup>1</sup> Department of Computer Science and Technology,  
East China Normal University, Shanghai, China  
xxli@cs.ecnu.edu.cn

<sup>2</sup> Westone Cryptologic Research Center, Beijing, China

<sup>3</sup> Department of Computer Science and Engineering,  
Shanghai Jiaotong University, Shanghai, China

<sup>4</sup> State Key Laboratory of Information Security, Institute of Information  
Engineering, Chinese Academy of Sciences, Beijing, China  
yuyu@yuyu.hk

<sup>5</sup> National Engineering Laboratory for Wireless Security, Xi'an University of Posts  
and Telecommunications, Xi'an, China

<sup>6</sup> National Network Emergency Response Technical Team/  
Coordination Center, Beijing, China

**Abstract.** In a *verifiable computation* (VC) scheme, a client asks a server to perform some outsourced computations, and the latter returns the results as its response. The results can be verified privately or publicly. Fiore and Gennaro (CCS 2012) constructed publicly verifiable protocols for secure outsourcing polynomials and matrix computations. *Batch verifiable computation* (BVC) schemes allow a client to outsource multiple functions on a same input, and thus much reduce the storage overhead at the server side without sacrificing the efficiency of verification. However, existing BVC schemes only support private verifiability (which only allows the client who outsources the computations to verify the results). In this paper, we propose BVC schemes with public verifiability, i.e., any third party can efficiently verify the results returned by the server without accessing secret key. To delegate  $s$  functions, our BVC schemes require a cloud storage of only  $1 + 1/s$  times the storage size needed by the  $s$  functions themselves. We extend our schemes to meet less cloud storage overhead as well.

**Keywords:** Batch verifiable computation · Public verifiability · Pseudorandom function · Storage overhead

## 1 Introduction

With the availability of cloud services, outsourcing expensive computations are getting widespread attentions [16, 17, 20, 21]. That is, a client with resource-constraint devices or weak computation abilities outsources computation

workloads to a cloud server and thus enjoys unlimited computing resources in a pay-per-use manner. Since the responses returned by the server may not be correct, they should be verified. This is known as *verifiable computation* (VC) [6, 7, 10, 15, 19]. Without verifying the correctness of the returned results, the server may abuse the inputs they received and then allure the client into accepting the incorrect answer for some reasons (e.g. saving computation or financial problem), which is a severe problem. Secure outsourcing computation constitutes one of the fundamental principles of the new cloud computing paradigm. In a VC scheme, a client outsources computations to a cloud server, who then performs heavy computations and returns to the client the results along with some proofs; and some verifier can then verify the correctness of the results. In a VC scheme with private verifiability, only the client who outsourced the computation can perform the verification. In a VC scheme with public verifiability, however, anyone can do this given the results and some public auxiliary information.

There exists several VC schemes for secure outsourcing computation, and some of them further consider practical solutions for specific functions (e.g. polynomials). We can see VC schemes for generic functions based on fully homomorphic encryption (FHE) [5, 8, 12, 13]. As FHE constructions require heavy cryptographic computation (at least for now), these schemes are mainly of theoretical interest and are too inefficient to be of any practical use. Benabbas et al. [2, 3] constructed practical VC schemes for specific functions such as polynomials, which does not rely on FHE schemes. In these schemes, the client firstly stores an encoding of a polynomial coefficients on a cloud server, and then computes an encoding of the input which is sent to the server; upon receiving the input, the server performs polynomial computation on the input; in addition, there is an efficient algorithm that allows the client to verify whether the value from the server is correct or not. Papamanthou et al. [11] proposed a VC scheme for outsourcing dynamic polynomials, which allows to update the number of the polynomial coefficients incrementally. Note that these schemes require the storage of the encoded functions at last twice of the outsourced function itself. Let the storage overhead be the ratio of the total cloud storage used by encoding the function, to the cloud storage required for the outsourced function itself. And thus we can see that all exiting VC schemes [2, 8, 9, 11] have the storage overhead no less than 2.

The notion of homomorphic signature was proposed by Johnson et al. [14]. The basic idea of homomorphic MACs is that a user can use a secret key to generate a set of tags  $\sigma_1, \dots, \sigma_n$  authenticating values  $D_1, \dots, D_n$ , respectively. Anyone can homomorphically execute a function  $f$  over  $\sigma_1, \dots, \sigma_n$  to generate a short tag  $\sigma$  that authenticates  $D$  as the output of  $f(D_1, \dots, D_n)$ . At first glance, homomorphic MACs seem to perfectly fit the problem of verifiable computations on (growing) outsourced data. Catalano and Fiore [18] realized a practical homomorphic MACs for polynomials. Even though the above scheme admits verification queries, the client's verification time is linear in the outsourced computation. Later the homomorphic MACs of Backes et al. [15] realized amortize verification, but this only admits polynomials with degree no more than 2.

Our VC constructions consider a scenario where a client outsources multiple functions (that belong to a function family) and wants to compute them on a same input. The traditional solution is to execute each function fed with the same input one by one, and this has the following drawbacks: (1) the client must compute the encoding of the input multiple times; (2) and verify the results for these functions one by one. Zhang et al. [1] first proposed the notion of *batch verifiable computation* (BVC) and constructed concrete schemes for securely outsourcing polynomials and matrix computations which reduce the storage overhead. More interestingly, their constructions allow the client to perform verification operation in a batch manner although this is done with the secret key (that is, only the client who outsourced the functions can do this). Nevertheless, publicly verifiable property is important in some contexts where the results have to be verified by several clients (who can not share a secret key). We give a positive answer to the problem by proposing BVC schemes with public verifiability (BVCP) and thus generalize their work.

## 1.1 Our Contribution

Our contributions include the following.

- We introduce the notion of BVC with public verifiability (BVCP), which allows any third party to verify the correctness of the server’s output.
- We propose several concrete schemes for outsourcing polynomials and matrix computations: (1) *Polynomials with total degree bounded* – a client outsources  $s$  polynomials  $f_1, \dots, f_s$  with total degree bounded (degree bounded in each monomial) to a server, and requests the value of  $f_1(\mathbf{x}), \dots, f_s(\mathbf{x})$  for some inputs  $\mathbf{x}$  from the server; (2) *Polynomials with bounded degree in each variable* – a client outsources  $s$  polynomials  $f_1, \dots, f_s$  with degree bounded in each variable to a server, and requests the value of  $f_1(\mathbf{x}), \dots, f_s(\mathbf{x})$  for some inputs  $\mathbf{x}$  from the server; (3) *Matrix function* – a client outsources  $s$   $m \times d$  matrixes  $\mathbf{f}_1, \dots, \mathbf{f}_s$  to a server, and requests the value of  $\mathbf{x} \cdot \mathbf{f}_1, \dots, \mathbf{x} \cdot \mathbf{f}_s$  from the server (note that this scheme can be extended to get a batch matrix multiplication, i.e., with a matrix  $\mathbf{f}'$  and computing the value of  $\mathbf{f}' \cdot \mathbf{f}_1, \dots, \mathbf{f}' \cdot \mathbf{f}_s$  by applying the scheme to each row of  $\mathbf{f}'$ ).
- We extend our last two schemes to enjoy a smaller storage overhead. When the outsourced computations are tera byte magnitude, smaller storage overhead can effectively save the storage of the server.

## 1.2 An Overview of Our Schemes

NOTATIONS. We use  $[n]$  to denote set  $\{1, \dots, n\}$ . We use bold for vectors (e.g.,  $\boldsymbol{\alpha}, \mathbf{a}$ ). Given two vectors  $\mathbf{a}$  and  $\mathbf{b}$ , we use  $\langle \mathbf{a}, \mathbf{b} \rangle$  to denote their inner product. Let  $x \stackrel{\$}{\leftarrow} S$  be the process of uniformly selecting  $x$  at random in a set  $S$ . Given a group  $\mathbb{G}$  of prime order, we use  $e(u, v)$  for a bilinear map, where  $u, v \in \mathbb{G}$ . We use  $F_k$  for a pseudorandom function (PRF) with closed-form efficiency, which will be introduced in Sect. 2.3.

It is folklore that both polynomial computations and matrix computations can be represented by inner product. To describe our schemes, here we use a technique for securely outsourcing the inner product computations. With the aim at computing the inner product between  $s$  vectors  $\mathbf{f}_l = \{f_{l,i}\}_{i \in \mathbb{I}} \in \mathbb{Z}_p^N$  for  $l \in [s]$  and a vector  $\mathbf{y} = (y_i)_{i \in \mathbb{I}} \in \mathbb{Z}_p^N$ , where  $p$  is a prime and  $\mathbb{I}$  is an order set of cardinality  $N$ , the client picks  $\alpha \leftarrow \mathbb{Z}_p$  and  $\mathbf{r} = (r_i)_{i \in \mathbb{I}} \leftarrow \mathbb{Z}_p^N$ , computes the tag  $t_i = \alpha f_{1,i} + \dots + \alpha^s f_{s,i} + r_i$  for every  $i \in \mathbb{I}$ , and stores  $(\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$  on the server. Define  $\boldsymbol{\alpha} = (\alpha, \dots, \alpha^s)$ . The server computes  $\boldsymbol{\rho}_1 = \langle \mathbf{f}_1, \mathbf{y} \rangle, \dots, \boldsymbol{\rho}_s = \langle \mathbf{f}_s, \mathbf{y} \rangle$  and returns  $\boldsymbol{\rho} = (\rho_1, \dots, \rho_s)$  together with a proof  $\pi = \sum_{i \in \mathbb{I}} t_i y_i$ . To verify the returned result, the client needs to compute  $\langle \mathbf{r}, \mathbf{y} \rangle$  and accepts the result only if  $\pi = \langle \boldsymbol{\alpha}, \boldsymbol{\rho} \rangle + \langle \mathbf{r}, \mathbf{y} \rangle$ . This technique has the following drawback: (1) it only admits private verifiability for the client must keep  $\alpha$  secretly; (2) the client has to store  $\mathbf{r}$  and the size of  $\mathbf{r}$  is independent of the outsourced functions. We adapt it to meet public verifiability and use PRFs with closed-form efficiency to enjoy a less storage without sacrificing efficiency. Below we outline our tricks for polynomials with total degree bounded, polynomials with bounded degree in each variable and matrix computations.

**POLYNOMIALS WITH TOTAL DEGREE BOUNDED.** Let  $f_1, \dots, f_s$  be  $m$ -variate polynomials with total degree no more than  $d$  over the finite field  $\mathbb{Z}_p$ , where  $m, d$  are integers. Let  $\mathbb{I} = \{0, 1, \dots, m\}^d$  and  $\mathbf{i} = (i_1, \dots, i_d) \in \mathbb{I}$ . For every  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}_p^m$ , we define  $\mathbf{x}_i = x_{i_1} \cdots x_{i_d}$  and  $x_0 = 1$ . For  $l \in [s]$ ,  $\mathbf{f}_l \stackrel{\text{def}}{=} (f_{l,i} : i \in \mathbb{I})$  contains the corresponding  $l$ -th coefficients of the polynomial and  $\rho_l \stackrel{\text{def}}{=} \sum_{i \in \mathbb{I}} \mathbf{x}_i \cdot f_{l,i}$ . To outsource these  $s$  functions, the client picks  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_s) \leftarrow \mathbb{Z}_p^s$ , then computes  $t_i = g^{\alpha_1 f_{1,i} + \dots + \alpha_s f_{s,i}} \cdot F_k(\mathbf{i})$  for every  $i \in \mathbb{I}$  and sends  $(\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$  to the server. The client computes  $\tau = e(\prod_{i \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}_i}, g)$  and keeps it publicly. Then the server computes  $\rho_1, \dots, \rho_s$  and returns the result  $\boldsymbol{\rho} = (\rho_1, \dots, \rho_s)$  together with a proof  $\pi = \prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}_i}$ . Any third party or the client itself can verify the result  $\boldsymbol{\rho}$  by computing the equation  $e(\pi, g) = e(g, g)^{\langle \boldsymbol{\alpha}, \boldsymbol{\rho} \rangle} \cdot \tau$ . To check the result, the verifier uses the parameters  $e(g, g)^{\alpha_1}, \dots, e(g, g)^{\alpha_s}$  (instead of  $\boldsymbol{\alpha}$  itself), which can be made public.

**POLYNOMIALS WITH BOUNDED DEGREE IN EACH VARIABLE.** Let  $f_1, \dots, f_s$  be  $m$ -variate polynomials with each variable's degree no more than  $d$  over the finite field  $\mathbb{Z}_p$ , where  $m, d$  are integers. Let  $\mathbb{I} = \{0, 1, \dots, d\}^m$  and  $\mathbf{i} = (i_1, \dots, i_m) \in \mathbb{I}$ . For every  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}_p^m$ ,  $\mathbf{x}^i \stackrel{\text{def}}{=} x_1^{i_1} \cdots x_m^{i_m}$ . For  $l \in [s]$ ,  $\mathbf{f}_l \stackrel{\text{def}}{=} (f_{l,i} : i \in \mathbb{I})$  contains the corresponding  $l$ -th coefficients of the polynomial and  $\rho_l \stackrel{\text{def}}{=} \sum_{i \in \mathbb{I}} \mathbf{x}^i \cdot f_{l,i}$ . To outsource these  $s$  functions, the client picks  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_s) \leftarrow \mathbb{Z}_p^s$ , then computes  $t_i = g^{\alpha_1 f_{1,i} + \dots + \alpha_s f_{s,i}} \cdot F_k(\mathbf{i})$  for every  $i \in \mathbb{I}$  and sends  $(\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$  to the server. The client also computes  $\tau = e(\prod_{i \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}^i}, g)$  and keeps it publicly. The server computes  $\rho_1, \dots, \rho_s$  and returns the result  $\boldsymbol{\rho} = (\rho_1, \dots, \rho_s)$  together with a proof  $\pi = \prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}^i}$ . Any third party or the client itself can verify the result  $\boldsymbol{\rho}$  by computing the equation  $e(\pi, g) = e(g, g)^{\langle \boldsymbol{\alpha}, \boldsymbol{\rho} \rangle} \cdot \tau$  with public parameters  $e(g, g)^{\alpha_1}, \dots, e(g, g)^{\alpha_s}$ .

**MATRIX FUNCTIONS.** Let  $\mathbf{f}_1, \dots, \mathbf{f}_s$  be  $m \times d$  matrix functions over the finite field  $\mathbb{Z}_p$ , where  $m, d$  are integers. Define  $\mathbf{f}_l = (f_{l,(i,j)} : (i,j) \in \mathbb{I})$ ,  $\mathbf{x} = (x_1, \dots, x_m)$  and  $\rho_l = (\rho_{l,1}, \dots, \rho_{l,s}) = (\sum_{i=1}^m x_i \cdot f_{l,(i,1)}, \dots, \sum_{i=1}^m x_i \cdot f_{l,(i,d)})$ , where  $\mathbb{I} = [m] \times [d]$ . To outsource these matrix functions, the client picks  $\alpha = (\alpha_1, \dots, \alpha_s) \leftarrow \mathbb{Z}_p^s$ , then computes  $t_{(i,j)} = g^{\alpha_1 f_{1,(i,j)} + \dots + \alpha_s f_{s,(i,j)}} \cdot F_k(i,j)$  for every  $(i,j) \in \mathbb{I}$  and sends  $(\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_{(i,j)}\}_{(i,j) \in [m] \times [d]})$  to the server. The client also computes  $\tau_j = e(\prod_{i=1}^m F_k(i,j)^{x_i}, g)$  for every  $j \in [d]$  and keeps it publicly. The server computes  $\rho_1, \dots, \rho_s$  and returns the results  $\rho_l$  for every  $l \in [s]$  together with a proof  $\pi = (\pi_1, \dots, \pi_d) = (\prod_{i=1}^m (t_{(i,1)})^{x_i}, \dots, \prod_{i=1}^m (t_{(i,d)})^{x_i})$ . Any third party or the client itself can verify the result  $\rho_l$  for every  $l \in [s]$  by computing the equations  $e(\pi_j, g) = e(g, g)^{\sum_{i=1}^s \alpha_i \rho_{l,i}} \cdot \tau_j$ , where  $j \in [d]$ . To check the results, the verifier can use the public parameters  $e(g, g)^{\alpha_1}, \dots, e(g, g)^{\alpha_s}$ .

### 1.3 Organization

Section 2 describes the cryptographic assumptions used in our schemes and provides a formal definition for publicly verifiable BVC and its security. Section 3 reviews three PRFs with closed-form efficiency used as building blocks in our schemes. We propose concrete schemes for three different functions followed by security proofs in Sect. 4. In Sect. 5, we present the efficiency of our schemes that are compared with existing work in [3], and extend the last two schemes to enjoy a less storage overhead. We conclude our work in Sect. 6.

## 2 Preliminaries

In this section, we first present cryptographic assumptions. Then we show a formal definition for BVC with public verifiability and describe desired properties. Finally, we review the formal definition of PRF with closed-form efficiency.

### 2.1 Cryptographic Assumptions

**Definition 1 (DLIN Assumption).** Let  $\lambda$  be a security parameter. Let  $\mathbb{G}$  be a group of prime order  $p$ ,  $g_0, g_1, g_2 \xleftarrow{\$} \mathbb{G}$  be generators, and  $r_0, r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ . We define the advantage of an adversary  $\mathcal{A}$  in deciding the Linear problem in  $\mathbb{G}$  as

$$\text{Adv}_{\mathcal{A}}(\lambda) = \left| \Pr[\mathcal{A}(p, g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, g_0^{r_1+r_2}) = 1] - \Pr[\mathcal{A}(p, g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, g_0^{r_0}) = 1] \right|.$$

We say that the Decision Linear (DLIN) Assumption holds in  $\mathbb{G}$  if for every probabilistic polynomial time (PPT) algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\lambda)$  such that  $\text{Adv}_{\mathcal{A}}(\lambda) \leq \text{negl}(\lambda)$ .

**Definition 2 (CDH Assumption).** Let  $\lambda$  be a security parameter. Let  $\mathbb{G}$  be a group of prime order  $p$ ,  $g \stackrel{\$}{\leftarrow} \mathbb{G}$  be a generator, and  $x, y \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ . We define the advantage of an adversary  $\mathcal{A}$  in computational Diffie-Hellman problem in  $\mathbb{G}$  as

$$Adv_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A}(\lambda, g, g^x, g^y) = g^{xy}]$$

We say that the CDH assumption holds in  $\mathbb{G}$  if for every PPT algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\lambda)$  such that  $Adv_{\mathcal{A}}(\lambda) \leq \text{negl}(\lambda)$ .

## 2.2 BVC with Public Verifiability

A BVCP scheme  $\mathcal{BVCP} = (\text{KeyGen}, \text{ProbGen}, \text{Comp}, \text{Verify})$  consists of four algorithms defined as below:

- $\text{KeyGen}(1^\lambda, f_1, \dots, f_s)$ : On input  $s$  functions  $f_1, \dots, f_s \in \mathcal{F}$  and a security parameter  $\lambda$ , this algorithm generates a secret key  $\text{SK}$  to be stored secretly, a public key  $\text{PK}$  and a verification key  $\text{VK}$  that is public to the verifiers.
- $\text{ProbGen}(\mathbf{x}, \text{SK})$ : On input  $\mathbf{x}$ , this algorithm returns a pair  $(\sigma, \tau)$ , where  $\sigma$  is an encoding of  $\mathbf{x}$  and  $\tau$  is an auxiliary information about  $\mathbf{x}$ .
- $\text{Comp}(\text{PK}, \sigma)$ : The server takes as input  $(\text{PK}, \sigma)$ , and returns a result  $\rho$  and a proof  $\pi$ .
- $\text{Verify}(\text{VK}, \rho, \pi, \tau)$ : On input  $\text{VK}, \rho, \pi$  and  $\tau$ , this algorithm outputs  $\rho$  if  $\rho$  is correct with respect to  $\mathbf{x}$  and an error  $\perp$  otherwise.

In our BVCP model, a client runs algorithm  $\text{KeyGen}(1^\lambda, f_1, \dots, f_s)$  for outsourcing  $s$  functions  $f_1, \dots, f_s \in \mathcal{F}$  to an untrusted server, then sends  $\text{PK}$  to the server, and stores  $\text{SK}$  secretly while keeps  $\text{VK}$  publicly. For any input  $\mathbf{x}$  in the domain of  $f_1, \dots, f_s$ , the client performs algorithm  $\text{ProbGen}(\mathbf{x}, \text{SK})$ . Then the server runs algorithm  $\text{Comp}(\text{PK}, \sigma)$  and returns a pair  $(\rho, \pi)$ . Later a verifier checks the correctness of  $\rho$  with algorithm  $\text{Verify}(\text{VK}, \rho, \pi, \tau)$ . Obviously, a scheme  $\mathcal{BVCP}$  should achieve three fundamental properties: correctness, security and efficiency.

**CORRECTNESS.** This property requires that a verifier can always successfully verify the result returned by an honest server. Formally, for any functions  $f_1, \dots, f_s$  and  $\mathbf{x}$ , if a client runs  $\text{KeyGen}, \text{ProbGen}$  algorithms honestly, and an honest server runs algorithm  $\text{Comp}$  to generate a valid pair  $(\rho, \pi)$ , the output of algorithm  $\text{Verify}$  is always  $\rho$ .

**SECURITY.** A scheme  $\mathcal{BVCP}$  is secure if a malicious server cannot convince a verifier to accept an incorrect result. Formally, a scheme  $\mathcal{BVCP}$  is said to be secure if for any PPT adversary  $\mathcal{A}$ , we have

$$Adv_{\mathcal{A}}^{\mathcal{BVCP}}(f_1, \dots, f_s, \mathbf{x}) \leq \text{negl}(\cdot)$$

where  $Adv_{\mathcal{A}}^{\mathcal{BVCP}}(f_1, \dots, f_s, \mathbf{x}) = \Pr[\text{Game}_{\mathcal{A}}(f_1, \dots, f_s, \mathbf{x}) = 1]$  is defined as the advantage that  $\mathcal{A}$  wins in the game  $\text{Game}_{\mathcal{A}}(f_1, \dots, f_s, \mathbf{x})$ . Below we narrate how  $\mathcal{A}$  plays with the challenger in the  $\text{Game}_{\mathcal{A}}(f_1, \dots, f_s, \mathbf{x})$ :



**Setup.** On input  $f_1, \dots, f_s$ , the challenger runs algorithm `KeyGen`, gives  $\text{PK}, \text{VK}$  to the adversary  $\mathcal{A}$ , and keeps  $\text{SK}$  secretly.

**Queries.** The adversary  $\mathcal{A}$  adaptively makes queries in a polynomial number. For  $j = 1, \dots, q$ :

- $\mathcal{A}$  picks  $\mathbf{x}_j$  in the domain of  $f_1, \dots, f_s$ , and gives it to the challenger.
- The challenger runs algorithm `ProbGen`( $\mathbf{x}_j, \text{SK}$ ), and sends  $\sigma_j, \tau_j$  to  $\mathcal{A}$ .
- $\mathcal{A}$  responds to the challenger with a pair  $(\tilde{\rho}_j, \tilde{\pi}_j)$ .
- The challenger runs `Verify`( $\text{VK}, \tilde{\rho}_j, \tilde{\pi}_j, \tau$ ), and outputs the result  $\tilde{\rho}_j$  or  $\perp$ .

**Forge.**  $\mathcal{A}$  chooses  $\mathbf{x}^*$  in the domain of  $f_1, \dots, f_s$  (note that  $\mathcal{A}$  behaves just like it has done in any one of the  $q$  queries). Then the challenger performs algorithm `ProbGen`( $\mathbf{x}^*, \text{SK}$ ), and gives  $(\sigma^*, \tau^*)$  to  $\mathcal{A}$ .  $\mathcal{A}$  picks a pair  $(\tilde{\rho}^*, \tilde{\pi}^*)$  and sends it to the challenger.

**Output.** The challenger performs the algorithm `Verify`. If the challenger outputs  $\tilde{\rho}^*$  or  $\perp$ , where  $\tilde{\rho}^* \neq (f_1(\mathbf{x}^*), \dots, f_s(\mathbf{x}^*))$ , the game outputs 1 otherwise 0.

**EFFICIENCY.** As outsourcing computations to a server, the client should not be involved in much computable resources. The efficiency property requires that if for any outsourced  $s$  functions  $f_1, \dots, f_s$ , the computable resources required by outsourcing all computations should be less than computing any single outsourced function, respectively.

### 2.3 PRF with Closed Form Efficiency

We use PRF with closed-form efficiency as our building block. A PRF with closed-form efficiency consists of two algorithms  $\Sigma = (\text{Kg}, \text{F})$ , which was first proposed by Benabbas et al. [2]. The key generation algorithm `Kg` takes as input the security parameter  $\lambda$  and some other parameters *params*, and outputs the secret key and parameters that can determine the domain  $\mathcal{X}$  and range  $\mathcal{Y}$  of  $\text{F}$ . On input  $x \in \mathcal{X}$ , the algorithm `F` exploits the secret key  $k$  to compute  $y = \text{F}_k(x)$ , where  $y \in \mathcal{Y}$ . Obviously, it must satisfy the fundamental property of PRF: pseudorandom. That is, for any PPT adversary  $\mathcal{A}$ , we have:

$$\left| \Pr[\mathcal{A}^{\text{F}_k(\cdot)}(1^\lambda, \text{pp}) = 1] - \Pr[\mathcal{A}^{R(\cdot)}(1^\lambda, \text{pp}) = 1] \right| \leq \text{negl}(\lambda),$$

where  $(k, \text{pp}) \leftarrow \text{Kg}(1^\lambda, \text{params})$ , and  $R : \mathcal{X} \rightarrow \mathcal{Y}$  is a random function.

Let `Eval` be an arbitrary computation that takes  $l$  random values  $v_1, \dots, v_l \in \mathcal{Y}$  and  $s$  arbitrary values  $x_1, \dots, x_s \in \mathcal{X}$ . Suppose that the best algorithm to compute `Eval`( $v_1, \dots, v_l, x_1, \dots, x_s$ ) requires time  $t$ . Let  $z_1, \dots, z_l$  be  $l$  values in the domain  $\mathcal{X}$  of  $\text{F}$ . We say that a PRF  $\Sigma = (\text{Kg}, \text{F})$  has closed form efficiency for `Eval` if there exists an algorithm `CFEval` such that `CFEval`( $k, x_1, \dots, x_s$ ) = `Eval`( $\text{F}_k(z_1), \dots, \text{F}_k(z_l), x_1, \dots, x_s$ ) and it only needs time  $\mathcal{O}(t)$ .

### 3 PRF with Closed-Form Efficiency Based on the DLIN Assumption

In this section, we introduce the building block of our schemes: PRF. We present different PRFs have the property of closed-form efficiency for outsourcing polynomials with bounded total degree in each monomial, polynomials with bounded degree in each variable and matrix computations.

#### 3.1 A PRF for Polynomials with Total Degree Bounded

In construction  $\Sigma_1$ , we will use an instantiation of Lewko-Waters function [4], namely, a PRF  $\Sigma_1 = (\text{Kg}, \text{F})$  for outsourcing polynomials with bounded total degree at most  $d$ . More details related to the PRF can be found in [3]. Let  $m, d$  be positive numbers. The construction of  $\Sigma_1$  follows:

- $\text{Kg}(1^\lambda, m, d)$ : Let  $\Lambda = (p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda, 2)$ ,  $k_0, l_0 \leftarrow \mathbb{Z}_p$  and a  $2 \times 2$  matrix  $K_{u,v} \leftarrow \mathbb{Z}_p^{2 \times 2}$  for every  $(u, v) \in [d] \times [m]$ . This algorithm outputs the secret key  $k = \{k_0, l_0\} \cup \{K_{u,v} : (u, v) \in [d] \times [m]\}$  and  $\text{pp} = (\Lambda, m, d)$ . The domain of  $\text{F}$  is  $\mathbb{I} = \{0, 1, \dots, m\}^d$  and the range is  $\mathbb{G}$ .
- $\text{F}$ : On input  $\mathbf{i} = (i_1, \dots, i_d)$ , each  $i_u$  is an integer in  $\{0, \dots, m\}$ . This algorithm outputs  $\text{F}_k(\mathbf{i}) = g^{\xi_{\mathbf{i}}}$ , where  $(\xi_{\mathbf{i}}, \eta_{\mathbf{i}}) = (k_0, l_0) \cdot \prod_{u=1}^m K_{u,i_u}$ .

Note that  $\Sigma_1$  is secure under the DLIN assumption. Define  $\text{Eval}(\{\text{F}_k(\mathbf{i})\}_{\mathbf{i} \in \mathbb{I}}, \mathbf{x}) = \prod_{\mathbf{i} \in \mathbb{I}} \text{F}_k(\mathbf{i})^{x_{\mathbf{i}}} = g^\xi$ , where

$$\begin{aligned} \xi &= \sum_{\mathbf{i} \in \mathbb{I}} (\xi_{\mathbf{i}}, \eta_{\mathbf{i}}) \cdot \mathbf{x}_{\mathbf{i}} = \sum_{i_1}^m \cdots \sum_{i_d}^m (k_0, l_0) \cdot \prod_{u=1}^d K_{u,i_u} \cdot x_{i_u} \\ &= (k_0, l_0) \prod_{u=1}^d (K_{u,0} + K_{u,1} \cdot x_1 + \cdots + K_{u,m} \cdot x_m). \end{aligned}$$

Without  $k$ , computing  $\text{Eval}(\{\text{F}_k(\mathbf{i})\}_{\mathbf{i} \in \mathbb{I}}, \mathbf{x})$  requires  $\mathcal{O}(|\mathbb{I}|) = \mathcal{O}((m+1)^d)$  exponentiations in  $\mathbb{G}$ . While given  $k$ , it only requires  $(4md + 4d)$  multiplications in  $\mathbb{Z}_p$ ,  $(4m + 2d)$  additions in  $\mathbb{Z}_p$  and one exponentiation in  $\mathbb{G}$  (note that the above computation about  $\xi$  can be represented by multiplication and addition in  $\mathbb{Z}_p$ ). Hence,  $\Sigma_1$  has closed-form efficiency for  $\text{Eval}$ .

#### 3.2 A PRF for Polynomials with Bounded Degree in Each Variable

In construction  $\Sigma_2$ , we will use a PRF with closed-form efficiency constructed by Fiore et al. [3], namely, a PRF  $\Sigma_2 = (\text{Kg}, \text{F})$  for polynomials with each variable's degree no more than  $d$ . Let  $m, a = \lceil \log(d+1) \rceil$  be positive numbers. The construction of  $\Sigma_2$  follows:

- $\text{Kg}(1^\lambda, m, a)$ : Let  $\Lambda = (p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda, 2)$ ,  $k_0, l_0 \leftarrow \mathbb{Z}_p$  and a  $2 \times 2$  matrix  $K_{u,v} \leftarrow \mathbb{Z}_p^{2 \times 2}$  for every  $(u, v) \in [m] \times [a]$ . This algorithm outputs  $k = \{k_0, l_0\} \cup \{K_{u,v} : (u, v) \in [m] \times [a]\}$  and  $\text{pp} = (\Lambda, m, a)$ . Generally, the domain of  $\text{F}$  is  $\mathbb{I} = \{0, 1, \dots, d\}^m$  and the range is  $\mathbb{G}$ .

- **F**: On input  $\mathbf{i} = (i_1, \dots, i_m)$ , this algorithm computes the bit representations  $i_u = (i_{u,1}, \dots, i_{u,a})$  for each  $u \in [m]$  and outputs  $F_k(\mathbf{i}) = g^{\xi \mathbf{i}}$ , where  $(\xi_{\mathbf{i}}, \eta_{\mathbf{i}}) = (k_0, l_0) \cdot \prod_{u=1}^m \prod_{v=1}^a K_{u,v}^{i_{u,v}}$ .

Note that  $\Sigma_2$  is secure under the DLIN assumption. Define  $\text{Eval}(\{F_k(\mathbf{i})\}_{\mathbf{i} \in \mathbb{I}}, \mathbf{x}) = \prod_{\mathbf{i} \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}^{\mathbf{i}}} = g^{\xi}$ , where

$$\begin{aligned} \xi &= \sum_{\mathbf{i} \in \mathbb{I}} (\xi_{\mathbf{i}}, \eta_{\mathbf{i}}) \cdot \mathbf{x}^{\mathbf{i}} = \sum_{i_1=0}^{2^a-1} \cdots \sum_{i_m=0}^{2^a-1} (k_0, l_0) \prod_{u=1}^m \prod_{v=1}^a K_{u,v}^{i_{u,v}} \cdot x_u^{i_{u,v} \cdot 2^{v-1}} \\ &= (k_0, l_0) \prod_{u=1}^m \prod_{v=1}^a (E + K_{u,v} \cdot x_u^{v-1}) \end{aligned}$$

and  $E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ . Without  $k$ , computing  $\text{Eval}(\{F_k(\mathbf{i})\}_{\mathbf{i} \in \mathbb{I}}, \mathbf{x})$  requires  $\mathcal{O}(|\mathbb{I}|) = \mathcal{O}((d+1)^m)$  exponentiations in  $\mathbb{G}$ . Given  $k$ , it only requires  $8ma$  multiplications in  $\mathbb{Z}_p$ ,  $6ma$  additions in  $\mathbb{Z}_p$  and one exponentiation in  $\mathbb{G}$  (note that the above computation about  $\xi$  can be represented by multiplication and addition in  $\mathbb{Z}_p$ ). Hence,  $\Sigma_2$  has closed-form efficiency for  $\text{Eval}$ .

### 3.3 A PRF for Matrix Functions

In construction  $\Sigma_3$ , we will use a PRF with closed-form efficiency constructed by Fiore et al. [3], namely, a PRF  $\Sigma_3 = (\text{Kg}, \text{F})$  for outsourcing matrix computations. Let  $m, d$  be positive numbers. The construction of  $\Sigma_3$  follows:

- **Kg**( $1^\lambda, m, d$ ): Let  $\Lambda = (p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda, 2)$ ,  $u_i, v_i \leftarrow \mathbb{G}$  for every  $i \in [m]$ ,  $k_j, l_j \leftarrow \mathbb{Z}_p$  for every  $j \in [d]$ . This algorithm outputs the secret key  $k = \{(u_i, v_i) : i \in [m]\} \cup \{(k_j, l_j) : j \in [d]\}$  and  $\text{pp} = (\Lambda, m, d)$ . Generally, the domain of **F** is  $\mathbb{I} = [m] \times [d]$  and the range is  $\mathbb{G}$ .
- **F**: On input  $\mathbf{i} = (i, j) \in \mathbb{I}$ , this algorithm outputs  $F_k(i, j) = u_i^{k_j} v_i^{l_j}$ .

Note that  $\Sigma_3$  is secure the DLIN assumption. Define  $\text{Eval}(\mathbf{y}, \mathbf{x}) = (\prod_{i=1}^m (y_{i,1})^{x_i}, \prod_{i=1}^m (y_{i,2})^{x_i}, \dots, \prod_{i=1}^m (y_{i,d})^{x_i})$ , where  $\mathbf{y} = (y_{i,j})_{m \times d} \in \mathbb{G}^{m \times d}$  is a  $m \times d$  matrix and  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}_p^m$ . Then

$$\begin{aligned} \text{Eval}(F_k(i, j)_{m \times d}, \mathbf{x}) &= \left( \prod_{i=1}^m F_k(i, 1)^{x_i}, \prod_{i=1}^m F_k(i, 2)^{x_i}, \dots, \prod_{i=1}^m F_k(i, d)^{x_i} \right) \\ &= \left( \left( \prod_{i=1}^m u_i^{x_i} \right)^{k_1} \left( \prod_{i=1}^m v_i^{x_i} \right)^{l_1}, \dots, \left( \prod_{i=1}^m u_i^{x_i} \right)^{k_d} \left( \prod_{i=1}^m v_i^{x_i} \right)^{l_d} \right). \end{aligned}$$

Without  $k$ , computing  $\text{Eval}(F_k(i, j)_{m \times d}, \mathbf{x})$  requires  $\mathcal{O}(md)$  exponentiations and multiplications in  $\mathbb{Z}_p$ . Given  $k$ , it only requires  $2(m+d)$  exponentiations and  $(2m-1)$  multiplications in  $\mathbb{Z}_p$ . Hence,  $\Sigma_3$  has closed-form efficiency for  $\text{Eval}$ .

## 4 Our Schemes

In this section, we propose concrete batch verifiable computation schemes with public verifiability for outsourcing different polynomial computations and matrix computations.

### 4.1 Polynomials with Total Degree Bounded

Let  $\mathcal{F}_1$  be the function family of all  $m$ -variate polynomials with total degree in each monomial no more than  $d$  over the finite field  $\mathbb{Z}_p$ , where  $m, d$  are integers. Let  $f_1, \dots, f_s \in \mathcal{F}_1$  be the outsourced polynomials. Let  $\mathbb{I} = \{0, 1, \dots, m\}^d$  and  $\mathbf{i} = (i_1, \dots, i_d) \in \mathbb{I}$ . For every  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}_p^m$ ,  $\mathbf{x}_i \stackrel{\text{def}}{=} x_{i_1} \cdots x_{i_d}$  and  $x_0 = 1$ . For  $l \in [s]$ ,  $\mathbf{f}_l \stackrel{\text{def}}{=} (f_{l,i} : \mathbf{i} \in \mathbb{I})$  contains the corresponding coefficients of  $l$ -th polynomials and  $\rho_l \stackrel{\text{def}}{=} \langle \mathbf{f}_l(\mathbf{x}), \mathbf{f}_l \rangle$ , where  $\mathbf{y} = (\mathbf{x}_i : \mathbf{i} \in \mathbb{I})$ . Below is our scheme  $\mathcal{BVC}\mathcal{P}_1$  for outsourcing  $f_1, \dots, f_s \in \mathcal{F}_1$  obtained by PRF  $\Sigma_1 = (\text{Kg}, \text{F})$ , which is instantiated in Sect. 3.1.

- $\text{KeyGen}(1^\lambda, f_1, \dots, f_s)$ : Pick  $(\text{pp}, k) \leftarrow \Sigma_1.\text{Kg}(1^\lambda, m, d)$  and a vector  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_s) \leftarrow \mathbb{Z}_p^s$ . Let  $h_l = e(g, g)^{\alpha_l}$  for every  $l \in [s]$ . This algorithm computes  $t_i = g^{\alpha_1 f_{1,i} + \dots + \alpha_s f_{s,i}} \cdot F_k(\mathbf{i})$  for every  $\mathbf{i} \in \mathbb{I}$ , then outputs  $\text{PK} = (\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$ ,  $\text{SK} = k$  and  $\text{VK} = (h_1, \dots, h_s)$ .
- $\text{ProbGen}(\mathbf{x}, \text{SK})$ : On input  $\mathbf{x}$  and  $\text{SK}$ , the client outputs  $\sigma = \mathbf{x}$  and  $\tau = e(\prod_{i \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}_i}, g)$ .
- $\text{Comp}(\text{PK}, \sigma)$ : The server computes  $\rho_l = \sum_{i \in \mathbb{I}} f_{l,i} \cdot \mathbf{x}_i$  for every  $l \in [s]$  and a proof  $\pi = \prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}_i}$ . Then the server outputs  $\boldsymbol{\rho} = (\rho_1, \dots, \rho_s)$  and  $\pi$ .
- $\text{Verify}(\text{VK}, \boldsymbol{\rho}, \pi, \tau)$ : A verifier checks the validity of  $\boldsymbol{\rho}$  by verifying the equation  $e(\pi, g) = \prod_{l=1}^s h_l^{\rho_l} \cdot \tau$ . The verifier outputs  $\boldsymbol{\rho}$  if the equation holds and an error  $\perp$  otherwise.

**Theorem 1.** *The proposed scheme  $\mathcal{BVC}\mathcal{P}_1$  is correct.*

*Proof.* Suppose that the server is honest. Giving the proof  $\pi = \prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}_i}$ , we have  $e(\pi, g) = e(g^{\sum_{l=1}^s \alpha_l \rho_l} \cdot \prod_{i \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}_i}, g)$  and  $\prod_{l=1}^s h_l^{\rho_l} \cdot \tau = e(g^{\sum_{l=1}^s \alpha_l \rho_l} \cdot \prod_{i \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}_i}, g)$ . Obviously, we get  $e(\pi, g) = \prod_{l=1}^s h_l^{\rho_l} \cdot \tau$ . Hence, the output of algorithm  $\text{Verify}$  is always  $\boldsymbol{\rho}$ .

**Theorem 2.** *The proposed scheme  $\mathcal{BVC}\mathcal{P}_1$  is secure under the DLIN assumption in  $\mathbb{G}$ .*

*Proof.* Let  $\text{Game}_0, \text{Game}_1$  be two security games. The game  $\text{Game}_0$  is a standard security game for scheme  $\mathcal{BVC}\mathcal{P}_1$  while  $\text{Game}_1$  is the same as  $\text{Game}_0$  except that  $\text{Game}_1$  uses a random function  $\Psi : \mathbb{I} \rightarrow \mathbb{G}$  instead of the function  $F_k$ . We define  $\epsilon_i$  be the probability that the adversary  $\mathcal{A}$  wins in  $\text{Game}_i$  for each  $i \in \{0, 1\}$ . To prove the security of scheme  $\mathcal{BVC}\mathcal{P}_1$ , we need to show  $\epsilon_0 \leq \text{negl}(\lambda)$ . If  $|\epsilon_0 - \epsilon_1|$  is not negligible, one can use adversary  $\mathcal{A}$  to break the security of

scheme  $\mathcal{BVCP}_1$ . Therefore,  $|\epsilon_0 - \epsilon_1|$  must be negligible. Obviously, we just need to show  $\epsilon_1 \leq \text{negl}(\lambda)$ . We prove the security of  $\mathcal{BVCP}_1$  by contradiction. Supposed that  $\epsilon_1$  is non-negligible. We show that there is a challenger that can simulate  $\mathcal{A}$  to solve the CDH problem (on inputs  $(g, g^\alpha, h)$ , the challenger outputs  $h^\alpha$ ), which should be harder than the DLIN problem. The following shows how the challenger plays with  $\mathcal{A}$  in game  $\text{Game}_1$ .

**Setup.** On input  $1^\lambda, f_1, \dots, f_s$ , the challenger performs similarly to the algorithm  $\text{KeyGen}$ : picks  $r \leftarrow [s]$  and computes  $h_r = e(g^\alpha, h)$ ; picks  $\alpha_l \leftarrow \mathbb{Z}_p$  and computes  $h_l = e(g, g)^{\alpha_l}$  for each  $l \in [s] \setminus r$ . Then the challenger picks  $t_i \leftarrow \mathbb{G}$  for every  $i \in \mathbb{I}$  and defines  $\text{PK} = (\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$ ,  $\text{SK} = \perp$  and  $\text{VK} = (h_1, \dots, h_s)$ . At last, the challenger sends  $\text{PK}, \text{VK}$  to  $\mathcal{A}$ .

**Query.** The adversary  $\mathcal{A}$  adaptively queries polynomial times. For  $j = 1, \dots, q$ ,  $\mathcal{A}$  and the challenger interact as below:

- $\mathcal{A}$  picks  $\mathbf{x}_j = (x_{j,1}, \dots, x_{j,m})$  and sends it to the challenger.
- On input  $\mathbf{x}_j$ , the challenger computes  $\sigma_j = \mathbf{x}_j$  and  $\tau_j = e(\prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}_j^i}, g) / (\prod_{l=1}^s h_l^{\sum_{i \in \mathbb{I}} f_{l,i} \cdot \mathbf{x}_j^i})$ . Then it sends them to  $\mathcal{A}$ .
- $\mathcal{A}$  picks  $\tilde{\rho}_j \in \mathbb{Z}_p^s$  and  $\tilde{\pi}_j \in \mathbb{G}$  and sends them to the challenger.
- The challenger verifies the correctness of  $\tilde{\rho}_j \in \mathbb{Z}_p^s$  by running algorithm  $\text{Verify}(\text{VK}, \tilde{\rho}_j, \tilde{\pi}_j, \tau_j)$ , and outputs the result to  $\mathcal{A}$ .

**Forgery.**  $\mathcal{A}$  picks  $\mathbf{x}^* \in \{\mathbf{x}_j\}_{j \in [q]}$  and outputs a tuple  $(\mathbf{x}^*, \tilde{\rho}^*, \tilde{\pi}^*)$  as its forgery.

Define  $(\rho^*, \pi^*)$  be the output of algorithm  $\text{Comp}$  under an honest server, where  $\rho^* = (\rho_1^*, \dots, \rho_s^*)$  and  $\pi^* \in \mathbb{G}$ . Since the scheme  $\mathcal{BVCP}_1$  is correct, we have that  $e(\pi^*, g) = \prod_{l=1}^s h_l^{\rho_l^*} \cdot \tau^*$ .  $\mathcal{A}$  wins the game  $\text{Game}_1$  only if  $\rho^* \neq \tilde{\rho}^*$  and  $e(\tilde{\pi}^*, g) = \prod_{l=1}^s h_l^{\tilde{\rho}_l^*} \cdot \tau^*$ . So we have  $e(\tilde{\pi}^*/\pi^*, g) = \prod_{l=1}^s h_l^{\tilde{\rho}_l^* - \rho_l^*}$ , which implies that  $e(\tilde{\pi}^*/\pi^*, g) \cdot \prod_{l \in [s] \setminus r} h_l^{-(\tilde{\rho}_l^* - \rho_l^*)} = e(h^{\alpha(\tilde{\rho}_r^* - \rho_r^*)}, g)$ . So the challenger can compute  $h^\alpha = (\tilde{\pi}^* \cdot (\pi^*)^{-1} \cdot \prod_{l \in [s] \setminus r} h_l^{-(\tilde{\rho}_l^* - \rho_l^*)})^{(\tilde{\rho}_r^* - \rho_r^*)^{-1}}$ .

Below we show the probability that the challenger uses  $\mathcal{A}$  to solve the CDH problem. At the beginning of the proof, we assume the probability that  $\mathcal{A}$  wins the game  $\text{Game}_1$  is  $\epsilon_1$ , which is non-negligible. For  $\rho^* \neq \tilde{\rho}^*$ , there is a nonempty set  $R \subseteq [s]$  such that  $\rho_{r^*}^* \neq \tilde{\rho}_{r^*}^*$  for any  $r^* \in R$ . In the **Setup** phase, the challenger chooses  $r$  uniformly random and independently. Hence, the probability that  $r \in R$  is no less than  $|R|/s$ , and even no less than  $1/s$ . The probability that the challenger learns  $h^\alpha$  is the same as  $\mathcal{A}$  wins in the game  $\text{Game}_1$  and  $r \in R$ , which is no less than  $\epsilon_1/s$ . Therefore, the challenger can solve the CDH problem with a non-negligible probability no less than  $\epsilon_1/s$ . This contradicts the hardness of CDH and thus the DLIN assumption. Therefore,  $\epsilon_1$  must be negligible, and then the scheme  $\mathcal{BVCP}_1$  is secure under the DLIN assumption.

**Theorem 3.** *The proposed scheme  $\mathcal{BVCP}_1$  is efficient.*

*Proof.* It is trivial that the computable resources invested by the client is independent of the size of the outsourced functions except the algorithm  $\text{KeyGen}$ .

Since we always outsource large functions, the algorithm **KeyGen** requires one-time expensive computable resources, but the resources can be amortized over all future executions of the same outsourced functions on different inputs. For the algorithm **ProbGen**, it needs roughly  $1/s$  times computable resources of outsourcing  $s$  functions individually. When the client wants to verify the correctness of  $\rho$ , it needs to run algorithm **Verify** and the required computable resources is independent of the size of the outsourced functions. Hence, our scheme  $\mathcal{BVC}\mathcal{P}_1$  is efficient.

### 4.2 Polynomials with Bounded Degree in Each Variable

Let  $\mathcal{F}_2$  be the function family of all  $m$ -variate polynomials with degree in each variable at most  $d$  over the finite field  $\mathbb{Z}_p$ , where  $m, d$  are integers. Define  $a = \lceil \log(d + 1) \rceil$ . Let  $f_1, \dots, f_s \in \mathcal{F}_2$  be the outsourced functions. Let  $\mathbb{I} = \{0, 1, \dots, d\}^m$  and  $\mathbf{i} = (i_1, \dots, i_m) \in \mathbb{I}$ . For every  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}_p^m$ ,  $\mathbf{x}^{\mathbf{i}} \stackrel{\text{def}}{=} x_1^{i_1} \cdots x_m^{i_m}$ . For every  $l \in [s]$ ,  $\mathbf{f}_l \stackrel{\text{def}}{=} (f_{l,i} : \mathbf{i} \in \mathbb{I})$  contains the corresponding coefficients of  $l$ -th polynomials and  $\rho_l = f_l(\mathbf{x}) = \langle \mathbf{y}, \mathbf{f}_l \rangle$ , where  $\mathbf{y} = (\mathbf{x}^{\mathbf{i}} : \mathbf{i} \in \mathbb{I})$ . Below is scheme  $\mathcal{BVC}\mathcal{P}_2$  obtained by PRF  $\Sigma_2$ , which is described in Sect. 3.2.

- **KeyGen**( $1^\lambda, f_1, \dots, f_s$ ): Pick  $(\text{pp}, k) \leftarrow \Sigma_2.\text{Kg}(1^\lambda, m, a)$  and a vector  $\alpha = (\alpha_1, \dots, \alpha_s) \leftarrow \mathbb{Z}_p^s$ . Let  $h_l = e(g, g)^{\alpha_l}$  for every  $l \in [s]$ . This algorithm computes  $t_i = g^{\alpha_1 f_{1,i} + \dots + \alpha_s f_{s,i}} \cdot F_k(\mathbf{i})$  for every  $\mathbf{i} \in \mathbb{I}$ , then outputs  $\text{PK} = (\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{\mathbf{i} \in \mathbb{I}})$ ,  $\text{SK} = k$  and  $\text{VK} = (h_1, \dots, h_s)$ .
- **ProbGen**( $\mathbf{x}, \text{SK}$ ): On input  $\mathbf{x}$  and  $\text{SK}$ , the client outputs  $\sigma = \mathbf{x}$  and  $\tau = e(\prod_{\mathbf{i} \in \mathbb{I}} F_k(\mathbf{i})^{\mathbf{x}^{\mathbf{i}}}, g)$ .
- **Comp**( $\text{PK}, \sigma$ ): The server computes  $\rho_l = \sum_{\mathbf{i} \in \mathbb{I}} f_{l,i} \cdot \mathbf{x}^{\mathbf{i}}$  for every  $l \in [s]$  and a proof  $\pi = \prod_{\mathbf{i} \in \mathbb{I}} (t_i)^{\mathbf{x}^{\mathbf{i}}}$ . Then the server outputs  $\rho = (\rho_1, \dots, \rho_s)$  and  $\pi$ .
- **Verify**( $\text{VK}, \rho, \pi, \tau$ ): A verifier outputs  $\rho$  if  $e(\pi, g) = \prod_{l=1}^s h_l^{\rho_l} \cdot \tau$  holds and an error  $\perp$  otherwise.

We can prove that  $\mathcal{BVC}\mathcal{P}_2$  is secure under the DLIN assumption in  $\mathbb{G}$  similarly to  $\mathcal{BVC}\mathcal{P}_1$ . Besides, it is easy to see that  $\mathcal{BVC}\mathcal{P}_2$  is correct and efficient.

### 4.3 Matrix Computations

Let  $\mathcal{F}_3$  be the family of all  $m \times d$  matrix functions over the finite field  $\mathbb{Z}_p$ , where  $m, d$  are integers. Let  $\mathbb{I} = [m] \times [d]$ . A client outsources  $s$  matrices  $\mathbf{f}_1, \dots, \mathbf{f}_s \in \mathcal{F}_3$  to a server and wants to compute  $\mathbf{x} \cdot \mathbf{f}_l$  for every  $l \in [s]$ , where  $\mathbf{x} = (x_1, \dots, x_m)$ . For every  $l \in [s]$ , define  $\mathbf{f}_l = (f_{l,(i,j)} : (i,j) \in \mathbb{I})$  and  $\rho_l = (\rho_{l,1}, \dots, \rho_{l,d}) = (\sum_{i=1}^m x_i \cdot f_{l,(i,1)}, \dots, \sum_{i=1}^m x_i \cdot f_{l,(i,d)})$ . We obtain the scheme  $\mathcal{BVC}\mathcal{P}_3$  for outsourcing  $\mathbf{f}_1, \dots, \mathbf{f}_s \in \mathcal{F}_3$  by running our BVCP model  $d$  times in parallel. Below is the construction of  $\mathcal{BVC}\mathcal{P}_3$  obtained by PRF  $\Sigma_3$ , which is described in Sect. 3.3.

- **KeyGen**( $1^\lambda, \mathbf{f}_1, \dots, \mathbf{f}_s$ ): Pick  $(\mathbf{pp}, k) \leftarrow \Sigma_3.\text{Kg}(1^\lambda, (m, d))$ , and pick a vector  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_s) \leftarrow \mathbb{Z}_p^s$ . Let  $h_l = e(g, g)^{\alpha_l}$  for every  $l \in [s]$ . This algorithm computes  $t_{(i,j)} = g^{\alpha_1 f_{1,(i,j)} + \dots + \alpha_s f_{s,(i,j)}} \cdot F_k(i, j)$  for every  $(i, j) \in \mathbb{I}$ , then outputs  $\text{PK} = (\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_{(i,j)}\}_{(i,j) \in \mathbb{I}})$ ,  $\text{SK} = k$  and  $\text{VK} = (h_1, \dots, h_s)$ .
- **ProbGen**( $\mathbf{x}, \text{SK}$ ): On input  $\mathbf{x}$  and  $\text{SK}$ , the client outputs  $\sigma = \mathbf{x}$  and  $\tau = (\tau_1, \dots, \tau_d)$ , where  $\tau_j = e(\prod_{i=1}^m F_k(i, j)^{x_i}, g)$ .
- **Comp**( $\text{PK}, \sigma$ ): The server computes  $\rho_l$  for every  $l \in [s]$  and  $\pi_j = \prod_{i=1}^m (t_{(i,j)})^{x_i}$  for every  $j \in [d]$  and outputs  $\boldsymbol{\rho} = (\rho_1, \dots, \rho_s)$  and  $\pi = (\pi_1, \dots, \pi_d)$ .
- **Verify**( $\text{VK}, \boldsymbol{\rho}, \pi, \tau$ ): The verifier checks the validity of  $\rho_l$  with  $e(\pi_j, g) = \prod_{i=1}^s h_i^{\rho_l^{i,j}} \cdot \tau_j$  for every  $j \in [d]$ . The verifier outputs  $\boldsymbol{\rho}$  if the above equations hold and an error  $\perp$  otherwise.

We can prove  $\mathcal{BVC}\mathcal{P}_3$  is secure under the DLIN assumption in  $\mathbb{G}$  similarly to  $\mathcal{BVC}\mathcal{P}_1$ . Besides, it is easy to see that  $\mathcal{BVC}\mathcal{P}_3$  is correct and efficient.

## 5 Performance Analysis and Extension

In this section, we analyze the storage overhead and computable resources of our schemes and make a comparison with schemes [3]. Besides, we extend our scheme  $\mathcal{BVC}\mathcal{P}_2$  as an example to make the server enjoy less storage for outsourcing  $s$  function computations on the same input.

### 5.1 Efficiency Analysis

We recall that the storage overhead denotes the ratio of the total cloud storage required by encoding the function, to the cloud storage required for the outsourced function itself. The storage overhead of the schemes proposed in [3] is 2, while our schemes achieve that with  $1 + 1/s$ . We take  $\mathcal{BVC}\mathcal{P}_1$  as an example. In the scheme  $\mathcal{BVC}\mathcal{P}_1$ , the public key  $\text{PK} = (\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$ . For every  $l \in s$ , where  $\mathbf{f}_l$  can be represented as  $(m + 1)^d$  elements in  $\mathbb{Z}_p$ ; for every  $i \in \mathbb{I}$ ,  $t_i$  is an element in a group  $\mathbb{G}$  of prime order  $p$ . To outsource  $s$  functions  $\mathbf{f}_1, \dots, \mathbf{f}_s$ , the storage overhead is  $(s(m + 1)^d + |\mathbb{I}|)/(m + 1)^d = 1 + 1/s$ .

We take  $\mathcal{BVC}\mathcal{P}_1$  as an example to analyze the communication overhead. In the algorithm **KeyGen**, the client stores  $\text{SK}$  (an element in  $\mathbb{Z}_p$ ) secretly, keeps  $\text{VK}$  (containing  $s$  bilinear map elements) publicly and sends  $\text{PK}$  to the server. The algorithm **ProbGen** takes  $\mathbf{x}$  (containing  $m$  elements in  $\mathbb{Z}_p$ ) as input, and outputs  $\sigma$  (containing  $m$  elements in  $\mathbb{Z}_p$ ) and  $\tau$  (a bilinear map). On input  $\text{PK}$  and  $\sigma$ , the server computes  $\boldsymbol{\rho}$  (containing  $s$  elements in  $\mathbb{Z}_p$ ) and a proof  $\pi$  (an element in a group  $\mathbb{G}$ ) in the algorithm **Comp**. Finally, the verifier stores  $\text{VK}, \boldsymbol{\rho}, \pi$  and  $\tau$ , checks the correctness of  $\boldsymbol{\rho}$  and outputs  $\boldsymbol{\rho}$  or  $\perp$  in the algorithm **Verify**.

Table 1 gives a description of the comparison between our schemes and [3]. Here we define  $A_p, E_p$  and  $M_p$  be the number of bit operations required for one addition, one exponentiation and multiplication modulo  $p$ . We denote by  $E_{\mathbb{G}}$  an exponentiation in  $\mathbb{G}$ . Let  $P$  be the number of bit operations required for a pairing computation. Let  $E_b, M_b$  and  $C_b$  be the number of bit operations required for

**Table 1.** Comparison with publicly verifiable VC schemes.

Functions	Schemes	Storage	Client Computation	Verify Computation
$\mathcal{F}_1$	$\mathcal{BVC}\mathcal{P}_1$	$1 + 1/s$	$(4md + 4d)M_p$ $+E_G + P$	$sE_b + sM_b$ $+P + C_b$
	[3]	2	$(4md + 4d)sM_p$ $+sE_G + sP$	$sE_b + sM_b$ $+sP + sC_b$
$\mathcal{F}_1$	$\mathcal{BVC}\mathcal{P}_2$	$1 + 1/s$	$8maM_p$ $+E_G + P$	$sE_b + sM_b$ $+P + C_b$
	[3]	2	$8masM_p$ $+sE_G + sP$	$sE_b + sM_b$ $+sP + sC_b$
$\mathcal{F}_1$	$\mathcal{BVC}\mathcal{P}_3$	$1 + 1/s$	$(2m + d - 2)dM_p$ $+(2m + 2d)dE_p$	$dsE_b + dsM_b$ $+dP + dC_b$
	[3]	2	$(2m + d - 2)dsM_p$ $+(2m + 2d)dsE_p$	$dsE_b + dsM_b$ $+dsP + dsC_b$

one exponentiation of pairing, one multiplication between two pairings and one comparison of two elements in pairings.

Both our schemes and [3] require one-time expensive KeyGen phase, where the cost can be amortized over future function computations. Therefore, we only consider the computable resources of other algorithms in these schemes at this point. In  $\mathcal{BVC}\mathcal{P}_1$ , the client needs to compute  $\tau$ , which requires  $(4md + 4d)M_p + (4m + 2d)A_p + E_G + P$  bit operations. Then the verifier requires  $sE_b + sM_b + P + C_b$  bit operations to verify the result. In  $\mathcal{BVC}\mathcal{P}_2$ , the client needs  $(8ma)M_p + (6ma)A_p + E_G + P$  bit operations to compute  $\tau$ . Then the verifier requires  $sE_b + sM_b + P + C_b$  bit operations to verify the result. In  $\mathcal{BVC}\mathcal{P}_3$ , the client needs to compute  $\tau$ , which requires  $(2m + d - 2)M_p + (2m + 2d)E_p$  bit operations. Then the verifier requires  $sE_b + sM_b + P + C_b$  bit operations to verify  $e(\pi_j, g) = \prod_{l=1}^s h_l^{\alpha_{l,j}} \cdot \tau_j$  for every  $j \in [d]$ . To verify the  $s$  outsourced functions, the verifier requires  $(sE_b + sM_b + P + C_b)d$  bit operations. We note that Table 1 considers all operations in our schemes and [3] except addition (requiring far less computable resources than other operations).

### 5.2 Extension

Since the outsourced functions may be tera byte magnitude, in such a way that the server needs quantity of storage to store the encoded functions. It is preferable that reducing the storage overhead when the client does not compute outsourced functions frequently. In our last two schemes, we reduce the storage overhead less than  $1 + 1/s$ . Here we just extend  $\mathcal{BVC}\mathcal{P}_2$  as an example.

We divide every outsourced function into  $q$  blocks, where  $q$  is a positive number. Let  $n = (d + 1)/q$ . Define the outsourced functions represented as  $\mathbf{f}_1 = (f_{(1,j),i} : j \in [q], i \in \mathbb{I}), \dots, \mathbf{f}_s = (f_{(s,j),i} : j \in [q], i \in \mathbb{I})$ , where  $\mathbb{I} = \{0, \dots, n - 1\} \times \{0, \dots, d\}^{m-1}$ . The client picks a vector  $\alpha = (\alpha_{l,j} \leftarrow \mathbb{Z}_p : l \in [s], j \in [q])$ ,



computes a tag  $t_i$  of the data blocks  $(f_{(l,j),i} : l \in [s], j \in [q])$  for every  $i \in \mathbb{I}$ , and stores  $(\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$  on the server. With the received input, the server computes  $(\{\rho_{l,j} = \sum_{i \in \mathbb{I}} f_{(l,j),i} \cdot \mathbf{x}^i\}_{l \in [s], j \in [q]})$  and a proof  $\pi = \prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}^i}$ . Finally, anyone can check the correctness of  $\rho$  with  $\text{vk}, \rho, \pi$  and  $\tau$ . Define  $a = \log(d+1)$  and  $b = \log n$ . Below we construct the detail scheme extended from  $\mathcal{BVC}\mathcal{P}_2$ .

- **KeyGen** $(1^\lambda, f_1, \dots, f_s)$ : Pick  $(\text{pp}, k) \leftarrow \Sigma'.\text{Kg}$ . Pick a vector  $\alpha = (\alpha_{l,j} \leftarrow \mathbb{Z}_p : l \in [s], j \in [q])$ . Let  $h_{l,j} = e(g, g)^{\alpha_{l,j}}$  for every  $l \in [s], j \in [q]$ . This algorithm computes  $t_i = g^{\sum_{l=1}^s \sum_{j=1}^q \alpha_{l,j} f_{(l,j),i} \cdot \mathbf{F}_k(i)}$  for every  $i \in \mathbb{I}$ , then outputs  $\text{PK} = (\mathbf{f}_1, \dots, \mathbf{f}_s, \{t_i\}_{i \in \mathbb{I}})$ ,  $\text{SK} = k$  and  $\text{VK} = (\{h_{l,j}\}_{l \in [s], j \in [q]})$ .
- **ProbGen** $(\mathbf{x}, \text{SK})$ : On input  $\mathbf{x}$  and  $\text{SK}$ , the client outputs  $\sigma = \mathbf{x}$  and  $\tau = e(\prod_{i \in \mathbb{I}} \mathbf{F}_k(i)^{\mathbf{x}^i}, g)$ .
- **Comp** $(\text{PK}, \sigma)$ : The server computes  $\rho_{l,j} = \sum_{i \in \mathbb{I}} f_{(l,j),i} \cdot \mathbf{x}^i$  for every  $l \in [s]$  and  $q \in [j]$ , and a proof  $\pi = \prod_{i \in \mathbb{I}} (t_i)^{\mathbf{x}^i}$ . Then the server outputs  $\rho = (\{\rho_{l,j}\}_{l \in [s], j \in [q]})$  and  $\pi$ .
- **Verify** $(\text{VK}, \rho, \pi, \tau)$ : A verifier outputs  $f_l(\mathbf{x}) = \sum_{j=1}^q x_1^{(j-1)n} (\sum_{i \in \mathbb{I}} f_{(l,j),i} \cdot \mathbf{x}^i)$  for every  $l \in [s]$  if  $e(\pi, g) = \prod_{l=1}^s \prod_{j=1}^q h_{l,j}^{\rho_{l,j}} \cdot \tau$  holds. Otherwise outputs  $\perp$ .

*Remark 1.* (1) Obviously, the extended scheme is correct, secure and efficient deriving from our scheme  $\mathcal{BVC}\mathcal{P}_2$ . (2) The storage overhead of the extended scheme is  $1 + 1/(sq)$ , and it will approach 1 when  $q$  is large enough. (3) The output of  $\text{Verify}(\text{VK}, \rho, \pi, \tau)$  is  $f_l(\mathbf{x}) = \sum_{j=1}^q x_1^{(j-1)n} (\sum_{i \in \mathbb{I}} f_{(l,j),i} \cdot \mathbf{x}^i)$  for every  $l \in [s]$  or  $\perp$ , which is slightly different from our model. (4)  $\Sigma'$  is defined below:

- **Kg** $(1^\lambda, m, a, b)$ : Let  $\Lambda = (p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda, 2)$ ,  $k_0, l_0 \leftarrow \mathbb{Z}_p$ , a  $2 \times 2$  matrix  $K_{1,v} \leftarrow \mathbb{Z}_p^{2 \times 2}$  for every  $v \in [b]$ ,  $K_{u,v} \leftarrow \mathbb{Z}_p^{2 \times 2}$  for every  $(u, v) \in \{2, \dots, m\} \times [a]$ . This algorithm outputs  $k = \{k_0, l_0\} \cup \{K_{1,v} : v \in [b]\} \cup \{K_{u,v} : (u, v) \in \{2, \dots, m\} \times [a]\}$  and  $\text{pp} = (\lambda, m, a)$ . Generally, the domain of  $\mathbf{F}$  is  $\mathbb{I} = \{0, \dots, n-1\} \times \{0, \dots, d\}^{m-1}$  and the range is  $\mathbb{G}$ .
- **F**: On input  $\mathbf{i} = (i_1, \dots, i_m)$ , this algorithm computes the bit representations  $i_1 = (i_{1,1}, \dots, i_{1,b})$  and  $i_u = (i_{u,1}, \dots, i_{u,a})$  for each  $u \in \{2, \dots, m\}$  and outputs  $\mathbf{F}_k(\mathbf{i}) = g^{\xi_i}$ , where  $(\xi_i, \eta_i) = (k_0, l_0) \prod_{v=1}^b K_{1,v}^{i_{1,v}} \prod_{u=2}^m \prod_{v=1}^a K_{u,v}^{i_{u,v}}$ .

## 6 Conclusion

In this paper, we generalize the batch verifiable computation notion into batch verifiable computation with public verifiability, where any third party is able to verify the computation result. By introducing three PRFs with closed-form efficiency, we construct three schemes for outsourcing different functions. Besides, we prove the security of our schemes under the DLIN assumption. Finally, we compare our schemes with the existed schemes and extend them to enjoy less storage required by encoding the functions.


**Acknowledgements.** Yu Yu (corresponding author) was supported by the National Natural Science Foundation of China (61572192, 61472249, 61572149, 61571191). Xiangxue Li (corresponding author) was supported by Science and Technology Commission of Shanghai Municipality (Grant No. 13JC1403502, 13JC1403500).

## References

1. Zhang, L.F., Safavi-Naini, R.: Batch verifiable computation of outsourced functions. *J. Des. Codes Crypt.* **77**, 563–585 (2015)
2. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) *CRYPTO 2011*. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
3. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012*, pp. 501–512. ACM (2012)
4. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 112–120. ACM (2009)
5. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: efficient verification via secure computation. In: Abramsky, S., Gavaille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) *ICALP 2010*. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)
7. Chung, K.-M., Kalai, Y.T., Liu, F.-H., Raz, R.: Memory delegation. In: Rogaway, P. (ed.) *CRYPTO 2011*. LNCS, vol. 6841, pp. 151–168. Springer, Heidelberg (2011)
8. 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)
9. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) *TCC 2012*. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
10. Parno, B., Howell, J., Gentry, C., et al.: Pinocchio: nearly practical verifiable computation. In: *2013 IEEE Symposium on Security and Privacy (SP)*, pp. 238–252. IEEE Press, Berkeley (2013)
11. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) *TCC 2013*. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (2013)
12. 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)
13. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS 2012*, pp. 309–325. ACM, New York (2012)
14. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) *CT-RSA 2002*. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)

15. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, pp. 863–874. ACM, New York (2013)
16. Chen, X., Li, J., Susilo, W.: Efficient fair conditional payments for outsourcing computations. *IEEE Trans. Inf. Forensics Secur.* **7**, 1687–1694 (2014)
17. Chen, X., Li, J., Ma, J., et al.: New algorithms for secure outsourcing of modular exponentiations. *IEEE Trans. Parallel Distrib. Syst.* **25**, 2386–2396 (2014)
18. Catalano, D., Fiore, D.: Practical homomorphic MACs for arithmetic circuits. In: Johansson, T., Nguyen, P.Q. (eds.) *EUROCRYPT 2013*. LNCS, vol. 7881, pp. 336–352. Springer, Heidelberg (2013)
19. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, pp. 113–122. ACM, New York (2008)
20. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 48–59. ACM, New York (2010)
21. Atallah, M.J., et al.: Secure outsourcing of scientific computations. *Adv. Comput.* **54**, 215–272 (2002)

# Accelerating Oblivious Transfer with Batch Multi-exponentiation

Yang Sun<sup>1,5</sup>, Qianhong Wu<sup>1,6</sup>, Jingwen Liu<sup>2</sup>, Jianwei Liu<sup>1</sup>, Xinyi Huang<sup>3</sup>,  
Bo Qin<sup>4,6</sup>, and Wei Hu<sup>2</sup>

<sup>1</sup> School of Electronic and Information Engineering, Beihang University,  
Beijing, China

<sup>2</sup> Potevio Information Technology Co., Ltd, Beijing, China

<sup>3</sup> Fujian Provincial Key Laboratory of Network Security and Cryptology,  
School of Mathematics and Computer Science, Fujian Normal University,  
Fuzhou, China

<sup>4</sup> Key Laboratory of Data Engineering and Knowledge Engineering  
(Renmin University of China) Ministry of Education,  
School of Information, Renmin University of China, Beijing, China  
`bo.qin@ruc.edu.cn`

<sup>5</sup> State Key Laboratory of Integrated Services Networks,  
Xidian University, Xi'an, China

<sup>6</sup> State Key Laboratory of Information Security,  
Institute of Information Engineering, Chinese Academy of Sciences,  
Beijing 100093, China

**Abstract.** More and more people use smart end devices to retrieve digital items and purchase on the Internet. Oblivious transfer (OT) is a fundamental tool to protect user privacy in such applications. Most existing works devote to improving the communication performance of OT protocols; few work has been done to improve the computation efficiency. Modular exponentiation is the most frequent operation in OT protocols. It is known that the computation cost of any OT protocol must be linear with the database size; speeding up the exponentiations is critical for OT protocols to be deployed in practice. To this end, we investigate batch multi-exponentiation algorithms and propose two new algorithms. Then we apply our batch multi-exponentiation algorithms to acceleration of OT protocols. Our approach is especially useful for the  $k$ -out- $n$  OT. We also exploit the algorithm to speed up simultaneous execution of 1-out- $n$  OT protocols which we called batch OT. We conduct a series of experiments and the experimental results show that our approach is effective and can significantly accelerate OT protocols.

**Keywords:** Batch oblivious transfer · Batch multi-exponentiation · Modular exponentiation

## 1 Introduction

With more and more smart mobile devices accessing to the Internet, people have an increasing reliance on smart mobile devices to retrieve digital items, conduct

online purchase and so on. Malicious websites and hackers may exploit the information about the user activities to launch various attacks, especially social engineering attack, including phishing, impersonation and cheating. Oblivious transfer (OT) is a fundamental cryptographic primitive [36] that can be used to protect user privacy in such applications. An OT protocol involves two parties, a sender  $\mathcal{S}$  and a receiver  $\mathcal{R}$ .  $\mathcal{R}$  requests some digital item possessed by  $\mathcal{S}$ , but doesn't want  $\mathcal{S}$  to know which item he chooses. In the original OT,  $\mathcal{S}$  sends a message to  $\mathcal{R}$ , and  $\mathcal{R}$  gets the message with probability 0.5 while  $\mathcal{S}$  doesn't know whether  $\mathcal{R}$  gets the message or not. It is shown that, the original OT protocol is equivalent of the following 1-out-2 OT, denoted by  $OT_2^1$ , which means that  $\mathcal{S}$  sends two messages to  $\mathcal{R}$ ,  $\mathcal{R}$  only obtains the one he requested before, while  $\mathcal{S}$  doesn't know  $\mathcal{R}$ 's choice. The concept of  $OT_n^1$  [32] extends  $OT_2^1$  by letting  $\mathcal{R}$  choose one from  $n$  messages, where  $n$  is usually used to model the database size if  $\mathcal{S}$  is a database server. The general extension is  $OT_n^k$  which means that  $\mathcal{R}$  requests  $k$  items from  $n$  ones possessed by  $\mathcal{S}$ .

Since the sender does not know the receiver's choice, the OT protocols can be used to protect the user privacy in e-commerce. In this scenario, a customer can purchase on the Internet without leaking the purchased item. Similar applications include protection of privacy in signing online contracts. Indeed, a concrete scheme has been proposed by employing  $OT_n^k$  to address protection in contracts from the same party [1]. For contracts request with different parties, a straightforward approach is to invoke  $OT_n^k$  or the  $OT_n^1$  for each party, yet it is inefficient.

It's essential to speed up the computation of the execution of the OT protocols as it is known that the computation cost of  $OT_n^1$  is linear with  $n$ . The basic operation of oblivious transfer is modular exponentiation. It has been shown that the computation cost must be linear with  $n$  which usually represents the database size and the total number of digital items/contracts in such applications. The existing modular exponentiation algorithms [7, 20, 21, 27, 28, 37, 41] mainly focus on the modular exponentiation and multi-exponentiation. Wu et al. [42] proposed several batch multi-exponentiation algorithms. These basic algorithms may be further improved for more efficient OT protocols.

## 1.1 Our Work

Batch cryptosystem [18, 22, 29] is a way to accelerate simultaneous execution of a number of cryptographic procedures. We focus on the batch algorithms to speed up oblivious transfer protocols. We propose new batch multi-exponentiation algorithms and improve the efficiency of oblivious transfer.

First, we study the way to accelerate the basic operation modular exponentiation. We propose two new batch multi-exponentiation algorithms common bit batch multi-exponentiation (CBBM) and simultaneous batch multi-exponentiation algorithm (TBI-SBM). Comparing with the basic batch multi-exponentiation (BBM) [42] algorithm and simultaneous batch multi-exponentiation (SBM) [42] algorithm, our algorithms are more efficient and save more than 25% time when the base number is small.

Then we apply the batch multi-exponentiation algorithm to  $k$ -out- $n$  oblivious transfer. We propose the batch oblivious transfer to accelerate the process of computation. Finally, we consider the situation that a user request different messages from different senders, a new batch oblivious transfer is introduced. Comparing with using the previous batch multi-exponentiation [42] to get the result, our algorithm is more efficient, save about 25% computation time.

## 2 Related Work

Rabin [36] proposed the concept of oblivious transfer, as a generalization,  $OT_2^1$  was suggested by Even et al. [16].  $OT_n^1$  oblivious transfer was introduced by Brassard et al. [6] under the name all or nothing disclosure of secrets. OT protocol is useful which can be used to construct protocols for secret key agreement [43], contract signing [16] and general secure multi-party computation [12, 19, 23–25]. Bellare and Micali [4] proposed practical OT in the honest-but-curious model, and the protocol of Naor and Pinkas [33] is secure in the half-simulation model. The first adaptive  $k$ -out- $n$  OT is proposed by Naor and Pinkas [32] and is secure in the half-simulation, similarly to [1]. Some other works concentrate on OT protocols in the random oracle model [10, 34, 40].

Communication and computation complexity is an important issue in OT protocols. Poor efficiency makes OT protocols difficult to be deployed in the real world. Modular exponentiation is the most common operation of many different cryptosystems, but it's also expensive operation. Many efforts focus have been made to this problem. A good survey of early efforts to speeding up modular exponentiation can be found in [20]. The most well-known algorithm may be the binary method [26], also called “square and multiply” method. There are also many other approaches [7, 21, 27, 28, 30, 37]. For multi-exponentiation, Shamir's trick described in [15] merges the process of square and multiplication. It is useful in the situation that the storage space is sufficient. Most multi-exponentiation [5, 13, 38, 44] algorithms focus on the representation of exponentiation to reduce the hamming weight. Batch exponentiation is an acceleration for multiple exponentiations with a fixed base. M'Raihi and Naccache [31] proposed the first batch exponentiation algorithm. Then Chung et al. [11] improved this basic batch exponentiation by a decremental combination strategy. Pipenger presented an almost optimal approach for fast computing batch multi-base exponentiations [35] by using the vector addition chain of the exponents. Unfortunately, it has been shown that to find such a shortest chain is an NP-complete problem [14] and it cannot be finished by modern computers in the general case. In 2015, Wu et al. [42] proposed a batch multi-exponentiation algorithm which can save 50% time than using the multi-exponentiation repeatedly. Subsequently, Sun et al. [39] improved this algorithm on elliptic curves. They changed the representation of exponent and exploit the feature that inverse on elliptic curves is fast.

Since Fiat's [18] work, different batch techniques have been proposed. Batch signature verification has been widely used to promote the verification of multiple

signatures [3, 8, 9, 17, 45]. Batch signing approach has been used in generation of blind signature [39]. Batch encryption is useful to generate a lot of ciphertexts simultaneously [42]. With the development of e-business, the requirement of calculating speed is more and more high. Oblivious transfer is an important tool for e-business. The well-known oblivious transfer [1, 6, 16, 36] are useful to solve the problem of e-business. But these algorithms are not efficient to solve a lot of oblivious transfer scheme. So we propose batch oblivious transfer to solve this problem. In this paper, we propose new batch multi-exponentiation algorithms to compute many multi-exponentiations simultaneously, which saves nearly 25 % computation overhead compared to previous batch algorithm. We also combine our new batch multi-exponentiation algorithm with oblivious transfer. Then we propose the first batch oblivious transfer which is more efficient.

### 3 Batch Multi-exponentiation

Here we describe our new batch multi-exponentiation algorithms called CBBM and TBI-SBM. Compared with the previous work [42], our algorithms are more efficient. In order to describe clearly, let  $r_i$  be a  $l$  bits string representing an exponent. Let  $r_{i,j}$  denote the  $j$ -th bit of  $r_i$ . Let  $i \in [a, b]$  represent an integer  $i$  ranging from  $a$  to  $b$ . We use  $\wedge$  to represent the bitwise AND operation and  $\oplus$  to represent the bitwise XOR operation. The smallest integer that is greater than or equal to  $\frac{n}{d}$  is denoted by  $\lceil \frac{n}{d} \rceil$ .

**Basic Idea.** We focus on the acceleration of multiple multi-exponentiations represented by  $c_i = g_1^{r_{i,1}} g_2^{r_{i,2}} \dots g_m^{r_{i,m}}$  for  $i \in [1, n]$ , where  $r_{i,1}, r_{i,2}, \dots, r_{i,m}$  are  $l$  bits long randomly picked exponents and  $g_1, g_2, \dots, g_m$  are fixed bases.

We denote every exponent  $r_{i,k}$  as

$$r_{i,k} = \sum_{j=0}^{l-1} r_{i,k,j} 2^j$$

where  $i \in [1, n], k \in [1, m]$  with  $r_{i,k,j} \in \{0, 1\}$ . Each exponentiation can be defined as

$$g_k^{r_{i,k}} = g_k^{\sum_{j=0}^{l-1} r_{i,k,j} 2^j} = \prod_{j=0}^{l-1} g_k^{r_{i,k,j} 2^j}$$

where  $i \in [1, n], k \in [1, m]$ .

For the tuples of multi-exponentiations

$$\begin{aligned} c_1 &= g_1^{r_{1,1}} g_2^{r_{1,2}} \dots g_m^{r_{1,m}} \\ c_2 &= g_1^{r_{2,1}} g_2^{r_{2,2}} \dots g_m^{r_{2,m}} \\ &\vdots \\ c_n &= g_1^{r_{n,1}} g_2^{r_{n,2}} \dots g_m^{r_{n,m}} \end{aligned} \tag{1}$$

Calculating  $c_i$  for each  $i \in [1, n]$  separately is a direct way to get the multiple results. Wu et al. [42] proposed the first batch multi-exponentiation algorithms

to accelerate the calculation of the above formula. Although their algorithm saves half of the calculation time, they only merge the process of square. We are interested in more efficient algorithms to compute multi-exponentiations in a batch to improve the efficiency.

We propose two new batch multi-exponentiation algorithms to get a more efficient way to get the result. We provide illustrative examples for these algorithms to demonstrate the processes. In our examples, for simplicity, we slightly abuse the exponentiation representation by denoting  $\prod_{j=0}^{l-1} g_k^{r_{i,k,j} 2^j} = g^{r_{i,k,0} \dots r_{i,k,l-1}}$  if it incurs no confusion. Our new batch multi-exponentiation algorithms are more efficient than the previous algorithms by merging the process of multiplications and squares.

### 3.1 Common Bit Batch Multi-exponentiation Algorithm

The BBM algorithm [42] reduces the number of squares. However, the number of multiplication operations is identical to that of applying simple multi-exponentiation algorithm. In this section, we concentrate on reducing the number of multiplications in our algorithm and propose a CBBM algorithm.

Note that any two exponents of the same column exponentiations have the common part. Instead of computing the common part separately, we can extract such part and calculate it once. This is the basic idea of our CBBM algorithm. The CBBM algorithm consists of two stages: the preparation stage and the calculation stage. In the preparation stage, the exponentiation of the same column is divided into a number of groups, each of which has 2 exponentiations. Then, the algorithm extracts the common part and the unique part of the two exponentiations. In the calculation stage, the algorithm calculates the common part of the two exponents, and the unique part for each of the exponents. Then, it multiplies the common part and each of the unique parts to get each result. Therefore, the common part is only computed once and the number of the multiplications is reduced.

We denote by  $r_{COMi}$  the common bits of exponents  $r_{2i}$  and  $r_{2i-1}$ , and by  $r_{NEWi}$  the unique exponent of  $r_i$ . The CBBM algorithm can be described as follows.

**Preparation Stage:** Assume that the batch number  $n$  is even. The algorithm first divides the exponentiations of the same column into  $n/2$  groups

$$\{g_k^{r_{1,k}}, g_k^{r_{2,k}}\}, \{g_k^{r_{3,k}}, g_k^{r_{4,k}}\} \dots \{g_k^{r_{n-1,k}}, g_k^{r_{n,k}}\}$$

where each group has 2 exponentiations. It then initializes  $\frac{n}{2}$  registers

$$r_{COM1,k}, r_{COM2,k}, \dots, r_{COM \frac{n}{2},k}$$

to store the common bits of the exponents

$$\{r_{1,k}, r_{2,k}\}, \{r_{3,k}, r_{4,k}\} \dots \{r_{n-1,k}, r_{n,k}\}$$



Since there are totally  $m$  columns,  $\frac{mn}{2}$  registers are required to store the common bits. The algorithm next computes the common bit exponent for each group as

$$r_{COMi,k} = r_{2i-1,k} \wedge r_{2i,k}$$

where  $i \in [1, \frac{n}{2}]$  and  $k \in [1, m]$ , and changes the original exponents to the unique exponents

$$r_{NEWi,k} = r_{i,k} \oplus r_{COM\lceil \frac{i}{2} \rceil, k}$$

where  $i \in [1, n]$  and  $k \in [1, m]$ .

**Calculation Stage:** After the preparation stage, the results can be represented as

$$g_k^{r_{i,k}} = g_k^{r_{NEWi,k}} \times g_k^{r_{COM\lceil \frac{i}{2} \rceil, k}}$$

where  $i \in [1, n], k \in [1, m]$ . The algorithm additionally initializes  $\frac{n}{2}$  registers

$$\{c_{COM1}, c_{COM2}, \dots, c_{COM\frac{n}{2}}\}$$

to store common results

$$\prod_{k=1}^m g_k^{r_{COM1,k}}, \prod_{k=1}^m g_k^{r_{COM2,k}} \dots \prod_{k=1}^m g_k^{r_{COM\frac{n}{2},k}}$$

and initializes registers  $c_1, \dots, c_n$  to store the unique results

$$\prod_{k=1}^m g_k^{r_{NEWi,k}}, i \in [1, n]$$

To get the final results, the algorithm simply multiplies  $c_i$  with the corresponding  $c_{COM\lceil \frac{i}{2} \rceil}$

$$c_i = c_i \times c_{COM\lceil \frac{i}{2} \rceil}$$

where  $i \in [1, n]$ . The detailed CBBM algorithm is shown in Algorithm 1.

**An Illustrative Example.** Our CBBM algorithm can be illustrated by the following example. Suppose that we compute the following multiple multi-exponentiations with binary exponents.

$$c_1 = g_1^{1010} g_2^{1011} c_2 = g_1^{1001} g_2^{1010}$$

In the preparation stage, the CBBM algorithm divides the exponentiations into the groups as

$$\{g_1^{1010}, g_1^{1001}\} \{g_2^{1011}, g_2^{1010}\}$$

Then it finds the common bit of exponents of each group as

$$r_{COM1,1} = r_{1,1} \wedge r_{2,1} = 1010 \wedge 1001 = 1000$$

$$r_{COM1,2} = r_{1,2} \wedge r_{2,2} = 1011 \wedge 1010 = 1010$$

**Algorithm 1.** CBBM algorithm**Input:**

the bases  $g_1, g_2, \dots, g_m$   
 the exponents  $r_{NEWi,k}, i \in [1, n], k \in [1, m]$ .  
 the common bits exponents  $r_{COMi,k}, 1 \leq i \leq \frac{n}{2}$

**Output:**

$c_1, c_2, \dots, c_n$

```

1: BEGIN
2:  $c_i = 1$  for every  $1 \leq i \leq n, c_{COMi} = 1$  for every  $1 \leq i \leq \frac{n}{2}$ 
3: for j=0 to l-1 do
4:   for k=1 to m do
5:     for i=1 to n do
6:       if  $r_{COM[\frac{j}{2}],k,i} = 1$  then
7:          $c_{COM[\frac{j}{2}]} = c_{COM[\frac{j}{2}]} \times g_k$ 
8:       end if
9:     end for
10:    if  $r_{NEWi,k,j} = 1$  then
11:       $c_i = c_i \times g_k$ .
12:    end if
13:  end for
14:   $g_k = g_k^2$ 
15: end for
16: for i=1 to n do
17:   $c_i = c_i \times c_{COM[\frac{i}{2}]}$ ;
18: end for
19: END
20: return  $c_1, c_2, \dots, c_n$ 

```

Next, the algorithm changes the original exponents to be the unique exponents

$$r_{NEW1,1} = r_{1,1} \oplus r_{COM1,1} = 1010 \oplus 1000 = 0010$$

$$r_{NEW2,1} = r_{2,1} \oplus r_{COM1,1} = 1001 \oplus 1000 = 0001$$

$$r_{NEW1,2} = r_{1,2} \oplus r_{COM1,2} = 1011 \oplus 1010 = 0001$$

$$r_{NEW2,2} = r_{2,2} \oplus r_{COM1,2} = 1010 \oplus 1010 = 0000$$

In the calculation stage, we have that

$$g_1^{r_{1,1}} = g_1^{r_{NEW1,1}} \times g_1^{r_{COM1,1}} = g_1^{0010} \times g_1^{1000}$$

$$g_1^{r_{2,1}} = g_1^{r_{NEW2,1}} \times g_1^{r_{COM1,1}} = g_1^{0001} \times g_1^{1000}$$

$$g_2^{r_{1,2}} = g_2^{r_{NEW1,2}} \times g_2^{r_{COM1,2}} = g_2^{0001} \times g_2^{1010}$$

$$g_2^{r_{2,2}} = g_2^{r_{NEW2,2}} \times g_2^{r_{COM1,2}} = g_2^{0000} \times g_2^{1010}$$

Therefore, the algorithm can correctly computes the results by running

$$c_1 = g_1^{1010} g_2^{1011} = g_1^{0010} g_2^{0001} \times g_1^{1000} g_2^{1010}$$

$$c_2 = g_1^{1001} g_2^{1010} = g_1^{0001} g_2^{0000} \times g_1^{1000} g_2^{1010}$$

We can see that  $g_1^{1000} g_2^{1010}$  is used twice to compute  $c_1$  and  $c_2$ , thus reducing the multiplication operations.

**Algorithm Complexity.** The CBBM algorithm reduces the number of multiplications by finding the common part of the exponents so that the common part is only calculated once. The number of squares is identical to that in the BBM algorithm. The following claim shows the complexity of the proposed CBBM algorithm.

*Claim.* Assume that  $n$  is even. The CBBM algorithm needs  $lm$  squares and  $\frac{3lmn}{8} + n$  multiplications on average to compute  $c_1, \dots, c_n$ , where  $m$  is the total number of bases,  $l$  is the maximal bit length for all the exponents  $r_{i,k}$  with  $i \in [1, n], k \in [1, m]$ .

*Proof.* The hamming weight of the exponent  $r_{i,k}$  on average is  $l/2$ . In the preparation stage, the common bit of 2 exponents are found. The hamming weight of common exponent is  $\frac{l}{4}$  on average, and the hamming weight of each unique exponent is  $\frac{l}{4}$  on average. Then it requires  $\frac{3l}{4}$  multiplications to calculate 2 results. For all exponents, the number of total multiplications is  $\frac{3lmn}{8}$ . To get the final results, we need another  $n$  multiplications to multiply the common results and the unique results, and  $l$  square operations are required for each column. Therefore, the total number of square operations we need is  $l \cdot m$ .

### 3.2 Two Bits Improved SMB

To further reduce the number of multiplications of SBM [42], we proposed a TBI-SBM algorithm, at the cost of more storage. The SBM algorithm handles only one bit of the same row's exponents simultaneously, while the TBI-SBM algorithm handles 2 bits of each exponent once ( $2m$  bits in total) by additionally requiring more pre-computations.

The algorithm also includes the preparation stage and the calculation stage.

**Preparation stage:** Assume that the bit length  $l$  is even. Pre-compute  $\prod_{i=1}^m g_i^{r_{i,j}}$  for all  $r_{i,j} \in \{0, 1, 2, 3\}$ . For certain two bits, there are  $2^{2m} - 1$  different cases (showed in Table 1) of the exponents in the same row. The results are organized as in Table 2.

**Calculation stage:** In the calculation stage, the algorithm first judges the certain bit of certain row's exponent equal to  $s_k$  for  $k \in [1, 2^{2m} - 1]$ . Then, it multiplies the corresponding pre-computed values  $f_k$  to the intermediate  $c_i$ . The detailed algorithm is described in Algorithm 2.

**An Illustrative Example.** To demonstrate the processes of our algorithm, we give an example with binary exponents as follows:

$$c_1 = g_1^{111010} g_2^{101011} = (g_1^2 g_2^3)^1 \cdot (g_1^2 g_2^2)^4 \cdot (g_1^3 g_2^2)^{16}$$

**Table 1.** Two bits exponent representation

	$r_{i,m,2j+1}r_{i,m,2j}, r_{i,m-1,2j+1}r_{i,2,2j}, \dots, r_{i,1,2j+1}r_{i,1,2j} \in [0, \frac{l}{2} - 1]$
$s_1$	00, 00, $\dots$ 01
$s_2$	00, 00, $\dots$ , 10
$\dots$	$\dots$
$s_{2^{2m-1}}$	11, 11 $\dots$ 11

**Table 2.** Pre-computation

$f_1$	$g_1$
$f_2$	$g_2$
$\dots$	$\dots$
$f_{2^{2m-1}}$	$g_m^3 \times g_{m-1}^3 \times \dots \times g_1^3$

$$c_2 = g_1^{100101} g_2^{101011} = (g_1^1 g_2^3)^1 \cdot (g_1^1 g_2^2)^4 \cdot (g_1^2 g_2^2)^{16}$$

In the same row, we consider the identical two bits of the exponents simultaneously, which means that two bits of the whole row need only one multiplication. To achieve this, not only the bases  $g_1, g_2$  need to be squared  $l$  times, but also the terms

$$g_1^2, g_1^3, g_2^2, g_1 g_2, g_1^2 g_2, g_1^3 g_2, g_2^2, g_1 g_2^2, g_1^2 g_2^2, g_1^3 g_2^2, g_2^3, g_1 g_2^3, g_1^2 g_2^3, g_1^3 g_2^3$$

(which is computed in stage 1) should be computed  $l$  times.

**Algorithm Complexity.** The TBI-SBM algorithm is an improvement of the SBM algorithm that is also suitable for the situations with a small number of bases. This algorithm needs more storage occupations than the SBM algorithm, but further reduces the number of multiplications. The following claim shows the number of squares and multiplications involved in the TBI-SBM algorithm.

*Claim.* The TBI-SBM algorithm requires  $(2^{2m} - 1)\frac{l}{2}$  squares and  $\frac{ln}{2} - \frac{ln}{2 \times 2^{2m}} + 2^{2m} - 1 - m$  multiplications on average to compute  $c_1, \dots, c_n$ , where  $m$  is the total number of bases,  $l$  is the maximal bit length of all the exponents  $r_{i,k}$  for  $i \in [1, n], k \in [1, m]$ .

*Proof.* In the preparation stage, the algorithm needs to prepare  $2^{2m} - 1$  values listed in Table 4. Computing each table entry requires one multiplication except  $f_1, f_2, \dots, f_m$ . Therefore, we need  $2^{2m} - 1 - m$  multiplications to get the table. The calculation stage requires both squares and multiplications. There are  $(2^{2m} - 1)$  values in the auxiliary table and each value needs be squared  $l$  times. Thus, the total number of squares is  $(2^{2m} - 1)l$ . The bit length of the exponents is  $l$ , and hence there is a probability  $\frac{1}{4^m}$  that two bits of all the exponents are both 0. So the total number of multiplications on average in the calculation stage is  $\frac{ln}{2} - \frac{ln}{2 \times 4^m}$  for  $n$  results.

---

**Algorithm 2.** TBI-SBM Algorithm

---

**Input:**

the bases  $g_1, g_2, \dots, g_k$   
the exponents  $r_{i,k}i \in [1, n], k \in [1, m]$

**Output:**

$c_1, c_2, \dots, c_n$

```

1: BEGIN
2:  $f = g_1 \times g_2$ 
3: for j=0 to  $\lceil \frac{l}{2} \rceil - 1$  do
4:   for i=1 to n do
5:     for k=1 to  $2^{2^m} - 1$  do
6:       if  $r_{i,1,2^j}r_{i,1,2^{j+1}}, r_{i,2,2^j}r_{i,1,2^{j+1}}, \dots, r_{i,m,2^j}r_{i,m,2^{j+1}} = s_k$  then
7:          $c_i = c_i \times f_k$ .
8:       end if
9:     end for
10:   end for
11:   for k=1 to  $2^{2^m} - 1$  do
12:      $f_k = (f_k^2)^2$ ;
13:   end for
14: end for
15: END
16: return  $c_1, c_2, \dots, c_n$ 

```

---

**3.3 Comparison between Algorithms**

In this section, we compare the complexity of our algorithms with previous work [42] evaluated by the number of squares and the number of multiplications. The results are shown in Table 3. The number of squares is determined by the length of exponents and the number of multiplications is determined by the hamming weight of the exponent.

**Table 3.** Simple multi-exponentiation,BBM,CBBM,SBM,TBI-SBM

	Multiplication	Square	Storage
Simple [2]	$\frac{lmn}{2}$	$ln$	$n$
BBM [42]	$\frac{lmn}{2}$	$lm$	$m + n$
CBBM	$\frac{3lmn}{8} + n$	$lm$	$m + \frac{(3+m)n}{2}$
SBM [42]	$ln - \frac{ln}{2^m} + 2^m - 1 - m$	$(2^m - 1)l$	$2^m - 1 + n$
TBI-SBM	$\frac{ln}{2} - \frac{ln}{2 \times 4^m} + 2^{2^m} - 1 - m$	$(2^{2^m} - 1)l$	$2^{2^m} - 1 + n$

**Comparison between BBM and CBBM algorithms:** Assume that  $n$  is even. The BBM algorithm needs  $lm$  squares and  $\frac{lmn}{2}$  multiplications. In contrast, the CBBM algorithm needs  $lm$  squares and  $\frac{3lmn}{8} + n$  multiplications. The number

of multiplications in the CBBM algorithm is about  $\frac{1}{4}$  less than the number of multiplications of BBM. The cost is more registers to store common exponent and common results.

**Comparison between SBM and TBI-SBM algorithms:** Assume that a square can be seen as a multiplication. The number of multiplications of TBI-SBM is

$$\frac{ln}{2} - \frac{ln}{2 \times 4^m} + 2^{2m} - 1 - m + (2^{2m} - 1)l$$

and the number of multiplications of SBM is

$$ln - \frac{ln}{2^m} + 2^m - 1 - m + (2^m - 1)l$$

Then if we want the TBI-SBM algorithm consumes less time than the SBM algorithm, we have to satisfy that

$$\frac{ln}{2} - \frac{ln}{2 \times 4^m} + 2^{2m} - 1 - m + (2^{2m} - 1)l < ln - \frac{ln}{2^m} + 2^m - 1 - m + (2^m - 1)l$$

meaning that

$$(2^{2m} - 2^m)(l + 1) + \left(\frac{1}{2^m} - \frac{1}{2^{2m+1}} - \frac{1}{2}\right)ln < 0$$

## 4 Batch Oblivious Transfer

In this section, we describe our batch oblivious transfer protocol. Batch oblivious transfer may be for the same sender or different senders. It processes the requests in a batch, rather than computing the answers one by one, in order to save the computation time. Batch oblivious transfer has three phases as the normal one. The most time-consuming operation in oblivious transfer is multi-exponentiation. Hence, our batch multi-exponentiation algorithm can be used to accelerate this process.

### 4.1 Wen-Guey Tzeng Oblivious Transfer

Tzeng oblivious transfer [40] have two parties, a sender( $\mathcal{S}$ ) and a receiver( $\mathcal{R}$ ).  $\mathcal{R}$  requests messages from  $\mathcal{S}$  and then  $\mathcal{S}$  responses to  $\mathcal{R}$  the encrypted messages.  $\mathcal{R}$  can obtain the messages which he requests before, but he doesn't know other messages. Now we review the protocol briefly. Choose two generators  $g$  and  $h$  from prime order- $q$  group  $G_q$ . It's also useful for the situation that  $G_q$  is the set of quadratic residues of  $Z_p^*$ , where prime  $p = 2q + 1$ , then any element in  $G_q \setminus 1$  is a generator of  $G_q$ . Assume  $\mathcal{S}$  has messages  $m_1, m_2 \dots, m_n$  which belong to  $G_q$ . The 1-out- $n$  OT protocol works as follows:

1.  $\mathcal{R}$  chooses the number of the message he want to get  $\alpha$ ,  $1 \leq \alpha \leq n$ , and the random number  $r, r \in Z_q$ , computes  $y = g^r h^\alpha$ , and then sends  $y$  to  $\mathcal{S}$ .

2. For each  $1 \leq i \leq n$ ;  $\mathcal{S}$  chooses random number  $k_i \in Z_q$ , computes  $c_i = (g^{k_i}, m_i(y/h^i)^{k_i})$ , then sends  $c_i, 1 \leq i \leq n$  to  $\mathcal{R}$ .
3.  $\mathcal{R}$  chooses  $c_\alpha = (a, b)$ , and computes  $m_\alpha = b/a^r$ .  
By this protocol,  $\mathcal{R}$  gets the message  $m_\alpha$ , Since  $c_\alpha = (a, b) = (g^{k_\alpha}, m_\alpha(y/h^\alpha)^{k_\alpha})$ ,  $b/a^r = m_\alpha(y/h^\alpha)^{k_\alpha}/(g^{k_\alpha})^r = m_\alpha$ .

$\mathcal{R}$  chooses  $k$  different messages from same  $\mathcal{S}$  called  $k$ -out- $n$  oblivious transfer. The detail is showed as follows:

1.  $\mathcal{R}$  chooses different number  $\alpha_1, \alpha_2 \cdots \alpha_k$ , where  $1 \leq \alpha_i \leq n, 1 \leq i \leq k$ ; which  $\alpha_i$  represents the number of the message that  $\mathcal{R}$  wants to get, then  $\mathcal{R}$  chooses the random number  $r_l$  of  $Z_q, 1 \leq l \leq k$ , computes  $y_l = g^{r_l} h^{\alpha_l}$  and sends  $y_l$  to  $\mathcal{S}$ .
2.  $\mathcal{S}$  chooses random number  $k_{i,l} \in Z_q, 1 \leq l \leq k, 1 \leq i \leq n$ ; computes  $c_{i,l} = (g^{k_{i,l}}, m_i(y_l/h^i)^{k_{i,l}})$ , and sends to  $\mathcal{R}$ .
3.  $\mathcal{R}$  chooses  $c_{\alpha_l,l} = (a, b)$ , and computes  $m_{\alpha_l} = b/a^{r_l}, 1 \leq l \leq k$ .

The  $OT_n^k$  scheme has the same correctness as the  $OT_n^1$  scheme.

### 4.2 Batch Oblivious Transfer

We focuses on the problem that  $\mathcal{R}$  requests many messages from the same sender  $\mathcal{S}$  or different senders  $\mathcal{S}$ . The protocol in which  $\mathcal{R}$  requests many messages from same  $\mathcal{S}$  is  $k$ -out- $n$  oblivious transfer. For  $\mathcal{R}$ , the most time-consuming process is to compute  $y_l, 1 \leq l \leq k$ . Our batch multi-exponentiation algorithm is used to accelerate this process. To generate  $y_l, 1 \leq l \leq k$ ,  $\mathcal{R}$  needs to compute  $y_l = g^{r_l} h^{\alpha_l}, 1 \leq l \leq k$ . This process can be accelerated as follows:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix} = \begin{pmatrix} g^{r_1} h^{\alpha_1} \\ g^{r_2} h^{\alpha_2} \\ \vdots \\ g^{r_k} h^{\alpha_k} \end{pmatrix} \leftarrow CBBM\left(\{g, h\}; \{r_l, \alpha_l : 1 \leq l \leq k\}\right),$$

In the situation in which one wants to buy different goods from different sellers,  $\mathcal{R}$  requests messages from different  $\mathcal{S}$ , and every  $\mathcal{S}$  receives one or more requests. The cases where  $\mathcal{R}$  requests one or more messages from each  $S_i, 1 \leq i \leq k$  can be processed similarly. We take one message as an example. Assume  $\mathcal{R}$  requests one message to each  $S_i, 1 \leq i \leq k$ . He needs to compute  $y_i = g^{r_i} h^{\alpha_i}, 1 \leq i \leq k$ . The result can also be obtained with batch multi-exponentiation. Although different senders  $\mathcal{S}$  have different messages, the bases  $g$  and  $h$  keep unchanged, Hence, the process can also be accelerated by batch multi-exponentiation algorithm.

### 4.3 Performance

We provide a theoretical analysis of the batch  $k$ -out- $n$  oblivious transfer with CBBM and compare it with previous batch multi-exponentiation algorithms

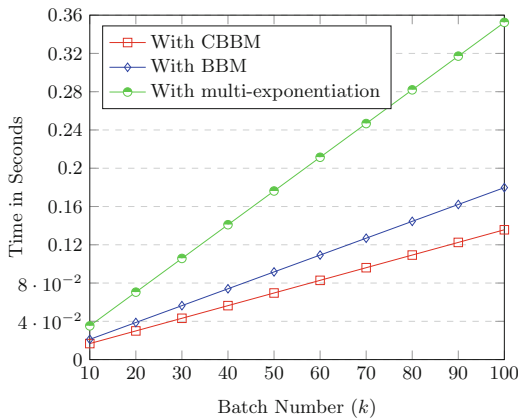
and oblivious transfer with simple multi-exponentiation [2]. The batch OT with different senders is similar to the batch  $k$ -out- $n$  OT. Hence, we only analyse batch  $k$ -out- $n$ OT. Batch multi-exponentiation algorithm is composed by square and multiplication. The number of multiplications and square is important index of comparison. The details are summarized in Table 4.

**Table 4.** Comparison of batch oblivious transfer scheme with CBBM, BBM and simple multi-exponentiation

Algorithm	Efficiency
With CBBM	$\frac{3lk}{4} + k$ multiplications + $2l$ squares
With BBM	$lk$ multiplications + $2l$ squares
With simple multi-exponentiation	$lk$ multiplications + $lk$ squares

The comparison shows that the number of square is the same as those of batch algorithms CBBM and BBM. The difference is the number of multiplications related to the message number  $k$  and exponent length  $l$ . The CBBM is more efficient than the BBM for all  $k$ . If the  $k$  is large enough, the CBBM saves nearly 25 % time than the BBM. Comparing the batch algorithms with simple multi-exponentiation, we can see that the batch algorithms reduce the number of squares. The number of squares doesn't increase linearly with the  $k$ .

We conducted a series of experiments to evaluate the practicality of batch oblivious transfer scheme with CBBM and compare it with the case where BBM and simple multi-exponentiation is used. We run the experiment on a PC with a 2.80 GHz CPU and 3 GB RAM, by using C programming language in the GMP and PBC library. We set  $|p| = 1024$  and  $l = |q| = 140$ , that is, all the random



**Fig. 1.** The performance of oblivious transfer with CBBM/BBM/simple multi-exponentiation (Color figure online)



exponents are 140-bits long, while  $g$  and  $h$  are 1024-bits long. The experiment are executed for 100 times to remove the experimental noise. In the experiments, the number  $k$  is set as 10, 20,  $\dots$ , 100 to see the performance improvements due to CBBM, BBM and simple multi-exponentiation (Fig. 1).

As we discussed, the receiver takes  $k$  resource-intensive multi-exponentiations. It can be seen that the performances for generating  $k$  request are linear with the number  $k$  of messages choice. Furthermore, computing  $k$  multi-exponentiations with  $m = 2$ , which are the main workloads of the receiver when generating  $k$  requests, by CBBM algorithm only requires roughly 75 % computation time of that by BBM. Comparing with the simple multi-exponentiation, more than 50 % computation time can be saved. Hence, CBBM algorithm is more efficient than the BBM algorithm and simple multi-exponentiation, and the batch oblivious transfer is a useful scheme.

## 5 Conclusion

This paper proposed two new batch multi-exponentiation algorithms to accelerate computations. Comparing with the previous algorithms, our new algorithms is more efficient when the base number is small. These algorithm was further used to speed up batch oblivious transfer scheme. In this way, multiple requests can be computed simultaneously. Both theoretical and experimental analyses show that our batch multi-exponentiation algorithm greatly improves the performance of the oblivious transfer scheme. This application also implies that our algorithms are useful to accelerate some other cryptographic schemes or protocols if many multi-exponentiations are involved.

**Acknowledgment.** This paper is partially supported by the National Key Basic Research Program (973 program) through project 2012CB315905, by the National High Technology Research and Development Program of China (863 Program) through project 2015AA017205, by the Natural Science Foundation of China through projects 61370190, 61173154, 61272501, 61402029, 61472429, 61202465 and 61532021, by the Beijing Natural Science Foundation through project 4132056, by the Guangxi natural science foundation through project 2013GXNSFB053005, the Innovation Fund of China Aerospace Science and Technology Corporation, Satellite Application Research Institute through project 2014-CXJJ-TX-10, the Open Project of Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University.

## References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Avanzi, R.M.: On multi-exponentiation in cryptography. Cryptology ePrint Archive, Report 2002/154 (2002)

3. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
4. Bellare, M., Micali, S.: Non-interactive oblivious transfer and applications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 547–557. Springer, Heidelberg (1990)
5. Bos, J.N.E., Coster, M.J.: Addition chain heuristics. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 400–407. Springer, Heidelberg (1990)
6. Brassard, G., Crépeau, C., Robert, J.M.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
7. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast exponentiation with precomputation. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 200–207. Springer, Heidelberg (1993)
8. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. *J. Cryptol.* **25**(4), 723–747 (2012)
9. Cheon, J.H., Kim, Y., Yoon, H.: A new ID-based signature with batch verification. Cryptology ePrint Archive, Report 2004/131 (2004)
10. Chu, C.-K., Tzeng, W.-G.: Efficient  $k$ -out-of- $n$  oblivious transfer schemes with adaptive and non-adaptive queries. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 172–183. Springer, Heidelberg (2005)
11. Chung, B., Hur, J., Kim, H., Hong, S.M., Yoon, H.: Improved batch exponentiation. *Inf. Process. Lett.* **109**(15), 832–837 (2009)
12. Crépeau, C., van de Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995)
13. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Complexity and fast algorithms for multiexponentiations. *IEEE Trans. Comput.* **49**(2), 141–147 (2000)
14. Downey, P., Leong, B., Sethi, R.: Computing sequences with addition chains. *SIAM J. Comput.* **10**(3), 638–646 (1981)
15. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
16. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. *Commun. ACM* **28**(6), 637–647 (1985)
17. Ferrara, A.L., Green, M., Hohenberger, S., Pedersen, M.Ø.: Practical short signature batch verification. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 309–324. Springer, Heidelberg (2009)
18. Fiat, A.: Batch RSA. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 175–185. Springer, Heidelberg (1990)
19. Goldwasser, S., Levin, L.A.: Fair computation of general functions in presence of immoral majority. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991)
20. Gordon, D.M.: A survey of fast exponentiation methods. *J. Algorithms* **27**(1), 129–146 (1998)
21. Hong, S.-M., Oh, S.-Y., Yoon, H.: New modular multiplication algorithms for fast modular exponentiation. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 166–177. Springer, Heidelberg (1996)
22. Hwang, M.S., Lin, I.C., Hwang, K.F.: Cryptanalysis of the batch verifying multiple RSA digital signatures. *Inform. Lith. Acad. Sci.* **11**(1), 15–19 (2000)

23. Kilian, J.: Founding cryptography on oblivious transfer. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 20–31. ACM (1988)
24. Kilian, J.: A general completeness theorem for two party games. In: Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, pp. 553–560. ACM (1991)
25. Kilian, J., Kushilevitz, E., Micali, S., Ostrovsky, R.: Reducibility and completeness in private computations. *SIAM J. Comput.* **29**(4), 1189–1208 (2000)
26. Knuth, D.E.: *The Art of Computer Programming. Seminumerical Algorithms*, vol. 2. Addison-Wesley Professional, Boston (2014)
27. Lim, C.H., Lee, P.J.: More flexible exponentiation with precomputation. In: Desmedt, Y.G. (ed.) *CRYPTO 1994*. LNCS, vol. 839, pp. 95–107. Springer, Heidelberg (1994)
28. Lou, D.C., Lai, J.C., Wu, C.L., Chang, T.J.: An efficient montgomery exponentiation algorithm by using signed-digit-recoding and folding techniques. *Appl. Math. Comput.* **185**(1), 31–44 (2007)
29. Hwang, M.-S., Lee, C.-C., Tang, Y.-L.: Two Simple batch verifying multiple digital signatures. In: Qing, S., Okamoto, T., Zhou, J. (eds.) *ICICS 2001*. LNCS, vol. 2229, pp. 233–237. Springer, Heidelberg (2001)
30. Montgomery, P.L.: Modular multiplication without trial division. *Math. Comput.* **44**(170), 519–521 (1985)
31. M’Raihi, D., Naccache, D.: Batch exponentiation: a fast DLP-based signature generation strategy. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security, CCS 1996, pp. 58–61. ACM, New York (1996)
32. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) *CRYPTO 1999*. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
33. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 448–457. Society for Industrial and Applied Mathematics (2001)
34. Ogata, W., Kurosawa, K.: Oblivious keyword search. *J. Complex.* **20**(2), 356–371 (2004)
35. Pippenger, N.: On the evaluation of powers and monomials. *SIAM J. Comput.* **9**(2), 230–250 (1980)
36. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical report TR-81, Aiken Computation Laboratory, Harvard University (1981)
37. de Rooij, P.: Efficient exponentiation using precomputation and vector addition chains. In: De Santis, A. (ed.) *EUROCRYPT 1994*. LNCS, vol. 950, pp. 389–399. Springer, Heidelberg (1995)
38. Solinas, J.: Low-weight binary representations for pairs of integers. Technical report, CORR 2001–41, Department of C&O, University of Waterloo (2001)
39. Sun, Y., Wu, Q., Qin, B., Wang, Y., Liu, J.: Batch blind signatures on elliptic curves. In: Lopez, J., Wu, Y. (eds.) *Information Security Practice and Experience*. LNCS, vol. 9065, pp. 192–206. Springer, Heidelberg (2015)
40. Tzeng, W.-G.: Efficient 1-out-n oblivious transfer schemes. In: Naccache, D., Paillier, P. (eds.) *PKC 2002*. LNCS, vol. 2274, pp. 159–171. Springer, Heidelberg (2002)
41. Wu, C.L., Lou, D.C., Lai, J.C., Chang, T.J.: Fast modular multi-exponentiation using modified complex arithmetic. *Appl. Math. Comput.* **186**(2), 1065–1074 (2007)
42. Wu, Q., Sun, Y., Qin, B., Hu, J., Liu, W., Liu, J., Ding, Y.: Batch public key cryptosystem with batch multi-exponentiation. *Future Gener. Comput. Syst.* (2015)

43. Yao, A.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, 1986, pp. 162–167. IEEE (1986)
44. Yen, S.M., Laih, C.S., Lenstra, A.K.: Multi-exponentiation (cryptographic protocols). *Comput. Digital Techn.* **141**(6), 325–326 (1994)
45. Zhang, C., Lu, R., Lin, X., Ho, P.H., Shen, X.: An efficient identity-based batch verification scheme for vehicular sensor networks. In: The 27th Conference on Computer Communications, INFOCOM 2008, pp. 816–824. IEEE, April 2008

# **Pseudo Random/One-way Function**

# CTM-sp: A Family of Cryptographic Hash Functions from Chaotic Tent Maps

Xun Yi<sup>(✉)</sup>, Xuechao Yang, Yong Feng, Fengling Han, and Ron van Schyndel

School of Science, RMIT University, Victoria 3000, Australia  
{xun.yi,xuechao.yang,yong.feng,fengling.han,ron.vanschyndel}@rmit.edu.au

**Abstract.** Hash functions are commonly used in checksums, check digits, fingerprints, randomization functions, error-correcting codes, and ciphers. In this paper, we give a family of hash functions on the basis of chaotic tent maps and the sponge construction, where input is absorbed into the hash state at a given rate, then an output hash is squeezed from it at the same rate. The family include four types of hash functions, which are designed to facilitate the implementation on 8-bit, 16-bit, 32-bit and 64-bit processors, respectively. Our experiments have shown that the family meet security criteria for design of hash functions and the performance of our hash functions is comparable to SHA-3.

**Keywords:** Hash function · Chaotic tent map · Sponge construction · Collusion resistant

## 1 Introduction

Hash function ( $H$ ) is a transformation that takes an input  $m$  and outputs a fixed-size string, which is called hash value  $h$  (that is,  $h = H(m)$ ). A hash value services as a compact representative image (sometime called digital fingerprint) of an input string, and can be used as if it were uniquely identifiable with that string. Hash functions are primarily used in hash tables to quickly locate a data record (e.g., a dictionary definition) given its search key (the headword). Specifically, the hash function is used to map the search key to an index; the index gives the place in the hash table where the corresponding record should be stored. Now hash functions are commonly used in checksums, check digits, fingerprints, randomization functions, error-correcting codes, and ciphers.

SHA-3, originally known as Keccak [5], is a cryptographic hash function designed by Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On October 2, 2012, Keccak was selected as the winner of the NIST hash function competition [11].

SHA-3 uses the sponge construction, where input is absorbed into the hash state at a given rate, then an output hash is squeezed from it at the same rate. The sponge construction is shown in Fig. 1, where  $P_1, P_2, \dots, P_i$  are inputs and  $Z_0, Z_1$  are outputs. The sponge function is a generalization of the concept of cryptographic hash function with infinite output and can perform all symmetric

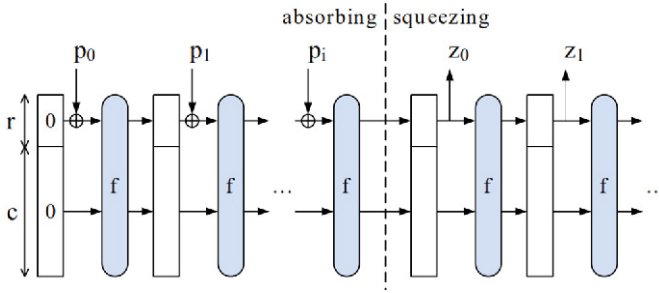


Fig. 1. Sponge construction

cryptographic functions, from hashing to pseudo-random number generation to authenticated encryption.

As primitive used in the sponge construction, the Keccak instances [5] call one of seven permutations, so called Keccak- $f[r+c]$ , with  $r+c = 5 \times 5 \times 2^\ell$  ( $\ell = 0, 1, 2, \dots, 6$ ) and the size of  $P_i$  and  $Z_j$  is  $r$  bits.  $r$  is called bitrate and  $c$  is called capacity. The largest permutation is Keccak- $f[1600]$ , but smaller (or more “lightweight”) permutations can be used in constrained environments. Each permutation consists of the iteration of a simple round function. The round function further consists of 5 invertible step mappings:

- $\theta$  for diffusion;
- $\rho$  for inter-slice dispersion;
- $\pi$  for disturbing horizontal/vertical alignment;
- $\chi$  for non-linearity;
- $\iota$  to break symmetry;

from (3-Dimension)  $5 \times 5 \times 2^\ell$  array of bits to (3-Dimension)  $5 \times 5 \times 2^\ell$  array of bits. Each round has  $12 + 2\ell$  iterations of the five sub-rounds.

The choice of operations in SHA-3 is limited to bitwise XOR, AND and NOT and rotations. The authors claim 12.5 cycles per byte [6] on an Intel Core 2 CPU. However, in hardware implementations it is notably faster than all other finalists [8].

Chaos and cryptography have some common features, the most prominent being sensitivity to variables’ and parameters’ changes. An important difference between chaos and cryptography lies on the fact that systems used in chaos are defined only on real numbers, while cryptography deals with systems defined on finite number of integers [13]. After certain adaption, chaos has been used to construct hash functions [14, 17].

Based on Baptista’s method [4], Wong developed a hash function [14], which is built on the number of iterations of one-dimensional logistic map needed to reach the region corresponding to the character, along with a look-up table updated dynamically.

Based on the simplest one-dimensional chaotic tent maps, Yi [17] proposed a hash function, which operates on a messages with arbitrary length to produce  $2\ell$ -bit hash value and can be easily implemented in both hardware and software.

In recent years, chaotic maps have been often used in the design of hash functions, e.g., [1, 3, 9, 12, 15, 18] and etc. However, some of the existing chaos-based hashing schemes have been shown to have security flaws in [2, 7, 16].

In this paper, we give a family of new hash functions on the basis of chaotic tent maps and the sponge construction, which are called CTM-sp hash functions. The family includes four types of hash functions, which are designed to facilitate the implementation on 8-bit, 16-bit, 32-bit and 64-bit processors, respectively.

## 2 Preliminaries

A chaotic tent map is an one-dimensional and piecewise linear map as follows:

$$F_\alpha : x_i = \begin{cases} x_{i-1} & (\text{if } 0 \leq x_{i-1} \leq \alpha) \\ \frac{1-\alpha}{1-\alpha}x_{i-1} & (\text{if } \alpha < x_{i-1} \leq 1) \end{cases} \tag{1}$$

where  $0 < \alpha < 1$ . This map transforms an interval  $[0,1]$  onto itself and contains only one parameter  $\alpha$ . A sequence computed by iterating  $F_\alpha$  from an arbitrary initial point in  $(0,1)$  acts chaotically because the function  $F_\alpha$  is expansionary everywhere in the interval  $(0,1)$ .

Based on  $F_\alpha$ , we define a map  $G_\alpha$  as follows:

For  $0 \leq \alpha < 1$ ,

$$G_\alpha : x_i = \begin{cases} F_{A(\alpha)}(x_{i-1}) & (\text{if } 0 < x_{i-1} < 1) \\ \beta & (\text{otherwise}) \end{cases} \tag{2}$$

where  $A(\alpha)$  is an affine mapping from  $[0, 1)$  to a sufficient small interval  $I$  around  $1/2$ , e.g.,  $A(\alpha) = 0.4 + 0.2 \cdot \alpha$ , and  $0 < \beta < 1$  is a constant.

## 3 Description of CTM-sp Hash Functions

First of all, a message with arbitrary length is broken into blocks  $P_1, P_2, \dots, P_i$ , each has  $r$  bits, and “padded” so that the size of the message is a multiple of  $r$ . The  $i$  blocks are hashed with the sponge construction as shown in Fig. 1. The core of the sponge construction is the function  $f$ , taking as input  $b = r + c$  bits and returning  $b$  bits, defined in this section.

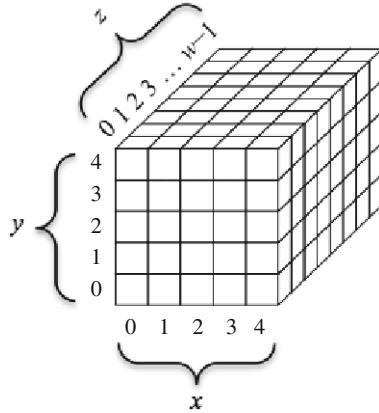
### 3.1 State

Like SHA-3 [5], the state permutation is comprised of  $b$  bits, which is arranged into a 3-dimension array, denoted as  $A$ , as shown in Fig. 2.

From Fig. 1, we can see that

$$b = xyz = 5 \times 5 \times w.$$





**Fig. 2.** State array in 3 dimensions

If  $S$  denotes a string that represents the state, the its bits are indexed from 0 to  $b - 1$ , so that

$$S = S[0]||S[1]|| \cdots ||S[b - 2]||S[b - 1]$$

The corresponding state array,  $A$ , is defined as follows:

For all triples  $(x, y, z)$  such  $0 \leq x \leq 4, 0 \leq y \leq 4$  and  $0 \leq z \leq w - 1$ ,

$$A[x, y, z] = S[x + 5y + 25z].$$

We consider 4 types of states,  $w = 8, 16, 32, 64$ , which are suitable for 8-bit, 16-bit, 32-bit and 64-bit CPU, respectively.

### 3.2 Substitution

The substitution  $S$  transforms a state  $A$  with  $b$  bits to a new state  $A^*$  with  $b$  bits, defined as follows.

Give a state  $A[x, y, z]$  where  $0 \leq x \leq 4, 0 \leq y \leq 4, 0 \leq z \leq w - 1$ , the substitution  $S$  is composed of the following steps:

1. For  $0 \leq x, y \leq 4$ , let

$$B[x, y] = (A[x, y, 0], A[x, y, 1], \cdots, A[x, y, w - 1])$$

2. For  $0 \leq y \leq 4$ , compute

$$C[y] = B[0, y] \oplus B[1, y] \oplus B[2, y] \oplus B[3, y] \oplus B[4, y]$$

where  $\oplus$  denotes the exclusive-OR of two  $w$ -bit blocks.

3. For  $0 \leq y \leq 4$ , transform  $B[0, y]$  and  $C[y]$  into fractions  $b[0, y]$  and  $c[y]$ . Assume that  $B[0, y] = (b_0, b_1, \dots, b_{w-1})$  and  $C[y] = (c_0, c_1, \dots, c_{w-1})$ , let

$$b[0, y] = 0.b_0b_1 \cdots b_{w-1},$$

$$c[y] = 0.c_0c_1 \cdots c_{w-1}$$

4. According to the chaotic tent map defined in Eq. 2, compute

$$d[y] = G_{c[y]}^\ell(b[0, y]) + c[y] \pmod{1}$$

where  $G_{c[y]}^\ell$  stands for  $\overbrace{G_{c[y]} \circ \cdots \circ G_{c[y]}}^\ell$  as defined in Algorithm 1,  $\ell = \log_2 w$  and  $\pmod{1}$  denotes an operation removing the integer part, e.g.,  $1.0101011101 \pmod{1} = 0.0101011101$ .

---

**Algorithm 1.** Function  $G_\alpha^\ell(x_0)$

---

**Input:**  $b[0, y], c[y], \ell$ .

**Output:**  $d[y]$

- 1: Let  $x_0 = b[0, y], \alpha = c[y]$
  - 2: For  $j = 1$  to  $\ell$
  - 3: If  $x_{j-1} \neq 0$
  - 4: Let  $x_j = G_\alpha(x_{j-1}) = F_{0.4+0.2\cdot\alpha}(x_{j-1})$
  - 5: Else
  - 6: Let  $x_j = \beta$
  - 7: End If
  - 8: Next  $j$
  - 9: Let  $d[y] = x_\ell + c[y] \pmod{1}$
  - 10: **return**  $d[y]$
- 

5. For  $0 \leq y \leq 4$ , assume that  $d[y] = 0.d_0d_1 \cdots d_{w-1}$  (note that we remove all bits after  $w$ ), let

$$A^*[0, y, z] = d_z$$

for  $z = 0, 1, 2, \dots, w - 1$ . In addition, let

$$A^*[x, y, z] = A[x, y, z] \oplus A^*[0, y, z]$$

for  $1 \leq x \leq 4, 0 \leq y \leq 4, 0 \leq z \leq w - 1$ .

The substitution transformation can be illustrated as in Fig. 3, where the first element is replaced and other four elements are XORed with the first element. CTM denotes Chaotic Tent Mapping and  $y = 0, 1, 2, 3, 4$ .

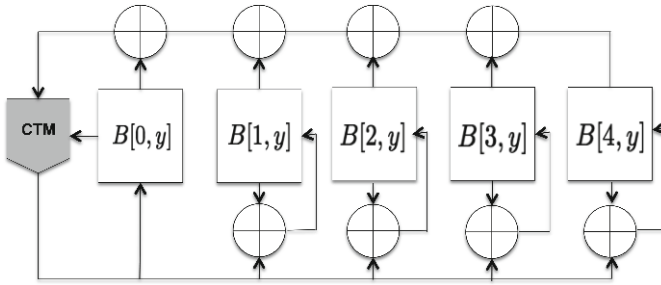


Fig. 3. Substitution transformation

### 3.3 Permutation

The permutation  $P$  transforms a state  $A$  with  $b$  bits to a new state  $A^*$  with  $b$  bits, defined as follows.

Give a state  $A[x, y, z]$  where  $0 \leq x \leq 4, 0 \leq y \leq 4, 0 \leq z \leq w - 1$ , the permutation  $P$  is composed of the following steps:

1. For  $0 \leq z \leq w - 1$ , let

$$B[x + 5y + 1, z] = A[x, y, z]$$

2. For  $0 \leq z \leq w - 1, 1 \leq t \leq 25$ , permute  $B[t, z]$  according to the following right table:

The output at the position  $(t, z)$  is denoted as  $C[t, z]$ .

3. For  $0 \leq x, y \leq 4, 0 \leq z \leq w - 1$ , let

$$A^*[x, y, z] = C[x + 5y + 1, z].$$

### 3.4 Function $f$

With reference to Fig. 1, the function  $f$  takes as input  $b$  bits and outputs  $b$  bits, as described in Algorithm 2, where  $S$  and  $P$  denote the substitution and permutation, respectively.

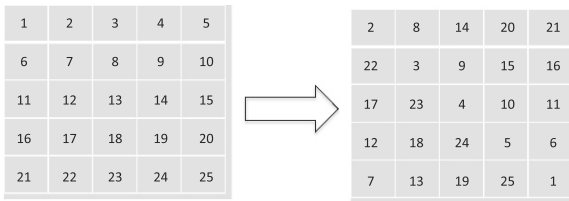


Fig. 4. Permutation transformation

---

**Algorithm 2.** Function  $f$

---

**Input:**  $A[x, y, z]$  where  $0 \leq x, y \leq 4, 0 \leq z \leq w - 1$ .  
**Output:**  $A^*[x, y, z]$  where  $0 \leq x, y \leq 4, 0 \leq z \leq w - 1$ .  
 1: For  $j = 1$  to 5  
 2: Let  $A^*[x, y, z] = P(S(A[x, y, z]))$  for any  $x, y, z$   
 3: Let  $A[x, y, z] = A^*[x, y, z]$  for any  $x, y, z$   
 4: Next  $j$   
 5:  $A^*[x, y, z] = A[y, x, z]$  for any  $x, y, z$   
 6:  $A[x, y, z] = A^*[x, y, z]$  for any  $x, y, z$   
 7: For  $j = 1$  to 5  
 8: Let  $A^*[x, y, z] = P(S(A[x, y, z]))$  for any  $x, y, z$   
 9: Let  $A[x, y, z] = A^*[x, y, z]$  for any  $x, y, z$   
 10: Next  $j$   
 11: **return**  $A^*[x, y, z]$  for any  $x, y, z$

---

From Algorithm 2, we can see the function  $f$  has ten rounds of iteration. With reference to Figs. 3 and 4, rows are hashed by substitutions and permutations in the first five rounds of Algorithm 2. After transposing  $x$  and  $y$ , columns are hashed by substitutions and permutations in the last five rounds. Unlike SHA-3, the function  $f$  does not have any permutation along with  $z$  axis. The hash in the direction of  $z$  axis is achieved by treating  $B[0, y]$  ( $y = 0, 1, 2, 3, 4$ ) as a decimal  $b[0, y]$  and iterating chaotic tent maps.

## 4 Theoretical Security Analysis

### 4.1 Distribution of $G_\alpha^\ell(x_0)$

$G_\alpha^\ell(x_0)$  plays a critical role in our hash round function as shown in Fig. 3. It is evolved from the chaotic tent map  $F_\alpha(x_0)$  and its features are determined by two parameters  $\alpha$  and  $\ell$ .

**Theorem 1** [17]. For any  $0 \leq \alpha < 1$ , the distribution of  $x_1 = G_\alpha(x_0)$  for randomly chosen  $0 < x_0 < 1$  is the standard uniform distribution  $U(0, 1)$ .

**Proof:** Considering that  $x_0$  is randomly chosen from  $(0, 1)$ ,  $x_0$  obeys the standard uniform distribution  $U(0, 1)$ . For any  $0 < \varepsilon < 1$ ,

$$\begin{aligned} Prob(x_1 < \varepsilon) &= Prob\left(\frac{x_0}{A(\alpha)} < \varepsilon, x_0 \leq A(\alpha)\right) \\ &\quad + Prob\left(\frac{1 - x_0}{1 - A(\alpha)} < \varepsilon, x_0 > A(\alpha)\right) \\ &= Prob(x_0 < A(\alpha)\varepsilon) \\ &\quad + Prob(x_0 > 1 - (1 - A(\alpha))\varepsilon) \\ &= A(\alpha)\varepsilon + 1 - (1 - (1 - A(\alpha))\varepsilon) = \varepsilon \end{aligned}$$

Therefore,  $x_1 = G_\alpha(x_0)$  obeys the standard uniform distribution  $U(0, 1)$ .  $\triangle$

Note:  $A(\alpha)\varepsilon < A(\alpha)$  and  $1 - (1 - A(\alpha))\varepsilon > A(\alpha)$  when  $0 < A(\alpha) < 1$  and  $0 < \varepsilon < 1$ .

**Corollary 1** [17]. For any  $0 \leq \alpha < 1$  and any  $\ell$ , the distribution of  $x_\ell = G_\alpha^\ell(x_0)$  for randomly chosen  $0 < x_0 < 1$  is the standard uniform distribution  $U(0, 1)$ .

**Proof:** Based on Theorem 1, this corollary can be proved by the method of induction.  $\Delta$

In Algorithm 1, we compute

$$d[y] = G_{c[y]}^\ell(b[0, y]) + c[y] \pmod{1}$$

for  $y = 0, 1, 2, 3$ . Assume that  $d[y] = 0.d_1d_2 \cdots d_{w-1}$ , we replace  $A[0, y, z]$  with  $d_z$  for  $z = 0, 1, \dots, w$ .

**Theorem 2.** For any  $0 \leq c[y] < 1$  and any  $\ell$ , the distribution of  $d[y] = G_{c[y]}^\ell(b[0, y]) + c[y] \pmod{1}$  for randomly chosen  $0 < b[0, y] < 1$  is the standard uniform distribution  $U(0, 1)$ .

**Proof:** Let  $a = c[y]$  and  $x = b[0, y]$ . Based on Corollary 1, we know  $Prob(G_a^\ell(x) < \varepsilon) = \varepsilon$ . Therefore, we have

$$\begin{aligned} & Prob(d[y] < \varepsilon) \\ &= Prob(G_a^\ell(x) + a - 1 < \varepsilon, G_a^\ell(x) + a \geq 1) \\ &\quad + Prob(G_a^\ell(x) + a < \varepsilon, G_a^\ell(x) + a < 1) \\ &= Prob(G_a^\ell(x) < \varepsilon + 1 - a, G_a^\ell(x) \geq 1 - a) \\ &\quad + Prob(G_a^\ell(x) < \varepsilon - a, G_a^\ell(x) < 1 - a) \\ &= (\varepsilon + 1 - a)(1 - (1 - a)) + (\varepsilon - a)(1 - a) \\ &= \varepsilon a + (1 - a)a + \varepsilon - \varepsilon a - a(1 - a) = \varepsilon \end{aligned}$$

Therefore,  $d[y] = G_a^\ell(x) + a \pmod{1}$  obeys the standard uniform distribution  $U(0, 1)$ .  $\Delta$

## 4.2 Diffusion Analysis of $f$

Given an input of  $b$  bits, a function  $f$  with the diffusion property outputs  $b$  bits, such that each output bit depends on all input bits.

The function  $f$  is composed of two basic transformations - substitution  $S$  and permutation  $P$ . With reference to Figs. 3 and 4, after a substitution, each element in the first column is replaced by the output of the chaotic tent map and becomes dependence of all elements of the same row. Then a permutation follows.

Two criteria for design of the permutation are: (1) after a permutation, the five output elements in each row come from different rows in the input; (2) after the first five rounds of iteration in Algorithm 2, all elements as shown in Fig. 4 are replaced by the outputs of the chaotic tent maps.

With reference to Fig. 4, we can see that the permutation meets the first criterion. In addition, each permutation shifts five columns to the left cyclically with positions of five elements in each column being changed. Therefore, the permutation also meets the second criterion.

In the end of the first five rounds of iteration in Algorithm 2, each column becomes dependent to all twenty-five elements in the original input. After transposing  $x$  and  $y$ , columns become rows. In the last five rounds of iteration in Algorithm 2, all twenty-five elements are replaced by the outputs of the chaotic tent maps again and become dependent to all twenty-five elements in the original input.

## 5 Experimental Security Analysis

We have implemented the function  $f$  with  $b = 5 \times 5 \times 2^\ell$  bits for  $\ell = 3, 4, 5, 6$  by using Java programming and performed the avalanche effect test, the correlation test and the statistical tests for the validation of random number generators and pseudo random number generators for cryptographic applications [10].

### 5.1 Avalanche Effect Test

The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly (e.g., half the output bits flip).

To check if the function  $f$  meets the avalanche effect, we have done the following test.

1. Randomly choose an input  $a$  with  $b (=5 \times 5 \times 2^\ell)$  bits and compute  $c = f(a)$  with Algorithm 2. Let  $d = 1$ .
2. Compute  $a^* = a \oplus d$  and  $c^* = f(a^*)$ .
3. Compute the change rate  $r = |c^* \oplus c|/b$  where  $|c^* \oplus c|$  denotes the Hamming weight.
4. If  $d \neq 2^b$ , then let  $d = d \cdot 2$  and go to (2). Otherwise, stop.

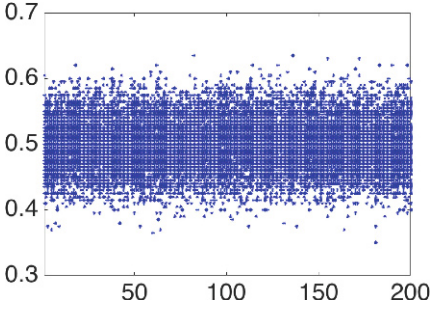
We have repeated the test for 100 times for  $\ell = 3, 4, 5, 6$  and found that the change rate is always around 50% as shown in Figs. 5, 6, 7 and 8.

Figures 5, 6, 7 and 8 show that avalanche effect becomes better and better with the increase of  $\ell$ .

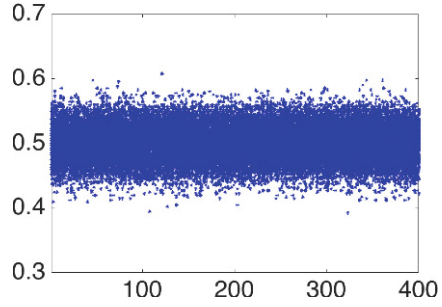
### 5.2 Correlation Test

A correlation coefficient is a statistical measure of the degree to which changes to the value of one variable predict change to the value of another.

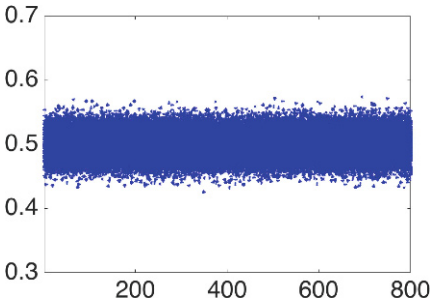
To check correlation relations among the bits of the hash input and output, we have done the following test.



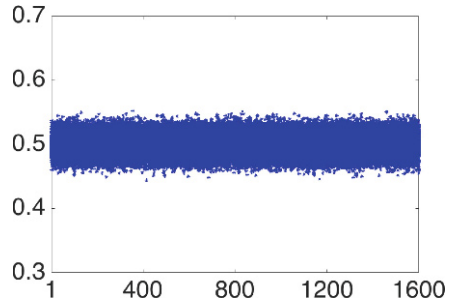
**Fig. 5.** Avalanche effect test ( $\ell = 3$ )



**Fig. 6.** Avalanche effect test ( $\ell = 4$ )



**Fig. 7.** Avalanche effect test ( $\ell = 5$ )

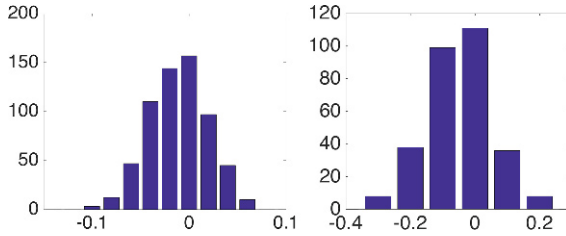


**Fig. 8.** Avalanche effect test ( $\ell = 6$ )

1. Randomly choose  $n$  ( $= 10000$ ) inputs  $a_1, a_2, \dots, a_n$ , each having  $b$  ( $= 5 \times 5 \times 2^\ell$ ) bits and compute  $c_i = f(a_i)$  for  $i = 1, 2, \dots, n$  with Algorithm 2. The size of  $c_i$  is also  $b$  bits.
2. For each  $a_i$  ( $i = 1, 2, \dots, n$ ), we partition it into 25 blocks,  $a_{i1}, a_{i2}, \dots, a_{im}$  with equal size  $2^\ell$  bits, where  $m = 25$ .
3. For each  $c_i$  ( $i = 1, 2, \dots, n$ ), we partition it into 25 blocks,  $c_{i1}, c_{i2}, \dots, c_{im}$  with equal size  $2^\ell$  bits, where  $m = 25$ .
4. For each pair  $(u, v)$  where  $1 \leq u, v \leq m$ , let  $x_i = a_{iu}$  and  $y_i = c_{iv}$  for  $i = 1, 2, \dots, n$  and compute the correlation coefficient between hash input and output

$$r_{uv} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

where  $\bar{x} = \sum_{i=1}^n x_i/n$  and  $\bar{y} = \sum_{i=1}^n y_i/n$ .

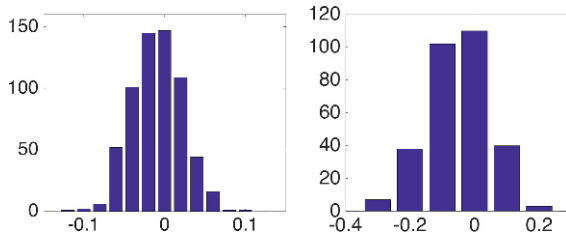


**Fig. 9.** Correlation test (where  $r_{uv}$  is on the left and  $r'_{uv}$  is on the right,  $\ell = 3$ )

5. For each pair  $(u, v)$  where  $u \neq v$  and  $1 \leq u, v \leq m$ , let  $x_i = c_{iu}$  and  $y_i = c_{iv}$  for  $i = 1, 2, \dots, n$  and compute the correlation coefficient between hash output and output

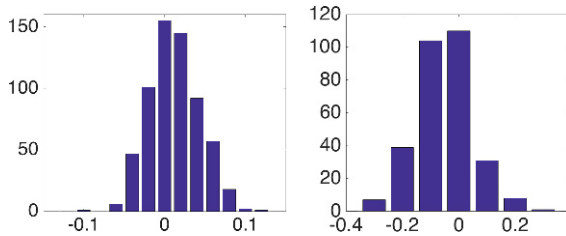
$$r'_{uv} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

where  $\bar{x} = \sum_{i=1}^n x_i/n$  and  $\bar{y} = \sum_{i=1}^n y_i/n$ .



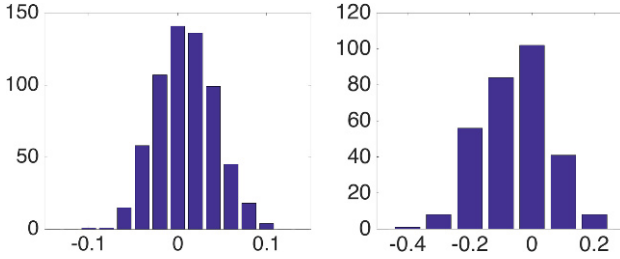
**Fig. 10.** Correlation test (where  $\ell = 4$ )

The correlation coefficients  $r_{uv}$  and  $r'_{uv}$  are shown in Figs. 9, 10, 11 and 12.



**Fig. 11.** Correlation test (where  $\ell = 5$ )





**Fig. 12.** Correlation test (where  $\ell = 6$ )

Figures 9, 10, 11 and 12 show that the correlation coefficients between input and output and between output and output are close to 0 and indicate the independence between input and output and between output and output.

### 5.3 NIST Statistical Tests

In 2010, a statistical test suite for random and pseudorandom number generators for cryptographic applications was given by NIST [11]. We have downloaded the software and tested the randomness properties of our hash output when  $\ell = 6$ . We collect a sequence of hash output up to  $100 \times 10,000 \times 1,600$  bits and run the software on the data. The statistical test results are shown in Fig. 13 (in Appendix), from which we can see our data passes all statistical tests.

## 6 Conclusion

In this paper, we have given a family of hash functions on the basis of chaotic tent maps and the sponge construction. Experiments have displayed good properties of our hash function.

We have implemented our hash algorithm with  $\ell = 6$  (i.e.,  $b = 1,600$  bits), 1,088-bit block size (i.e.,  $r = 1,088$  bits) and 256-bit hash output on a MacBook Pro, Intel Core i7, 2 GHz, 4 Cores, 8 GB memory with Java version 8 Update 65 (build 1.8.0\_65-b17). Hash of 10 MB data takes about 100 s. We test SHA3-256 on the same platform and hash of 10 MB data also takes about 100 s. The performance of our hash algorithm is comparable to SHA-3. Like SHA-3, most operations in our hash algorithm can be carried out in parallel. On implementation, this feature can be used to improve the performance of our hash algorithm significantly.

# Appendix

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <10000.txt>												
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
5	12	10	9	8	10	18	11	7	0.289667	100/100	Frequency	
11	11	12	3	9	10	10	11	15	8	0.474986	99/100	BlockFrequency
9	8	7	16	7	13	13	10	6	11	0.401199	100/100	CumulativeSums
8	6	16	7	17	7	9	11	12	7	0.129620	100/100	CumulativeSums
6	13	15	3	7	16	10	15	8	7	0.032923	99/100	Runs
7	4	15	14	6	14	9	7	16	8	0.051942	98/100	LongestRun
9	12	8	8	8	7	11	16	13	8	0.574903	100/100	Rank
15	14	12	12	7	7	15	6	6	6	0.122325	99/100	FFT
11	10	7	7	6	13	12	14	10	10	0.699313	98/100	NonOverlappingTemplate
8	8	12	3	13	12	9	14	13	8	0.319084	100/100	NonOverlappingTemplate
8	14	5	12	16	10	6	12	9	8	0.275709	98/100	NonOverlappingTemplate
13	9	11	9	10	13	8	11	9	7	0.935716	98/100	NonOverlappingTemplate
11	7	12	7	13	3	12	12	13	10	0.366918	98/100	NonOverlappingTemplate
9	13	9	11	5	7	13	12	13	8	0.616305	99/100	NonOverlappingTemplate
5	10	11	9	13	13	8	9	6	16	0.334538	100/100	NonOverlappingTemplate
7	7	12	8	11	13	10	9	12	11	0.897763	98/100	NonOverlappingTemplate
5	14	10	6	4	10	10	24	9	8	0.000555	100/100	NonOverlappingTemplate
16	10	9	14	14	8	6	9	9	5	0.236810	98/100	NonOverlappingTemplate
13	9	7	9	4	17	13	8	13	7	0.137282	99/100	NonOverlappingTemplate
12	9	6	11	9	8	13	12	8	12	0.851383	99/100	NonOverlappingTemplate
11	8	8	11	8	13	9	10	13	9	0.946308	100/100	NonOverlappingTemplate
13	8	11	10	12	6	6	9	14	11	0.657933	98/100	NonOverlappingTemplate
9	11	15	4	9	13	5	14	12	8	0.202268	99/100	NonOverlappingTemplate
8	7	11	13	7	8	12	11	10	13	0.834308	98/100	NonOverlappingTemplate
6	20	9	11	10	6	10	8	9	11	0.122325	99/100	NonOverlappingTemplate
12	7	7	13	13	12	12	6	10	8	0.657933	100/100	NonOverlappingTemplate
11	11	14	10	9	7	11	11	10	6	0.867692	99/100	OverlappingTemplate
9	13	9	7	11	9	11	10	15	6	0.699313	98/100	Universal
5	13	7	9	11	16	8	10	8	13	0.366918	99/100	ApproximateEntropy
2	4	6	5	7	9	5	8	7	9	0.568055	61/62	RandomExcursions
4	4	7	6	6	10	6	6	8	5	0.834308	62/62	RandomExcursions
1	5	8	6	8	6	7	6	8	7	0.671779	62/62	RandomExcursions
6	6	6	6	7	4	8	2	11	6	0.500934	62/62	RandomExcursions
5	2	5	5	6	10	6	7	9	7	0.568055	61/62	RandomExcursions
8	6	4	8	6	5	4	12	6	3	0.324180	62/62	RandomExcursions
5	6	5	5	5	9	3	3	14	7	0.066882	62/62	RandomExcursions
5	3	6	7	3	10	6	7	7	8	0.637119	61/62	RandomExcursions
7	4	6	6	13	3	7	5	7	4	0.232760	61/62	RandomExcursionsVariant
6	4	6	7	11	7	7	4	6	4	0.671779	61/62	RandomExcursionsVariant
4	7	7	6	11	3	5	5	7	7	0.602458	61/62	RandomExcursionsVariant
8	6	6	7	2	6	7	6	7	7	0.911413	59/62	RandomExcursionsVariant
10	5	3	5	8	7	3	9	7	5	0.468595	61/62	RandomExcursionsVariant
10	8	1	3	8	5	4	9	10	4	0.082177	61/62	RandomExcursionsVariant
6	7	8	2	3	9	6	6	8	7	0.602458	60/62	RandomExcursionsVariant
4	8	7	7	4	9	4	7	5	0.834308	60/62	RandomExcursionsVariant	
4	6	10	4	8	10	2	8	6	4	0.253551	60/62	RandomExcursionsVariant
6	6	6	9	3	6	6	9	5	6	0.862344	62/62	RandomExcursionsVariant
8	6	3	5	3	8	5	5	8	8	0.568055	62/62	RandomExcursionsVariant
8	6	9	10	12	9	6	16	10	14	0.401199	100/100	Serial
10	9	9	6	6	10	10	13	15	12	0.616305	99/100	Serial
7	11	9	9	9	12	11	11	8	13	0.955835	98/100	LinearComplexity

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 96 for a sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately = 59 for a sample size = 62 binary sequences.

For further guidelines construct a probability table using the MAPLE program provided in the addendum section of the documentation.

Fig. 13. NIST statistical tests

## References

1. Akhavan, A., Samsudin, A., Akhshani, A.: Hash function based on piecewise non-linear chaotic map. *Chaos Solitons Fract.* **42**(2), 1046–1053 (2009)
2. Alvarez, G., Montoya, F., Romera, M., Pastor, G.: Cryptanalysis of dynamic look-up table based chaotic cryptosystems. *Phys. Lett. A* **326**, 211–218 (2004)

3. Amin, M., Faragallah, O., El-Latif, A.: Chaos-based hash function (CBHF) for cryptographic applications. *Chaos Solitons Fract.* **42**, 767–772 (2009)
4. Baptista, M.S.: Cryptography with chaos. *Phys. Lett. A* **240**(1/2), 50–54 (1998)
5. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak sponge function family. <http://keccak.noekeon.org/>
6. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V.: Implementation overview version 3.2. <http://keccak.noekeon.org/Keccak-implementation-3.2.pdf>
7. Guo, W., Wang, X., He, D., Cao, Y.: Cryptanalysis on a parallel keyed hash function based on chaotic maps. *Phys. Lett. A* **373**, 2306–3201 (2009)
8. Guo, X., Huang, S., Nazhandali, L., Schaumont, P.: Fair and comprehensive performance evaluation of 14 second round sha-3 ASIC implementations. In: NIST 2nd SHA-3 Candidate Conference (2010)
9. Kano, A., Ghebleh, M.: A fast and efficient chaos-based keyed hash function. *Commun. Nonlinear Sci. Numer. Simul.* **18**(1), 109–123 (2013)
10. NIST Special Publication 800-22 rev1, A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications (2008)
11. NIST selects winner of secure hash algorithm (SHA-3) competition. <http://www.nist.gov/itl/csd/sha-100212.cfm>
12. Ren, H., Wang, Y., Xie, Q., Yang, H.: A novel method for one-way hash function construction based on spatio temporal chaos. *Chaos Solitons Fract.* **42**(4), 2014–2022 (2009)
13. Stojanovski, T., Kocarev, L.: Chaos-based random number generators - part I: analysis. *IEEE Trans. Circu. Syst. I Fundam. Theor. Appl.* **48**(3), 281–288 (2001)
14. Wong, K.W.: A combined chaotic cryptographic and hashing scheme. *Phys. Lett. A* **307**, 292–298 (2003)
15. Xiao, D., Liao, X., Deng, S.: Parallel keyed hash function construction based on chaotic maps. *Phys. Lett. A* **372**, 4682–4688 (2008)
16. Xiao, D., Peng, W., Liao, X., Xiang, T.: Collision analysis of one kind of chaos-based hash function. *Phys. Lett. A* **374**, 1228–1231 (2010)
17. Yi, X.: Hash function based on chaotic tent maps. *IEEE Trans. Circ. Syst. II Express Brief* **52**(6), 354–357 (2005)
18. Zhang, J., Wang, X., Zhang, W.: Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter. *Phys. Lett. A* **362**, 439–448 (2007)

# One-Key Compression Function Based MAC with Security Beyond Birthday Bound

Avijit Dutta, Mridul Nandi, and Goutam Paul<sup>(✉)</sup>

Indian Statistical Institute, Kolkata 700 108, India  
avirocks.dutta13@gmail.com, mridul.nandi@gmail.com,  
goutam.paul@isical.ac.in

**Abstract.** Gaži et al. [CRYPTO 2014] analyzed the NI-MAC construction proposed by An and Bellare [CRYPTO 1999] and gave a tight birthday-bound of  $O(\ell q^2/2^n)$ , as an improvement over the previous bound of  $O(\ell^2 q^2/2^n)$ . In this paper, we design a simple extension of NI-MAC, called NI<sup>+</sup>-MAC, and prove that it has security bound beyond birthday (BBB) of order  $O(q^2 \ell^2/2^{2n})$  provided  $\ell \leq 2^{n/4}$ . Our construction not only lifts the security of NI-MAC beyond birthday, it also reduces the number of keys from 2 (NI uses 2 independent keys) to 1. Before this work, Yasuda had proposed [FSE 2008] a single fixed-keyed compression function based BBB-secure MAC with security bound  $O(\ell q^2/2^{2n})$  that uses an extra mask, and requires a storage space to store the mask. However, our proposed construction NI<sup>+</sup> does not require any extra mask and thereby reduces the state size compared to Yasuda's proposal [FSE 2008] with providing the same order of security bound for light-weight applications.

**Keywords:** Beyond birthday · MAC · NI · Structure-graph

## 1 Introduction

In symmetric key paradigm, MAC (Message Authentication Code) is used for preserving message integrity and message origin authentication. The design of a MAC should not only consider achieving security, but also target attaining efficiency. In the literature, three different approaches of designing a MAC exists: (a) universal hash function based MAC, a popular example of which is UMAC [8], (b) a compression function based MAC, like NMAC [2], HMAC [2], NI [1] etc. (c) Block cipher based MAC, such as CBC MAC [4], PMAC [9], OMAC [18] etc. Most of the popular MACs are block cipher based MACs, but each one of them suffers from the same problem - security is guaranteed up to the *birthday bound*. When the block length of the underlying block cipher is 128-bit, then birthday bound does not seem to be a problem, as we are guaranteed to have 64 bits of security which is well acceptable for many practical applications. But when we deal with 64-bit block cipher (e.g. HIGHT [17], PRESENT [10]) as used in many light weight crypto devices (e.g. RFID, smartcard) then birthday bound problem becomes the main bottleneck.

**NMAC and HMAC.** NMAC and its variant HMAC [2] is the first re-keying compression function based MAC where a key is appended to a message and then the appended message is hashed using Merkle-Damgård technique. It has been standardized in [24] and has become popular and widely used in many network protocols like SSH, IPsec, TLS etc. Bellare et al. in [2] proves that NMAC is a secure PRF based on the assumptions (i)  $f$  is a secure PRF and (ii)  $\text{Casc}^f$  is a WCR (weakly collision resistant). HMAC, when instantiated with MD4 or SHA-1, plays the role of  $\text{Casc}^f$  and both have been found not to satisfy the WCR property [38, 39] and hence the security of HMAC [2] stands void. To restore the PRF security of NMAC, Bellare in [6] investigates the proof and drops the assumption (ii). Kobitz and Menezes in [23] criticizes the way [6] discusses the practical implication of their result against uniform and non-uniform reductions used in the proof.

Dodis et al. in [12] investigates the indistinguishable property of HMAC from a keyed random oracle. In a recent line of researches, generic attack against iterated hash based MAC are being investigated [26, 31–33]. More recently, Gaži et al. in [15] showed a tight bound on NMAC. There is also a recent result [16] on the generic security analysis of NMAC and HMAC with input whitening.

Yasuda in [41] had proposed a novel way of iterating a compression function dedicated for the use of MAC which is more efficient than standard HMAC to process data much faster. In [43] Yasuda has showed that classical sandwiched construction with Merkle-Damgård iteration based hashing provides a secure MAC which is an alternative for HMAC, useful in situation where the message size is small and high performance is required. A new secret-prefix MAC based on hash functions is presented in [46] which is similar to HMAC but does not require the second key.

U. Maurer et al. in [28] has presented a MAC construction namely PDI, that transforms any Fixed-Input Length (FIL) MAC to Alternative Input Length (AIL) MAC and investigated the tradeoff between the efficiency of MAC and the tightness of its security reduction. In [29], construction of AIL MAC from a FIL MAC with a single key was presented which is better than NI [1].

**Beyond Birthday Secure MAC.** We discuss two types of MACs in this category - one is block cipher based and the other one is compression function based.

- *Block Cipher Based Beyond Birthday Secure MAC.* Recently, many MAC constructions have been proposed with security beyond the birthday barrier without degrading the performance. The first attempt was made in ISO 9797-1 [3] without security proof. But Algorithm 4 of ISO 9797-1 was attacked by Joux et al. [21] that falsified the security bound. Algorithm 6 of ISO 9797-1 was proven to be secure against  $O(2^{2n/3})$  queries with restrictions on the message length [47]. In [47], Yasuda also presented SUM-ECBC, a 4-key rate-1/2 construction with beyond birthday bound security. In 2011, Yasuda improved the number of keys

and rate over SUM-ECBC and proposed a 3-key rate-1 PMAC\_Plus construction [48] with beyond birthday security. In 2012, Zhang et al. [51] proposed a 3key version of f9 MAC (3kf9) that achieves BBB security.

There is also another deterministic MAC mode that provides security beyond the birthday bound. Given an  $n$ -bit to  $n$ -bit fixed-key block cipher with MAC security  $\epsilon$  against  $q$  queries, Dodis et al. [13] have designed a variable-length MAC achieving  $O(\epsilon q \text{poly}(n))$  MAC security. However, this design requires even longer keys and more block cipher invocations. By parity method, Bellare et al. present MACRX [3] with BBB security, conditioned on the input parameters being random and distinct. In [19], Jaulmes et al. proposed a randomized MAC that provides BBB security based on the ideal model (or possibly based on tweakable block cipher). Another BBB secure randomized construction called generic enhanced hash-then-MAC has been proposed in [30] by Minematsu. In [25], the authors propose a tweakable block-cipher based two-key rate-2 BBB-secure MAC with security margin of  $O(q^2 \ell^2 / 2^n)$ . Recently Datta et al. in [11] unify PMAC\_Plus and 3kf9 in one key setting with beyond birthday security.

- *Compression Function Based Beyond Birthday Secure MAC.* Besides the block cipher based BBB MAC constructions, Yasuda in [42] proposed a compression function based MAC construction - Multi-lane HMAC, that achieves BBB security. In [45] Yasuda presented a double pipe mode operation (Lucks Construction [27]) for constructing AIL MAC from a FIL MAC that achieves BBB security. This work is further extended to provide full security in [49]. In [44] Yasuda has proposed a fixed single keyed compression function based cascaded MAC in which, for an  $l$  blocks message, one needs to compute  $l$  many different masks where the masks are generated from a single mask  $\Delta_0$  using the field multiplication. The security of the scheme has been proved to be  $O(\ell q^2 / 2^{2n})$ . Further improvement on [44] is followed in [50].

**Fixed-Key MAC.** An et al. in [1] proposed a fixed-keyed compression function based MAC called NI-MAC. The construction of NI-MAC is similar to that of NMAC [2], the only difference is that NI-MAC uses two independent keyed compression functions  $f_{K_1}, f_{K_2}$ . The motivation of designing NI was to avoid constant re-keying on multi-block messages in NMAC and to allow for a security proof starting by the standard switch from a PRF to a random function, followed by information-theoretic analysis.

We mention here that the security proof technique for re-keying compression function based MAC is completely different from that of fixed-keyed compression function based MAC. The security of the former scheme is proved using reduction argument, whereas that of the latter is proved by replacing the fixed-keyed compression function with a random function.

Gaži et al. in [15] revisited the proof of NI-MAC and gave a tight birthday bound of  $O(\frac{\ell q^2}{2^n})$ , a better bound than earlier  $O(\frac{\ell^2 q^2}{2^n})$ .

**Our Contributions.** We have the following two main contributions.

(1) We propose a fixed key compression function based MAC  $\text{NI}^+$  with rate<sup>1</sup>  $b/(b+n)$ , which is an extension of existing NI-MAC, that achieves beyond-birthday security of security bound  $O(q^2\ell^2/2^{2n})$ , where  $b$  is the block length and  $n$  is the number of output bits. Our proposed construction not only lifts the security of NI beyond birthday (Sect. 4), but also reduces the number of required keys from two (NI uses two independent keys) to one.

(2) Yasuda in [44] proposed a rate-1, one pass mode BBB secure MAC with a beyond birthday security bound of  $O(\ell q^2/2^{2n})$ . The construction uses a keyed-compression function  $f_k$  from  $b'$  bits to  $n$  bits and a  $b'$ -bit mask  $\Delta_0$  where one needs to store the mask value. Note that, the assumption in the construction [44] is  $b' \geq 2n$ . Now, for processing a message of  $l$  blocks, one needs to compute the masks  $\Delta_1, \Delta_2, \dots, \Delta_l$  which are computed from  $\Delta_0$  using field multiplication. The state size of the Yasuda's proposed construction is  $2(b'+n)$ , as one needs to store the  $b'$ -bit masking value and the  $b'$ -bit checksum value along with two  $n$  bits partial outputs<sup>2</sup>. In this regard, our construction  $\text{NI}^+$  is a rate- $b/b+n$  single-keyed compression function based MAC that uses a keyed-compression function  $f_k$  from  $(b+n)$ -bits to  $n$  bits, where  $b > n$ . Our construction does not use any mask and therefore the state size of  $\text{NI}^+$  is reduced to  $(b+2n)$  as one needs to store  $b$ -bit checksum value along with two  $n$  bit partial outputs.

However, to compare the state-size of Yasuda's construction with our design, one needs to consider the compression functions with the same input size in both the scheme, i.e., one needs to replace input size ( $b'$ ) of the compression function used in the construction proposed in [44] by  $b+n$ , which gives the state size of Yasuda's scheme to  $2(b+n)+2n = 2(b+2n)$ , which is twice of our state size. Though reducing this state-size to  $2n$  bits was placed as an open problem in [44, Section 7], our construction has slightly improved the state size, albeit with the cost of an extra factor of  $\ell$  in the security bound. However, we note that this bound is comparable to that of [44] for light-weight applications in which  $\ell$  is usually assumed to be small.

In the following table we show different parameters and the security bound of known stateless and deterministic BBB secure MACs. We write BC to denote block cipher based MAC in which the underlying primitive is a block cipher and  $\text{CF}_{rk}$  denotes re-keying compression function based MAC in which the underlying primitive is a compression function (e.g. HMAC),  $\text{CF}_{fk}$  denotes fixed-keyed compression function based MAC (e.g. NI).

<sup>1</sup> Rate  $\triangleq \frac{b}{rs}$ , where  $b$  is the size of message block,  $s$  is the total size of the function without the key part and  $r$  is the total number of function calls to process a single message block.

<sup>2</sup> In [44] author has mistakenly stated the state size for the construction is  $b'+2n$  bits, without considering the state size required for storing the  $b'$ -bit mask, thus eventually state size becomes  $2(b'+n)$ .

Construction	Type	# Keys	Rate	Security Bound	State size (#bits)
SUM-ECBC [47]	BC	4	1/2	$O(\ell^3 q^3 / 2^{2n})$	$2n$
PMAC_Plus [48]	BC	3	1	$O(\ell^3 q^3 / 2^{2n})$	$4n$
3kf9 [51]	BC	3	1	$O(\ell^3 q^3 / 2^{2n})$	$2n$
1kf9 [11]	BC	1	1	$O(q^3 \ell^4 / 2^{2n})$	$2n$
1k_PMAC+ [11]	BC	1	1	$O(q^3 \ell^4 / 2^{2n})$	$4n$
$L$ -Lane ( $L = 2$ ) HMAC [42]	CF <sub>rk</sub>	3	1/2	$O(\ell^2 q^2 / 2^{2n})$	$2n$
1-pass mode [44]	CF <sub>fk</sub>	1	1	$O(\ell q^2 / 2^{2n})$	$(2b + 4n)$
NI <sup>+</sup> [This paper]	CF <sub>fk</sub>	1	$b/(b + n)$	$O(\ell^2 q^2 / 2^{2n})$	$(b + 2n)$

## 2 Preliminaries

In this section, we briefly discuss the notations and definitions used in this paper. We also state some existing basic results. We denote  $|S|$  as the cardinality of set  $S$ . Let  $x \xleftarrow{\$} S$  denote that  $x$  is chosen uniformly at random from  $S$ .  $[n]$  denotes the set of integers  $\{1, 2, \dots, n\}$ .  $(s)_n$  denotes the last  $n$  bit substring of  $b$  bit string  $s$ . Let  $M$  be a binary string over  $\{0, 1\}$ . Length of  $M$  in bits is denoted by  $|M|$ . When  $|M| \bmod b \neq 0$ , we pad  $10^d$  to  $M$  to make  $|M| \bmod b = 0$  where  $d = n - 1 - |M| \bmod b$  and  $b$  denotes the block length of  $M$ .  $M_1 || M_2 || \dots || M_l$  denotes the partition of message  $M$  after  $M$  is padded, where each  $M_i \in \{0, 1\}^b$  and  $l$  denotes the number of blocks of  $M$ .  $\ell$  denotes the maximum number of blocks in a message. By a  $q$ -set or a  $q$ -tuple  $x := (x_i : i \in I)$  for an index set  $I$ , we mean a set or a tuple of size  $q$ . When all elements  $x'_i s$  are distinct we write  $x \in \text{dist}_q$ .

**Random Functions.** Let  $\text{Func}(A, B)$  denote the set of all functions from  $A$  to  $B$ . A **random function**  $F$  is a function which is chosen from  $\text{Func}(A, B)$  following some distribution, not necessarily uniform. In particular, a function  $\rho_n$  is said to be a uniform random function, if  $\rho_n$  is chosen uniformly at random from the set of all functions from a specified finite domain  $\mathcal{D}$  to  $\{0, 1\}^n$ . Throughout the paper we fix a positive integer  $n$ . We will specify a uniform random function by performing *lazy sampling*. In lazy sampling, initially the function  $\rho$  is undefined at every point of its domain. We maintain a set  $\text{Dom}(\rho)$  that grows dynamically to keep the record of already defined domain points of  $\rho$ .  $\text{Dom}(\rho)$  is initialized to be empty. If  $x \notin \text{Dom}(\rho)$  then we will choose  $y \xleftarrow{\$} \{0, 1\}^n$  and add  $x$  in  $\text{Dom}(\rho)$ . In this regard,  $x$  is said to be *fresh*. On the other hand, if  $x \in \text{Dom}(\rho)$  (i.e.  $x = x'$ ) then  $y \leftarrow f(x')$ . In this regard  $x$  is said to be *covered*.

**Security Definitions.** We consider that an adversary  $\mathcal{A}$  is an oracle algorithm with access to its oracle  $\mathcal{O}(\cdot)$  and outputs either 1 or 0. Accordingly, we write  $\mathcal{A}^{\mathcal{O}(\cdot)} = 1$  or 0. The resource of  $\mathcal{A}$  is measured in terms of the time complexity  $t$  which takes into account the time it takes to interact with its oracle  $\mathcal{O}(\cdot)$  and the time for its internal computations, query complexity  $q$  takes into account the number of queries asked to the oracle by the adversary, data complexity  $\ell$  takes into account the maximum number of blocks in each query.



**Pseudo-Random Function.** We define **distinguishing advantage** of an oracle algorithm  $\mathcal{A}$  for distinguishing two random functions  $F$  from  $G$  as

$$\text{Adv}_{\mathcal{A}}(F ; G) := \Pr[\mathcal{A}^F = 1] - \Pr[\mathcal{A}^G = 1].$$

We define PRF-advantage of  $\mathcal{A}$  for an  $n$ -bit construction  $F$  by

$$\text{Adv}_F^{\text{prf}}(\mathcal{A}) := \text{Adv}_{\mathcal{A}}(F ; \rho_n).$$

We call  $\mathcal{A}$  a  $(q, \ell, t)$ -distinguisher if it makes at most  $q$  queries with at most  $\ell$ -blocks in each query and runs in time at most  $t$ . We write  $\text{Adv}_F^{\text{prf}}(q, \ell, t) = \max_{\mathcal{A}} \text{Adv}_F^{\text{prf}}(\mathcal{A})$  where maximum is taken over all  $(q, \ell, t)$ -distinguisher  $\mathcal{A}$ . In an information theoretic situation we also ignore the time parameter  $t$ . We call a keyed construction  $F$  is  $(q, \ell, \epsilon)$ -prf if  $\text{Adv}_F^{\text{prf}}(q, \ell) \leq \epsilon$ . Informally,  $F$  is called a secure PRF, if  $\epsilon$  is negligible,

**Collision-Free and Cover-Free.** Now we define some other information-theoretic security advantages (in which there is no presence of an adversary). Let  $H$  be a random function which outputs two  $n$  bit blocks, denoted by  $(\Sigma, \Theta) \in (\{0, 1\}^n)^2$ . For a  $q$ -tuple of distinct messages  $\mathcal{M} = (M^1, \dots, M^q)$ , we write  $H(M^i) = (\Sigma^i, \Theta^i)$ . For a  $q$ -tuple of pairs  $(\Sigma^i, \Theta^i)_i$ , we say that

1. A tuple  $(\Sigma^i, \Theta^i)_i$  is **collided** if  $\exists i, j \in [q]$  such that  $\Sigma^i = \Sigma^j$  and  $\Theta^i = \Theta^j$  for some  $j \neq i$ . Otherwise the tuple is said to be **collision-free**.
2. A tuple  $(\Sigma^i, \Theta^i)_i$  is **covered** if  $\exists i, j \in [q]$  such that  $\Sigma^i = (M^j_{\alpha})|_n$  and  $\Theta^i = Y^j_{\alpha-1}$  where  $\alpha \in [l_i]$  or  $\alpha \in [l_j]$  and  $j$  could be equal to  $i$ ,  $M^j_{\alpha}$  denotes the  $\alpha^{th}$  block of  $j^{th}$  message  $M^j$  and  $Y^j_{\alpha-1}$  is a  $n$  bit binary string that denotes the output of  $(\alpha - 1)^{th}$  block corresponding to  $j^{th}$  message  $M^j$ . Otherwise the tuple is said to be **cover-free**.

**Definition 1.** We define  $(q, \ell)$ -collision advantage and  $(q, \ell)$ -cover-free advantage as

$$\text{Adv}_F^{\text{coll}}(q, \ell) = \max_{M \in \text{dist}_q} \Pr[(\Sigma_i, \Theta_i)_i \text{ is not collision - free }].$$

$$\text{Adv}_F^{\text{cf}}(q, \ell) = \max_{M \in \text{dist}_q} \Pr[(\Sigma_i, \Theta_i)_i \text{ is not cover - free }].$$

Clearly,  $\text{Adv}_F^{\text{coll}}(q, \ell) \leq \frac{q^2}{2} \text{Adv}_F^{\text{coll}}(2, \ell)$ . Similarly,  $\text{Adv}_F^{\text{cf}}(q, \ell) \leq \frac{q^2}{2} \text{Adv}_F^{\text{cf}}(2, \ell)$ . So it would be sufficient to concentrate on a pair of messages while bounding collision free or cover-free advantages. We say that a construction  $F$  is  $(q, \ell, \epsilon)$ -xxx if  $\text{Adv}_F^{\text{xxx}}(q, \ell) \leq \epsilon$  where xxx denotes either **collision-free** or **cover-free**.

**Structure Graphs.** In this section, we briefly revisit the structure graph analysis [5,15]. A more formal analysis of structure graph is available in the full version [14] of this paper.

Consider a cascaded construction with a uniform random function  $f$ , that works on a message  $M = M_1 || M_2 || \dots || M_l$  of length  $l$  blocks as follows:

$$Y_0 = \mathbf{0}, \text{ and } Y_i = f(Y_{i-1}, M_i) \text{ for } i = 1, \dots, l,$$

where  $M_i$  is the  $i^{\text{th}}$  block of message  $M$ .

Informally, for a set of any two fixed distinct messages  $\mathcal{M} = \{M^1, M^2\}$  and a uniformly chosen random function  $f$ , we construct the structure graph  $\mathcal{G}^f(\mathcal{M})$  with  $\{0, 1\}^n$  as the set of nodes as follows. We follow the computations for  $M^1$  first, block by block, and next those of  $M^2$ . In the process of the computation of a message  $M$ , we create nodes labelled by the values  $y_i$  of the intermediate chaining variables  $Y_i$  where  $0 \leq i \leq l-1$  ( $l$  being the number of message blocks of  $M$ ) and edges  $(y_i, y_{i+1})$  which is labelled by the  $(i+1)^{\text{th}}$  block of message i.e.  $M_{i+1}$ . In this process, if we arrive at a vertex already labelled, while not following an existing edge, we call this event an  $f$ -collision.<sup>3</sup> The sequence of alternating vertices and edges corresponding to the computations for a message  $M^j$  is called an  $M^j$ -walk or more generally a message walk, denoted by  $W^j$ .

Let  $\mathcal{G}(\mathcal{M})$  denote the set of all structure graphs corresponding to the set of messages  $\mathcal{M}$  (by varying  $f$  over a function family). For a fixed graph  $G \in \mathcal{G}(\mathcal{M})$ , let  $f\text{Coll}(G)$  denote the set of all  $f$ -collisions in  $G$ . We state the following results.

**Proposition 1.** [15, Lemma 2] *For a fixed graph  $G$ ,  $\Pr_f[\mathcal{G}^f(\mathcal{M}) = G] \leq 2^{-n|f\text{Coll}(G)|}$ .*

**Proposition 2.**  $\Pr[G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}) : |f\text{Coll}(G)| \geq 3] \leq \frac{27\ell^6}{2^{3n}}$ , where  $\ell$  is the maximum number of blocks of the messages in  $\mathcal{M}$ .

Proof of Proposition 2 can be found in Appendix A.

It is to be noted that for CBC-MAC analysis [5],  $f(\alpha, \beta)$  is taken as  $\pi(\alpha \oplus \beta)$  and for the NI-MAC analysis [15],  $f(\alpha, \beta)$  is taken as  $\rho(\alpha || \beta)$ , where  $\pi$  is a random permutation over  $n$  bits and  $\rho$  is a random function from  $b+n$  bits to  $n$  bits, where  $b$  is the message block-length and  $n$  is the length of the chaining variable as well as the tag.

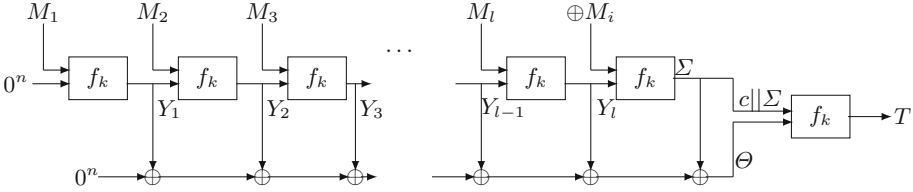
### 3 Proposed Construction of NI<sup>+</sup> for Beyond-Birthday Secure MAC

We present the schematic diagram of NI<sup>+</sup> in Fig. 1 followed by the description in Algorithm 1. Let  $f_k : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  be a keyed function from  $b+n$  bits to  $n$  bits where  $b > n$  where  $b$  refers to the block length of a message block and  $n$  refers to the output length in bits. Let  $M \in \{0, 1\}^{bl}$ . So we can write  $M = (M_1, M_2, \dots, M_l)$  where each  $M_i \in \{0, 1\}^b$ . We define a checksum block  $CS = \oplus_{i=1}^l M_i$ . We denote  $\mathbf{Casc}^{\text{fk}}(M) := f_{k_1}(\dots(f_k(f_k(0, M_1), M_2), \dots, M_l))$ . Output of  $\mathbf{Casc}^{\text{fk}}(M)$  and the checksum block  $CS$  is passed through the same function  $f_k$  and the output is denoted as  $\Sigma$ . We obtain  $\Theta$  by xoring all the

<sup>3</sup> We use the term collision and accident interchangeably.

intermediate chaining values (i.e.  $\oplus_{i=1}^l Y_i \oplus \Sigma$ ). We concatenate a fixed  $b - n$  bit string  $c = 10^{b-n-1}$  with the  $2n$  bit string  $\Sigma || \Theta$  to match the input size of  $f_k$  and then the entire concatenated  $b + n$  bit string (i.e.  $c || \Sigma || \Theta$ ) is passed through  $f_k$  and finally outputs the tag  $T$ . We sometimes denote  $CS$  by  $M_{l+1}$ .

Note that,  $NI^+$  is similar to that of NI up to  $\mathbf{Casc}^{fk}(M)$  except the following differences.



**Fig. 1.** Construction of  $NI^+$  MAC

```

Input:  $f_k : k \xleftarrow{\$} \mathcal{K}, M \leftarrow \{0, 1\}^*, c \leftarrow 10^{b-n-1}$ 
Output:  $T \in \{0, 1\}^n$ 
1  $M_1 || M_2 || \dots || M_l \leftarrow M || 10^*$ ; //  $l$  is the number of message blocks in  $M$ 
2  $Z \leftarrow 0^n; Y \leftarrow 0^n$ ;
   for  $i = 1$  to  $l$  do
3   |  $Y \leftarrow f_k(M_i, Y); Z \leftarrow Z \oplus Y$ ;
   end
4  $CS \leftarrow \oplus_{i=1}^l M_i$ ;
5  $Y \leftarrow f_k(CS, Y); Z \leftarrow Z \oplus Y; \Sigma \leftarrow Y; \Theta \leftarrow Z$ ;
6 Return  $T \leftarrow f_k(c || \Sigma, \Theta)$ ;
    
```

**Algorithm 1.** Algorithm for  $NI^+$  MAC

(a) In NI construction,  $b$ -bit encoding of  $|M|$  and the last message block output  $Y_\ell$  is passed through a different keyed compression function  $f_{k_2}$ . In  $NI^+$ , we substitute the  $b$ -bit length encoding by the checksum block  $CS$ . Moreover,  $CS$  and  $Y_\ell$  is passed through the same keyed compression function.

(b) NI is a two fixed-keyed compression function based MAC.  $NI^+$  is a single fixed-keyed compression function based MAC.

(c) NI provides only birthday bound ( $\ell q^2 / 2^n$ ) security.  $NI^+$  provides beyond birthday bound security ( $q^2 \ell^2 / 2^{2n}$ ) when  $\ell \leq 2^{n/4}$ .

Schematic diagram of NI is given in Appendix B.

**Remark 1.** We note that the beyond birthday security is not possible to achieve if we just keep the original structure of NI-MAC and output  $\Sigma$  as the last block output (i.e.  $\Sigma = f_{K_2}(|M|, Y_l)$ ) and  $\Theta$  as the sum of all intermediate chaining variables (i.e.  $\Theta = \oplus_{i=1}^l Y_i \oplus \Sigma$ ) as the birthday bound attack is followed from Prennneel and Oorschot’s attack [34].

## 4 Security Analysis of NI<sup>+</sup>-MAC

Gaži et al. in [15] have shown that the advantage of NI-MAC is bounded above by  $\frac{q^2}{2^n} \left( \ell + \frac{64\ell^4}{2^n} \right)$ . In this section we analyze the advantage of our construction NI<sup>+</sup>-MAC and show that the advantage of NI<sup>+</sup>-MAC achieves beyond birthday bound security; better than that of NI-MAC. Thus we have the following theorem.

**Theorem 1.** *Let  $f : \{0, 1\}^k \times \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a  $(\epsilon, t, q)$  secure PRF. Then NI<sup>+</sup> be a  $(\epsilon', t', q, \ell)$  secure PRF, where*

$$\epsilon' \leq \epsilon + \frac{q}{2^n} + \frac{q^2}{2^{2n}} + \frac{q^2\ell^2}{2^{2n}} + \frac{q^2\ell^4}{2^{3n}} + \frac{27q^2\ell^6}{2^{3n}},$$

such that  $t = t' + \tilde{O}(\ell q)$ . Moreover, if  $\ell \leq 2^{n/4}$  then,

$$\epsilon' \leq \epsilon + \frac{q}{2^n} + \frac{q^2\ell^2}{2^{2n}}.$$

*Proof.* Let  $\mathcal{A}$  be a adaptive PRF-adversary against NI<sup>+</sup> running in time  $t$  and asking at most  $q$  queries, each of length at most  $\ell$  blocks. NI<sup>+</sup> uses a single keyed function  $f_k$ . Now if we replace  $f_k$  by a uniformly distributed random function  $r$  such that  $r \xleftarrow{\$} \text{Func}(\{0, 1\}^b \times \{0, 1\}^n, \{0, 1\}^n)$  and call the resulting construction NI <sub>$r$</sub> <sup>+</sup>, then using the standard reduction from information theoretic setting to complexity theoretic setting we have,

$$\text{Adv}_{\text{NI}^+}^{\text{prf}} \leq \epsilon + \text{Adv}_{\text{NI}_r^+}^{\text{prf}}.$$

Therefore to prove Theorem 1, we only need to prove

$$\text{Adv}_{\text{NI}_r^+}^{\text{prf}} \leq \frac{q}{2^n} + \frac{q^2}{2^{2n}} + \frac{q^2\ell^2}{2^{2n}} + \frac{q^2\ell^4}{2^{3n}} + \frac{27q^2\ell^6}{2^{3n}}.$$

Consider the following Game as shown in Algorithm 2 where the adversary  $\mathcal{A}$  queries to oracle  $O$  with distinct messages  $M^i$  and obtains the response  $T^i$ .

Note that Game  $G_0$  truly simulates a uniform random function and  $G_1$  simulates the actual construction NI <sub>$r$</sub> <sup>+</sup>. Therefore using the fundamental lemma of game-playing technique [7], we have the following.

$$\begin{aligned} \text{Adv}_{\text{NI}_r^+}^{\text{prf}} &= |\Pr[\mathcal{A}^{G_1} = 1] - \Pr[\mathcal{A}^{G_0} = 1]| \\ &\leq \Pr[\mathcal{A}^{G_1} \text{ sets } \mathbf{badsigma} \vee \mathcal{A}^{G_1} \text{ sets } \mathbf{bad}] \\ &\leq \Pr[\mathcal{A}^{G_1} \text{ sets } \mathbf{badsigma}] + \Pr[\mathcal{A}^{G_1} \text{ sets } \mathbf{bad}]. \end{aligned} \quad (1)$$

Therefore, we evaluate now the probability  $\Pr[\mathcal{A}^{G_1} \text{ sets } \mathbf{bad}]$ . To evaluate this, let us define a double block function  $\mathcal{H}_f(M) := (\Sigma, \Theta)$  with respect to a uniform random function  $f$ . Recall that the tuple  $\mathcal{H}_f(M^i) := (\Sigma^i, \Theta^i)_i, \forall i \in [q]$  is said to be collision-free if  $\forall i$ , either  $\Sigma^i \neq \Sigma^j$  or  $\Theta^i \neq \Theta^j$  or both  $\forall j \neq i$ . Similarly, the

```

1 initialize: badsigma, bad  $\leftarrow$  false;
2 On the  $j^{\text{th}}$  query  $M^j$ ;
3  $M_1^j || M_2^j || \dots || M_l^j \leftarrow M^j || 10^*$   $\leftarrow$  Partition( $M^j$ ),  $Y_0 = 0$ ;
4 for  $i = 1$  to  $l$ ;
5   if  $((M_i^j, Y_{i-1}^j) \in \text{Dom}(f))$   $Y_i^j \leftarrow f(M_i^j, Y_{i-1}^j)$ ;
6   Else  $Y_i^j \leftarrow \{0, 1\}^n$  and  $f(M_i^j, Y_{i-1}^j) \leftarrow Y_i^j$ ;
7    $\text{Dom}(f) \leftarrow \text{Dom}(f) \cup (M_i^j, Y_i^j)$ ;
8 if  $((\oplus_{i=1}^l M_i^j, Y_l^j) \in \text{Dom}(f))$   $Y_{l+1}^j \leftarrow f(\oplus_{i=1}^l M_i^j, Y_l^j)$ ;
9 Else  $Y_{l+1}^j \leftarrow \{0, 1\}^n$  and  $f(\oplus_{i=1}^l M_i^j, Y_l^j) \leftarrow Y_{l+1}^j$ ;
10  $\text{Dom}(f) \leftarrow \text{Dom}(f) \cup (\oplus_{i=1}^l M_i^j, Y_l^j)$ ;
11  $\Sigma^j \leftarrow Y_{l+1}^j$ ,  $\Theta^j \leftarrow \oplus_{i=1}^{l+1} Y_i^j$ ;
12 if  $(\Sigma^j = 0)$  badsigma  $\leftarrow$  true;
13  $T^j \xleftarrow{\$} \{0, 1\}^n$ ;
14 if  $((\Sigma^j, \Theta^j) = (\Sigma^i, \Theta^i))$  for some  $i \in \{1, 2, \dots, j-1\}$ , or  $(c || \Sigma^j, \Theta^j) = (M_s^*, Y_{s-1}^*)$  such that  $s \in [l_i + 1]$  or  $s \in [l_j + 1]$ ,  $* \in \{i, j\}$ ;
15   if (bad);
16     Coll( $i, j$ )  $\leftarrow$  true, bad  $\leftarrow$  true;
17     if  $((\Sigma^j, \Theta^j) = (\Sigma^i, \Theta^i))$   $T^j \leftarrow f(\Sigma^i, \Theta^i)$ ;
18     Else  $T^j \leftarrow f(M_s^*, Y_{s-1}^*)$ ;
19 Return  $T^j$ ;

```

**Algorithm 2.** Game  $G_0$  is without boxed statement and  $G_1$  is with boxed statement.

tuple  $(\Sigma^i, \Theta^i)_i$  is said to be cover-free if  $\forall i$ , either  $\Sigma^i \neq (M_\alpha^j)_{|n}$  or  $\Theta^i \neq Y_{\alpha-1}^j$  or both. Therefore, it is then easy to see that,

$$\begin{aligned} \Pr[A^{G_1} \text{ sets } \mathbf{bad}] &\leq \mathbf{Adv}_H^{\text{coll}}(q, \ell) + \mathbf{Adv}_H^{\text{cf}}(q, \ell) \\ &\leq \frac{q^2}{2} (\mathbf{Adv}_H^{\text{coll}}(2, \ell) + \mathbf{Adv}_H^{\text{cf}}(2, \ell)). \end{aligned} \quad (2)$$

Now we state the following lemma, the first three cases of which bound the collision-free advantage and the last three cases bound the cover-free advantage of function  $H_f(\cdot)$ .

**Notation:** Let  $E_{\text{coll}}$  denotes the collision event (i.e.  $\Sigma^i = \Sigma^j \wedge \Theta^i = \Theta^j$ ) and  $E_{\text{cf}}$  denotes the covered event (i.e.  $\Sigma^i = x \wedge \Theta^i = Y_t^s$ ) for some  $n$  bit constant  $x$  and  $s \in [q], t \in [l_s]$ .  $W^i$  denotes the walk graph corresponding to message  $M^i$ .  $\mathbf{Y}$  denotes the vector of intermediate computations (i.e.  $(Y_1, Y_2, \dots, Y_l)$ ) for a  $l$  block message.  $l_i$  and  $l_j$  denote the message length in number of blocks of  $M^i$  and  $M^j$  respectively. When  $M^i$  is not a prefix of  $M^j$  or  $M^j$  is not a prefix of  $M^i$ ,  $p$  denotes longest common prefix (LCP) of  $M^i$  and  $M^j$ . That means  $M_{p+1}^i \neq M_{p+1}^j$  and  $M_\alpha^i = M_\alpha^j$  where  $1 \leq \alpha \leq p$ . Let  $\mathcal{G}(\mathcal{M}^i, \mathcal{M}^j)$  denotes the set of all structure graphs corresponding to two fixed messages  $\mathcal{M}^i$  and  $\mathcal{M}^j$ .  $\mathcal{G}^a \subset \mathcal{G}(\mathcal{M}^i, \mathcal{M}^j)$  be the set of all structure graphs with accident  $a$  where, in this

paper, we consider  $a = 0, 1, 2$ . Moreover, when  $a = 1$  or  $2$  we denote  $\mathcal{G}_{nl}^a \subset \mathcal{G}_a$  be the set of all structure graphs such that none of the two message walks  $W^i, W^j$  contains a loop.  $\mathcal{G}_l^a$  denotes the set of all remaining structure graphs. Moreover,  $\mathcal{G}^a = \mathcal{G}_{nl}^a \sqcup \mathcal{G}_l^a$  for  $a = 1, 2$ .

**Lemma 1.** *Let us consider  $G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}^i, \mathcal{M}^j)$ , where  $M^i$  and  $M^j$  are any two distinct messages, each of length at most  $\ell$  blocks and a particular  $n$  bit constant  $x$ , we have the followings.*

$$\text{Case (A): } \Pr[E_{\text{coll}} \wedge |fColl(G)| = 0] \leq \frac{1}{2^{2n}}.$$

$$\text{Case (B): } \Pr[E_{\text{coll}} \wedge |fColl(G)| = 1] \leq \frac{\ell^2}{2^{2n}}.$$

$$\text{Case (C): } \Pr[E_{\text{coll}} \wedge |fColl(G)| = 2] \leq \frac{\ell^4}{2^{3n}}.$$

$$\text{Case (D): } \Pr[E_{\text{cf}} \wedge |fColl(G)| = 0] \leq \frac{1}{2^{2n}}.$$

$$\text{Case (E): } \Pr[E_{\text{cf}} \wedge |fColl(G)| = 1] \leq \frac{\ell^2}{2^{2n}}.$$

$$\text{Case (F): } \Pr[E_{\text{cf}} \wedge |fColl(G)| = 2] \leq \frac{\ell^4}{2^{3n}}.$$

We present the proofs of one case for the collision-free advantage, namely, Case (A) in Appendix C and one case for the cover-free advantage, namely, Case (D) in Appendix D. The complete proof is available in the full version [14] of this paper.

**Resume the proof of Theorem 1:** Now we have all the materials to prove Theorem 1. It is easy to see the followings two results.

$$\text{Adv}_{\text{H}}^{\text{coll}}(2, \ell) \leq \sum_{k=0}^2 \Pr[E_{\text{coll}} \wedge |fColl(G)| = k] + \Pr[|fColl(G)| \geq 3],$$

$$\text{Adv}_{\text{H}}^{\text{cf}}(2, \ell) \leq \sum_{k=0}^2 \Pr[E_{\text{cf}} \wedge |fColl(G)| = k] + \Pr[|fColl(G)| \geq 3].$$

Therefore, we have the following results.

$$\text{Adv}_{\text{H}}^{\text{coll}}(2, \ell) \leq \frac{1}{2^{2n}} + \frac{\ell^2}{2^{2n}} + \frac{\ell^4}{2^{3n}} + \frac{27\ell^6}{2^{3n}}, \quad (3)$$

$$\text{Adv}_{\text{H}}^{\text{cf}}(2, \ell) \leq \frac{1}{2^{2n}} + \frac{\ell^2}{2^{2n}} + \frac{\ell^4}{2^{3n}} + \frac{27\ell^6}{2^{3n}}. \quad (4)$$

Equation (3) follows from Case (A),(B) and (C) of Lemma 1. Similarly, Eq. (4) follows from Case (D),(E) and (F) of Lemma 1.

Substituting Eqs. (3) and (4) into Eq. (2) we obtain

$$\Pr[\mathcal{A}^{G_1} \text{ sets bad}] \leq \frac{q^2}{2^{2n}} + \frac{q^2\ell^2}{2^{2n}} + \frac{q^2\ell^4}{2^{3n}} + \frac{27q^2\ell^6}{2^{3n}}.$$

Moreover it is easy to see that  $\Pr[\mathcal{A}^{G_1} \text{ sets badsigma}] \leq \frac{q}{2^n}$ . Substituting these two probability expressions back to Eq. (1) gives the required bound.  $\square$

## 5 Conclusion

In this paper, we have proposed a single fixed-key compression function based MAC NI<sup>+</sup>, a variant of NI-MAC, that achieves BBB security with less number of keys than NI-MAC. Our construction is different from Yasuda’s proposed single-fixed key compression function based MAC construction in which we have been able to slightly reduce the state size from  $2(b + 2n)$  bits to  $(b + 2n)$  bits which was posed as an open problem in [44] to reduce the state size to  $2n$  bits.

**Acknowledgements.** The authors are thankful to the Project *Centre of Excellence in Cryptology* (CoEC) of Indian Statistical Institute for partial support towards this research work.

## A Proof of Proposition 2

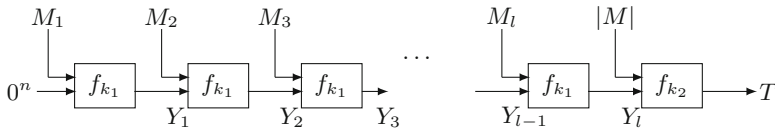
We prove that  $\Pr[G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}) : |fColl(G)| \geq 3] \leq \frac{27\ell^6}{2^{3n}}$ , where  $\ell$  is the total number of blocks of the messages in  $\mathcal{M}$  where  $\ell \leq 2^{n/2}$ .

$$\begin{aligned} \Pr[G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}) : |fColl(G)| \geq 3] &= \sum_{i=3}^{\infty} \Pr[G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}) : |fColl(G)| = i] \\ &\leq \sum_{i=3}^{\infty} \sum_{H \in \mathcal{G}^i(\mathcal{M})} \Pr[G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}) : G = H] \leq \sum_{i=3}^{\infty} \sum_{H \in \mathcal{G}^i(\mathcal{M})} \frac{1}{2^{in}} \text{ (from Proposition 1)} \\ &\leq \sum_{i=3}^{\infty} \frac{|\mathcal{G}^i(\mathcal{M})|}{2^{in}}. \end{aligned}$$

Now, note that the a graph  $G$  is uniquely determined by its number of collisions. Therefore,  $|\mathcal{G}^i(\mathcal{M})| \leq \left(\frac{2\ell(2\ell+1)}{2}\right)^i \leq (3\ell^2)^i$ . Now let  $a$  denotes  $\frac{3\ell^2}{2^n}$ . Assuming  $\ell \leq 2^{n/2}$ , we can write,  $\Pr[G \stackrel{\$}{\leftarrow} \mathcal{G}(\mathcal{M}) : |fColl(G)| \geq 3] \leq \frac{a^3}{(1-a)} \leq \frac{27\ell^6}{2^{3n}}$ .  $\square$

## B Diagram of NI

See Fig. 2



**Fig. 2.** Construction of NI-MAC

### C Proof of Lemma 1, Case (A)

We provide the proof of  $\Pr[E_{\text{coll}} \wedge |f\text{Coll}(G)| = 0] \leq \frac{1}{2^{2n}}$  here. We fix a structure graph  $H \in \mathcal{G}^0$  and then analyse the probability of the event  $E_{\text{coll}}$  with respect to  $H$  in a case-by-case basis.

**Case (A.1)** When  $M^i$  or  $M^j$  is not a prefix of each other, we recall that  $p$  be the LCP of  $M^i$  and  $M^j$ . Therefore, all  $Y_\alpha^i$  and  $Y_\beta^j$  are distinct where  $p + 1 \leq \alpha \leq l_i, p + 1 \leq \beta \leq l_j$ . Moreover,  $Y_\alpha^i \neq Y_\alpha^j, p + 1 \leq \alpha \leq \min\{l_i, l_j\}$  as the number of collisions in  $H$  is 0. Therefore, we have,

$$\Pr[E_{\text{coll}} \wedge G = H] = \Pr[\Theta^i = \Theta^j \wedge G = H | \Sigma^i = \Sigma^j] \cdot \Pr[\Sigma^i = \Sigma^j].$$

It is obvious that  $\Pr[\Sigma^i = \Sigma^j] \leq \frac{1}{2^{n-2\ell}}$  and the event  $\Theta^i = \Theta^j \wedge G = H$  conditioned on the event  $\Sigma^i = \Sigma^j$  implies a non trivial equation on  $\mathbf{Y}$  as we will obtain  $Y_{p+1}^i$  and  $Y_{p+1}^j$  for which  $\Theta^i \oplus \Theta^j = 0$  would become non-trivial. Thus,  $\Pr[\Theta^i = \Theta^j \wedge G = H | \Sigma^i = \Sigma^j] \leq \frac{1}{2^{n-2\ell}}$ . Therefore,  $\Pr[E_{\text{coll}} \wedge G = H] \leq \frac{1}{2^{2n}}$ , assuming  $\ell \leq 2^{n-1}$ .

**Case (A.2)** Consider either of the two messages is a prefix of other (w.l.o.g  $M^j$  is a prefix of  $M^i$ ). Since  $l_i > l_j$  therefore,  $p = l_j$ . Since the number of collision in  $H$  is 0,  $Y_{p+1}^i, \dots, Y_{l_i}^i$  are all distinct with each other and with  $Y_1^j, \dots, Y_{l_j}^j$ . This implies that  $Y_{l_i}^i \neq Y_{l_j}^j$  as depicted in Fig. 3. Therefore, the probability of  $\Theta^i = \Theta^j \wedge G = H$  conditioned on the event  $\Sigma^i = \Sigma^j$  will be  $O(1/2^n)$  as we will obtain two random variables  $Y_{l_i}^i$  and  $Y_{l_j}^j$  for which  $\Theta^i \oplus \Theta^j = 0$  would become non-trivial. Moreover,  $\Pr[\Sigma^i = \Sigma^j] \leq \frac{1}{2^n}$ . Therefore again,  $\Pr[E_{\text{coll}} \wedge G = H] \leq \frac{1}{2^{2n}}$ .

Since,  $\mathcal{G}^0 = 1$ , we have,  $\Pr[E_{\text{coll}} \wedge |f\text{Coll}(G)| = 0] \leq \frac{1}{2^{2n}}$ .  $\square$



Fig. 3. Structure graph with 0 accident

### D Proof of Lemma 1, Case (D)

We present the proof of  $\Pr[E_{\text{cf}} \wedge |f\text{Coll}(G)| = 0] \leq \frac{1}{2^{2n}}$  here. We fix a structure graph  $H \in \mathcal{G}^0$  and then analyse the probability of the event  $E_{\text{cf}}$  with respect to  $H$  in a case-by-case basis.

**Case (i)** Let  $p$  be the LCP of  $M^i$  and  $M^j$ . Therefore,  $Y_\alpha^i = Y_\alpha^j$  where  $1 \leq \alpha \leq p$  and  $Y_\beta^i \neq Y_\beta^j$  where  $p + 1 \leq \beta \leq \min\{l_i, l_j\}$  as the number of accident in  $H$  is 0. Moreover, if  $l_i > l_j$  then all  $Y_\beta^i$  would have been distinct as  $|f\text{Coll}(G)| = 0$  where  $l_j + 1 \leq \beta \leq l_i$ . Note that, it is also true that  $Y_{l_i}^i \neq Y_{l_j}^j$ . Therefore, we have the following set of equations:

$$Y_{l_i+1}^i = x, \tag{5}$$

$$Y_1^i \oplus Y_2^i \oplus \dots \oplus Y_{l_i+1}^i + Y_t^s = 0, \tag{6}$$



where  $s$  could be either  $i$  or  $j$  and  $t \in [l_i + 1]$  or  $t \in [l_j + 1]$ . For each of these cases one can easily check that the above system of equation has rank 2. Therefore,  $\Pr[E_{cf} \wedge G = H] \leq \frac{1}{2^{2n}}$ .

**Case (ii).** Without loss of generality let us consider that  $M^j$  is a prefix of  $M^i$ . Since  $l_i > l_j$  therefore,  $p = l_j$ . Since, number of collisions in  $H$  is 0,  $Y_{p+1}^i, \dots, Y_{l_i}^i$  are all distinct with each other and with  $Y_1^j, \dots, Y_{l_j}^j$ . This implies that  $Y_{l_i}^i \neq Y_{l_j}^j$  as depicted in Fig. 3. Therefore, the set of equations (Eqs. (5) and (6)) has the full rank. Therefore, again we have,  $\Pr[E_{cf} \wedge G = H] \leq \frac{1}{2^{2n}}$ .

Therefore from the above two cases we have,  $\Pr[E_{cf} \wedge G = H] \leq \frac{1}{2^{2n}}$  for any non-zero  $n$  bit constant  $x$ . Moreover  $|\mathcal{G}^0| \leq 1$ . So  $\Pr[E_{cf} \wedge |fColl(G)| = 0] \leq \frac{1}{2^{2n}}$ .

## References

1. An, J.H., Bellare, M.: Constructing vil-macs from fil-macs: message authentication under weakened assumptions. In: Wiener [40], pp. 252–269
2. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
3. Bellare, M., Goldreich, O., Krawczyk, H.: Stateless evaluation of pseudorandom functions: security beyond the birthday barrier. In: Wiener [40], pp. 270–287
4. Bellare, M., Kilian, J., Rogaway, P.: The security of cipher block chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
5. Bellare, M., Pietrzak, K., Rogaway, P.: Improved security analyses for CBC macs. In: Shoup [35], pp. 527–545
6. Bellare, M.: New proofs for NMAC and HMAC: security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
7. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay [36], pp. 409–426
8. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: fast and secure message authentication. In: Wiener [40], pp. 216–233
9. Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Knudsen [22], pp. 384–397
10. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., 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)
11. Datta, N., Dutta, A., Nandi, M., Paul, G., Zhang, L.: One-key double-sum MAC with beyond-birthday security. Cryptology ePrint Archive, Report 2015/958 (2015). <http://eprint.iacr.org/>
12. Dodis, Y., Ristenpart, T., Steinberger, J., Tessaro, S.: To hash or not to hash again? (In)Differentiability results for  $H^2$  and HMAC. In: Canetti, R., Safavi-Naini, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 348–366. Springer, Heidelberg (2012)
13. Dodis, Y., Steinberger, J.: Domain extension for MACs beyond the birthday barrier. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 323–342. Springer, Heidelberg (2011)

14. Dutta, A., Nandi, M., Paul, G.: One-Key Compression Function Based MAC with Security beyond Birthday Bound. Cryptology ePrint Archive, Report 2015/1016, 20 October 2015. <http://eprint.iacr.org/>
15. Gaži, P., Pietrzak, K., Rybár, M.: The exact PRF-security of NMAC and HMAC. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 113–130. Springer, Heidelberg (2014)
16. Gaži, P., Pietrzak, K., Tessaro, S.: Generic security of NMAC and HMAC with input whitening. Cryptology ePrint Archive, Report 2015/881, 2015. <http://eprint.iacr.org/>
17. Hong, D., Sung, J., Hong, S.H., Lim, J.-I., Lee, S.-J., Koo, B.-S., Lee, C.-H., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J.-S., Chee, S.: 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)
18. Iwata, T., Kurosawa, K.: OMAC: one-key CBC MAC. In: Johansson [20], pp. 129–153
19. Jaulmes, É., Joux, A., Valette, F.: On the security of randomized CBC-MAC beyond the birthday paradox limit: a new construction. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 237–251. Springer, Heidelberg (2002)
20. Johansson, T. (ed.): FSE 2003. LNCS, vol. 2887. Springer, Heidelberg (2003)
21. Joux, A., Poupard, G., Stern, J.: New attacks against standardized macs. In: Johansson [20], pp. 170–181
22. Knudsen, L.R. (ed.): EUROCRYPT 2002. LNCS, vol. 2332. Springer, Heidelberg (2002)
23. Kobitz, N., Menezes, A.: Another look at HMAC. *J. Math. Cryptology* **7**(3), 225–251 (2013)
24. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997
25. Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable blockciphers with beyond birthday-bound security. In: Canetti, R., Safavi-Naini, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 14–30. Springer, Heidelberg (2012)
26. Leurent, G., Peyrin, T., Wang, L.: New generic attacks against hash-based MACs. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 1–20. Springer, Heidelberg (2013)
27. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
28. Maurer, U.M., Sjödin, J.: Domain expansion of MACs: alternative uses of the FIL-MAC. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 168–185. Springer, Heidelberg (2005)
29. Maurer, U.M., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
30. Minematsu, K.: How to thwart birthday attacks against MACs via small randomness. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 230–249. Springer, Heidelberg (2010)
31. Naito, Y., Sasaki, Y., Wang, L., Yasuda, K.: Generic state-recovery and forgery attacks on ChopMD-MAC and on NMAC/HMAC. In: Sakiyama, K., Terada, M. (eds.) IWSEC 2013. LNCS, vol. 8231, pp. 83–98. Springer, Heidelberg (2013)
32. Thomas Peyrin, Y., Sasaki, L.W.: Generic related-key attacks for HMAC. In: Wang and Sako [37], pp. 580–597

33. Peyrin, T., Wang, L.: Generic universal forgery attack on iterative hash-based MACs. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 147–164. Springer, Heidelberg (2014)
34. Preneel, B., van Oorschot, P.C.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
35. Shoup, V. (ed.): CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
36. Vaudenay, S. (ed.): EUROCRYPT 2006. LNCS, vol. 4004. Springer, Heidelberg (2006)
37. Wang, X., Sako, K. (eds.): ASIACRYPT 2012. LNCS, vol. 7658. Springer, Heidelberg (2012)
38. Wang, X., Yin, Y.L., Hongbo, Y.: Finding collisions in the full SHA-1. In: Shoup [35], pp. 17–36
39. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
40. Wiener, M. (ed.): CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)
41. Yasuda, K.: Boosting merkle-damgård hashing for message authentication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 216–231. Springer, Heidelberg (2007)
42. Yasuda, K.: Multilane HMAC— security beyond the birthday limit. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 18–32. Springer, Heidelberg (2007)
43. Yasuda, K.: “Sandwich” is indeed secure: how to authenticate a message with just one hashing. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 355–369. Springer, Heidelberg (2007)
44. Yasuda, K.: A one-pass mode of operation for deterministic message authentication— security beyond the birthday barrier. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 316–333. Springer, Heidelberg (2008)
45. Yasuda, K.: A double-piped mode of operation for MACs, PRFs and PROs: security beyond the birthday barrier. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 242–259. Springer, Heidelberg (2009)
46. Yasuda, K.: HMAC without the “Second” key. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 443–458. Springer, Heidelberg (2009)
47. Yasuda, K.: The sum of CBC MACs is a secure PRF. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 366–381. Springer, Heidelberg (2010)
48. Yasuda, K.: A new variant of PMAC: beyond the birthday bound. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 596–609. Springer, Heidelberg (2011)
49. Yasuda, K.: On the full MAC security of a double-piped mode of operation. IEICE Trans. **94–A**(1), 84–91 (2011)
50. Yasuda, K.: A parallelizable prf-based MAC algorithm: well beyond the birthday bound. IEICE Trans. **96**(1), 237–241 (2013)
51. Zhang, L., Wenling, W., Sui, H., Wang, P.: 3kf9: enhancing 3GPP-MAC beyond the birthday bound. In: Wang and Sako [37], pp. 296–312

# **Cloud Storage Security**

# Towards Efficient Fully Randomized Message-Locked Encryption

Tao Jiang<sup>1,4</sup>, Xiaofeng Chen<sup>1(✉)</sup>, Qianhong Wu<sup>2</sup>, Jianfeng Ma<sup>1</sup>, Willy Susilo<sup>3</sup>,  
and Wenjing Lou<sup>4</sup>

<sup>1</sup> State Key Laboratory of Integrated Service Networks (ISN),  
Xidian University, Xi'an, China

jiangt2009@gmail.com, xfchen@xidian.edu.cn, jfma@mail.xidian.edu.cn

<sup>2</sup> School of Electronic and Information Engineering, Beihang University,  
Beijing, China

qianhong.wu@buaa.edu.cn

<sup>3</sup> Centre for Computer and Information Security Research,  
School of Computing and Information Technology, University of Wollongong,  
Wollongong, Australia

wsusilo@uow.edu.au

<sup>4</sup> Department of Computer Science,  
Virginia Polytechnic Institute and State University, Blacksburg, VA, USA  
wjlou@vt.edu

**Abstract.** Cross-user data deduplication will greatly reduce the storage cost of storage service provider. Motivated by secure data deduplication, Abadi et al. extended the work Message-Locked Encryption (MLE) and introduced the primitive of MLE2 with nice security properties. However, their fully randomized scheme (R-MLE2) requires the inefficient equality-testing algorithm to identify all duplicate ciphertexts. Thus, an interesting open problem is how to reduce the overhead of R-MLE2 and propose an efficient construction for R-MLE2. In this paper, we introduce a new primitive called  $\mu$ R-MLE2, which gives a partial positive answer to this open problem. Our main trick is to use the client-assistant way based on *static* or *dynamic* decision trees. The advantage gained from it is that by interacting with clients, the server will reduce the time complexity of deduplication equality test from linear time to efficient logarithmic time over the whole database items.

**Keywords:** Deduplication · Convergent encryption · Message-locked encryption · Interactive protocol

## 1 Introduction

With the rapid growing of cloud storage service, such as cloud storage [8, 9, 17], encryption becomes an important technique for protecting the confidentiality of data. Although data encryption provides an important guarantee over the security and privacy of clients' data, it limits the manners of the accessibility

and availability of the encrypted data. Thus, it is important to design efficient scheme to support secure and efficient computation outsourcing [5, 6] and storage outsourcing [7]. Data deduplication enables cloud data storage systems to find and remove duplicate data without compromising its availability. The goal of data deduplication is to store more data in less space by storing and maintaining files (blocks in fine-grained deduplication manner) into a single copy, where the redundant copies of data are replaced by a reference to this copy. It means that, data deduplication storage system could reduce the storage size of  $u$  clients, who share the same data copy  $m$ , from  $\mathcal{O}(u \cdot |m|)$  to  $\mathcal{O}(u + |m|)$  if some implementation-dependent constants are hidden. Also, clients do not need to upload their data to the cloud storage server when there has been one copy stored, which will not only greatly reduce the communication cost of clients and cloud server, but also save the network bandwidth.

When the data from different clients is encrypted with their private secret keys, it is difficult to conduct ciphertext data deduplication among clients. A secure cross-client deduplication scheme should enable a storage server to detect data deduplication over the data encrypted by different clients, while efficiently prevent the practical attacks [10, 16, 19] from poor deduplication scheme. Douceur et al. [7] proposed the first solution for secure and efficient data deduplication, and they call it *convergent encryption*. This idea promoted many significant applications, where various schemes [3, 12] are implemented or designed based on convergent encryption. Recently, Bellare et al. [4] define a new primitive, Message-Locked Encryption (MLE), which brought rigor to security deduplication, and captured various security aspects of MLE. Also, they constructed several schemes and provided some detailed analysis over them. To strengthen the notions of security by considering plaintext distributions depend on the public parameter, Abadi et al. [1] proposed two approaches (fully random scheme and deterministic scheme) that are secure even for lock-dependent message in realistic. It answered the question: Can message-locked encryption be secure for lock-dependent message? The tag randomization design makes the fully random scheme, R-MLE2 for short, satisfy the standard secure notion of data confidentiality. Also, the overhead in the length of the ciphertext is only additive and independent of the message length.

However, as the open problem described in [1], the R-MLE2 scheme is not efficient in the deduplication process because of the comparison of the randomized tag introduced. It is important to maintain tags for sub-linear deduplication time, since for large data sets linear scans are prohibitive, particularly if they involve a linear number of cryptographic operations. In this paper, we ask whether the R-MLE2 scheme can be much more efficient (with logarithmic or nearly logarithmic deduplication test overhead) in data deduplication for large database while also keep the security properties of the deduplication scheme? We adopt client-server interaction based on random decision tree, mutable tree and self-generation tree to improve the efficiency of our schemes, and design two (static/dynamic) efficient R-MLE2 schemes ( $\mu$ R-MLE2). Both of the designed schemes support efficient data equality test while keeping the security of clients' data by allowing a small number of interactions.

## 1.1 Related Works

Convergent encryption [7] ensures data privacy in deduplication. It is a *deterministic* scheme, where a ciphertext  $C = E(k, m)$  is an encryption over message  $m$  under a message-dependent key  $k = h(m)$ , where  $h$  is a cryptographic hash function and  $E$  is a block cipher. In the deterministic scheme, identical plaintexts will be mapped to one ciphertext. When a client uploads the encrypted plaintext to a server, the server can find the duplicate ciphertext and store only one copy of each data. In this cross-user secure deduplication scheme, the clients need not to coordinate their actions or consider the existence of other clients who hold the same data copy.

Bellare et al. [4] formalized this primitive as message-locked encryption, and explored its application in space-efficient secure outsourced storage. An MLE scheme  $\mathcal{MLE} = (\text{P}, \text{K}, \text{E}, \text{D}, \text{T})$  is composed of five polynomial time algorithms. In  $\mathcal{MLE}$ , the parameter generation algorithm  $\text{P}$  is used to generate the public parameter. The key generation algorithm  $\text{K}$  is used to generate the message-derived key. On inputting a key and a message the encryption algorithm  $\text{E}$  outputs the ciphertext. The decryption algorithm  $\text{D}$  reverses the process, whose output is used to compute the ciphertext/plaintext, and the tag generation algorithm  $\text{T}$  is used to generate the tag of the ciphertext. In the scheme, tag generation maps the ciphertext to a tag and identical plaintext result in one equal tag.

To enhance the security of deduplication and protect the data confidentiality, Bellare et al. [3] showed how to protect the data confidentiality by transforming the predictable message into an unpredictable message. In their system, a third party called key server is introduced to generate the file tag for duplicate check. Recently, Liu et al. [14] designed a secure deduplication scheme without additional independent servers. Li et al. [12] addressed the key management issue in block-level deduplication by distributing these keys across multiple servers after encrypting the files. Li et al. [13] considered the hybrid cloud architecture consisting of a public cloud and a private cloud and efficiently solved the problem of deduplication with differential privileges. Yuan et al. [20] proposed a deduplication system in the cloud storage to reduce the storage size of the tags for integrity check. Recently, Bellare and Keelveedhi [2] proposed a new primitive iMLE, which adopted interaction as a new ingredient to provide privacy for messages that are both correlated and dependent on the public system parameters.

Abadi et al. [1] provided stronger security guarantee for secure deduplication. The first approach was to avoid using tags that are derived deterministically from the message. They designed a fully randomized scheme that supported equality test over ciphertext. More precisely, there were three components in the fully randomized scheme, namely a payload, a tag and a proof of consistency. The tag they designed for plaintext  $m$  is computed as  $\tau = (g^r, g^{rh(m)})$ , where  $g$  is the generator of a bilinear group,  $h$  is a sufficient strong collision-resistant function, and  $r$  is a randomly chosen number. Given two tags  $\tau_1 = (g_1, h_1)$  and  $\tau_2 = (g_2, h_2)$ , the equality-testing algorithm verifies  $e(g_1, h_2) \stackrel{?}{=} e(g_2, h_1)$ . The second approach was a deterministic scheme. It was made secure subject to the condition

where the distributions were efficiently samplable using at most  $q$  queries to the random oracle. Thus, the security of the second approach was guaranteed by limiting the computational power of the adversarial message distributions.

## 1.2 Our Contributions

Building on the above insight, we make several contributions, as follows:

1. This is the first attempt to solve the open problem pointed out by [1] “the first scheme (R-MLE2) requires a pairwise application of the equality-testing algorithm to identify all duplicate ciphertexts”. We reduce the linear pairing comparison times of the R-MLE2 to nearly logarithmic times.
2. By adopting client-server interaction, we construct two deduplication decision tree structures: *static deduplication decision tree* and *dynamic deduplication decision tree*. The static one is suitable for static data, while the dynamic one, based on the self-generation tree, allows data update such as data insertion, deletion, and modification.
3. We provide the security and theoretical performance analysis for the proposed schemes, which show that our scheme is both secure and efficient.

## 2 Preliminaries and Notation

### 2.1 Notation

The set of binary string of length  $n$  is denoted as  $\{0, 1\}^n$ , and the set of all finite binary strings are denoted as  $\{0, 1\}^*$ . We denote the bit length of a given binary string  $s$  as  $|s|$ . Given two binary strings  $s_1$  and  $s_2$ , the concatenation is written as  $s_1||s_2$ . The notation  $[1, n]$  denotes the integer set  $\{1, \dots, n\}$  with  $n \in \mathbb{N}$ . We denote the output  $x$  of an algorithm  $\mathcal{A}$  as  $x \leftarrow \mathcal{A}$ . Sampling uniformly random from a set  $X$  is denoted as  $x \stackrel{R}{\leftarrow} X$ . Also,  $A \leftarrow B$  is used to denote the communication between two entity  $A$  and  $B$ . Throughout,  $\lambda$  is denoted as the security parameter, and  $h(\cdot)$  is modeled as hash function.

### 2.2 Bilinear Pairings

Let  $\mathbb{G}$  and  $\mathbb{G}_T$  be two cyclic multiplicative groups of prime order  $p$ ,  $g$  be a generator of  $\mathbb{G}$ . A bilinear pairing is a map  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  with the following properties:

- Bilinear:  $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$  for all  $u, v \in \mathbb{G}$ , and  $a, b \in \mathbb{Z}_p^*$
- Non-degenerate:  $\hat{e}(g, g) \neq 1$ .
- Computable: It is efficient to compute  $\hat{e}(u, v)$  for all  $u, v \in \mathbb{G}$ .

### 2.3 Decision Trees

A decision tree is a decision support tree-like model, where the decision process walks the tree from the root. The tree nodes, correspond to partitioning rules, are used to decide which branch to take until a leaf node is encountered.



### 2.4 MLE for Lock-Dependent Message

A message-locked encryption for lock-dependent messages MLE2 [1] is a six-tuple  $\Pi = (\text{PPGen}, \text{KD}, \text{Enc}, \text{Dec}, \text{EQ}, \text{Valid})$  defined below.

- The parameter generation algorithm  $\text{PPGen}$  takes as input  $1^\lambda$  and returns public parameters  $pp$ .
- The key derivation function  $\text{KD}$  takes as input public parameters  $pp$ , a message  $m$ , and outputs a message-derived key  $k_m$ .
- The encryption algorithm  $\text{Enc}$  takes as input public parameters  $pp$ , a message  $m$ , and a message-derived key  $k_m$ . It outputs a ciphertext  $c$ .
- The decryption algorithm  $\text{Dec}$  takes as input public parameters  $pp$ , ciphertext  $c$ , and a secret key  $k$  and outputs either a message  $m$  or  $\perp$ .
- The equality algorithm  $\text{EQ}$  takes as input public parameters  $pp$ , and two ciphertexts  $c_1$  and  $c_2$  and outputs 1 if both ciphertexts are generated from the same underlying message.
- The validity-test algorithm  $\text{Valid}$  takes as input public parameters  $pp$  and a ciphertext  $c$  and outputs 1 if the ciphertext  $c$  is a valid ciphertext.

## 3 Security Model and Definitions

Our system consists of the clients and a cloud storage server as shown in Fig. 1. The clients (or data owners), will outsource their encrypted data to the untrusted cloud storage server. We consider the following models and basic properties.

**Definition 1.** ( $\mu\text{R-MLE2}$ ) An efficient fully random message-locked encryption scheme with randomized tag is an eight-tuple of polynomial-time algorithms  $\Pi = (\text{PPGen}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Treelnit}, \text{EQ}, \text{Valid}, \text{Dedup})$  run by a client and a deduplication server.

- $pp \leftarrow \text{PPGen}(1^\lambda)$ : The parameter generation algorithm takes  $1^\lambda$  as input and outputs the public parameter  $pp$ .
- $k_m \leftarrow \text{KeyGen}(pp, m)$ : The key generation algorithm takes the public parameters  $pp$  and a message  $m$  as inputs, and outputs a message-derived key  $k_m$ .

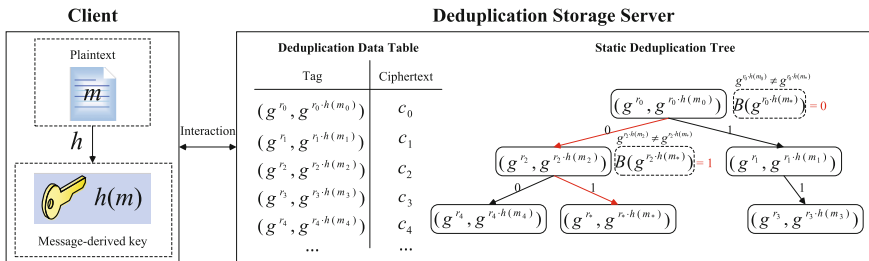


Fig. 1. The general network structure of  $\mu\text{R-MLE2}$

- $c \leftarrow \text{Enc}_{pp}(k_m, m)$ : The encryption algorithm takes the public parameters  $pp$  and the message derived key  $k_m$  as inputs, and returns a ciphertext  $c$ .
- $m \leftarrow \text{Dec}_{pp}(k_m, c)$ : The algorithm takes the public parameter  $pp$  and the message derived key  $k_m$  as inputs. If the algorithm runs successfully, it will return the plaintext  $m$ . Otherwise, it will return  $\perp$ .
- $ts \leftarrow \text{Treelnit}(1^\lambda)$ : The tree initialization algorithm takes  $1^\kappa$  as input, and outputs the tree state  $ts$  of the current database.
- $\{0, 1\} \leftarrow \text{EQ}_{pp}(\tau_1, \tau_2)$ : The equality-testing algorithm takes the public parameter  $pp$  and the tags  $\tau_1$  and  $\tau_2$  of two ciphertexts as inputs, and outputs 1 if the tags of the ciphertexts are generated from identical messages.
- $\{0, 1\} \leftarrow \text{Valid}_{pp}(c)$ : The validity-testing algorithm takes public parameters  $pp$  and the ciphertext  $c$  as input. It outputs 1 if the ciphertext  $c$  is a valid input and 0 otherwise.
- $\{0, 1\} \leftarrow \text{Dedup}_{pp}(st, \tau_1, \tau_2)$ : The data deduplication algorithm takes the public parameters  $pp$ ,  $\tau_1$ , and tag  $\tau_2$  as inputs. It returns whether a duplicate data copy has been found.

Intuitively, with a client holding message  $m'$  and its corresponding tag  $\tau'$ , the scheme should direct to the identical data copy if a duplicate value is stored in the storage server. We consider that the server stores a sequence of data  $\{c_1, \dots, c_n\}$  and the corresponding tag values  $\{\tau_1, \dots, \tau_n\}$ . The tree states evolve after each storing (there is no duplication data copy stored in the storage server), from  $ts_0$  to  $ts_n$ , where  $ts_0$  is the initial state. We define the following properties.

**Definition 2.** (Correctness). A  $\mu$ -RMLE2 scheme for the plaintext domain  $D$  is correct if for all security parameter  $\lambda$ , tag equality test algorithm  $\{0, 1\} \leftarrow \text{EQ}_{pp}(\tau_i, \tau_j)$ , and deduplication algorithm  $\{0, 1\} \leftarrow \text{Dedup}_{pp}(st, \tau)$  for all data sequence  $c_1, \dots, c_n$  and tag sequence  $\tau_1, \dots, \tau_n$ , for all tree states  $ts$ , we have

$$\text{Dedup}_{pp}(st, \tau_i) = \text{Dedup}_{pp}(st, \tau_j) \text{ for all steps, and finally get } \text{EQ}_{pp}(\tau_i, \tau_j) = 1 \text{ iff } m_i = m_j.$$

We now define the security of our scheme, which intuitively says that the scheme must not leak anything besides the bits for deduplication path choosing in the deduplication test tree. The security definition is the Path-PRV-CDA2. The definition says that an adversary cannot distinguish between two test sequences of values as long as the sequences have the same tree path.

**Path-PRV-CDA2 Security Game.** The security game between a client and an adversary  $Adv$  for security parameter  $\lambda$  proceeds as follows:

The client and the server run  $\mu$ R-MLE2 as constructed, the client and the adversary engage in a number of rounds of interaction (not larger than the height of deduplication decision tree), where the client randomly samples message from real or rand mode.

- At round  $i$ , the client will send 1-bit path decision value to the adversary.
- With the additional bit information, the adversary conducts the Path-PRV-CDA2 game,  $\text{Exp}_{\Pi, \mathcal{A}}^{\text{mode}}(\lambda)$ , as defined in [1].
- The adversary outputs  $b$ .

We provide our secure definition of Path-PRV-CDA2 security based on the definition PRV-CDA2 security presented in [1].

**Definition 3 (Path-PRV-CDA2 security).** A  $\mu R$ -MLE2 scheme  $\Pi$  is Path-PRV-CDA2 secure if for any probabilistic polynomial-time adversary  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\lambda)$  such that

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{Path-PRV-CDA2}}(\lambda) \stackrel{\text{def}}{=} |\Pr [\text{Exp}_{\Pi, \mathcal{A}}^{\text{real}}(\lambda) = 1] - \Pr [\text{Exp}_{\Pi, \mathcal{A}}^{\text{rand}}(\lambda) = 1]| \leq \text{negl}(\lambda),$$

where for each mode  $\in \{\text{real}, \text{rand}\}$  and  $\lambda$  the experiment is from  $\text{Exp}_{\Pi, \mathcal{A}}^{\text{mode}}(\lambda)$ .

We also define the efficiency and the dynamic properties for our schemes.

**Definition 4. (Efficiency)** We say a  $\mu R$ -MLE2 scheme is efficient, if the scheme is of asymptotically optimal over deduplication test, namely sublinear equality-test time.

**Definition 5. (Dynamic)** We say a  $\mu R$ -MLE2 scheme is dynamic, if the scheme can be efficiently added, deleted and changed after the initial outsourcing.<sup>1</sup>

## 4 The $\mu R$ -MLE2 Constructions

### 4.1 High-Level Description

Abadi et al. [1] proposed a construction for building fully randomized message-locked encryption scheme based on entropy-based DDH assumption. In the scheme, the “payload” is used to store the encryption of message using some underlying randomized encryption scheme, and the tag is generated from the message. There is a proof of consistency, which proves that the payload and the tag correspond to the same message. A tag for a message  $m$  is computed as  $\tau = (g^r, g^{r \cdot h(m)})$ . Given two tags  $\tau_1 = (g^{r_1}, g^{r_1 \cdot h(m_1)})$  and  $\tau_2 = (g^{r_2}, g^{r_2 \cdot h(m_2)})$  the equality algorithm verifies  $\hat{e}(g^{r_1}, g^{r_2 \cdot h(m_2)}) \stackrel{?}{=} \hat{e}(g^{r_2}, g^{r_1 \cdot h(m_1)})$ .

However, the server needs to conduct data equality test over the whole database, which is inefficient in practical utilization. To solve this problem, we provide an efficient scheme. The main trick is that we adopt an interactive way to construct decision tree structures over the deduplication database, where the client who wants to store data needs to conduct a number of interactions with the server to verify whether the data is a duplicate copy. More precisely, the server maintains a decision tree, which stores the storage states of the current database. A client, who wants to store data, will interact with the server, where the server provides the tree state and the client provides a path decision over the decision tree in each communication round. Trivially, given the private key  $h(m)$  and some relevant information, the client computes and sends a 1-bit path decision to the server in each step. When there is no duplicate data stored, the

<sup>1</sup> We assume the specific operation over the outsourced data is step by step.

node pointer of the state tree will move a null child node of a leaf node. Then, the server will store the data and update the decision tree state.

The first decision tree, based on which the scheme conducts data deduplication test without pairing computation, is a static tree. It means that deduplication scheme based on the static deduplication decision tree is efficient. However, it does not efficiently support the data insertion and deletion, based on which the deduplication is difficult to support data update. Note that efficient decision tree update is important in practical applications, we design a deduplication decision which supports efficient tree update. Our main trick is to use a self-generation tree constructed from a public seed, where the server verifies whether the data form is identical to the current node in the tree and returns the result to the client then indecently provides the server which node path to take in the next step.

### 4.2 The Proposed $\mu$ R-MLE2 Schemes

In this section, we present the detail construction of our efficient randomized MLE2 ( $\mu$ R-MLE2) based on the definition of fully randomized MLE2 [1]. The scheme  $\mu$ R-MLE2  $\Pi = (\text{PPGen}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Treelnit}, \text{EQ}, \text{Valid}, \text{Dedup})$  is polynomial in the security parameter. Since our schemes are based on R-MLE2, we omit some construction details of the algorithms in [1], and provide only the related three algorithms as follows:

- Tree initialization algorithm  $ts \leftarrow \text{Treelnit}(1^\lambda)$ : It initializes server state  $st$  with *static/dynamic* deduplication decision tree and returns  $st$ .
- Equality-testing algorithm  $\{0, 1\} \leftarrow \text{EQ}_{pp}(\tau_1, \tau_2)$ : On input  $\tau_1 = (g_1, h_1) \in \mathbb{G}^2$  and  $\tau_2 = (g_2, h_2) \in \mathbb{G}^2$ , the algorithm verifies  $\hat{e}(g_1, h_2) \stackrel{?}{=} \hat{e}(g_2, h_1)$  and outputs 1 if and only if  $\hat{e}(g_1, h_2) = \hat{e}(g_2, h_1)$ .
- Data deduplication algorithm  $\{0, 1\} \leftarrow \text{Dedup}_{pp}(st, \tau_1, \tau_2)$ : On input the public parameters  $pp$ ,  $\tau_1$ , and tag  $\tau_2$ , it calls the algorithm EQ at each node in a tree path until the leaf node if  $0 \leftarrow \text{EQ}$ . It outputs 0 when all the EQ test output 0. Otherwise it output 1.

We design two equality test algorithms based on the static and dynamic deduplication decision tree, respectively.

**Static Deduplication Decision Tree.** Figure 1 provides an example about storing  $m^*$  based on static deduplication decision tree. A client, with message  $m_*$ , wants to conduct secure deduplication. It generates the corresponding tag  $\tau = (g^{r_*}, g^{r_* \cdot h(m_*)})$  over message  $m_*$ . Also, the storage server stores the deduplication data table and maintains its corresponding deduplication decision tree.

As shown in Algorithm 1, the tag comparison query path is generated according to the data storage sequence of clients, which only allows the server to add data at the leaf nodes. Also, it can be called client-oriented deduplication scheme because the deduplication is conducted at the client side. We need to remark

---

**Algorithm 1.** Equality test over *static* deduplication decision tree

---

- 1.1 Client $\leftarrow$ Server: The client asks for the deduplication of new data  $m_*$ , and the server returns the tag of the current node  $(g^{r_i}, g^{r_0 \cdot h(m_i)})$ . (Initially, the current node is the root of the tree and its tag is  $(g^{r_0}, g^{r_0 \cdot h(m_0)})$ ).
  - 1.2 Client: The client computes  $g^{r_i \cdot h(m_*)}$  and verifies  $g^{r_i \cdot h(m_*)} \stackrel{?}{=} g^{r_i \cdot h(m_i)}$ .
  - 1.3 Client $\rightarrow$ Server: If  $g^{r_i \cdot h(m_*)} = g^{r_i \cdot h(m_i)}$ , the client sends “duplication find” to the server. Otherwise, it computes  $b = B(g^{r_i \cdot h(m_*)}) \in \{0, 1\}$  and sends  $b$  to the server.
  - 1.4 Server: The server moves the current pointer of the tree according to  $b$ . If  $b = 0$ , the server moves the pointer to its left child. Otherwise, it moves the current pointer to its right child. Then, return to step 1.1. The algorithm stops, when the server receives “duplication find”, or when the server needs to move the pointer to an empty node.
- 

that, the static scheme allows clients to get the detail construction of the deduplication decision tree stored in the storage server, which allows malicious attackers to monitor the data deduplication activity of the storage server, such as the exactly time when a new data is uploaded to the storage server.

**Dynamic Deduplication Decision Tree.** Figure 2 provides an example about storing  $m^*$  based on dynamic deduplication decision tree. The dynamic deduplication decision tree is a self-generation tree, where the seed  $s_0$  of the tree is a public parameter generated by the server. The left(right) child of node with  $s_{0b_1b_2\dots b_i}$  is  $s_{0b_1b_2\dots b_i} = h(s_{0b_1b_2\dots b_{i-1}}||0)(s_{0b_1b_2\dots b_i} = h(s_{0b_1b_2\dots b_{i-1}}||1))$ . The scheme, as shown in Algorithm 2, can be called server-oriented deduplication scheme, where the deduplication test is conducted by the storage server. It hides the deduplication decision tree structure stored in the storage server, which will defeat the security problem in static deduplication scheme.

---

**Algorithm 2.** Equality test over *dynamic* deduplication decision tree

---

- 2.1 Client $\rightarrow$ Server: The client asks for the deduplication of new data  $m_*$  and sends the server  $\tau = (g^{r_*}, g^{r_* \cdot h(m_*)})$  and  $b_i$ . (Initially,  $b = -1$ , which means that the current node is the root of the tree and the corresponding tag is  $\tau = (g^{r_0}, g^{r_0 \cdot h(m_0)})$ ).
  - 2.2 Server: The server verifies whether  $e(g^{r_*}, g^{r_i \cdot h(m_i)}) = e(g^{r_i}, g^{r_* \cdot h(m_*)})$ .
  - 2.3 Server $\rightarrow$ Client: The server returns 1 when the equation holds, and 0 otherwise.
  - 2.4 Client: When the client receives 0 from the server, the client computes  $s_{0b_1\dots b_i} = h(s_{0b_1b_2\dots b_{i-1}}||b_i)$ . Then it computes  $b_i = B(h(m)||s_{0b_1\dots b_{i-1}})$ . (The initial seed is  $s_0$ ).
  - 2.5 Client $\rightarrow$ Server: The client sends  $b_{i+1}$  to the server.
  - 2.6 Server: The server moves the current pointer over the tree according to  $b_{i+1}$ . If  $b_{i+1} = 0$ , the server moves the pointer to its left child. Otherwise, it moves the pointer to its right child. Then, go to step 2.3. The algorithm 2 stops as described in algorithm 1.
-

The tree construction relies on the self-generated hash value  $s_{0b_1\dots b_i}$  in self-generation tree, and clients can generate the tree themselves. More precisely, the path decision of dynamic deduplication scheme is decided according to  $b = B(h(m)||s_{0b_1\dots b_i})$ , which is independent of data storage sequence. The complement and deletion operations are obvious. To insert a value at a certain position, firstly conduct the complement operation and move the existing data to a leaf node. Then, insert the specified data to the current position. To delete a node, firstly delete the current node. Then, choose an appropriate leaf node of the deleted node and insert it to the position of the deleted node. With dynamic deduplication tree, we are able to improve the efficiency of our scheme by conducting tree balancing as illustrated in Appendix B. To further reduce the communication round, the client could compute and send multiple bits to the server. Also, the client could reduce some computational overhead of the hash value generation by storing some hash elements of the self-generation tree. The security analysis of our proposed scheme will be given in the full version of this paper.

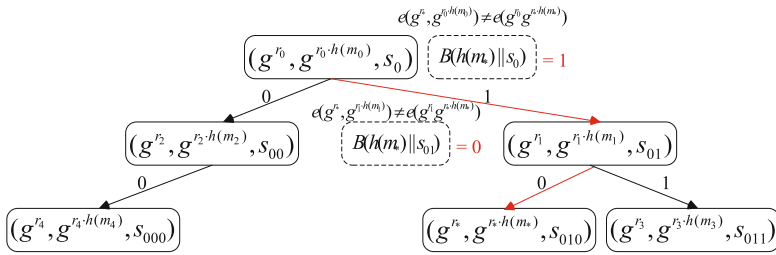


Fig. 2. Dynamic deduplication tree

## 5 Performance Analysis

We provide theoretical efficiency analysis and comparison of the proposed schemes. In Appendix A, we discuss the theoretical results over our decision tree. Also, Appendix B provide the tree optimization from deduplication tree balancing. Based on these analyzes, Tables 1 and 2 illustrate the comparison among the three schemes in terms of both asymptotic computation and communication complexity and actual execution time. In our computation analysis, **Hash** denotes hash operation mapping a bit-string to a designed length value, **Mul** denotes a multiplication operation, **Exp** denotes the exponentiation operation in  $\mathbb{G}$ , and **Pair** denotes the pairing operation. In our communication analysis, we consider the bit-length of the content that needed to transfer between the client and the server. In our analysis, **SND** and **RCV** denote the overhead of sending and receiving a message with a certain length respectively.

**Table 1.** Computation overhead of data deduplication

Scheme	Client	Server
R-MLE2	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash}$	$2\mathbf{Pair} \cdot O(n)$
$\mu\mathbf{R}$ -MLE2 (Static)	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash} + (\mathbf{Exp} + \mathbf{Hash}) \cdot O(h)$	$\emptyset$
$\mu\mathbf{R}$ -MLE2 (Dynamic)	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash} + (2\mathbf{Hash}) \cdot O(h)$	$2\mathbf{Pair} \cdot O(h)$

We omit lightweight string comparison overhead and some common computational overhead among the three scheme such as key generation, data encryption and validity testing.

As shown in Tables 1 and 2, the client needs to conduct one-time tag generation and transmission. Then, the server will use pairing computation over the whole database to realize the equality test. More precisely, the computation and communication overhead of the server are  $O(1)$  and  $O(n)$ , respectively.

The scheme based on static deduplication decision tree needs to fetch the tags in the path of deduplication decision tree and verify whether there is a duplicate copy of data stored in the server. Then, it sends a 1-bit path decision information to the server for choosing the path over the deduplication decision tree. The server just provides the tag values for the client according to the 1-bit path decision information each time. Since the static scheme is client side equality test, it does not need the expensive pairing computation. Instead, compared with the dynamic scheme, the scheme based on the static one needs much more communication overhead in each communication round.

The scheme based on dynamic deduplication decision tree, will greatly reduce both the communication and computation over of the client. More precisely, the client will only compute the path decision bit with the seeds of self-generation tree and send it to the server each time. The server will conduct the expensive equality test based on bilinear pairing. The maximum communication rounds of our schemes are decided by the deduplication decision tree height  $h$ , while not the whole data items  $n$  stored in the server. Actually, with the self-generation tree, the client could send multiple bits to help the server to conduct path decision each time, which will further reduce the communication rounds of the scheme. Compared with the scheme based on static deduplication decision tree, the client conducts lightweight hash operations and leaves the expensive pairing computation to the server.

*Remark 1.* The maximum equality test times of our schemes are the height of the deduplication decision tree. According to the theoretical analysis of the deduplication decision tree height in the Appendix A, our scheme is efficient. As the tree updating and the tree balancing discussion in the Appendix B, it is obvious that the scheme based on the dynamic deduplication decision tree is a dynamic scheme. For data deletion, the server could directly delete the relation between the deduplication decision tree and the data item. For data insertion, if a node is empty, the server will just insert the data. Otherwise, the data owner need help the server to move the data to a leaf node of the tree according to the deduplication policy and then insert the data element to the designed node.

**Table 2.** Data deduplication communication overhead

Scheme	Communication bits (Client)	Communication rounds
R-MLE2	$\mathbf{SND}( g^r  +  g^{r \cdot h(m)} )$	$O(1)$
$\mu$ R-MLE2 (Static)	$\mathbf{SND}(x) + \mathbf{RCV}(y)^*$	$O(h)$
$\mu$ R-MLE2 (Dynamic)	$\mathbf{SND}(x) + \mathbf{RCV}(O(h))^*$	$O(h)$

We omit the ciphertext uploading overhead among the three scheme. We do not provide the communication overhead of the server, because the communication content between the server and client is asymmetric. The server-side communication overhead is  $\mathbf{SND}(\mathbf{RCV})$ , when the client-side communication overhead is  $\mathbf{RCV}(\mathbf{SND})$ .

\*  $x = |g^r| + |g^{r \cdot h(m)}| + O(h)$  and  $y = (|g^r| + |g^{r \cdot h(m)}|)O(h)$  and  $1 \leq x \leq h$ .

## 6 Conclusion

We explore intractive avenues to extend the efficiency of fully randomized secure deduplication scheme. We construct two interactive schemes based on static and dynamic deduplication decision tree structures, respectively. The scheme based on the static deduplication decision tree does not allow the tree to update, while the scheme the dynamic deduplication decision tree is constructed based on the designed self-generation tree, which allows the server to conduct tree update and some other optimization such as tree balancing based on deduplication access frequency of the user. We also provide the security and performance analysis of our scheme, which show that our scheme is Path-PRV-CDA2 secure and it achieves several orders of magnitude higher performance than the state-of-the-art scheme in practical data deduplication.

**Acknowledgement.** We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (No. 61272455, and No. 61370190), China 111 Project (No. B08038), Doctoral Fund of Ministry of Education of China (No.20130203110004), Program for New Century Excellent Talents in University (No. NCET-13-0946), the Fundamental Research Funds for the Central Universities (No. BDY151402), and National High Technology Research and Development Program (863 Program) of China (No.2015AA016007); Besides, this work is also supported by US National Science Foundation under grant (CNS-1217889 and CNS-1446479).

## Appendix

### A Tree Height

Most operations on a deduplication decision tree take time directly proportional to the height of the tree, so it is desirable to keep the height small. A binary tree with height  $h$  can contain at most  $2^{h+1} - 1$  nodes. It follows that for a tree with  $n$  nodes and height  $h$  where  $h \geq \lfloor \log_2^n \rfloor$ .



We model the hash function as random oracle in this paper. Since the input  $g^{r \cdot h(m)}$  each time is random in our first deduplication decision tree construction, we consider  $g^{r \cdot h(m)}$  as a random bit-string. Then, we also consider  $b = B(g^{r \cdot h(m)})$  as random bit.

In the self-generation binary tree, we consider the adopted hash function from a family of hash functions which maps the value of each key from some universe  $U$  into  $m$ .  $H = h : U \rightarrow [m]$ , where  $\forall x, y \in U, x \neq y : \Pr_{h \in H} [h(x) = h(y)] \leq \frac{1}{m}$ . Then, we can model the values in the self-generation tree as random values. Finally, we model  $b = B(s || h(m))$  as random bit.

The two deduplication decision trees constructed are similar to the random binary trees widely studied to provide information useful in evaluating algorithms based on this storage structure. We model the hash value as the random input and adopt the compressed binary  $b$  to make path decision.

If we construct a binary search tree from a sequence of  $n$  different numbers by inserting them in the random order into an initially empty tree as shown in our two constructions. Let  $H_n$  be the height of the constructed tree on  $n$  nodes. As  $n$  approaches  $\infty$ , there exists constants  $\alpha = 4.311\dots$  and  $\beta = 1.953\dots$ , such that the expected value  $E(H_n) = \alpha \ln n - \beta \ln \ln n + O(1)$ , and the variety of  $H_n$  is  $Var(H_n) = O(1)$  [18]. Here,  $\ln$  is the natural logarithm and  $\log$  is the base 2 logarithm. The expect height of the two deduplication decision trees are of logarithmic equality-testing time, which means that our schemes are efficient.

## B Deduplication Decision Tree Balancing

An optimal binary search tree (BST) is usually used to provide the smallest possible search time for a given sequence of accesses. We consider the server side optimization over our dynamic deduplication decision tree, because it could gradually collect and record the deduplication access frequency of all the elements stored in the database. Actually, it is not practical for us to conduct tree balancing over the static deduplication decision tree, because we can not predict the storing sequence and frequency affect the structure of deduplication decision tree. Also, all the first data owners relevant to the node's children need to take part into the tree balancing procedure in static scheme. Thus, we just consider tree balancing of dynamic scheme, where the tree can be modified at any time, typically by permitting tree rotations.

By generalizing the problem, we consider not only the frequencies with which a successful search is completed, but also the frequencies where unsuccessful searches occur. In our deduplication tree, we consider there are  $n$  elements  $B_1, \dots, B_n$  and  $2n + 1$  frequencies  $\beta_1, \dots, \beta_n, \alpha_1, \dots, \alpha_n$  with  $\sum \beta_i + \sum \alpha_j = 1$ , where  $\beta_i$  is the frequency of encountering element  $B_i$ , and  $\alpha_j$  is the frequency of encountering an element which lies between  $B_j$  and  $B_{j+1}$  as defined in [11]. In the dynamic deduplication decision tree with  $n$  interior nodes and  $n + 1$  leaves, as defined in [15], the weighted path length of the tree is

$$P = \sum_{i=1}^n \beta_i (b_i + 1) + \sum_{j=0}^n \alpha_j a_j. \quad (1)$$

Let  $n = 2^k - 1$ ,  $\beta_i = 2^{-k} + \varepsilon_i$ , with  $\sum_{i=1}^n \varepsilon_i = 2^{-k}$  and  $\varepsilon_1 > \varepsilon_2 > \dots > \varepsilon_n > 0$  for  $1 \leq i \leq n$  and  $\alpha_j = 0$  for  $1 \leq j \leq n$ . In a balanced tree for the above frequency distribution, as shown in [15], its weighted path length is

$$P \leq 2^{-(k-1)} \sum_{i=1}^n (b_i + 1) \leq 2^{-(k-1)} \sum_{i=1}^n 2^{(l-1)} \cdot l \leq 2 \cdot \log n. \quad (2)$$

If we get  $\beta_i > \beta_j$  where  $\beta_i$  is the frequency of encountering element  $B_i$  and  $\beta_j$  is the frequency of  $B_i$ 's child node  $B_j$ . We get the weighted path length sum of the two node is  $P$ . We exchange the position of element  $B_i$  and  $B_j$ , the sum of the weighted path length of the two node is  $P'$ . Since the distance of node  $B_i$  from the root is always smaller than that of its children, we have  $b_i < b_j$ . Then, we get

$$\begin{aligned} P - P' &= \beta_i(b_i + 1) + \beta_j(b_j + 1) - \beta_i(b_j + 1) - \beta_j(b_i + 1) \\ &= (\beta_i - \beta_j)(b_i - b_j) < 0 \end{aligned} \quad (3)$$

Equation 3 shows that we will get smaller weighted path length, if we move the element with larger frequency closer to the root. Thus, the server will be able to optimize the tree structure by moving element closer to the root in our scheme based on dynamic deduplication decision tree.

## References

1. Abadi, M., Boneh, D., Mironov, I., Raghunathan, A., Segev, G.: Message-locked encryption for lock-dependent messages. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 374–391. Springer, Heidelberg (2013)
2. Bellare, M., Keelveedhi, S.: Interactive message-locked encryption and secure deduplication. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 516–538. Springer, Heidelberg (2015)
3. Bellare, M., Keelveedhi, S., Ristenpart, T.: Dupless: server-aided encryption for deduplicated storage. In: Proceedings of the USENIX Security Symposium, pp. 179–194, DC, USA, August 2013
4. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 296–312. Springer, Heidelberg (2013)
5. 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)
6. Chen, X., Li, J., Weng, J., Ma, J., Lou, W.: Verifiable computation over large database with incremental updates. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part I. LNCS, vol. 8712, pp. 148–162. Springer, Heidelberg (2014)
7. Douceur, J., Adya, A., Bolosky, W., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of IEEE International Conference on Distributed Computing Systems, pp. 617–624, Macau, China, June 2002

8. Dropbox: Dropbox. <https://www.dropbox.com/>, your stuff, anywhere
9. Google: Google drive. <http://drive.google.com>, all your files, ready where you are
10. Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services: deduplication in cloud storage. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 40–47. CA, USA, January 2010
11. Knuth, D.E.: Optimum binary search trees. *J. Acta Inform.* **1**, 14–25 (1971)
12. Li, J., Chen, X., Li, M., Li, J., Lee, P., Lou, W.: Secure deduplication with efficient and reliable convergent key management. *IEEE Trans. Parallel Distrib. Syst.* **25**, 1615–1625 (2013)
13. Li, J., Chen, X., Li, M., Li, J., Lee, P.P.C., Lou, W.: A hybrid cloud approach for secure authorized deduplication. *IEEE Trans. Parallel Distrib. Syst.* 1–12 (2014)
14. Liu, J., Asokan, N., Pinkas, B.: Secure deduplication of encrypted data without additional independent servers. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 874–885. CO, USA, October 2015
15. Mehlhorn, K.: Nearly optimal binary search trees. *J. Acta Inform.* **5**, 287–295 (1975)
16. Mulazzani, M., Schrittwieser, S., Leithner, M., Huber, M., Weippl, E.R.: Darkclouds on the horizon: using cloud storage as attack vector and online slackspace. In: Proceedings of USENIX Security Symposium, pp. 65–76. CA, USA, August 2011
17. NetApp: Netapp. <http://www.netapp.com/us/products/platform-os/dedupe.aspx>, universal Storage System
18. Reed, B.: The height of a random binary search tree. *J. ACM* **50**, 306–332 (2003)
19. Stanek, J., Sorniotti, A., Androulaki, E., Kencl, L.: A secure data deduplication scheme for cloud storage. In: Proceedings of Financial Cryptography, pp. 99–118. CA, USA, March 2014
20. Yuan, J., Yu, S.: Secure and constant cost public cloud storage auditing with deduplication. In: Proceedings of IEEE Conference on Communications and Network Security, pp. 145–153, MD, USA, October 2013

# Secure and Traceable Framework for Data Circulation

Kaitai Liang<sup>1</sup>, Atsuko Miyaji<sup>2,3</sup>, and Chunhua Su<sup>2(✉)</sup>

<sup>1</sup> Department of Computer Science, Aalto University,  
School of Science and Technology, P.O. Box 15400, 00076 Aalto, Finland  
`kaitai.liang@aalto.fi`

<sup>2</sup> Department of Information and Communications Technology,  
Graduate School of Engineering,  
Osaka University, 1-1 Yamadaoka, Suita, Osaka 565-0871, Japan  
`{miyaji,su}@comm.eng.osaka-u.ac.jp`

<sup>3</sup> Crest, Japan Science and Technology Agency, Honcho,  
Kawaguchi-shi, Saitama 332-0012, Japan

**Abstract.** To date the rapid growth of big data processing and its circulation among multiple organizations incur both promising prospects and security challenges for the corresponding technologies, such as data management, data analysis and so on. Efficient and secure data traceability is of critical importance for big data circulation, especially for cloud service applications which are not fully trusted and the risk of leakage of sensitive personal information. In this paper, we propose a framework for mutual traceability for data circulation and secure outsourced computation in data-centric cloud service. Our construction is built on top of searchable attribute-based proxy re-encryption. We enable both data owner and data user to trace their data circulation or perform privacy-preserving feedback. Specifically, the system enables data owners to efficiently distribute and trace his/her data to a specified group of cloud service providers who match a security/privacy policy and meanwhile, the data, maintaining its traceable property, can be further updated after being shared. The new mechanism is applicable to many real-world big data applications. Finally, our framework is proved chosen ciphertext secure in the random oracle model.

**Keywords:** Data circulation · Data traceability · Data privacy · Cloud services security

## 1 Introduction

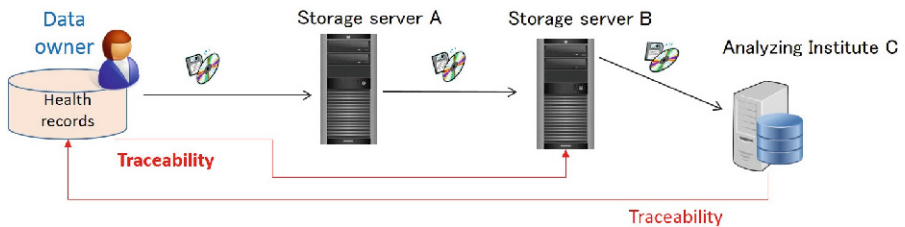
A great amount of companies and research institutes have been using big data technology for the purposes of marketing and research. The circulation of big data among different parties that becomes a common form of big data utilization. However, the circulation may bring security and privacy risks for personal sensitive information. From the privacy view point, data owners are usually reluctant

to distribute their data in big data circulation without the aids of security and privacy-enhancing technologies. In this paper, we focus on the security and privacy violation problem during the data circulation (including the data collection and distribution phases), we provide a secure and mutual traceable framework for big data utility among different data owners and cloud service providers.

### 1.1 Motivation and Problem Statement

Uploading a huge amount of data to remote clouds for big data analysis that helps not only individuals but also companies to make better decision on the next move. Companies, hospitals and research institutions prefer to store commercial and scientific data to clouds to achieve more efficiency in data analysis. After service consumers upload their data, says to a hospital sever, the data may be transferred to more advanced medical institute for further diagnosis or experimental study. This will unavoidably encounter with many unpredictable security and privacy challenges.

In this paper, we assume there are three main parties in the big data circulation context, namely data owner, data storage server and data analyzing institute. A data owner's data is circulated among data storage server and analyzing institute. We design secure and privacy-preserving framework that can ensure the security of data circulation and the traceability over data owner's respective private data sets which allow outsourced storage and outsourced computation service to be executed with confidentiality over encrypted data. It is important to implement a secure and traceable data circulation framework for clouds storage and computing scenario. Protecting the privacy of data owner but also supporting efficient secure data distributing in the context of big data that is an interesting and unsolved problem in the literature (Fig. 1).



**Fig. 1.** Framework for mutual traceability

We introduce the following three security properties which will be further used in this paper.

- Correctness: a data owner should be able to verify the integrity and the completeness of any computation results which are returned by a data analyzing institute.

- Confidentiality and Privacy: The data being stored on a server should not be abused during the transition either between data owner and storage service provider, or between the storage service provider and data analysis institute.
- Secure mutual traceability: a data owner should be able to trace his/her data distribution and meanwhile, a data analysis institute should be able to return the result of data computation to the data owner with the help of data traceability. But any malicious party should not be able to perform traceability on data distribution.

We make use of attribute-based encryption and proxy encryption in this paper to achieve our goals with a significant reason that the technologies can support fine-grained expressiveness over encryption. After distributing huge volumes of data to cloud servers, the data owner usually takes two actions - one is *data traceability* for checking out the route of data circulation, and the other is *authorized data distribution* for deciding who can be granted the rights of using the data. Traditional attribute-based encryption technology guarantees the confidentiality of the data, but barely supports data traceability. An interesting and possible behavior of a data owner is that he/she updates the data traceability after distributing the encrypted data to other organizations. Consider an encrypted virus sample is tagged with (“Material for data owner A”). After being shared with medical researchers in a data analyzing institute B, the researchers may change the tag as (“Sample at data analyzing institute B”). Since traditional attribute-based technique cannot support such update traceability, we need a new framework to solve the problem.

## 1.2 Related Work

There are many related works on functional encryption, such as the seminal attribute-based encryption proposed by Sahai and Waters [15]. Inspired by the first attribute-based encryption, Goyal *et al.* [10] proposed a key-policy attribute-based encryption whereby ciphertexts are associated with attributes and secret keys are associated with access policies. Many variants of attribute-based encryption, such as [13, 16], have been proposed later. However, the above naive approaches cannot be applied on data traceability quite well and moreover, the existing schemes need additional decryption/encryption burden for data owner who is required to be on-line all the time. The cost for the data owner will become more cumbersome when the number of data tracing is increasing. Our work is also related to verifiable computation with the notion of fully homomorphic encryption (FHE) was first put forward by Rivest *et al.* [14]. However, only in the past few years there have candidate FHE schemes been proposed. The first such scheme was constructed by Gentry [8]. The notion of non-interactive verifiable computation was introduced by Gennaro *et al.* [9]. It enables a computationally weak client to outsource the computation of a function to a server, which returns the result of the function evaluation as well as a non-interactive proof that the computation is carried out correctly.

### 1.3 Our Contribution

To efficiently construct a secure and traceable encrypted data system, we combine the techniques of attribute-based encryption, proxy re-encryption and verifiable computation. Specifically, our construction idea is mainly inspired by the technology introduced in [12] supporting keyword search and data sharing. Our contributions are as follows.

- We investigate one of the most important security issues for big data utilization - traceable data circulation. Our scheme guarantees that the traceability of outsourced encrypted data can be remained after its distribution among multiple parties and organizations.
- We employ Attribute-Based Keyword Search supporting keyword search [17] and overcome its technical limitation in the construction method of trapdoor token (used for searching and decryption). A trapdoor token consists of a data owner’s “re-randomized” secret key. We also prove that our framework is chosen ciphertext secure in the Random Oracle Model (ROM). Our framework is the first of its type supporting the mutual traceability but also secure data circulation (encrypted data).
- Inspired by [12], our scheme is extended to support data update, so that encrypted data can be further updated before delivering to other parties. This property brings convenience to data owner (who can gain access to the data) in the sense that the traceability can be controlled. Our system has good efficiency in traceability and data decryption.

The remainder of this paper is organized as follows. In Sect. 2, we outline the complexity assumption which are used in our framework. We provide detailed construction and implementation of our secure and traceable framework in Sect. 3. In Sect. 4, we describe the security model for our framework and present the security analysis. Finally, we draw conclusion in Sect. 5.

## 2 Preliminaries

In this section, we give a brief introduction for the security assumption to used in our construction. Let  $BSetup$  be an algorithm that on input the security parameter  $\lambda$ , outputs the parameters of a bilinear map as  $(q, g, \hat{g}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ , where  $\mathbb{G}_1$ ,  $\mathbb{G}_2$  and  $\mathbb{G}_T$  are multiplicative cyclic groups of prime order  $q$ , where  $q \in \Theta(2^\lambda)$ , and  $g$  is a random generator of  $\mathbb{G}_1$ ,  $\hat{g}$  is a random generator of  $\mathbb{G}_2$ . The mapping  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  has three properties: (1) *Bilinearity*: for all  $a, b \in_R \mathbb{Z}_q^*$ ,  $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab}$ ; (2) *Non-degeneracy*:  $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$ , where  $1_{\mathbb{G}_T}$  is the unit of  $\mathbb{G}_T$ ; (3) *Computability*:  $e$  can be efficiently computed. Note that  $\mathbb{G}_1$  and  $\mathbb{G}_2$  are not the same.

**Definition 1.** *Asymmetric Decisional  $l$ -BDHE Assumption*

The asymmetric decisional  $l$ -BDHE assumption is that all PPT algorithms  $\mathcal{A}$  given the vector  $\mathbf{y} = (g, g^s, g^a, \hat{g}, \hat{g}^a, \dots, \hat{g}^{a^l}, \hat{g}^{a^{l+2}}, \dots, \hat{g}^{a^{2l}})$ , have an advantage negligible in  $\lambda$  of distinguishing  $e(g, \hat{g})^{a^{l+1}s}$  from a random element  $R$  in  $\mathbb{G}_T$ . The advantage of  $\mathcal{A}$  is defined as  $|Pr[\mathcal{A}(\mathbf{y}, e(g, \hat{g})^{a^{l+1}s}) = 0] - Pr[\mathcal{A}(\mathbf{y}, R) = 0]|$ , where the probability is taken over the choice of  $a, s \in_R \mathbb{Z}_q^*$ ,  $R \in_R \mathbb{G}_T$ , the generators  $g, \hat{g}$ , the random bits consumed by  $\mathcal{A}$ .

**Definition 2.** *One-time Symmetric Encryption?*

We let  $\mathcal{K}_D$  be the key space  $\{0, 1\}^{poly(1^\lambda)}$ , and  $SY$  be a symmetric encryption scheme, where  $poly(1^\lambda)$  is the fixed polynomial size (bound) with respect to the security parameter  $k$ . The encryption algorithm  $S.Enc$  intakes a key  $K \in \mathcal{K}_D$  and a message  $M$ , outputs a ciphertext  $C$ . The decryption algorithm  $S.Dec$  intakes  $K$  and  $C$ , outputs  $M$  or a  $\perp$ .

*Target Collision Resistant Hash Function.* Target Collision Resistant (TCR) hash function was introduced by Cramer and Shoup [6]. A TCR hash function  $H$  guarantees that given a random element  $x$  which is from the valid domain of  $H$ , a PPT adversary  $\mathcal{A}$  cannot find  $y \neq x$  such that  $H(x) = H(y)$ . We let  $Adv_{H, \mathcal{A}}^{TCR} = Pr[H(x) = H(y) \wedge x \neq y | x, y \in DH]$  be the advantage of  $\mathcal{A}$  in successfully finding collisions from a TCR hash function  $H$ , where  $DH$  is the valid input domain of  $H$ . If a hash function is chosen from a TCR hash function family,  $Adv_{H, \mathcal{A}}^{TCR}$  is negligible. Here, in our construction, we use cryptographic hash function and we can make the output into integer, binary bits and arbitrary truncated length according to the usage.

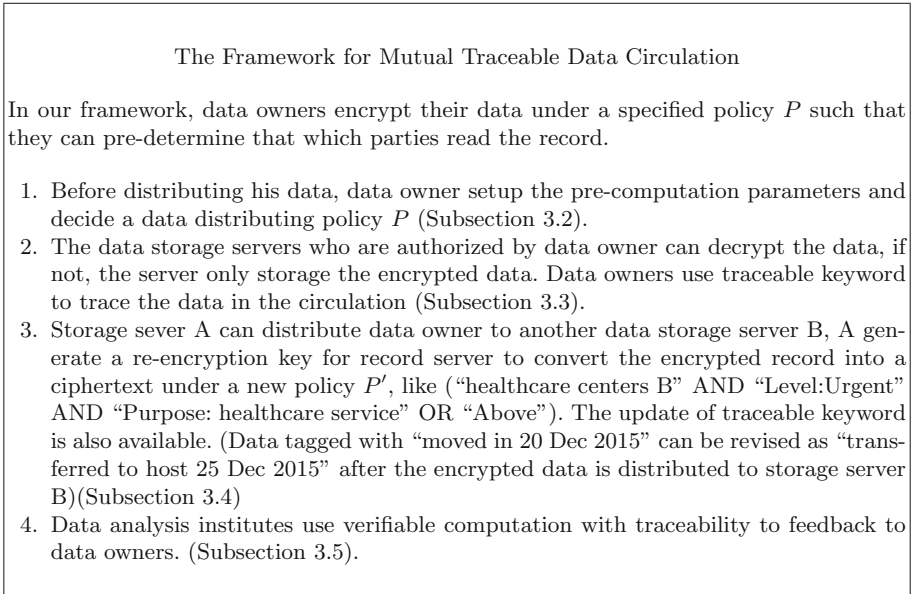
### 3 Our Secure and Traceable Framework for Data Circulation

Before providing the detailed construction, we give a general description of our scheme. We choose an ABE system with fast decryption [11] as one of our building blocks, which can provide expressiveness for data share and keyword search compared to other encryption systems. To achieve the privacy of the traceability, we first extend the concrete ABE system [11] into the asymmetric pairings group. Under the property of asymmetric pairings, one cannot tell whether a given ciphertext contains a keyword or not even he can make pairing computations from the ciphertext components. Our framework allows a token related to a keyword to be constructed via an interaction between Private Key Generator (PKG) and a system user (who specifies the keyword). The construction of the token is somewhat similar to that of the secret key of the user. However, the token (related to a keyword) will not enable its holder (i.e. a cloud server) to decrypt the ciphertext associated with the same keyword. This is a necessary requirement for searchable encryption, i.e. a trapdoor token for a keyword cannot deliver decryption ability to cloud server.



### 3.1 General Description

We construct our framework using several sub-schemes in the key-policy attribute-based setting. Secret key is associated with access policy which is predetermined by data owner, and ciphertext is tagged with attribute set for further usage in data circulation. As of [11], the data owner uses an LSSS access structure  $P = (M, \rho)$  to represent a policy,  $U$  to denote an attribute universe whereby  $|U|$  is a polynomial in  $1^\lambda$ . We let data owner uses  $Tg$  to denote either a single tracing tag including a group of multiple keywords and can be arbitrary length. Our system provides a fine-grained traceability to data owner after his distribution, for example, authorized data analysis institutes try to feedback the data analysis result to a patient in health risk as (“Tanaka”, “Kanazawa”, “Born in 1980”). Meanwhile, this search pattern is hidden from a server administrator such that the privacy of the patient can be guaranteed. It also supports secure data circulation. The general descriptions of our framework are as follows (Fig. 2):

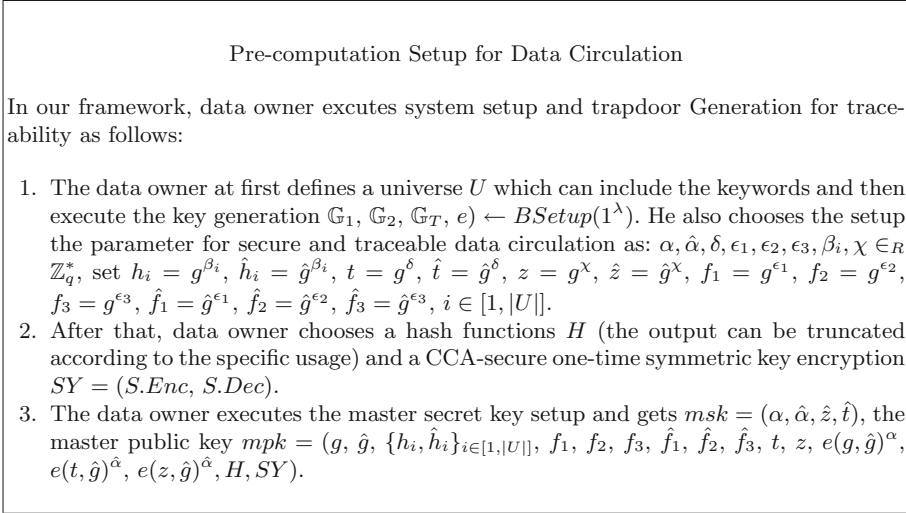


**Fig. 2.** General description of our traceable framework

### 3.2 Framework Setup for Data Owners

In our Framework, the traceability trapdoor is generated via the collaboration between a fully trusted PKG and data owner. When the key holder needs a traceable trapdoor, he first issues the corresponding request (with keyword(s)) to the PKG, the PKG then returns a related intermediate component. Finally,

the key holder re-randomizes the component to become a “real” trapdoor. In our current architecture, we assume all master secret keys (the one for secret key generation, and the other one for trapdoor generation) are known by the PKG only. That is why the secret key holder needs a interaction with the PKG when generating a trapdoor. The system can be extended to allow a secret key holder to generate a trapdoor on his own without any help of PKG. This requires the secret key holder to know  $\hat{\alpha}$ , namely,  $\hat{\alpha}$  is chosen by the secret key holder as one of his secret information (Fig. 3).



**Fig. 3.** Generation of traceable keyword

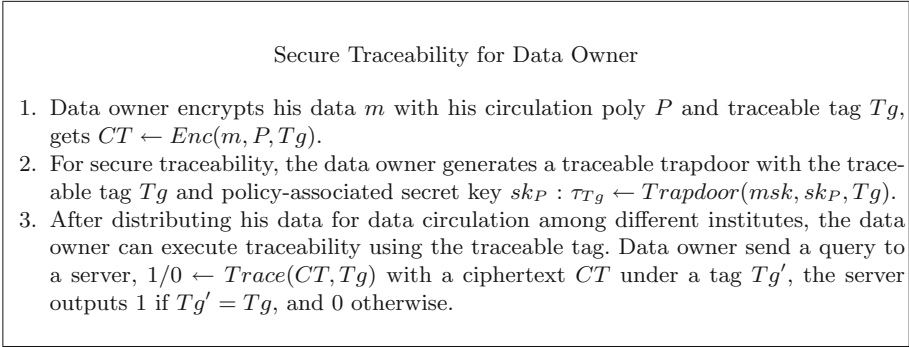
For detailed implementation, data owner can execute the keys generation as follows:  $KeyGen(msk, (M, \rho))$ . Let  $M$  be an  $l \times n$  matrix, and  $\rho$  be the function that associates rows of  $M$  to attributes. Choose a random vector  $(\mathbf{v}) = (\alpha, y_2, \dots, y_n) \in \mathbb{Z}_q^{*n}$  which to be used to share  $\alpha$ . For  $i = 1$  to  $l$ , compute  $\phi_i = (\mathbf{v}) \cdot M_i$ , where  $M_i$  is the vector related to  $i$ -th row of  $M$ . Choose  $r_1, \dots, r_l \in_R \mathbb{Z}_q^*$ , and set  $sk_{(M, \rho)}$  as

$$D_i = \hat{g}^{\phi_i} \cdot \hat{h}_{\rho(i)}^{r_i}, R_i = \hat{g}^{r_i}, \forall d \in \Gamma/\rho(i), Q_{i,d} = \hat{h}_d^{r_i},$$

where  $i \in [1, l], \Gamma$  is the set of distinct attributes in  $M$  (i.e.  $\Gamma = \{d : \exists i \in [1, l], \rho(i) = d\}$ ), and  $\Gamma/\rho(i)$  denotes the set  $\Gamma$  with the element  $\rho(i)$  removed if present. Note  $sk_{(M, \rho)}$  implicitly includes  $(M, \rho)$ .

### 3.3 Secure Data Distribution with Traceability

After generation the pre-computation, data owner can trace his data flow in the data circulation among specific authorized institutes (Fig. 4).



**Fig. 4.** Generation of traceable keyword

To transfer the private data from one data analyzing institute to another institute, the previous institute must perform the re-encryption. For more detailed in computing the encrypted data for further usage, data owner can encrypt its ciphertext  $CT$  as

$$\begin{aligned}
 A &= (m||\sigma) \oplus H(e(g, \hat{g})^{\alpha s}), B = g^s, \{C_x = h_x^s\}_{x \in S}, \\
 D &= e(t^{H(Tg)}z, \hat{g})^{\hat{\alpha} s}, E_1 = f_1^s, \\
 E_2 &= (f_2^{H(A, B, \{C_x\}_{x \in S}, D, E_1)} f_3)^s,
 \end{aligned}$$

where  $\sigma \in \{0, 1\}^\lambda$ ,  $m \in \{0, 1\}^\lambda$  and  $s = H(m, \sigma)$ .  $CT$  implicitly contains  $S$ .

After that, data owner generates the traceable trapdoor  $Trapdoor(msk, sk_{(M, \rho)}, TK)$  by the collaboration between a secret key holder and the fully trusted PKG. The PKG chooses a random vector  $\mathbb{V} = (\hat{\alpha}, \hat{y}_2, \dots, \hat{y}_n) \in \mathbb{Z}_q^{*n}$  which to be used to share  $\hat{\alpha}$ . For  $i = 1$  to  $l$ , it sets  $\hat{\phi}_i = (\mathbb{V}) \cdot M_i$ . It further sends the following values to data owner.

$$\begin{aligned}
 \tau_{1,i} &= (\hat{t}^{H_3(TK)} \hat{z})^{\hat{\phi}_i} \cdot \hat{h}_{\rho(i)}^{\hat{r}_i}, \tau_{2,i} = \hat{g}^{\hat{r}_i}, \\
 \forall d \in \Gamma/\rho(i), \tau_{3,i,d} &= \hat{h}_d^{\hat{r}_i},
 \end{aligned}$$

where  $\hat{r}_1, \dots, \hat{r}_l \in_R \mathbb{Z}_q^*$ ,  $i \in [1, l]$ ,  $\Gamma$  is the set of distinct attributes in  $M$ .

Data owner re-randomizes the values and gets  $\tau_{Tg}$  as:

$$\begin{aligned}
 \tau_{1,i} &= \tau_{1,i} \cdot \hat{h}_{\rho(i)}^{\xi_i}, \tau_{2,i} = \tau_{2,i} \cdot \hat{g}^{\xi_i}, \\
 \forall d \in \Gamma/\rho(i), \tau_{3,i,d} &= \tau_{3,i,d} \cdot \hat{h}_d^{\xi_i},
 \end{aligned}$$

sets the token  $\tau_{Tg}$  as  $T_1 = (\hat{t}^{H(TK)} \hat{z})^{\xi_l}$  and  $T_2 = \hat{g}^{\xi_l}$ , where  $\xi_i \in_R \mathbb{Z}_q^*$ . Note  $(M, \rho)$  implicitly includes in the token.

Data owner use  $Trace(CT, \tau_{Tg})$  to trace his data. Parse  $CT$  as  $(S, A, B, \{C_x\}_{x \in S}, D, E_1, E_2)$ , and  $\tau_{Tg}$  as  $(\tau_{1,i}, \tau_{2,i}, \tau_{3,i,d})$ . Suppose  $S$  associated with  $CT$  satisfies  $(M, \rho)$  associated with  $\tau_{Tg}$ , there exists a set of constants  $\{w_i\}_{i \in I} \in \mathbb{Z}_q^*$

so that  $\forall i \in I, \rho(i) \in S$  and  $\sum_{i \in I} w_i M_i = (1, 0, \dots, 0)$ . Given an original ciphertext  $CT$  associated with a keyword set  $Tg'$  and a token  $\tau_{Tg}$ , one can verify that

$$\frac{e(B, \prod_{i \in I} (\tau_{1,i} \prod_{x \in \Delta / \rho(i)} \tau_{3,i,x})^{w_i})}{e(\prod_{x \in \Delta} C_x, \prod_{i \in I} \tau_{2,i}^{w_i})} \stackrel{?}{=} D.$$

### 3.4 Secure Circulation Scheme

In this subsection, we construct a scheme of secure data circulation for data owners whose data are distributed among multiple institute (In Fig. 5).

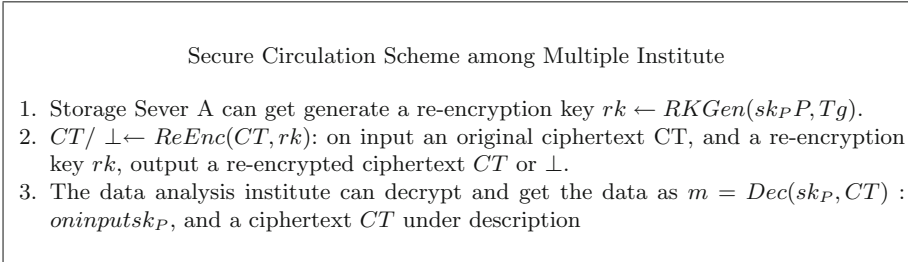


Fig. 5. Secure data circulation scheme

$RKGen(sk_{(M,\rho)}, S, TK)$ . Choose  $\gamma \in_R \mathbb{Z}_q^*$ ,  $\theta_1, \theta_2 \in_R \{0, 1\}^\lambda$ , and set  $rk$  as

$$\begin{aligned} rk_{1,i} &= D_i^{H(\theta_1)} \hat{f}_1^\gamma, rk_2 = \hat{g}^\gamma, rk_{3,i} = R_i^{H(\theta_1)}, \\ rk_{4,i} &= \forall d \in \Gamma / \rho(i) (Q_{i,d})^{H(\theta_1)}, \\ rk_5 &= (\theta_1 || \theta_2) \oplus H(e(g, \hat{g})^{\alpha \check{s}}), rk_6 = g^{\check{s}}, \\ rk_{7,x} &= (h_x^{\check{s}})_{x \in S}, rk_8 = e(t^{H(TK)} z, \hat{g})^{\hat{\alpha} \check{s}}, \\ rk_9 &= (f_2^{H(rk_5, rk_6, rk_{7,x}, rk_8)} f_3)^{\check{s}}, \end{aligned}$$

where  $sk_{(M,\rho)} = (D_i, R_i, \forall d \in \Gamma / \rho(i) Q_{i,d})$ ,  $i \in [1, l]$ ,  $\check{s} = H(\theta_1, \theta_2)$ . Note  $rk$  includes  $(M, \rho)$  and  $S$ .

Data owner parses  $CT$  as  $(S, A, B, \{C_x\}_{x \in S}, D, E_1, E_2)$ ,  $rk$  as  $((M, \rho), S', rk_{1,i}, rk_2, rk_{3,i}, rk_{4,i}, rk_5, rk_6, \{rk_{7,x}\}_{x \in S}, rk_8, rk_9)$ ,  $i \in [1, l]$ .

i. Check the validity of  $CT$  as

$$e(B, \hat{f}_1) \stackrel{?}{=} e(E_1, \hat{g}), \tag{1}$$

$$e(\prod_{x \in S} C_x, \hat{g}) \stackrel{?}{=} e(B, \prod_{x \in S} \hat{h}_x). \tag{2}$$

$$e(B, \hat{f}_2^{H_4(A, B, \{C_x\}_{x \in S}, D, E_1)} \hat{f}_3) \stackrel{?}{=} e(E_2, \hat{g}). \tag{3}$$

If one of the equations does not hold, output  $\perp$ . Else, proceed.

- ii. If  $S$  associated with  $CT$  satisfies  $(M, \rho)$  associated with  $rk$ , let  $I \subset \{1, 2, \dots, l\}$  be a set of indices and  $\{w_i\}_{i \in I} \in \mathbb{Z}_q^*$  be a set of constants so that  $\forall i \in I, \rho(i) \in S$  and  $\sum_{i \in I} w_i M_i = (1, 0, \dots, 0)$ , and define  $\Delta = \{x : \exists i \in I, \rho(i) = x\}$ . Compute

$$\begin{aligned} \Omega &= \frac{e(B, \prod_{i \in I} (rk_{1,i} \prod_{x \in \Delta / \rho(i)} rk_{4,x})^{w_i})}{e(E_1, rk_2^{\sum_{i \in I} w_i}) e(\prod_{x \in \Delta} C_x, \prod_{i \in I} rk_{3,i}^{w_i})} \\ &= \frac{e(g^s, \prod_{i \in I} \hat{g}^{\phi_i H_5(\theta_1) w_i} \hat{f}_1^{\gamma w_i} \prod_{x \in \Delta} \hat{h}_x^{w_i H_5(\theta_1) r_i})}{e(f_1^s, \hat{g}^{\gamma \sum_{i \in I} w_i}) e(\prod_{x \in \Delta} h_x^s, \prod_{i \in I} \hat{g}^{r_i H_5(\theta_1) w_i})} \\ &= e(g^s, \hat{g}^{\alpha H(\theta_1)}). \end{aligned}$$

- iii. Compute the re-encrypted ciphertext as  $T_1 = S.Enc(CT || \Omega, H_6(S.Key))$ ,  $T_2 = (rk_5, rk_6, \{rk_{7,x}\}_{x \in S'}, rk_8, rk_9)$ ,  $T_3 = (T_{3,1} = (S.Key || \theta_3) \oplus H_2(e(g, \hat{g})^{\alpha \tilde{s}}), T_{3,2} = g^{\tilde{s}}, T_{3,3,x} = (h_x^{\tilde{s}})_{x \in S'}, T_{3,4} = (f_2^{H_4(T_{3,1}, T_{3,2}, T_{3,3,x})} f_3)^{\tilde{s}})$ , where  $S.Key, \theta_3 \in_R \{0, 1\}^\lambda, \tilde{s} = H_1(S.Key, \theta_3)$ .

### 3.5 Traceable Outsourced Computation Between Data Owner and Analysis Institute

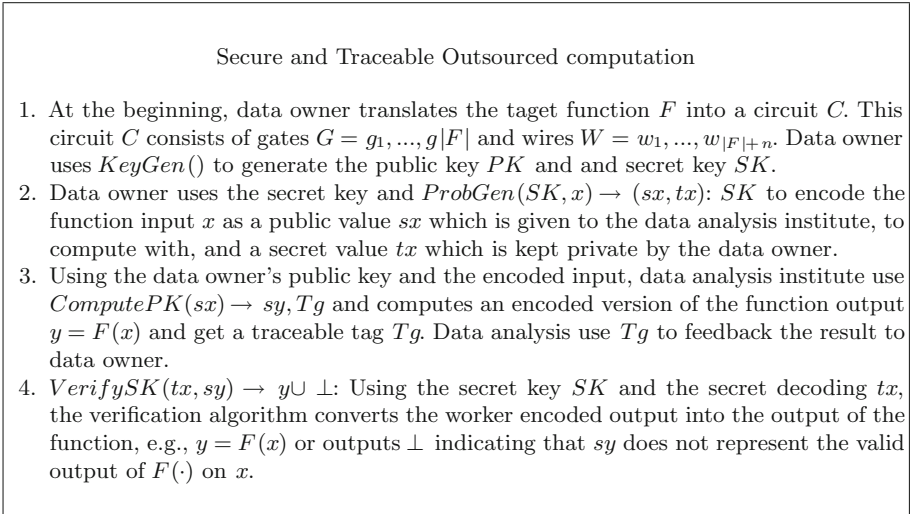
Here in this subsection, we construct a protocol for data analysis institute remotely perform data mining for data owner using his outsourced data, and provide feedback via traceability. A verifiable computation scheme  $\mathcal{VC} = (\text{Key-Gen}, \text{ProbGen}, \text{Compute}, \text{Verify})$  consists of the four algorithms such as in [9]. Our scheme captures the idea of an outsourceable verifiable computation scheme which is more efficient than computing the function in an amortized sense, since the cost of encoding the function can be amortized over many different input computations (In Fig. 6).

This sub-protocol is correct if the problem generation algorithm produces values that allows data analysis to compute values that will verify successfully and correspond to the evaluation of  $F$  on those inputs.

## 4 Security Analysis of Our Scheme

Here in this section, we provide formal proofs for our framework. Before the security analysis, we have to set up the adversary model. In the model, we make adversary strategy to break the privacy and security of our proposed framework and show the successful probability is negligible. The adversary models below including the selective chosen ciphertext security for original ciphertext and re-encrypted ciphertext, and the keyword privacy models.

We suppose  $H$  are hash functions (operated as oracles in the simulation),  $SY$  is a CCA-secure one time symmetric encryption. At the beginning, A outputs a challenge description  $\lambda^* \{0, 1\}^*$  Setup. B runs  $Setup(1^\lambda, U)$  and returns  $mpk$  to A. A is given access to the following oracles.



**Fig. 6.** Secure data circulation scheme

1.  $\mathcal{O}_{sk}(\mu)$ : given a description  $\mu$ , output  $sk \leftarrow KeyGen(msk, \mu)$ .
2.  $\mathcal{O}_{token}(\mu, Tg)$ : given  $\mu$  and a traceable tag  $Tg$ , output  $\tau_{Tg} \leftarrow Trapdoor(msk, sk_\mu, Tg)$ , where  $sk_\mu \leftarrow KeyGen(msk, \mu)$ .
3.  $\mathcal{O}_{test}(CT, Tg)$ : given  $CT$  and  $Tg$ , output  $1/0 \leftarrow Test(CT, \tau_{Tg})$ , where  $Tg \leftarrow Trapdoor(msk, sk_\mu, Tg)$ .
4.  $\mathcal{O}_{rk}(\mu, v, Tg)$ : on input  $\mu$ , a new description for ciphertext  $v$ , and a new keyword  $Tg$ , output  $rk \leftarrow RKGen(sk_\mu, v, Tg)$ .
5.  $\mathcal{O}_{re}(CT, \mu, v, Tg')$ : on input  $\mu$  an original ciphertext  $CT$ ,  $v$  and  $Tg'$ , output  $CT \leftarrow ReEnc(CT, rk)$ .
6.  $\mathcal{O}_{dec}(\mu, CT)$ : on output  $m$ .

**Theorem 1.** *An data circulation framework is selective chosen ciphertext secure if no PPT adversary  $A$  can break the security of [12].*

The detailed security proof will be presented in the full version of this paper. Here we provide a short security proof sketch. We assume each of the  $|U|$  attributes is a unique integer between 1 and  $|U|$  (like [11]). If there exists a PPT adversary  $\mathcal{A}$  can break the IND-sCCA-Or security of our scheme, we can construct a PPT algorithm  $\mathcal{B}$  to break the asymmetric decisional  $|U|$ -BDHE assumption.  $\mathcal{B}$  is given a problem instance of asymmetric decisional  $|U|$ -BDHE. We can show that gap of probabilities between breaking our framework and breaking our security assumption is negligible.

$$\begin{aligned}
& |Pr[b = b' : (\nu^*, state_1) \leftarrow \mathcal{A}(1^\lambda, U); \\
& (mpk, msk) \leftarrow Setup(1^\lambda, U); \\
& (m_0^*, m_1^*, Tg^*, state_2) \leftarrow A^{O_1}(mpk, state_1); \\
& b \in_R \{0, 1\}; CT^* \leftarrow ReEnc(Enc(m_b^*, \nu, Tg), rk); \\
& b' \leftarrow A^{O_1}(CT^*, state_2)] - \frac{1}{2}|,
\end{aligned}$$

In the adversarial game,  $state_1, state_2$  are the state information,  $\nu, Tg$  are chosen by adversary  $\mathcal{A}$ ,  $Enc(m_b^*, \nu, KW)$  is generated by  $\mathcal{B}$  as well as  $rk \leftarrow RKGen(sk_\mu, \nu^*, KW^*)$ ,  $\mu$  matches  $\nu$ . For  $\mathcal{O}_{sk}$ , if  $\mathcal{A}$  issues  $\mu$  satisfying  $\nu^*$ ,  $\mathcal{B}$  outputs  $\perp$ . There is no restriction for re-encryption key queries, namely,  $\mathcal{A}$  can obtain any re-encryption key. This is the reason why we ignore re-encryption oracle here. For decryption query, if the issued ciphertext is the challenge one, output  $\perp$ . We then can prove that for each challenge bit  $b$ , the probability of successful guess is negligible using all the answers from the accessible oracles above. An adversary for defining the indistinguishability in the CCA model can invoke the oracles as many times as they wish. Thus, the adversary can obtain multiple targeted  $Tg$  and ciphertexts  $CT$ .

## 5 Conclusions

In this paper, we propose a new secure and traceable framework for data owners, data storage service providers and data analysis institutes to execute secure mutual traceability for data circulation. Our framework is based on attribute-based proxy re-encryption with addition traceable tag searching, we proposed concrete constructions satisfying the new security and privacy requirement in big data circulation. We also proved the our scheme to be CCA secure in the ROM model. The framework is the first of its type to integrate searchable attribute-based encryption with attribute-based proxy re-encryption, which is applicable to many real-world applications.

**Acknowledgement.** Atsuko Miyaji is supported by Grant-in-Aid for Scientific Research (C) KAKENHI (5K00183) and (15K00189) and Japan Science and Technology Agency (JST); Chunhua Su is supported Grant-in-Aid for Scientific Research KAKENHI (B) 15K16005.

## References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM TISSEC **9**(1), 1–30 (2006)
2. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)

3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. 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)
5. 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)
6. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**(1), 167–226 (2004)
7. Ducas, L.: Anonymity from asymmetry: new constructions for anonymous HIBE. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 148–164. Springer, Heidelberg (2010)
8. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
9. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
10. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
11. Hohenberger, S., Waters, B.: Attribute-based encryption with fast decryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 162–179. Springer, Heidelberg (2013)
12. Liang, K., Susilo, W.: Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. *IEEE Trans. Inf. Forensics Secur.* **10**(9), 1981–1992 (2015)
13. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)
14. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. *Found. Secure Comput.* **32**(4), 169–178 (1978)
15. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
16. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
17. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Canada, pp. 522–530 (2014)



# Public Cloud Data Auditing with Practical Key Update and Zero Knowledge Privacy

Yong Yu<sup>1,2(✉)</sup>, Yannan Li<sup>1</sup>, Man Ho Au<sup>3</sup>, Willy Susilo<sup>4</sup>,  
Kim-Kwang Raymond Choo<sup>5</sup>, and Xinpeng Zhang<sup>1</sup>

<sup>1</sup> School of Computer Science and Engineering,  
University of Electronic Science and Technology of China, Chengdu 611731, China  
yuyong@uestc.edu.cn

<sup>2</sup> Guangxi Colleges and Universities Key Laboratory of Cloud Computing  
and Complex Systems, Guilin University of Electronic Technology,  
Guilin 541004, China

<sup>3</sup> Department of Computing, The Hong Kong Polytechnic University,  
Kowloon, China

<sup>4</sup> School of Computing and Information Technology,  
Center for Computer and Information Security Research, University of Wollongong,  
Wollongong, NSW 2522, Australia

<sup>5</sup> School of Information Technology and Mathematical Sciences,  
University of South Australia, Adelaide, Australia

**Abstract.** Data integrity is extremely important for cloud based storage services, where cloud users no longer have physical possession of their outsourced files. A number of data auditing mechanisms have been proposed to solve this problem. However, how to update a cloud user's private auditing key (as well as the authenticators those keys are associated with) without the user's re-possession of the data remains an open problem. In this paper, we propose a key-updating and authenticator-evolving mechanism with zero-knowledge privacy of the stored files for secure cloud data auditing, which incorporates zero knowledge proof systems, proxy re-signatures and homomorphic linear authenticators. We instantiate our proposal with the state-of-the-art Shacham-Waters auditing scheme. When the cloud user needs to update his key, instead of downloading the entire file and re-generating all the authenticators, the user can just download and update the authenticators. This approach dramatically reduces the communication and computation cost while maintaining the desirable security. We formalize the security model of zero knowledge data privacy for auditing schemes in the key-updating context and prove the soundness and zero-knowledge privacy of the proposed construction. Finally, we analyze the complexity of communication, computation and storage costs of the improved protocol which demonstrates the practicality of the proposal.

## 1 Introduction

Cloud storage, which enables cloud users to move their data from local storage systems to the cloud, is an important service offered by cloud computing [1].

This kind of new storage service has many advantages such as relieving users' burden of data management and maintenance, universal data access with independent geographical locations and avoiding capital cost on hardware and software [2]. Currently, an increasing number of users prefer to store their data in the cloud such as Amazon S3, Google cloud storage and Microsoft Skydrive [3].

However, at the meantime, cloud storage also brings a number of challenging security problems [4] despite of its appealing features. Due to losing physical possession and control of their outsourced data, cloud users would worry that their data might be tampered or deleted due to the following reasons [5]. Firstly, data loss could happen in any infrastructure regardless of the reliability measures the service providers would take. Secondly, in cloud storage, the cloud server may not be fully trusted. As a result, the server may discard the data which has been rarely accessed for financial reasons but claims that the data are well stored, or hides data loss incidents to maintain their reputation. Cloud Security Alliance (CSA) conducted a systematic investigation into reported vulnerabilities in cloud computing such as outages, downtimes, and data loss. CSA also released a white paper [6] in 2013 which revealed that the top three threats were "Insecure Interfaces & APIs", "Data Loss & Leakage" and "Hardware Failure". These three threats accounted for 64% of all cloud outage incidents while "Data Loss & Leakage" accounted for 25%. Therefore, guaranteeing the integrity of the data, or data auditing, in cloud is a highly desirable security requirement for secure cloud storage.

Traditional cryptographic approaches to checking data integrity such as hash functions or digital signatures are not directly applicable for cloud storage since a copy of the original file is required in the verification procedure of these primitives. Downloading entire file from cloud is not economical because it leads to unacceptable communication cost. In comparison with traditional cryptographic techniques for data integrity, secure auditing is much more practical for verifying data storage correctness. In 2007, Ateniese et al. proposed the notion of provable data possession (PDP) [7,8] to check the integrity of remote data. Ateniese et al. [9] also demonstrated how to construct data auditing schemes from homomorphic identification protocols. Juels et al. presented the concept of proof of retrievability (POR) [10], in which error-correcting codes and spot-checking technologies are utilized to achieve the properties of possession and retrievability of data files. Shacham and Waters proposed compact proofs of retrievability [11]. One of their constructions built from the BLS signature and secure in the random oracle model achieves the shortest query and response among all the proof-of-retrievability schemes with public verifiability. Because of the elegant properties of BLS signature [12,13], this construction was widely employed to build auditing schemes with additional properties, such as privacy-preserving public auditing [14–17,20], data dynamic auditing [14,18,19,21,22], efficient user revocation [23–26], etc. Note that Wang et al. [26] proposed a novel public auditing mechanism for the integrity of shared data with efficient user revocation using the similar primitive called proxy re-signatures. While employing similar tools, our idea of making use of proxy re-signature is different from that of Wang et al.

In Wang et al.'s scheme [26], it is the cloud server who re-signs the blocks since the cloud server has the re-signing key. However, in our security model, we assume the cloud server is not fully trusted and the key update is to be conducted by the user. In particular, a user may choose to update his key since it could be compromised, we assume the attacker (which is the cloud server in the model), can obtain the user's old key (say  $sk$ ). This security requirement makes construction in our model non-trivial. Specifically, it rules out the straightforward solution of releasing a re-signing key defined as  $\Delta = sk'/sk$ , where  $sk'$  is the new user secret key, for many of the discrete logarithm based scheme including the aforementioned schemes from BLS signature. The reason is that the cloud server can find out the user's new key ( $sk'$ ) from  $sk$  and  $\Delta$  easily. To date, there are two main mechanisms for data auditing, namely mechanisms based on public key infrastructure (PKI) and those based on pseudorandom functions (PRF). The proposals based on PRF are privately verifiable, which means only the data owner can check the integrity of the outsourced data. In comparison, auditing schemes with public verifiability allow external auditors to verify the data integrity on-demand. In this paper, we consider publicly verifiable auditing schemes only, as they are more practical in many applications. The key technique to realize public verifiability is the homomorphic authenticators introduced by Ateniese et al. [7].

In the PKI system, a certificate authority (CA) is involved, whose primary role is to digitally sign and issue certificates for certified users. Public-key cryptography makes use of certificates to address the issue of impersonation. A certificate is an electronic document used to associate the identity of an individual, a company or any other entity with a public key for a specified validity period. A critical problem in PKI is how to deal with the problem of user secret key exposure or expiration. Key expiration indicates certificate of a user is no longer valid while key exposure brings serious security threat to a valid user. As a consequence, for practical purpose, certificate revocation and re-issuing are critical aspects of maintaining the security of the PKI which underpins secure communication on the internet. A certificate may be worth revoking when it has had its private key compromised, the owner of the certificate no longer controls the domain for which it was issued, or the certificate was mistakenly signed. Without the ability to revoke a certificate, a CA has no direct means of marking a certificate as untrusted before the expiry of the certificate, which could be a few years away. In cloud storage, when a user needs to change his/her public key due to various reasons, a dilemma arises on how to verify the integrity of the outsourced data hosted in the cloud since old authenticators stored in the cloud are not valid any longer under the updated public key. A trivial solution for this problem is downloading all the file blocks and authenticators from the cloud, re-computing the authenticators for each block and uploading the blocks and authenticators to the cloud again. This approach will lead to unacceptable communication cost on both the cloud user and the cloud server and at the same time, bring significant computation cost on the cloud users. Addressing the issue

of efficient key-updating and authenticator-evolving in data auditing protocols is highly essential to make cloud storage really practical.

Another important issue in cloud data auditing is privacy, including identity privacy and data privacy. Wang et al. [27,28] developed ring signatures and group signatures based authenticators to realize that the identity of the signer on each block in shared data is kept private from public verifiers. Data privacy requires that the auditing process should not reveal knowledge of the challenged files to the third party auditor. Note that merely encrypting the data is not a viable solution, since the data on the cloud are usually non-static. Homomorphic encryption schemes may solve this issue in theory but current schemes are far from practical. The early auditing schemes are not privacy preserving. In response to a challenge, the cloud may release  $\mu_i = \sum_{(i,v_i)} v_i m_i$  where  $m_i$  are the challenged blocks and  $v_i$  are the challenge generated by the auditor. Hence, each  $\mu_i$  reveals partial information of the data block. By challenging those block repetitively, the auditor will learn the data. The first privacy-preserving public auditing scheme was proposed by Wang et al. in 2010 [16]. They modified the response in Shacham-Waters scheme by “blinding” the linear combination of the sampled file blocks, such that the auditor cannot recover the file blocks from the responses sent by the cloud server. Unfortunately, Xu et al. [29] found that the protocol is vulnerable to attacks by a malicious cloud server and an outside attacker. Subsequently, Wang et al. improved their scheme [17] such that it is secure against forgery attack. We argue that in some real world applications, it is not sufficient that the auditor cannot derive the whole file blocks. For example, the auditor can launch an offline guessing attack to learn which file among a number of files is stored on the cloud. Wang et al. proposed the notion of “zero knowledge public auditing” to resist off-line guessing attack. However, as a key point to capture this kind of privacy, a formal security model is missing in their work. Yu et al. [30] recently enhanced the privacy of remote data integrity checking protocols for secure cloud storage, but their model does not cover the key update scenario.

**Contributions.** In this paper, we formalize the security model of “zero knowledge data privacy” for auditing protocols supporting key update, and propose a technique to settle the key update problem efficiently. To be more specific, we utilize a simple zero-knowledge proof of discrete logarithm to make Shacham-Waters protocol [11] reveal no any knowledge of the outsourced data to the verifier in the context of key update. At the same time, the cloud user does not need to download his data from the cloud when his key changes or expires. Instead, the user downloads only the authenticators, which are much shorter in size compared with the entire file. The user can efficiently evolve authenticators, which reduces the communication and computation overheads drastically. We also prove soundness of the new protocol in the random oracle model under a new and reasonable security model. We finally analyze the complexity of the our construction.

## 2 Preliminaries

In this section, we review the publicly verifiable cloud storage system model and formalize the security model of data auditing protocols with key update in cloud storage.

### 2.1 System Model of Data Auditing with Key Update

The cloud data auditing architecture with public verifiability [7, 11] is illustrated in Fig. 1. Three kinds of entities are involved in the scenario, namely cloud users or data owners, the cloud server and a third party auditor (TPA). A cloud user generates data files and stores large amount of data on the remote cloud server without keeping a local copy. The cloud server has significant storage space and computation resources and provides data storage services for cloud users. TPA can be an organization managed by the government, which has expertise and capabilities that cloud users do not have and is trusted to check the integrity of the hosted data on behalf of cloud users upon request. Each entity has his own obligations and benefits. The cloud server may be self-interested, and for his own benefits, such as to maintain its reputation, the server might even decide to hide data corruption incidents to others. Moreover, in the current time period, the cloud server can obtain the data owner's secret key in previous time periods. However, we assume that the cloud server has no incentives to reveal the hosted data to TPA because of regulations and financial incentives. TPA is responsible for checking the integrity of the cloud data on behalf the cloud users in case that they have no time, resources or feasibility to monitor their data, and returns the auditing report to the cloud user. In an auditing scheme with zero knowledge privacy, the TPA cannot learn any information of the stored data during the auditing process.

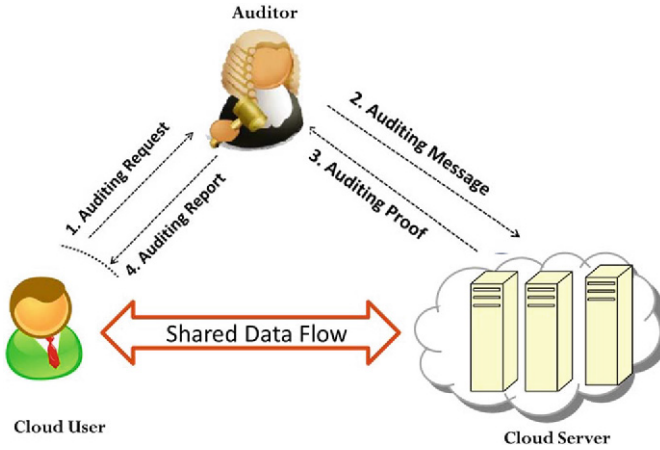
### 2.2 System Components and Its Security

A public auditing scheme with key-updating and authenticator-evolving is a collection of four algorithms, namely,  $\text{CrsGen}$ ,  $\text{KeyGen}$ ,  $\text{AuthGen}$ ,  $\text{AuthUpdate}$ , and an interactive proof system  $\text{Proof}$  between a prover and a verifier.

**$\text{CrsGen}(1^k)$ :** This algorithm takes as input a security parameter  $k$  and outputs a common reference string  $\text{crs}$ , which is an implicit input to all algorithms described below.

**$\text{KeyGen}(\text{crs})$ :** On input  $\text{crs}$ , the algorithm generates a public key  $pk$  and a secret key  $sk$  for the cloud user. The user publishes  $pk$  and keeps  $sk$  secret. Note that this algorithm is also used for key update.

**$\text{AuthGen}(sk, F)$ :** It takes as input the secret key  $sk$  and a file  $F = (m_1, \dots, m_n)$ , and outputs a set of authenticators  $\{D_i\}$  for this file and a set of public verification parameter  $\phi$ , which will be used for checking the data integrity in the proof phase.



**Fig. 1.** System model of publicly verifiable data auditing

**Proof**( $\mathcal{P}, \mathcal{V}$ ): This is an interactive protocol between the prover ( $\mathcal{P}$ ) and the verifier ( $\mathcal{V}$ ). The common input to ( $\mathcal{P}, \mathcal{V}$ ) is the public key  $pk$  and the public verification parameter  $\phi$ .  $\mathcal{P}$  has additional input the file  $F = (m_1, \dots, m_n)$  and a set of authenticators  $\{D_i\}$  of this file. At the end of the protocol,  $\mathcal{V}$  outputs a bit 1 or 0 to indicate if the stored file is kept intact or not. For notational convenience, we use  $\mathcal{P} \Leftrightarrow \mathcal{V}(pk, \phi) = 1$  to indicate that  $\mathcal{V}$  outputs 1 at the end of the interaction with  $\mathcal{P}$ . We omit the parameters  $(pk, \phi)$  when the context is clear.

**AuthUpdate**( $sk, pk, sk', pk', \{D_i\}, \phi$ ): On input a new key pair  $(pk', sk')$  and a set of authenticators  $\{D_i\}$  valid under the old key pair  $(pk, sk)$ , and the public verification parameter  $\phi$ , this algorithm outputs a set of evolved authenticators  $\{D'_i\}$  and parameter  $\phi'$  that are valid under the new key pair.

Completeness, soundness and data privacy are three security requirements for the proposed privacy preserving public auditing scheme. Completeness means that when interacting with the cloud server who keeps the data unchanged, the interactive protocol Proof will always result in  $\mathcal{P} \Leftrightarrow \mathcal{V} = 1$  when the cloud server and the TPA follow the protocol honestly.

Soundness says that any prover who can convince a verifier it is storing the data file is actually storing that file. Based on the security model due to Shacham and Waters [11], in the following, we define the security model for a public auditing scheme with key-updating and authenticator-evolving against a malicious server, wherein two entities are involved, i.e., an adversary who behaves as the untrusted server and a challenger who represents a data owner or the auditor. The goal of the adversary is to deceive the auditor, that is, generating a valid response without storing the original file. The key difference between our model and that of the previous PDP or PoR models is the capturing of key-updating and authenticator-evolving operations.

**Soundness.** To capture soundness, we first recap the definition of an  $\epsilon$ -admissible prover defined in [11]. An algorithm  $\mathcal{P}'$  is  $\epsilon$ -admissible with respect to  $(pk, \phi)$  if  $\Pr[\mathcal{P}' \Leftrightarrow \mathcal{V}(pk, \phi) = 1] \geq \epsilon$ . That is, it is able to convincingly answer an  $\epsilon$  fraction of verification challenges from an honest TPA running  $\mathcal{V}$ .

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

- *Init.* Challenger  $\mathcal{C}$  initialises a counter  $\text{cnt} = 0$  and an empty table store. It invokes  $\text{Crsgen}$  on input a security parameter  $k$  to obtain  $\text{crs}$ . It then invokes  $\text{KeyGen}$  to generate a key pair  $(pk_{\text{cnt}}, sk_{\text{cnt}})$ .  $\text{crs}, pk_{\text{cnt}}$  are given to the adversary  $\mathcal{A}$ . We assume  $\mathcal{A}$  can always retrieve the current value of  $\text{cnt}$  and  $\text{store}$ .
- *AuthQuery.* We can safely assume a file  $F$  will be unique for each  $\text{AuthQuery}$ . In case duplicated  $F$  is submitted,  $\mathcal{C}$  can return the result stored in the table store to  $\mathcal{A}$ .  $\mathcal{A}$  submits a file  $F = (m_1, \dots, m_n)$ .  $\mathcal{C}$  runs the  $\text{AuthGen}$  algorithm with input  $(sk_{\text{cnt}}, F)$  to compute corresponding authenticators  $\{D_i\}$  and public verification parameter  $\phi_F$ .  $\mathcal{C}$  adds a row  $(F, \{D_i\}, \phi_F)$  to the table store. Recall that  $\mathcal{A}$  has read access to  $\text{store}$  and thus returned value is omitted.
- *ProofQuery.*  $\mathcal{A}$  can involve itself in a Proof protocol with  $\mathcal{C}$ . Here,  $\mathcal{C}$  will play the role of the TPA ( $\mathcal{V}$ ) with input  $(pk_{\text{cnt}}, \phi_F)$ . The output of  $\mathcal{V}$  is sent to  $\mathcal{A}$  upon termination of the protocol.
- *Update.* When  $\mathcal{A}$  invokes this query,  $\mathcal{C}$  first sets  $\text{cnt} = \text{cnt} + 1$ . Next,  $\mathcal{C}$  invokes  $\text{KeyGen}$  again to obtain a new key pair  $(pk_{\text{cnt}}, sk_{\text{cnt}})$ . For each row  $(F, \{D_i\}, \phi_F)$  in table  $\text{store}$ ,  $\mathcal{C}$  invokes  $\text{AuthUpdate}(sk_{\text{cnt}-1}, pk_{\text{cnt}-1}, sk_{\text{cnt}}, pk_{\text{cnt}}, \{D_i\}, \phi_F)$  to obtain  $\{D'_i\}, \phi'_F$ . Update the row to  $(F, \{D'_i\}, \phi'_F)$ . After that,  $\mathcal{C}$  returns  $sk_{\text{cnt}-1}$  to  $\mathcal{A}$ .
- *Challenge.* At some point,  $\mathcal{A}$  outputs the description of a prover algorithm  $\mathcal{P}^*$ .

We first recap the concept of extractor. An extractor  $\text{Ext}$  is an algorithm which takes as input an  $\epsilon$ -admissible prover  $\mathcal{P}'$ , the public key  $pk$  and the verification parameter  $\phi$ , outputs a file  $F$ . We denote  $\text{Ext}(\mathcal{P}', pk, \phi) = F$ .

Now we are ready to define soundness. Specifically, we require that for all PPT algorithm  $\mathcal{A}$  which outputs an  $\epsilon$ -admissible prover  $\mathcal{P}^*$  with respect to  $(pk_{\text{cnt}}, \phi)$ , there exists an extractor  $\text{Ext}$  such that  $(\text{Ext}(\mathcal{P}^*, pk_{\text{cnt}}, \phi), pk_{\text{cnt}}, \phi)$  is a row in the table store.

In other words, when an algorithm can pass the Proof protocol with probability greater than  $\epsilon$ , it stores the original file in some format. The stored file in whatever format can be recovered back to the original file using the extractor algorithm.

**Zero Knowledge Data Privacy.** This property captures that the TPA learns no knowledge about the content except a random file name of the stored file based on the publicly available information. To further strengthen the property, we allow the TPA to select two equal length files,  $F_0 = (m_{0,1}, \dots, m_{0,n})$  and  $F_1 = (m_{1,1}, \dots, m_{1,n})$  and requires that given a set of meta-data as well as interaction with the cloud server, the TPA cannot obtain any knowledge about the file content of  $F_0$  and  $F_1$ . A formal security model is described as follows.

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

- *Init.* Challenger  $\mathcal{C}$  initialises a counter  $\text{cnt} = 0$ , counter  $j = 1$  and an empty table **store**. It invokes  $\text{CrsGen}$  on input a security parameter  $k$  to obtain  $\text{crs}$ . It then invokes  $\text{KeyGen}$  to generate a key pair  $(pk_{\text{cnt}}, sk_{\text{cnt}})$ .  $\text{crs}, pk_{\text{cnt}}$  are given to the adversary  $\mathcal{A}$ . We assume  $\mathcal{A}$  can always retrieve the current value of  $\text{cnt}$  and **store**.  $\mathcal{C}$  initialises an empty row  $R^* = (0, \perp, \perp, \perp)$ , which is called the challenge row.
- *AuthQuery*( $F$ ). We can safely assume  $F$  will be unique for each  $\text{AuthQuery}$ . In case duplicated  $F$  is submitted,  $\mathcal{C}$  can return the result stored in the table **store**.  $\mathcal{A}$  submits a file  $F = (m_1, \dots, m_n)$ .  $\mathcal{C}$  runs the  $\text{AuthGen}$  algorithm with input  $(sk_{\text{cnt}}, F)$  and obtains corresponding authenticators  $\{D_i\}$  and public verification parameter  $\phi_F$ .  $\mathcal{C}$  adds a row  $(j, F, \{D_i\}, \phi_F)$  to the table **store** and increases  $j$  by one.
- *GenChallenge*. This query can only be issued once. At some point,  $\mathcal{A}$  submits two equal-length files  $F_0$  and  $F_1$  (assuming again they are distinct to all inputs to the  $\text{AuthQuery}$ ).  $\mathcal{C}$  flips a fair coin  $b \in_R \{0, 1\}$  and runs the  $\text{AuthGen}$  algorithm with input  $(sk_{\text{cnt}}, F_b)$  and obtains the corresponding authenticators  $\{D_i\}$  and public verification parameter  $\phi_{F_b}$ .  $\mathcal{C}$  sets the challenge row  $R^*$  to be  $(0, F_b, \{D_i\}, \phi_{F_b})$ .  $\phi_{F_b}$  is returned to  $\mathcal{A}$ .
- *ProofQuery*( $\hat{j}$ ).  $\mathcal{A}$  engages  $\mathcal{C}$  in the  $\text{Proof}$  protocol. Note that  $j$  is specified by  $\mathcal{A}$  and refers to the  $j$ -th row in the table **store** or the challenge row if  $\hat{j} = 0$  (assume  $\text{GenChallenge}$  query has been issued).  $\mathcal{A}$  plays the role of the TPA and  $\mathcal{C}$  plays the role of the prover with input  $(pk_{\text{cnt}}, F, \{D_i\}, \phi_F)$  where  $(j, F, \{D_i\}, \phi_F)$  is a row in table **store** if  $j > 0$  or  $(pk_{\text{cnt}}, F_b, \{D_i\}, \phi_{F_b})$  if  $j = 0$  and the  $\text{GenChallenge}$  query has been issued.
- *Update*. When  $\mathcal{A}$  invokes this query,  $\mathcal{C}$  first sets  $\text{cnt} = \text{cnt} + 1$ . Next,  $\mathcal{C}$  invokes  $\text{KeyGen}$  algorithm again to obtain a new key pair  $(pk_{\text{cnt}}, sk_{\text{cnt}})$ . For each row  $(F, \{D_i\}, \phi_F)$  in table **store**  $\cup R^*$ ,  $\mathcal{C}$  invokes  $\text{AuthUpdate}(sk_{\text{cnt}-1}, pk_{\text{cnt}-1}, sk_{\text{cnt}}, pk_{\text{cnt}}, \{D_i\}, \phi_F)$  to obtain  $\{D'_i\}, \phi'_F$ . Update the row to  $(F, \{D'_i\}, \phi'_F)$ . After that,  $\mathcal{C}$  returns  $sk_{\text{cnt}-1}$  to  $\mathcal{A}$ .
- *Guess*. At some point,  $\mathcal{A}$  outputs a guess bit  $b'$ .

$\mathcal{A}$  wins the game if  $b = b'$ . The advantage of  $\mathcal{A}$  is the probability that it wins in the aforementioned game minus  $0.5^1$ . A public cloud data auditing scheme is called zero knowledge data private if the advantage of any polynomial-time adversary  $\mathcal{A}$  is negligible.

### 2.3 Zero-Knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge protocol [31] enables a prover ( $\mathcal{P}$ ) to convince a verifier ( $\mathcal{V}$ ) that some statement is true but the verifier learns nothing except the validity of the statement. In the following, we review a simple but powerful zero-knowledge proof of discrete logarithm instantiation introduced by

<sup>1</sup> This is to offset the winning probability based on random guessing.



Schnorr in his identification scheme [32], which plays an important role in our protocol. This zero-knowledge proof of knowledge allows  $\mathcal{P}$  to prove the knowledge of  $x \in Z_p$  such that  $y = g^x$  for some  $y \in G$  to  $\mathcal{V}$ .

**Commitment.**  $\mathcal{P}$  picks a random  $\rho \in Z_p$ , computes  $T = g^\rho$  and sends  $T$  to  $\mathcal{V}$ .

**Challenge.**  $\mathcal{V}$  selects a random  $c \in \{0, 1\}^\lambda$  and sends  $c$  back to  $\mathcal{P}$ .

**Response.**  $\mathcal{P}$  computes  $z = \rho - cx \pmod{p}$  and returns  $z$  to  $\mathcal{V}$ .

**Verify.**  $\mathcal{V}$  accepts the proof if and only if  $T = y^c g^z$ .

### 3 Our Construction

Our cloud data auditing protocol derives from compact proofs of retrievability due to Shacham and Waters [11]. Their construction utilizes a homomorphic authenticator based on the short signature scheme [12], which leads to the shortest query and response of any proof of retrievability with public verifiability. For the key-changing and authenticator-evolving, we are inspired by the idea of proxy re-signature [33], which enables a semi-trusted proxy to transform Alice's signature on a message  $m$  into Bob's signature on the same message  $m$ , to evolve the user's authenticators without downloading the file when his/her key changes. Regarding data privacy, we make use of zero knowledge proof of a discrete logarithm [34], such that both the aggregate authenticator  $\sigma$  and the combinations  $\mu_i$  of the challenged blocks in the response are randomized. As a consequence, the TPA learns no information of the content of the stored file except a file name randomly picked by the data owner<sup>2</sup>. The details of the protocol are as follows.

**CrsGen**( $1^k$ ). On input a security parameter  $\lambda$ , this algorithm outputs a large prime  $p$  and  $G, G_T$ , two multiplicative cyclic groups of the same order  $p$ .  $g$  is a generator of  $G$ .  $e : G \times G \rightarrow G_T$  denotes a bilinear map and  $H_0, H_1 : \{0, 1\}^* \rightarrow G$  represent two collision resistant cryptographic hash functions. In addition, this algorithm picks randomly  $h, u_1, u_2, \dots, u_s \in G$  and computes  $\eta = e(g, h)$ . The common reference string  $\text{crs}$  is  $(k, p, G, G_T, g, e, H_0, H_1, h, u_1, \dots, u_s, \eta)$ .

**KeyGen**( $\text{crs}$ ). On input the common reference string  $\text{crs}$ , a data owner (cloud user) generates a signing key pair  $(\text{spk}, \text{ssk})$ ,  $\text{spk} = g^{\text{ssk}}$  and another key pair  $(\alpha, v)$  for generating authenticators of file blocks, where  $\alpha \in Z_p$  and  $v = g^\alpha$ . The secret key of the data owner is  $\text{sk} = (\alpha, \text{ssk})$  and the public key is  $\text{pk} = (\text{spk}, v)$ .

For notational convenience, we use  $\eta_i$  to represent  $e(u_i, v)$  for  $i = 1$  to  $s$ . Note that  $\eta_1 = e(u_1, v), \dots, \eta_s = e(u_s, v)$  can be pre-computed by the relevant parties given the public key  $v$  and  $\text{crs}$ .

**AuthGen**( $\text{sk}, F$ ). Given a file  $F$ , the data owner firstly applies erasure codes such as RS code to obtain a processed file  $F'$ , and splits  $F'$  into  $n$  blocks. Each block is further fragmented into  $s$  sectors  $\{m_{ij}\}_{1 \leq i \leq n, 1 \leq j \leq s}$ , which is an element of  $Z_p$ . The data owner selects a file name  $F_n$  from a sufficiently large domain.

<sup>2</sup> This is reasonable since the data owner tells TPA which file will be audited in auditing request phase.

Let  $t_0$  be  $F_n||n$ . The data owner computes  $t = (H_0(t_0))^{ssk}$  and denotes the file tag  $ft = t_0||t$ . Then for each  $i, 1 \leq i \leq n$ , the owner computes an authenticator  $\sigma_i$  for block  $i$  as

$$\sigma_i = (H_1(F_n||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^\alpha.$$

Finally, the data owner stores

$$ft || \{m_{ij}\}_{(1 \leq i \leq n, 1 \leq j \leq s)} || \{\sigma_i\}_{1 \leq i \leq n}$$

to cloud. (Note that there exists strict access control policies that who can access the stored files and authenticators.) **Proof**( $P(F, \{\sigma_i\}, ft), V(pk)$ ). This is a 5-move interactive proof protocol executed between a prover (cloud server) and a verifier (TPA) as follows.

1. The TPA picks a random integer  $c$  and  $k, \psi \in Z_p$ , computes  $\Psi = g^k h^\psi$ . For  $1 \leq i \leq c$ , the TPA selects a random  $v_i \in Z_p$ . The commitment  $\Psi$  and the challenge  $chal = \{i, v_i\}_{1 \leq i \leq c}$ , which locates the positions of the challenged blocks in this auditing process, are sent to the cloud server.
2. Upon receiving  $(chal, \Psi)$ , the cloud server firstly chooses  $r, \rho_r, \rho_1, \dots, \rho_s \in Z_p$  randomly, then computes

$$II = \prod_{(i, v_i) \in chal} \sigma_i^{v_i} \cdot h^r, T = \eta^{\rho_r} \eta_1^{\rho_1} \dots \eta_s^{\rho_s}$$

and forwards  $(T, II)$  to the TPA.

3. The TPA sends  $(k, \psi)$  to the server.
4. The server checks if  $\Psi \stackrel{?}{=} g^k h^\psi$ . If the equation does not hold, the server aborts. Otherwise, it computes  $z_r = \rho_r - kr, \mu_i = \sum_{(i, v_i) \in chal} v_i m_{ij}, z_i = \rho_i - k\mu_i (1 \leq i \leq s)$  and sends  $(z_r, z_1, \dots, z_s)$  to the TPA.
5. The TPA verifies the file tag  $ft$  firstly by checking if the following equation holds,

$$e(g, t) = e(sp_k, H_0(t_0)).$$

If the verification fails, reject by emitting *False*. Otherwise, the TPA verifies if

$$\left( \frac{e(II, g)}{e\left(\prod_{(i, v_i) \in chal} H_1(F_n||i)^{v_i}, v\right)} \right)^k \stackrel{?}{=} \frac{T}{\eta^{z_r} \eta_1^{z_1} \dots \eta_s^{z_s}}.$$

**KeyUpdate**( $pk, sk$ ). The data owner can change his/her key pair  $(sk, pk)$  by generating a new key and as a consequence, the data owner has a updated public key  $pk' = (sp_{k'}, v')$  and secret key  $sk' = (\alpha', ssk')$  where  $sp_{k'} = g^{ssk'}$  and  $v' = g^{\alpha'}$ .

**AuthEvolve**( $pk, sk, pk', sk', ft, \sigma_i$ ). The data owner downloads  $ft||\{\sigma_i\}_{1 \leq i \leq n}$  from the cloud and evolves the file tag and block authenticators as follows.

1. Compute  $t' = t^{\frac{ssk'}{ssk}}$  and let  $ft' = t_0||t'$ .
2. Compute  $\sigma'_i = \sigma_i^{\frac{\alpha'}{\alpha}}$  for  $1 \leq i \leq n$ .
3. Upload  $ft'||\{\sigma'_i\}_{1 \leq i \leq n}$  to the cloud.

## 4 Security of the New Protocol

In this section, we show the proposed protocol achieves the properties of completeness, soundness and zero-knowledge privacy. The key update and authenticator evolving are valid as well. Completeness shows the correctness of the protocol and soundness guarantees the security against an untrusted server.

### 4.1 Completeness

The correctness of the scheme before key update is straightforward to verify with the properties of bilinear pairings. Below we demonstrate the verification still works after the key pair changes. If both the data owner and the server are honest, we have

$$\begin{aligned} t' &= t^{\frac{ssk'}{ssk}} \\ &= ((H_0(t_0))^{ssk})^{\frac{ssk'}{ssk}} \\ &= H_0(t_0)^{ssk'} \end{aligned}$$

Thus,  $e(g, t') = e(sp'k', H_0(t_0))$  holds.

Regarding the evolved authenticators,

$$\begin{aligned} \sigma'_i &= \sigma_i^{\frac{\alpha'}{\alpha}} \\ &= ((H_1(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^{\alpha})^{\frac{\alpha'}{\alpha}} \\ &= (H_1(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^{\alpha'} \end{aligned}$$

Accordingly,

$$e(\sigma', g) = e\left(\prod_{(i, v_i) \in chal} H_1(Fn||i)^{v_i} \cdot \prod_{j=1}^s u_j^{\mu_j}, v'\right)$$

holds as well.

## 4.2 Soundness

An auditing scheme has the property of soundness if any cheating prover who can convince a verifier that it is storing a file  $F$  is actually storing that file. In other words, from the cheating prover, there exists an extractor algorithm to extract the file blocks. The soundness proof of our protocol depends on the soundness of the protocol from Shacham and Waters. Specifically, we prove that if there is an adversary who can violate the soundness of our construction, we can construct another algorithm to break the soundness of the Shacham and Waters scheme [11] (referred to as SW scheme hereafter).

*Proof.* (Sketch) Suppose there is an adversary  $\mathcal{A}$  that can break the soundness of our scheme, we construct a simulator  $\mathcal{B}$  that can break the soundness of SW scheme.

- $\mathcal{B}$  is given a public key of the SW scheme. The value  $h$  will be generated by  $\mathcal{B}$  in such a way that the discrete logarithm of  $h$  to the base  $g$ , an element of the public key of the SW scheme, is known.  $h$  and the corresponding elements in  $pk^*$  is treated as the crs. Only the component  $(v, ssk)$  in the public key of the SW scheme is treated as  $pk^*$ . Let  $n$  be the number of update query made by  $\mathcal{A}$ .  $\mathcal{B}$  picks a random index  $\hat{i} \in \{1, \dots, n\}$  and set  $pk_{\hat{i}} = pk^*$ . For  $i \in \{1, \dots, n\} \setminus \{\hat{i}\}$ ,  $\mathcal{B}$  picks a random  $sk_i$  and computes  $pk_i = g^{sk_i}$ .  $pk_1$  is given to  $\mathcal{A}$  as the first public key of the system.
- (Queries at period other than  $\hat{i}$ ). For period not equal to  $\hat{i}$ ,  $\mathcal{B}$  is in possession of the private key and can thus answer all the queries from the adversary.
- (Update query from  $cnt = \hat{i} - 1$ ). When  $\mathcal{A}$  invokes the update query when  $cnt = \hat{i} - 1$ ,  $\mathcal{B}$  does not have the corresponding secret key  $sk_{\hat{i}}$  and will thus answer the query by re-computing all the authenticators using the AuthQuery to the underlying SW scheme. Note that the resulting authenticators are distributed as if it is computed from the update query.
- (Update query from  $cnt = \hat{i}$ ). When  $\mathcal{A}$  invokes the update query when  $cnt = \hat{i}$ ,  $\mathcal{B}$  answers the query by re-computing all the authenticators using the secret key  $sk_{\hat{i}+1}$ . The resulting authenticators are distributed as if it is computed from the update query.
- (Output). Finally,  $\mathcal{A}$  outputs a prover  $P^*$ . With probability  $1/n$ ,  $P^*$  is output during the period  $\hat{i}$ .  $\mathcal{B}$  aborts otherwise. Now we show how  $\mathcal{B}$  outputs a prover  $P'$  to the underlying SW scheme.
- (Construction of  $P'$ ). In an execution of  $P^*$ ,  $\mathcal{B}$  obtains  $(T, \Pi, k, z_r, z_1, \dots, z_s)$ .  $\mathcal{B}$  rewinds  $P^*$  to the third step and replies with a different value  $k'$  (this is possible since  $\mathcal{B}$  knows the discrete logarithm of  $h$  to base  $g$  and can thus provide the corresponding value  $\psi'$ ). Now the protocol finishes with a different transcript  $(T, \Pi, k', z'_r, z'_1, \dots, z'_s)$ . From these two set of equations,  $\mathcal{B}$  can compute the underlying values  $(r, \mu_1, \dots, \mu_s)$ .  $\mathcal{B}$  computes  $\sigma = \Pi/h^r$  and outputs  $(\sigma, \mu_1, \dots, \mu_s)$  on behalf of the prover  $P'$ . Note that if  $P^*$  is  $\epsilon$ -admissible,  $P'$  is  $\epsilon^2$ -admissible to the underlying SW scheme. In other words, if SW scheme is sound, our scheme is sound too.

### 4.3 Zero Knowledge Privacy

The new protocol achieves zero knowledge privacy. That is, the data auditing process does not leak any information of the outsourced data. To prove this property, we construct a simulator  $\mathcal{S}$  for the interaction between the data owner and the cloud server.

*Proof.* (Sketch) We show that the probability that  $\mathcal{A}$  can output the guess bit correctly with probability negligibly close to  $1/2$ . The idea is to demonstrating how  $\mathcal{A}$  is given a challenge that is independent of the underlying file  $F_0$  or  $F_1$ . Recall that the two files are of the same length and will be using the same name  $fn$  since the number of blocks and the filename are supposed to be known to the TPA.

To simulate a proof query related to the challenge file,  $\mathcal{S}$  follows the protocol honestly until step 3 upon receiving the values  $(k, \psi)$ .  $\mathcal{S}$  rewinds  $\mathcal{A}$  to step 2 and picks a random  $\Pi, z_r, z_1, \dots, z_s$  and computes

$$T = \left( \frac{e(\Pi, g)}{e\left(\prod_{(i, v_i) \in chal} H_1(Fn || i)^{v_i}, v\right)} \right)^{k \eta^{z_r} \eta_1^{z_1} \dots \eta_s^{z_s}}.$$

$\mathcal{S}$  sends  $T, \Pi$  to  $\mathcal{A}$ . Now  $\mathcal{A}$  returns with the same  $(k, \psi)$  since this pair is fixed by its first message  $\Psi$ .  $\mathcal{S}$  replies with  $(z_r, z_1, \dots, z_s)$ .

It can be seen that the value passes the verification and is distributed correctly. In addition, the view of  $\mathcal{A}$  is independent to  $F_0$  and  $F_1$  and thus the probability that  $\mathcal{A}$  answers correctly must be  $1/2$ .

## 5 Complexity Analysis

In this section, we report the complexity analysis of communication, computation and storage costs of the improved protocol.

### 5.1 Parameter Selection

The typical selection of the security parameter  $\lambda$  is 80. Due to the public verification of the proposed protocol,  $p$  should be a  $2k = 160$ -bit prime, and the elliptic curve should be chosen so that discrete logarithm is  $2^k$ -secure. For values of  $\lambda$  up to 128, pairing-friendly elliptic curves of prime order due to Barreto and Naehrig [35] can be employed.  $n \gg k$  denotes the number of blocks in a file. Regarding the choice of erasure codes, we follow the suggestions of the SW scheme. That is, traditional RS style erasure codes can be applied to our construction, but the encoding the decoding procedures take  $O(n^2)$  time. For the server that is not malicious, a system code, in which the first  $m$  blocks of the encoded file are the file itself, can be used for much more efficient public retrievability.

## 5.2 Performance Analysis

**Communication Cost.** In the Proof phase, the TPA sends  $\Phi$  and  $chal$  to the cloud server, which is of binary length  $\log_2 c + (c+1)\log_2 p$ . We can shorten this challenge dramatically by selecting a pseudo-random permutation to compute the locations  $i$  of the challenged blocks and a pseudo-random function to calculate the random challenge values  $v_i$ . In this circumstance, the TPA sends only keys of the pseudo-random permutation and pseudo-random function, which is of  $\log_2 p$  bits each. In the second step, the cloud server returns two points of elliptic curves,  $T$  and  $\Pi$ , to the TPA, which is of 320 bits. In the third step, the TPA sends  $(k, \phi)$  to the server, which is of binary length  $2\log_2 p$ . In the next step, the cloud server sends  $(z_r, z_1, \dots, z_s)$  to the TPA, which is of binary length  $(s+1)\log_2 p$ .

**Storage Cost.** In terms of the storage cost of the protocol, both the files and the corresponding metadata including the file tag and block authenticators need to be stored on cloud server due to public verifiability. Our scheme enjoys the advantage of flexible tradeoff between storage and communication in [11]. That is, a parameter  $s$  is used to give a tradeoff between response length and storage overhead. Each block is composed of  $s$  elements of  $Z_p$  called sectors. Each block instead of each sector has an authenticator, reducing the storage overhead to  $(1 + \frac{1}{s})\times$ . The data owner only needs to store the public key  $pk = (spk, v)$ , the private key  $sk = (\alpha, ssk)$ , so the storage cost of the data owner is almost  $2\log_2 p + 320$  bits. The TPA needs to store the public key of a user, which is of binary length 320 bits. During the auditing process, the TPA stores  $k, \phi, chal, \Phi, T, z_r, z_1, \dots, z_s$  for validating a response from the server, which is of binary length  $(c+s+3)\log_2 p + 320$  bits in total.

**Computation Cost.** We report the computation cost from the viewpoint of the data owner, the cloud server and the TPA. We only consider the expensive operations including bilinear maps, exponentiations and multiplications in  $G$  and  $G_T$ .  $T_{pair}$  denotes the time cost of computing a bilinear map of two points of an elliptic curve, and ignore some highly efficient operations, say, computing a hash function.  $T_{expG}$  and  $T_{expGT}$  stand for the time cost of an exponentiation in  $G$  and  $G_T$  respectively.  $T_{mulG}$  and  $T_{mulGT}$  denote the time overhead of a multiplication in  $G$  and  $G_T$  respectively.

The dominating computation of the data owner is generating authenticators for file blocks as  $\sigma_i = (H_1(Fn||i) \cdot \prod_{j=1}^s u_j^{m_{ij}})^\alpha$ , in which the time cost is  $(s+1)T_{expG} + sT_{mulG}$  for one authenticator. Regarding the authenticators evolving, the data owner needs to perform  $n+1$  exponentiations in  $G$ , and the time cost is  $(n+1)T_{expG}$ . As a consequence, the main computation cost of the data owner during the protocol is  $(sn+2n+1)T_{expG} + nsT_{mulG}$ . To generate and verify a proof, the TPA needs to compute  $\Phi$  and validate the file tag and the response, and thus the total computation cost of the TPA is  $(4T_{pair} + (c+2)T_{expG} + (s+2)T_{expGT})$ . To produce a proof, the cloud server has to compute  $\Pi, T$  and  $z_r, z_1, \dots, z_s$ , the primary computation cost of the server is  $(c+2)T_{expGT} + (c+1)T_{mulGT} + (s+1)T_{expGT} + sT_{mulGT}$ .

## 6 Conclusion

In this paper, we investigate two important issues of secure cloud data auditing to make cloud storage more practical: (1) how to evolve the authenticators of outsourced files efficiently when a cloud user's key changes, and (2) how to preserve the privacy of the stored files in auditing protocols with key update. We formalized the security model of soundness and zero knowledge data privacy for cloud data auditing process supporting key update. We also provided a concrete construction by cooperating the well-known Shacham-Waters scheme [11] and several novel cryptographic techniques. We proved the security including soundness and zero knowledge privacy of the proposed scheme in the new security model. The performance analysis shows that the new scheme is efficient and can be used in practice.

**Acknowledgements.** This work is supported by the Fundamental Research Funds for the Central Universities under Grant ZYGX2015J059.

## References

1. Mell, P., Grance, T.: Draft NIST working definition of cloud computing, 3 June 2009. <http://csrc.nist.gov/groups/SNC/cloud-computing/index.html>
2. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: Proceedings of the 33rd International Conference on Very Large Databases (VLDB 2007), pp. 782–793 (2007)
3. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generat. Comput. Syst.* **25**(6), 599–616 (2009)
4. Cloud Security Alliance. Top threats to cloud computing (2010)
5. Yang, K., Jia, X.H.: Data storage auditing service in cloud computing: challenges, methods and opportunities. *World Wide Web* **15**(4), 409–428 (2012)
6. Common Sense Alliance (2010). [http://www.cert.uy/wps/wcm/connect/975494804fdf89eaabbdab1805790cc9/Cloud\\_Computing\\_Vulnerability\\_Incidents.pdf/?MOD=AJPERES](http://www.cert.uy/wps/wcm/connect/975494804fdf89eaabbdab1805790cc9/Cloud_Computing_Vulnerability_Incidents.pdf/?MOD=AJPERES)
7. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X.: Provable data possession at untrusted stores. In: Proceedings of ACM Conference on Computer, Communications Security 2007, pp. 598–609 (2011)
8. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z.N.J., Song, D.: Remote data checking using provable data possession. *ACM Trans. Inf. Syst. Secur.* **14**, 1–34 (2011)
9. 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)
10. Juels, A., Pors, Jr., B.S.K.: Proofs of retrievability for large files. In: Proceedings of CCS 2007, Alexandria, VA, USA, pp. 584–597. ACM, November 2007
11. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)

12. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
13. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. *J. Cryptol.* **17**, 297–319 (2004)
14. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009)
15. Cui, H., Mu, Y., Au, M.H.: Proof of retrievability with public verifiability resilient against related-key attacks. *IET Inf. Secur.* **9**(1), 43–49 (2015)
16. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: Proceedings of IEEE INFOCOM 2010, San Diego, CA, pp. 525–533 (2010)
17. Wang, C., Chow, S.S., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **62**, 362–375 (2013)
18. Lier, S., Wörsdörfer, D., Gesing, J.: Business models and product service systems for transformable, modular plants in the chemical process industry. In: Meier, H. (ed.) Product-Service Integration for Sustainable Solutions. LNPE, vol. 6, pp. 227–238. Springer, Heidelberg (2013)
19. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **24**(9), 1717–1726 (2013)
20. Wang, H.: Proxy provable data possession in public clouds? *IEEE Trans. Serv. Comput.* **6**(4), 551–559 (2013)
21. Barsoum, A.F., Hasan, M.A.: Provable multicopy dynamic data possession in cloud computing systems. *IEEE Trans. Inf. Forensics Secur.* **10**(3), 485–497 (2015)
22. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: ACM CCS, pp. 325–336 (2013)
23. Yuan, J., Yu, S.: Public integrity auditing for dynamic data sharing with multi-user modification. *IEEE Trans. Inf. Forensics Secur.* **10**(8), 1717–1726 (2015)
24. Wang, B.Y., Li, B.C., Li, H.: Public auditing for shared data with efficient user revocation in the cloud. In: INFOCOM, pp. 2904–2912 (2013)
25. Yuan, J.W., Yu, S.C.: Efficient public integrity checking for cloud data sharing with multi-user modification. In: IEEE INFOCOM, pp. 2121–2129 (2014)
26. Wang, B.Y., Li, B.H., Li, H.: Panda: public auditing for shared data with efficient user revocation in the cloud. *IEEE Trans. Serv. Comput.* **8**(1), 92–106 (2015)
27. Wang, B.Y., Li, B.C., Li, H.: Oruta: privacy-preserving public auditing for shared data in the cloud. *IEEE Trans. Cloud Comput.* **2**(1), 43–56 (2014)
28. Wang, B., Li, B., Li, H.: Knox: privacy-preserving auditing for shared data with large groups in the cloud. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 507–525. Springer, Heidelberg (2012)
29. Xu, C., He, X., Abraha-Weldemariam, D.: Cryptanalysis of Wang’s auditing protocol for data storage security in cloud computing. In: Liu, C., Wang, L., Yang, A. (eds.) ICICA 2012, Part II. CCIS, vol. 308, pp. 422–428. Springer, Heidelberg (2012)
30. Yu, Y., Au, M.H., Mu, Y., Tang, S., Ren, J., Susilo, W., Dong, L.: Enhanced privacy of a remote data integrity checking protocol for secure cloud storage. *Int. J. Inf. Secur.* **14**(4), 307–318 (2015)
31. Goldwasser, S., Micali, S., Racko, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18**(1), 186–208 (1989)
32. Schnorr, C.P.: Efficient signature generation by smart cards. *J. Cryptol.* **4**(3), 161–174 (1991)



33. Ateniese, G., Hohenberger, S.: Proxy re-signatures: new definitions, algorithms, and applications. In: Proceedings of ACM Conference on Computer and Communications Security, pp. 310–319 (2005)
34. Camenisch, J.L., Stadler, M.A.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
35. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)

# **Password/QR Code Security**

# Exploiting the Error Correction Mechanism in QR Codes for Secret Sharing

Yang-Wai Chow<sup>1</sup>(✉), Willy Susilo<sup>1</sup>, Guomin Yang<sup>1</sup>, James G. Phillips<sup>2</sup>,  
Ilung Pranata<sup>3</sup>, and Ari Moesriami Barmawi<sup>4</sup>

<sup>1</sup> Centre for Computer and Information Security Research,  
School of Computing and Information Technology, University of Wollongong,  
Wollongong, Australia

{caseyc,wsusilo,gyang}@uow.edu.au

<sup>2</sup> Department of Psychology, Auckland University of Technology,  
Auckland, New Zealand

jphillip@aut.ac.nz

<sup>3</sup> School of Design, Communication and Information Technology,  
University of Newcastle, Callaghan, Australia

ilung.pranata@newcastle.edu.au

<sup>4</sup> School of Computing, Telkom University, Bandung, Indonesia  
mbarmawi@melsa.net.id

**Abstract.** This paper investigates a novel approach to secret sharing using QR codes. The proposed QR code secret sharing approach exploits the error correction mechanism inherent in the QR code structure, to distribute and encode information about a secret message in a number of shares. Each share in the scheme is constructed from a cover QR code, and each share itself is a valid QR code which can be scanned and decoded by a QR code reader. The secret message can be recovered by combining the information contained in the QR code shares. Since each share is a valid QR code, the proposed scheme has the advantage of reducing the likelihood of attracting the attention of potential attackers. In addition, the shares can be distributed via public channels without raising suspicion. Moreover, shares do not have to be stored or transmitted electronically, as QR codes can be distributed via printed media.

**Keywords:** Error correction · QR code · Secret sharing

## 1 Introduction

A secret sharing scheme refers to a method by which a dealer encodes a secret into a number of shares and distributes these shares to a group of participants. Individually, the shares reveal no information about the secret. The secret can only be reconstructed when information from an authorized number of shares is combined [2]. By splitting and encoding a secret into a number of shares, secret sharing overcomes the problem of storing a secret in a single information-carrier, which can easily be lost or damaged [19, 21]. Secret sharing schemes are important tools that have found many applications in cryptography and distributed

computing [2]. In addition, secret sharing is also regarded as a mechanism that can be used to transfer secret information via public channels in cryptography [25]. This paper investigates a novel approach to secret sharing by distributing and encoding a secret into a set of Quick Response (QR) codes.

The concept of secret sharing was first introduced independently by Blakley [3] and Shamir [18]. The secret sharing schemes that they proposed are known as  $k$ -out-of- $n$ , or  $(k, n)$ , threshold schemes. In a  $(k, n)$  secret sharing scheme, a secret is divided into  $n$  shares, where  $n > 1$ , and  $k$ , or more, shares are required to reconstruct the secret. Even complete knowledge of any  $k-1$ , or fewer, shares will reveal no information about the secret. Since its inception, many different secret sharing schemes have been proposed [1, 2, 7, 8, 16, 17, 19–22, 25]. Some schemes require complex numerical computation, whilst others require little or no computation [21]. Of the varying approaches, image sharing can be seen as a subset of the general secret sharing problem as the secret is a concealed image [22]. One of the popular ways of secret image sharing is known as visual cryptography [17]. Visual cryptography is a method of encoding and distributing a binary image into a number of shares, each to be printed on separate transparencies. When the qualified number of shares are stacked together, the human visual system can recover the secret image without the need for any computation.

Each share in visual cryptography looks like a random pattern of pixels. In an extension to visual cryptography, known as extended visual cryptography, each share is encoded using a meaningful cover image [1]. This means that when viewed individually, a meaningful, albeit noisy, image is visible on each share. The advantage of encoding the secret image into shares containing ‘innocent-looking’ meaningful cover images is that it reduces the likelihood of attracting the attention of attackers [20].

While the advantage of these visual secret sharing schemes is that decryption can be performed by the human visual system without any computation, these approaches suffer from a number of problems, including pixel expansion, contrast issues, share misalignment and the visual quality of the reconstructed image [4, 8, 16, 26]. Secret image sharing techniques have also been proposed to share and hide a secret image by distributing and embedding the information required to reconstruct the secret image in a number of digital images, each of which is a meaningful cover image [19]. However, these techniques hide information in digital images. Therefore, decryption must be performed on the digital image shares, which necessitates that the shares must be distributed via electronic means.

**Our Contribution.** This paper introduces a novel approach to secret sharing by distributing and encoding a secret message into a number of QR code shares. The proposed approach exploits QR code error correction redundancy, which is an inherent feature of the QR code structure. The advantage of this approach is that each share is a meaningful QR code, which individually does not reveal the secret message. The secret message can be recovered by combining the information contained in the QR code shares. Since each QR code share can be scanned

and decoded by any standard QR code reader, this means that the shares can be distributed via public channels without raising suspicion. In addition, since QR codes are meant to be scanned by a QR code reader, the shares do not have to be stored or transmitted electronically and can be distributed via printed media. Furthermore, the shares can be constructed using any artistic QR code method as long as it can be scanned and read. Therefore, each QR code share can be constructed using a different artistic QR code scheme in order to increase the secret sharing subterfuge by reducing the likelihood of attracting the attention of potential attackers.

## 2 Related Work

The QR code is a two-dimensional code that was invented by the company Denso Wave [10]. In recent years, QR codes have seen widespread adoption due to its convenience and ease of use, as any smartphone equipped with a camera and QR code reader can retrieve the information encoded within a QR code. The application of the QR code in the area of information security has previously been proposed for a number of different purposes. For example, QR codes have been used for authenticating visual cryptography shares [23], e-voting authentication [11] and for digital watermarking [14].

QR codes have also been used in the area of data hiding and steganography. Among the work conducted in this area, Wu et al. [24] proposed a data embedding approach for hiding a QR code in a digital image. Their purpose was to camouflage the appearance of a QR code in an image so as not to degrade the visual quality of the picture. Huang et al. [12] developed a reversible data hiding approach for images with QR codes. The purpose of their method was to avoid a QR code from degrading the quality of the image or concealing information contained in the image. In their approach, using reversible data hiding a portion of an image is hidden in the rest of the image. This portion is replaced with a QR code. Once the QR code has been scanned, it will be removed from the image and the original image will be restored using the data that was previously hidden in the rest of the image. A nested image steganography scheme was proposed by Chen and Wang [6] using QR codes. In their approach, two types of secret data, in the form of text (lossless) and image (lossy), are embedded into a cover image. The text portion of the secret data is embedded using a QR code. A similar approach was also reported in Chung et al. [9].

Unlike approaches that first convert a secret into a QR code before embedding it into a cover image, Lin et al. [15] introduced a scheme for concealing secret data in a cover QR code. Their approach capitalizes on the error correction property of a QR code in order to conceal secret data. The amount of secret data that can be concealed depends on the QR code version and error correction level that is used. Bui et al. [5] also investigated the problem of hiding secret information in a QR code. They argue that previous approaches to embedding secret messages in QR codes use bit embedding which is vulnerable to modification attacks. Consequently, they propose a method of using Reed-Solomon code and list decoding to hide a secret message in a QR code.

### 3 Background

The International Organization for Standardization (ISO) has established a standard for the QR code (ISO/IEC18004) [13]. This section outlines the basic QR code structure and error correction feature as defined by the ISO standard.<sup>1</sup>

#### 3.1 The QR Code Structure

A QR code symbol is constructed as a two-dimensional array of light and dark squares, referred to as modules. There are forty sizes of QR code symbol versions ranging from version 1 to version 40. Each QR code symbol version is comprised of a different number of modules, and as such different QR code versions give rise to different data capacities. Version 1 is made up of  $21 \times 21$  modules, and each successive version increases by 4 additional modules per side, up to version 40 which is made up of  $177 \times 177$  modules. The appropriate version to use depends on the amount and the type of data (alphanumeric, binary, Kanji or a combination of these) to be encoded as well as the error correction level. The error correction level will be described in Sect. 3.2 to follow.

The QR code structure consists of encoding regions and function patterns [13]. An example of the encoding regions and function patterns for a QR code version 7 symbol is illustrated in Fig. 1. As can be seen from the figure, a QR code symbol is surrounded by an empty region known as the quiet zone, which should have the same reflectance value as the light modules. Function patterns do not encode data, they consist of the finder patterns, separators, timing patterns and alignment patterns. There are three identical finder patterns located at each corner, except the bottom right corner, of the symbol. These are used by a QR code reader to recognize the QR code and to determine the rotational orientation of the symbol. The separators are one module wide, and are constructed from light modules to separate the finder patterns from the encoding region. Timing patterns are alternating light and dark modules used to determine module coordinates in the symbol. Alignment patterns allow QR code readers to compensate for image distortion. Different QR code versions have a different number of alignment patterns.

#### 3.2 Encoding and Error Correction

The encoding region consists of data codewords and error correction codewords, format information and version information (version information is only used in QR code version 7 and above). Message data is encoded as a bit stream which is divided into a sequence of codewords. All codewords are 8-bits in length. The codewords are divided into a number of error correction blocks, based on the QR code version and error correction level, and an appropriate number of error correction codewords are generated for each block. Error correction allows

---

<sup>1</sup> For a comprehensive description of the QR code structure and error correction mechanism, please refer to the ISO standard (ISO/IEC18004) [13].

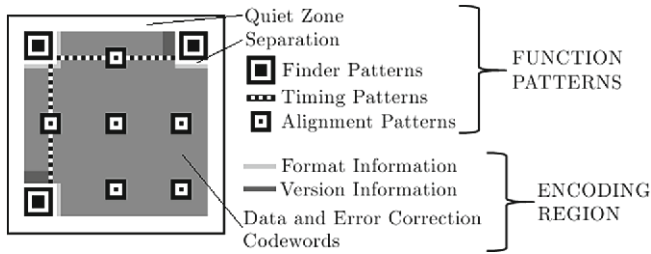


Fig. 1. QR code version 7 structure.

correct decoding of the message in the event that part of the symbol is dirty or damaged. This error correction feature has also been exploited to embed art or other information in QR code symbols.

The QR code employs Reed-Solomon error control coding for error detection and correction [13]. There are four error correction levels that the user can select. Each level provides a different error correction capacity (L  $\sim$  7%, M  $\sim$  15%, Q  $\sim$  25% and H  $\sim$  30%). Higher error correction levels improve the recovery capacity, but also increases the amount of data to be encoded. This means that if the same message were to be encoded using a higher error correction level, a larger QR code version may be required. The number of data codewords, error correction blocks and error correction codewords depend on the QR code version and error correction level. Table 1 shows these characteristics for QR code version 4 and 5. In the table, the error correction codewords for each block is given as  $(c, k, r)$ , where  $c$  is the total number of codewords,  $k$  is the number of data codewords and  $r$  is the error correction capacity. Note that some QR code versions have blocks with different  $(c, k, r)$  values for certain error correction levels. For example, it can be seen in Table 1 that QR code version 5 with an error correction level of Q has a total of 4 error correction blocks. The  $(c, k, r)$  values for the first 2 blocks are (33, 15, 9) while the values for the next 2 blocks are (34, 16, 9).

The codewords from the blocks are encoded in an interleaved manner, with the error correction codewords appended to the end of the data codeword sequence. This is done to minimize the possibility that localized damage will cause the QR code to become undecodable. Figure 2 shows the data codeword and error correction codeword arrangement for QR code version 4, with an error correction level of H. After encoding the codewords, a data mask is applied to the encoding region. There are a total of eight data mask patterns. The purpose of the data mask is to balance the light and dark modules, as well as to minimize the occurrence of undesirable patterns that may potentially confuse a QR code reader. Note that the data mask is not applied to the function patterns.

The format information is a 15-bit sequence consisting of 5 data bits and 10 error correction bits. Two copies of this information are encoded in a QR code symbol, as can be seen in Fig. 1. Of the 5 data bits, the first 2-bits indicate the error correction level and the next 3-bits indicate the data mask pattern used in the QR code symbol. Error correction for the format information is performed

**Table 1.** Error correction characteristics for QR code version 4 and 5 [13].

Version	Total codewords	Error correction level	Number of blocks	Error correction codewords per block <sup>a</sup> (c, k, r)
4	100	L	1	(100, 80, 10)
		M	2	(50, 32, 9)
		Q	2	(50, 24, 13)
		H	4	(25, 9, 8)
5	134	L	1	(134, 108, 13)
		M	2	(67, 43, 12)
		Q	2	(33, 15, 9)
			2	(34, 16, 9)
		H	2	(33, 11, 11)
			2	(34, 12, 11)

<sup>a</sup>c = total number of codewords, k = number of data codewords, r = error correction capacity.

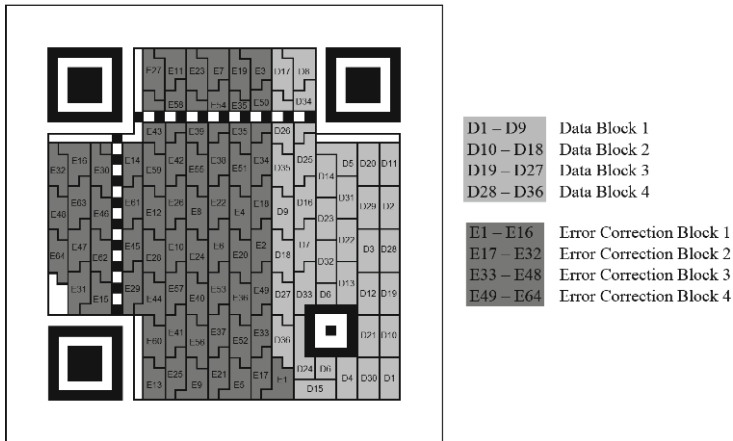
using the Bose-Chaudhuri-Hocquenghem (15, 5) code, which allows for an error of 3-bits, or less, to be corrected. The 15-bit long data bit string is XORed with a specific mask to ensure that the format information bit sequence does not contain all zeros for any combination of data mask pattern and error correction level.

Version information is only contained in QR code version 7 and above. It consists of an 18-bit sequence consisting of 6 data bits and 12 Golay error correction bits. The (18, 6) Golay code allows up to 3-bit errors to be corrected. Its purpose is to convey version information to a QR code reader, for example, for QR code version 7 the 6-bit data string is 000111. QR code version 7 and above contain two copies of this information, as depicted in Fig. 1.

## 4 Proposed QR Code Secret Sharing (QRCSS) Scheme

The error correction mechanism in the QR code makes it possible to manipulate some of the codewords, while still maintaining a QR code symbol that can be correctly decoded. The approach proposed in this paper is an  $(n, n)$  secret sharing scheme which will be referred to as QR Code Secret Sharing (QRCSS). The idea behind QRCSS is to exploit the error correction redundancy in the QR code structure, in order to take a QR code containing a secret message and to distribute and encode its information into  $n$  QR code shares. Each share is a valid and decodable QR code in themselves, which are each constructed from a meaningful cover QR code. The secret message can be recovered by first XORing the light and dark modules contained in the encoding region of the  $n$  QR code shares and adding the function patterns. This will produce a resulting QR code that when decoded will reveal the secret message. In the proposed scheme, the cover





**Fig. 2.** Data and error correction codeword arrangement for QR code version 4 with error correction level H.

QR codes and the secret QR code must be of the same QR code version with the same error correction level. This is so that all the codewords and codeword blocks are at exactly the same locations across all the QR code symbols.

#### 4.1 Number of Shares

An indication of the minimum number of QR code shares,  $n$ , required to share a secret QR code can be determined using the approximate recovery capacity previously discussed in Sect. 3.2. For example, to share a secret QR code using the error correction level H (i.e. 30%), the minimum number of QR code shares required is 3. This is because up to  $\sim 30\%^2$  of the codewords per block in each QR code share can be manipulated, which allows up to  $\sim 30\% \times 3 \approx 90\%$  of the codewords in the secret QR code to be distributed across the QR code shares. This means that the error in the reconstructed secret QR code is at least  $\sim 10\%$ . As long as the error in the reconstructed secret QR code is no more than its error correction capacity, which is  $\sim 30\%$ , the secret can be recovered. It can easily be seen that it is not possible to share the secret using only 2 QR code shares, as the reconstructed secret QR code will contain a minimum of  $\sim 40\%$  error, which overwhelms the error correction capacity and hence cannot be decoded correctly. This also means that the greater the number of shares used to share a secret, the smaller the amount of error that has to be introduced in each QR code share. An indication of the minimum number of shares required for the proposed QRCSS scheme for the respective error correction level is as follows: L ( $n \geq 14$ ), M ( $n \geq 6$ ), Q ( $n \geq 4$ ), and H ( $n \geq 3$ ).

<sup>2</sup> In practice, it is not advisable to introduce this much error in the QR code shares, because any damage or dirt may make the QR code symbol undecodable.

## 4.2 Codewords

The QR code error correction levels of L  $\sim$  7%, M  $\sim$  15%, Q  $\sim$  25% and H  $\sim$  30%, are merely approximate values. The exact error correction capacity depends on the QR code characteristics, which varies between the different QR code versions and error correction levels, as can be seen from the characteristics for QR code version 4 and 5 that were previously shown in Table 1.

The error correction capacity per block,  $r$ , for a QR code represents the maximum number of codewords that can be in error per block. In other words, a QR code cannot be decoded if more than  $r$  codewords per block contain errors. This means that to split the secret QR code across the QR code shares, the maximum number of codewords that can be changed across the QR code shares per block is  $r \times n$ . However, since  $c$  is the total number of codewords per block, there is no point changing more codewords than  $c$ , even in the case where  $r \times n$  is greater than  $c$ . Hence, the maximum number of codewords that may be changed should be the smaller of the two values. This will be referred to as  $m$ , where  $m = \min(r \times n, c)$ .

In order for a QR code to be correctly decoded, there must be a minimum number of codewords per block that cannot contain any errors. This value will be referred to as  $l$ , where  $l = c - r$ . This also means that in the reconstructed secret QR code, each block must have  $l$ , or more, correct codewords in order to decode the reconstructed QR code and to reveal the secret message.

The proposed QRCSS scheme requires that  $m > l$  for each block. Otherwise, it will not be possible to split information required to reconstruct the secret QR code across the shares, while maintaining valid QR code shares. This also means that the number of codewords that have to be altered per block,  $t$ , can lie anywhere between  $l$  and  $m$ . Nevertheless, in practice it is best to evenly distribute the error among the shares and the reconstructed secret. Given that  $l$  is the minimum number of codewords per block that have to be changed across the shares, to evenly distribute the error among the shares, at least  $\frac{l}{n}$  codewords per block have to be changed for each share. However, using this value would mean that the reconstructed secret has absolutely no error tolerance. As such, in the proposed QRCSS scheme, the error correction capacity for each block of the reconstructed secret was taken as the ratio between the total error correction capacity of the shares and minimum number of codewords that had to be changed, i.e.  $\frac{r \times n}{l}$  rounded to the nearest integer. Hence, to balance between the error correction capacity of the reconstructed secret and the number of errors that have to be introduced in the shares,  $t = \min(l + \text{round}(\frac{r \times n}{l}), m)$ , since  $t$  ranges between  $l$  and  $m$ . In general, the larger the value of  $n$ , the larger the error correction capacity of the reconstructed secret, for cases where  $t < m$ . If  $t = m$ , the error correction capacity of the reconstructed secret QR code is equal to the error correction capacity of the original secret QR code, i.e.  $r$ .

Let  $e$  be the maximum number of codewords in a block to change per share, or in other words the maximum number of errors that will be introduced to a block per share in order to distribute the secret. To evenly distribute the error across the shares, the value of  $e$  should be  $\frac{t}{n}$  rounded up to the nearest integer,

i.e.  $e = \text{round\_up}(\frac{t}{n})$ . This means that the larger the value of  $n$ , the smaller the amount of error per share  $e$ . So while the original cover QR code error capacity per block is  $r$ , the resulting error capacity per block for each QR code share in this QRCSS scheme is  $r - e$ . For the reconstructed secret QR code, since  $t$  codewords per block were altered in the shares,  $c - t$  codewords per block will be in error when reconstructed. So the error correction capacity per block in the reconstructed secret QR code is  $r - (c - t)$ .

### 4.3 Format and Version

The format information for a QR code contains 15-bits and allows for an error of 3-bits, or less. The first 2-bits in the bit sequence represent the error correction level, which in the proposed QRCSS scheme must be the same for all the shares and the secret. Hence, these 2-bits do not have to be altered as only the next 13-bits may vary between the individual shares and the secret. The format information allows for an error of up to 3-bits in each share to be corrected. In addition, the reconstructed secret itself allows for a 3-bit format information error to be corrected. As such, to evenly distribute the error across the shares, the maximum number of format information bits to change per share  $f$ , should be  $\frac{13}{n+1}$  rounded up to the nearest integer, i.e. if  $n = 3$  then  $f = 3$ , otherwise  $f = \text{round\_up}(\frac{13}{n+1})$ .

QR code version 7 and above have additional version information. However since the proposed QRCSS scheme uses QR codes of the same version, the version information modules are the same for all QR code shares. As such, the unaltered version information simply has to be added to the reconstructed secret QR code.

In the proposed scheme, the choice of QR code version used to share a secret has a direct bearing on its security. The security of the QRCSS scheme is governed by a security parameter  $\lambda$ . The size of  $\lambda$  depends on the data capacity of a QR code. The more data a QR code can contain, the larger  $\lambda$  will be. Hence, the QR code version has to be taken into consideration when determining security of the scheme. This is because the larger the QR code version, the higher its data capacity.

### 4.4 Algorithm

The proposed  $(n, n)$  QRCSS scheme can formally be described as follows. The algorithm takes one secret QR code,  $S$ , and  $n$  cover QR codes,  $C_1, C_2, \dots, C_n$ , as input and outputs  $n$  QR code shares,  $S_1, S_2, \dots, S_n$ . The minimum required value of  $n$  for each error correction level was previously discussed in Sect. 4.1. Each QR code share  $S_1$  to  $S_n$  is a valid QR code that when decoded by a QR code reader will produce the same information as their respective cover QR codes  $C_1$  to  $C_n$ . Pseudocode for the proposed QRCSS scheme is presented in Algorithm 1. It should be noted that the input QR codes can be generated using any standard QR code generator.

To recover the secret, XOR the modules in the encoding regions of all the shares  $S_1 \oplus S_2 \oplus \dots \oplus S_n$  and add the function patterns to produce a reconstructed QR code,  $S^r$ . When decoded by a QR code reader,  $S^r$  will produce the same message as  $S$ .

## 5 Analysis and Discussion

A secret sharing scheme can be evaluated based on a number of properties, including its security, reconstruction precision, computation complexity and storage requirement [21]. This section presents an analysis and discussion of the QRCSS scheme based on these properties.

A program implementing the proposed  $(n, n)$  QRCSS scheme, using the pseudocode shown in Algorithm 1, was developed. An example of the results produced by the QRCSS scheme is depicted in Figs. 3 and 4<sup>3</sup>. The example shown in these figures is for the case where  $n = 3$ , using QR code version 4 with error correction level H. Figure 3(a)–(c) show the three cover QR codes,  $C_1$ ,  $C_2$  and  $C_3$ , while Fig. 4(a) shows the original QR code containing a secret message,  $S$ , which is to be encoded into the shares. Figure 3(d)–(f) show the QR code shares resulting from the proposed QRCSS scheme, i.e.  $S_1$ ,  $S_2$  and  $S_3$ , which were obtained from their respective covers. Note that these shares are valid QR codes that can be read by any standard QR code reader. Figure 4(b) in turn shows the reconstructed QR code,  $S^r$ , which was obtained by XORing the modules in the encoding regions of all the shares,  $S_1 \oplus S_2 \oplus S_3$ , and appending the function patterns. The secret message can be recovered by decoding  $S^r$  using a QR code reader.

A description of the characteristics for the example results shown in Figs. 3 and 4 is as follows. The codewords in QR codes of version 4 and error correction level H are divided into 4 error correction blocks, with the characteristics of  $c = 25$ ,  $k = 9$  and  $r = 8$  for all blocks (refer to Table 1). Using the proposed algorithm,  $l = 17$  and  $m = 24$ , while  $t = 20$  and  $e = 7$  for each block. As such, a maximum of 7 codewords per block were altered in  $S_1$ ,  $S_2$  and  $S_3$  (i.e. a total of 20 codewords were changed per block, so two of the shares had 7 codewords altered while the remaining share had 6 codewords altered). This means that each block in  $S_1$ ,  $S_2$  and  $S_3$  has an error correction capacity of at least 1 and each block in  $S^r$  has an error correction capacity of 3. Hence, even if there are some errors introduced to  $S^r$ , due to damage or other reasons, the secret message can still be recovered.

Figure 3(g)–(i) show the respective difference images between the cover QR codes and the resulting QR code shares, while Fig. 4(c) show the differences between the secret QR code and the reconstructed secret QR code. In the difference images, gray represent no change, while white represents a change in the reflectance value from a dark module in the cover QR code to a light module in

<sup>3</sup> Please note that all QR codes presented in this paper are valid QR codes that can be decoded by a QR code reader.

**Algorithm 1.** QRCS pseudocode

---

```

function GENERATESHARES()
  Input: A secret QR code,  $S$ , and  $n$  cover QR codes,  $C_1, C_2, \dots, C_n$ 

  /* Create  $n$  shares and copy contents of each cover to the respective share */
  for  $i$  from 1 to  $n$  do
    Create  $S_i$ 
     $S_i \leftarrow C_i$ 
  end for

  /* Data and error correction codewords */
  for each QR code block do /* Some QR code versions have 2 different block sizes */
    Calculate  $l$  and  $m$ 
    if  $m \leq l$  then
      Quit /* Scheme requires  $m > l$ , otherwise input is not valid */
    else
      Determine  $t$  and  $e$ 
      for  $i$  from 1 to  $t$  do
        Randomly select a codeword,  $w$ , within the block that has not previously been
changed
        Select a random share,  $S_x$ , where  $1 \leq x \leq n$ , that has not exceeded the value
of  $e$ 

        for each module  $j$  in  $w$  do /* where  $1 \leq j \leq 8$  (a codeword contains 8 modules)
*/
          if  $S_1^j \oplus S_2^j \oplus \dots \oplus S_n^j \neq S^j$  then
            /*  $S_k^j$  denotes the  $j$ -th module of  $w$  in the  $k$ -th share */
            /*  $S^j$  denotes the  $j$ -th module of  $w$  in the secret */
            Flip the reflectance value of  $S_x^j$ 
          end if
        end for
      end for
    end if
  end for

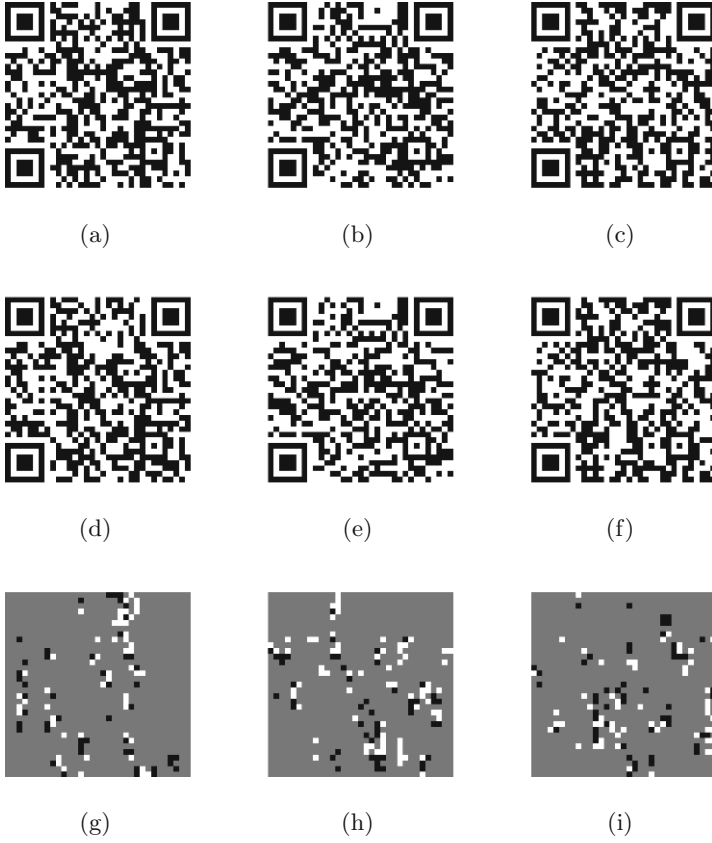
  /* Format information */
  Determine  $f$  /* The maximum number of format information bits to change per share
*/

  for  $j$  from 3 to 15 do /* For each module of the format information, except the first
2-bits */
    if  $S_1^j \oplus S_2^j \oplus \dots \oplus S_n^j \neq S^j$  then
      Select a random share,  $S_x$ , where  $1 \leq x \leq n$ , that has not exceeded the value of  $f$ 
      Flip the reflectance value of  $S_x^j$ 
    end if
  end for

  Output  $S_1, S_2, \dots, S_n$ 
end function

```

---



**Fig. 3.** Example of a (3, 3) QRCSS; (a)  $C_1$ , (b)  $C_2$ , (c)  $C_3$ , (d)  $S_1$ , (e)  $S_2$ , (f)  $S_3$ , (g) Difference image between  $C_1$  and  $S_1$ , (h) Difference image between  $C_2$  and  $S_2$ , (i) Difference image between  $C_3$  and  $S_3$ .



**Fig. 4.** Secret and reconstructed secret for the example shown in Fig. 3. (a) Original QR code containing the secret message,  $S$ . (b) Reconstructed secret QR code,  $S^r$ . (c) Difference image between  $S$  and  $S^r$ .

the QR code share, and conversely black represents a change from a light module in the cover to a dark module in the resulting share. The difference images indicate the error that was deliberately embedded into the QR codes for the purpose of sharing the secret.

## 5.1 Properties

From a computational complexity point of view, the QRCSS scheme is not computationally complex, as encoding and decoding in the scheme is based on the simple Boolean XOR operation. In addition, despite the deliberate error incorporated in the shares by the proposed scheme, the scheme is still able to handle a small amount of error when decoding the reconstructed secret QR code. The greater the number of shares used to distribute the secret, the smaller the amount of error that has to be encoded into each of the QR code shares. This will also increase the recoverability of the secret, because the reconstructed secret QR code will contain less errors.

Since the secret is encoded as a QR code, the size of the secret message is governed by the QR code data capacity. The size of each share in the proposed scheme is the same as the size of the secret QR code. One of the advantages of the QRCSS scheme is that the shares do not have to be stored in digital form, as they can be distributed on printed media.

## 5.2 Security

Individual QR code shares by themselves do not reveal the secret message. However, since shares and the secret QR code use the same QR code version and error correction level, an adversary who can identify a QR code share will have this knowledge. In addition, given any QR code share, it is trivial to obtain its cover QR code. This can be done by simply decoding the information in the QR code share, then using this information to generate a QR code of the appropriate version and error correction level. This will produce the original cover QR code. As such, it is easy to find the differences between the original cover QR code and the modified QR code. This property also makes it possible to identify potential QR code shares. Based on the information about the differences between the original and the modified QR code, the value of  $n$  can be inferred. Nevertheless, it is not possible to reconstruct the secret with this information alone.

**Brute Force Attack.** In light of the fact that in QRCSS the secret message is encoded as a QR code, an adversary can adopt a brute force strategy akin to a dictionary attack. This is because the secret must be in the form of a valid QR code. Thus, if an attacker has information about the version and error correction level of the secret QR code, the attacker can go through all the possibilities of valid QR codes with the same version and error correction level. Let  $S'$  denote a valid, or in other words 'meaningful', QR code of the known version and error correction level, and  $|S'|$  be the cardinality of all the valid QR codes with the

same version and error correction level. The probability of success for this attack will be bounded by  $\frac{1}{|S'|}$ . The size of  $|S'|$  is governed by the size of data that a QR code can contain, which is determined by the specific QR code version used to encode the message. Hence, the larger the secret QR code, the larger  $|S'|$  will be, which in turn lowers the success of an attack. Let  $d$  be the number of data codewords for a QR code of a given version and error correction level. Since each codeword contains 8 modules,  $|S'| = 2^{8d}$ .

**Collusion Attack.** It is conceivable that  $n - 1$  participants may collude and combine the information from their shares together in an attempt to find  $S$ .

Since  $S_1 \oplus S_2 \oplus \dots \oplus S_n = S^r$ , in a collusion attack where  $n - 1$  participants combine their shares, they will have  $S_1, S_2, \dots, S_{n-1}$ . Let  $S'_n$  denote a possible  $S_n$ , or in other words the attacker's 'guess' of  $S_n$ . Furthermore, let  $S'^r$  denote a valid QR code that when decoded will give the same information as  $S'$ , and let  $\mathbf{S}^r$  represent the set of all valid QR codes created by modifying the modules of all QR codes in  $\mathbf{S}'$ . This is in line with the fact that the modules in the reconstructed secret QR code,  $S^r$ , are not the same as the secret QR code,  $S$ . Hence, even though when decoded  $S'^r$  and  $S'$  will give the same information, the modules in  $S'^r$  and  $S'$  are not the same. This means that the entries in  $\mathbf{S}^r$  must be created by going through all the possible modifications of all QR codes in  $\mathbf{S}'$  that will produce valid QR codes. Therefore, the cardinality of  $\mathbf{S}^r$ ,  $|S'^r|$ , will be very much larger than  $|S'|$ .

$|S'^r|$  can be determined based on the specific QR code version and error correction level. Each block of a QR code has  $c$  codewords and an error capacity of  $r$  codewords. Each codeword contains 8 modules. This means that a block can correctly be decoded as long as there are  $r$ , or less, codewords in  $c$  that are in error. However, as previously shown in Table 1, some QR code versions have 2 different block characteristics,  $B_1$  and  $B_2$ . Let  $c_1$  and  $c_2$  denote the number of codewords per block, and let  $r_1$  and  $r_2$  represent the error correction capacity per block, for blocks of type  $B_1$  and  $B_2$  respectively. In addition, let  $n_1$  and  $n_2$  represent the number of blocks of type  $B_1$  and  $B_2$  for that specific QR code. For a QR code of a given version and error correction level,  $|S'^r|$  can be determined as follows:

$$|S'^r| = \left[ \left( \sum_{i=0}^{r_1} \binom{c_1}{i} \right) \cdot 2^{8n_1} \cdot \left( \sum_{j=0}^{r_2} \binom{c_2}{j} \right) \cdot 2^{8n_2} \right] \cdot |S'|$$

A collusion attack may take the following form. By going through the entries in  $\mathbf{S}^r$  using  $S_1 \oplus S_2 \oplus \dots \oplus S_{n-1} \oplus S'^r = S'_n$ , if  $S'_n$  is not a valid QR code, i.e. cannot be decoded, then  $S'^r$  can be removed from  $\mathbf{S}^r$ . Thus, reducing the space of potential reconstructed secret QR codes,  $|S'^r|$ , to contain less possibilities. Nevertheless, it should be noted that this space will still be very large as  $|S'^r|$  is very much larger than  $|S'|$ . Let  $|S''|$  denote this reduced space. It is obvious that the value of  $|S''|$  will increase for the cases of  $n - 2, n - 3$ , etc.



*Security Parameter.* The security underlying the QRCSS scheme is therefore governed by the security parameter  $\lambda = \min(|S'|, |S''|)$ . Clearly, larger QR code versions will increase the size of  $\lambda$ , thereby increasing the security of the scheme.

**Concealment.** One of the primary advantages of this scheme stems from the fact that since each share is a meaningful QR code in itself, this will reduce the likelihood of attracting the attention of potential attackers. In addition, since the modules in QR codes do not have to be black and white squares, it would be advantageous if each QR code share were to be constructed using different artistic QR code schemes. The QRCSS scheme will work as long as the modules in each share can be scanned. The use of different artistic schemes will not only increase the secret sharing subterfuge by using meaningful innocent-looking QR code shares, but will also improve concealment as shares will appear to be unrelated.

## 6 Conclusion and Future Work

This paper presents a novel approach to secret sharing using QR codes. In QR code secret sharing, a QR code containing a secret message is distributed and encoded into a set of meaningful QR code shares. The proposed approach uses the error correction feature, which is an inherent part of the QR code structure, to distributed and hide information about the secret. Each share is a valid QR code which contains meaningful information when scanned individually. Hence, this reduces the likelihood of attracting the attention of potential attackers when distributed via public channels. When all shares are made available, the secret message can be recovered. Unlike a number of other secret sharing schemes where the shares are digital images, which have to be stored and transmitted electronically, the QR code shares in this approach can be distributed using printed media.

## References

1. Ateniese, G., Blundo, C., De Santis, A., Stinson, D.R.: Extended capabilities for visual cryptography. *Theor. Comput. Sci.* **250**(1–2), 143–161 (2001)
2. Beimeel, A.: Secret-sharing schemes: a survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) *IWCC 2011*. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011)
3. Blakley, G.: Safeguarding cryptographic keys. In: *Proceedings of the 1979 AFIPS National Computer Conference*, pp. 313–317 (1979)
4. Blundo, C., D'Arco, P., Santis, A.D., Stinson, D.R.: Contrast optimal threshold visual cryptography schemes. *SIAM J. Discrete Math.* **16**(2), 224–261 (2003)
5. Bui, T.V., Vu, N.K., Nguyen, T.T., Echizen, I., Nguyen, T.D.: Robust message hiding for QR code. In: *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pp. 520–523. IEEE (2014)
6. Chen, W.-Y., Wang, J.-W.: Nested image steganography scheme using QR-barcode technique. *Opt. Eng.* **48**(5), 057004 (2009)

7. Chow, Y.-W., Susilo, W., Au, M.H., Barmawi, A.M.: A visual one-time password authentication scheme using mobile devices. In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) ICICS 2015. LNCS, vol. 8958, pp. 243–257. Springer, Heidelberg (2015)
8. Chow, Y.-W., Susilo, W., Wong, D.S.: Enhancing the perceived visual quality of a size invariant visual cryptography scheme. In: Chim, T.W., Yuen, T.H. (eds.) ICICS 2012. LNCS, vol. 7618, pp. 10–21. Springer, Heidelberg (2012)
9. Chung, C.-H., Chen, W.-Y., Tu, C.-M.: Image hidden technique using QR-barcode. In: Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2009. IHH-MSP 2009, pp. 522–525. IEEE (2009)
10. Denso Wave Incorporated. QRcode.com. <http://www.qrcode.com/en/>
11. Falkner, S., Kieseberg, P., Simos, D.E., Traxler, C., Weippl, E.: E-voting authentication with QR-codes. In: Tryfonas, T., Askoxylakis, I. (eds.) HAS 2014. LNCS, vol. 8533, pp. 149–159. Springer, Heidelberg (2014)
12. Huang, H.-C., Chang, F.-C., Fang, W.-C.: Reversible data hiding with histogram-based difference expansion for QR code applications. *IEEE Trans. Consum. Electron.* **57**(2), 779–787 (2011)
13. International Organization for Standardization: Information technology – automatic identification and data capture techniques – QR code 2005 bar code symbology specification. ISO/IEC 18004:2006 (2006)
14. Lee, H.-C., Dong, C.-R., Lin, T.-M.: Digital watermarking based on JND model and QR code features. In: Pan, J.-S., Yang, C.-N., Lin, C.-C. (eds.) Advances in Intelligent Systems and Applications. Smart Innovation, Systems and Technologies, vol. 2, pp. 141–148. Springer, Heidelberg (2013)
15. Lin, P.-Y., Chen, Y.-H., Lu, E.J.-L., Chen, P.-J.: Secret hiding mechanism using QR barcode. In: 2013 International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 22–25. IEEE (2013)
16. Liu, F., Guo, T., Wu, C.K., Qian, L.: Improving the visual quality of size invariant visual cryptography scheme. *J. Vis. Commun. Image Represent.* **23**(2), 331–342 (2012)
17. Naor, M., Shamir, A.: Visual cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)
18. Shamir, A.: How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979)
19. Thien, C.-C., Lin, J.-C.: Secret image sharing. *Comput. Graph.* **26**(5), 765–770 (2002)
20. Wang, D., Yi, F., Li, X.: On general construction for extended visual cryptography schemes. *Pattern Recognit.* **42**(11), 3071–3082 (2009)
21. Wang, D., Zhang, L., Ma, N., Li, X.: Two secret sharing schemes based on boolean operations. *Pattern Recognit.* **40**(10), 2776–2785 (2007)
22. Weir, J., Yan, W.: A comprehensive study of visual cryptography. *Trans. Data Hiding Multimed. Secur.* **5**, 70–105 (2010)
23. Weir, J., Yan, W.Q.: Authenticating visual cryptography shares using 2D barcodes. In: Shi, Y.Q., Kim, H.-J., Perez-Gonzalez, F. (eds.) IWDW 2011. LNCS, vol. 7128, pp. 196–210. Springer, Heidelberg (2012)
24. Wu, W.-C., Lin, Z.-W., Wong, W.-T.: Application of QR-code steganography using data embedding technique. In: Park, J.J., Barolli, L., Xhafa, F., Jeong, H.-Y. (eds.) Information Technology Convergence: Security, Robotics, Automations and Communication. Lecture Notes in Electrical Engineering, vol. 253, pp. 597–605. Springer, Dordrecht (2013)

25. Yan, W., Wier, J., Kankanhalli, M.S.: Image secret sharing. In: Cimoto, S., Yang, C.-N. (eds.) *Visual Cryptography and Secret Image Sharing*, pp. 381–402. CRC Press, Boca Raton (2012)
26. Yang, C.-N., Peng, A.-G., Chen, T.-S.: MTVSS: (M)isalignment (T)olerant (V)isual (S)ecret (S)haring on resolving alignment difficulty. *Signal Process.* **89**(8), 1602–1624 (2009)

# Password Requirements Markup Language

Moritz Horsch<sup>(✉)</sup>, Mario Schlipf, Johannes Braun, and Johannes Buchmann

Technische Universität Darmstadt, Hochschulstraße 10, 64283 Darmstadt, Germany  
{horsch,schlipf,jbraun,buchmann}@cdc.informatik.tu-darmstadt.de

**Abstract.** Passwords are the most widely used authentication scheme for granting access to user accounts on the Internet. In order to choose strong passwords, security experts recommend the usage of password generators. However, automatically generated passwords often get rejected by services, because they do not fulfill the services' password requirements. Users need to manually look up the password requirements for each individual service and configure the password generator accordingly. This inconvenience induces users not to employ password generators and rather stick to weak passwords. We present a solution that enables generators to automatically create passwords in accordance with services' password requirements. First, we introduce the Password Requirements Markup Language (PRML). It enables uniformly specified Password Requirements Descriptions (PRDs) for services. PRDs can be automatically processed by password generators and allow the generation of strong valid passwords without user interaction. Second, we present a crawler for the automatized extraction of password requirements from services' websites and the creation of the corresponding PRDs. This crawler allowed us to create PRDs of 72,124 services. Third, we describe a centralized and a decentralized approach for the provision of the PRDs to password generators. Finally, we present a password generator which uses PRDs and requires nothing but a service' URL in order to generate a strong and valid password for the service.

## 1 Introduction

User accounts at Internet services contain a multitude of personal data such as emails, pictures, and payment information. To restrict access to legitimate users, user accounts are usually protected by passwords. To prevent the various known attacks against passwords, such as brute-force [16, 28], dictionary [10, 27], and social engineering [12] attacks, it is essential to use strong passwords. However, users often tend to use passwords that are simple and easy to remember [5, 9, 13, 15, 29].

To assist users in creating strong passwords, security experts recommend to use password generators and store these passwords in password managers. A password generator creates random and strong passwords based on a predefined set of password generation rules. These rules can be modified with respect to the length and the allowed character sets for the generated passwords. However, in practice such passwords are often rejected by services because they do not

comply with the various password requirements of services. For instance, the generated passwords are too short, too long, or do not contain a special character. One possible solution would be that all services agree on a single set of requirements [7,24]. Yet, this seems unrealistic in the light of the multitude of different password implementations and security needs out there. Also, a common set of generation rules that fit to the password requirements of all services is not realizable due to the enormous diversity of requirements. Wang et al. [26] mention that such a set does even not exist for a set of 50 services. Currently, the only solution for users is to look up the password requirements of each service manually and configure the password generator accordingly. This is error-prone, very inconvenient for users, and not infrequently prevents users from employing password generators at all.

In this paper, we present a solution that closes the gap between the wide diversity of password requirements on the service-side and the single default password generation rule set on the user-side (password generator). It enables password generators to automatically create passwords that comply with services' requirements. We introduce the Password Requirements Markup Language (PRML) which is intended to uniformly specify a Password Requirements Description (PRD) for a service. This uniform description facilitates password generators to automatically process the individual password requirements of services. The password generation rules can then be adjusted accordingly. This removes the burden of a manual look-up of password requirements and configuration of the password generator. Users just need to provide the generator with an identifier of the service (e.g. its URL) for which they need a secure password in order to get one that fulfills the service's password requirements.

We also provide a set of tools to create, distribute, and to make use of PRDs. We present the Password Requirements Crawler (PRC), an application that is capable of extracting password requirements from services' websites by Natural Language Processing techniques and converting them into standardized PRDs. Up to now, the PRC was used to create PRDs of 72,124 services. We additionally implement the PRD Distribution Service (PRDDS), a central web service to make the PRDs available to password generators. As an alternative, we present a decentralized approach to distribute PRDs. This tool set is complemented by a password generator that makes use of PRDs. It solely requires the URL of a service to retrieve the corresponding PRD from the PRDDS and generate a secure password in accordance with the password requirements of services.

This paper is organized as follows: Sect. 2 contains related work. Section 3 presents PRML and Sect. 4 the PRC. Section 5 reports on the large-scale generation of PRDs. We describe the PRDDS and the decentralized solution in Sect. 6. Finally, we present our password generator in Sect. 7 and conclude the paper in Sect. 8. The Appendix A includes an exemplary PRD. Appendix B provides more technical background of the PRC.

## 2 Related Work

Password generators exist as web (e.g. random.org [19]) and stand-alone applications (e.g. PWGen [25]) as well as an integral part of almost all password managers. None of the existing password generators takes the password requirements of services into account. Users need to look up the requirements of each service and configure the generator manually.

Shay et al. [22] present a language and simulation model for password policies, which describe the creation and management of passwords. Their language is based on the generic authentication policy language AuthSL [23], but it is not capable of expressing password requirements of real services. The simulation model is later enhanced by technical and human factors [21], but it still focuses on simulating policies instead of expressing password generation requirements of existing services.

The website [passrequirements.com](http://passrequirements.com) [2] lists the password requirements of several Internet services. Due to the lack of standardized requirement descriptions and the limitation to only 64 services, the website cannot serve as a basis for autonomic retrieval of password requirements.

## 3 Password Requirements Markup Language

The Password Requirements Markup Language (PRML) specifies a framework for PRDs for Internet services. A PRD is a standardized description of the service's password requirements. It provides all required details for the automatic generation of secure passwords compliant to these requirements. We encode PRDs in XML and provide a full XML Schema<sup>1</sup>. XML is well-specified and supported by many programming languages, which enables an easy integration of PRDs into password generators.

We conducted an analysis of password requirements for 200 representative services<sup>2</sup>, highly relevant to our daily online lives, in order to provide a comprehensive and representative specification for PRDs. Based on the results we identified common password requirements and specified PRML. In the following we describe how password requirements can be described by PRML. An example can be found in Appendix A.

A PRD is represented by a `<prd>` XML element and it has two attributes: `url` and `version`. The `url` specifies for which URL (i.e. service) the PRD is valid. The `version` number allows applications to differentiate between different versions of a PRD and to update it. Furthermore, a PRD specifies the following password requirements:

<sup>1</sup> <http://www.passwordpolicy.info/prml.xsd>.

<sup>2</sup> The Alexa Top 500 US list [6] reduced by websites with pornographic and illegal content, non-english websites, and websites that do not have or allow the creation of user accounts (e.g. banking websites).

- *Character Sets.* The `<characterSets>` element defines a list of allowed character sets. Each set is specified by a `<characterSet>` element and defines a name (e.g. numbers) and a list of characters (e.g. 0..9).
- *Properties.* The `<properties>` element defines further restrictions for the character sets and the password as a whole:
  - *Minimum and Maximum Occurrences.* In the `<characterSettings>` element, for each character set a minimum (`<minOccurs>`) and a maximum (`<maxOccurs>`) of occurrences can be defined.
  - *Positions.* A `<positionRestriction>` element is used to restrict the occurrences (min ( `<minOccurs>`) and a max (`<maxOccurs>` occurrences) of a character set to a single or a range of positions ( `<positions>`).
  - *Sequences.* The `<maxConsecutive>` element specifies the number of allowed consecutive identical characters.
  - *Minimum and Maximum Password Length.* The `<minLength>` and `<maxLength>` elements specify the minimum and maximum length of the password, respectively.

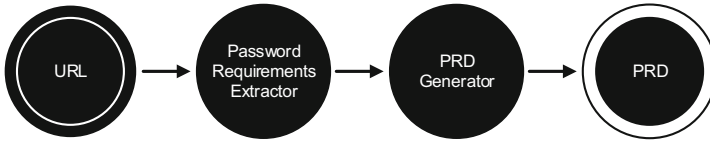
## 4 Password Requirements Crawler

PRDs for services are required to enable password generators to create passwords compliant with services’ password requirements. Creating PRDs manually for the huge amount of services that exists on the Internet is not feasible. We present the Password Requirements Crawler (PRC). It automatically extracts the password requirements from a service’s website and translates them into a PRD.

The password requirements are extracted from the sign up page of a service. This is the common location where users are confronted with them while choosing a password for their new user account.

The description and presentation of password requirements are extremely diverse. Services describe their requirements in natural language instead of using a common or structured format. For instance, a maximum password length of 10 characters can be expressed by “not more than 10”, “do not use 11 or more”, or “at most 10 characters”. This diversity makes it extremely challenging to identify and extract password requirements. For instance, regular expressions would be too narrow to cover all the different expressions. We apply Natural Language Processing technologies [11] to precisely identify and extract password requirements from a website of a service.

Figure 1 illustrates the processing sequence of the PRC’s components during PRD generation. The PRC consists of two components and takes as input the URL of a service’s sign up page and outputs a PRD. First, the Password Requirements Extractor (PRE) retrieves the rendered HTML document of a sign up page and extracts password requirements. Second, the Password Requirements Description Generator (PRDG) creates a PRD based on the PRE’s output. Sections 4.1 and 4.2 describe the PRE and the PRDG in detail. Section 4.3 provides an evaluation of the PRC.

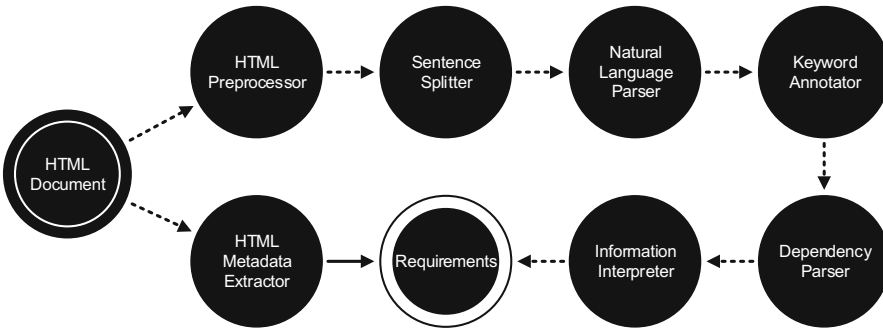


**Fig. 1.** Processing sequence for generating a Password Requirements Description.

### 4.1 Password Requirements Extractor

The Password Requirements Extractor (PRE) extracts password requirements from a HTML document. It gets as input the URL of a sign up page, downloads it, and extracts the password requirements. It outputs the extracted requirements in a uniform description, which is then used by the PRDG to create the PRD for the service.

Figure 2 provides an overview of the processing steps. Password requirements are extracted in two ways. On the one hand, Natural Language Processing technologies<sup>3</sup> are used to analyze the textual content of the document (cf. dashed path in Fig. 2). On the other hand, information provided by the document’s source code is analyzed by the HTML Metadata Extractor (cf. solid path in Fig. 2). Both results are finally combined. We describe the requirements extraction in the following.



**Fig. 2.** Overview of the password requirements extraction performed by the PRE.

*HTML Preprocessor (HP).* The HP removes content that in general does not contain password requirements from the HTML document, such as the header, images and drop-down lists. Furthermore, it adds sentence delimiters to list items, text blocks, and paragraphs in order to improve the subsequent detection of sentences in the content. The HP outputs the plain-text of the HTML document, without any HTML markup.

<sup>3</sup> Our implementation is based on the Apache UIMA framework [8,14].



*Sentence Splitter (SS)*. The SS takes as input the loose collection of words and punctuation marks from the HP and splits it into sentences. It detects beginnings and ends of sentences and annotates them with a delimiter.

*Natural Language Parser (NLP)*. The NLP determines the grammatical structure of each sentence, provided by the SS. It detects groups of words that belong together and identifies the category of each word (verb, subject, or object). It outputs a dependency tree for each sentence, which describes the sentence's grammatical structure and the textual relation of its words.

*Keyword Annotator (KA)*. The KA searches for keywords to identify sentences that potentially contain password requirements. Relevant keywords were identified based on 20 representative websites and extended by linguistically related words (cf. Appendix B for details). In essence, we use keywords related to character sets like “uppercase” and “numbers” and keywords focusing on password lengths such as “at least” and “more than”. The KA outputs a list of sentences containing these keywords.

*Dependency Parser (DP)*. The DP combines the results from the NLP and KA and puts both into a relation. For the example “use 6 to 12 characters” the dependency tree provides the information that 6 and 12 are numbers and the KA identified the word “characters” as a keyword. The DP now determines the relation between the numbers 6 and 12 and labels it as a number range.

*Information Interpreter (II)*. The II finally takes all information together and extracts the password requirements from the sentences. The example sentence “use not more than 16 characters” is interpreted as follows. The number “16” refers to the object “characters” and the keyword “more than” indicates that at least 16 characters must be used. Then, the negation “not” is considered and the sentence is interpreted as such that the maximum password length is 16. The II outputs the extracted requirements in a structured format. All the different expressions of password requirements used by services are now boiled down to a uniform format.

*HTML Metadata Extractor (HME)*. Beside the textual analysis of the document, the HME analyzes the HTML source code in order to extract additional information about the password requirements. The HTML standard specifies an optional `minlength` and `maxlength` attribute for input fields. If present, the HME stores these values to the final list of extracted requirements as the minimum and maximum password length. In case of password lengths found by both II and HME, the higher minimum length and lower maximum length are used.

## 4.2 Password Requirements Description Generator

The Password Requirements Description Generator (PRDG) takes as input the requirements found by the PRE and creates a PRD. The requirements stated by

services are often incomplete. A prominent example are special characters. Some services allow them or even require them, but they usually do not list them. To mitigate this problem the PRDG uses the following default fall-back settings for creating PRDs:

- *Missing Character Sets.* If a service does not name the allowed character sets, the allowed character set is set to letters and numbers.
- *Missing Password Lengths.* If a service does not specify the min and/or max password length these values are also undefined in the PRD.
- *Missing Special Characters.* If a service allows or requires special characters but does not name them, we define the following special characters in the PRD: . : , ; - + \* ! ? % & =.

We evaluated these default settings with the above introduced set of 200 services<sup>4</sup> and did not observe any incompatibilities.

### 4.3 Evaluation

We evaluated the PRC to verify that it extracts password requirements correctly. For the set of 200 representative services<sup>4</sup> we first manually looked up the sign up pages and the password requirements. Second, we run the PRC to create PRDs for the same services. Finally, we compared our manually extracted requirements with the results of the PRC and classified the PRDs in three categories:

- *Exact.* A PRD exactly describes the service’s password requirements.
- *Correct.* A PRD contains requirements with inaccurate values (e.g. smaller maximum password length). However, the PRD defines requirements that create passwords which are accepted by the service.
- *Incorrect.* A PRD contains requirements with incorrect values, which cause the generation of passwords that are not accepted by the service.

The results of the evaluation are listed in Table 1. For 167 out of 200 (83.5%) services the PRC perfectly extracted all password requirements. For additional 13 (8%) of the services, some password requirements were extracted inaccurately, but the PRDs can still be considered as correct. In most cases the inaccuracy is caused by a different value for minimum/ maximum occurrence of characters (cf. Table 2). For example, the PRC created a PRD for paypal.com that requires at least one number *and* one special character, whereas the actual requirement only demands at least one number *or* a special character. This does not significantly affect the security of the generated passwords, but it does not exactly match PayPal’s password requirements. Other inaccuracies that occurred were lower maximum password lengths or greater minimum password lengths. For instance, the PRC extracted a minimum length of 6 characters instead of 4 for the website bleacherreport.com. 17 (8.5%) of the 200 PRDs were classified as incorrect. For instance, the website cisco.com requires at least one uppercase and one lowercase

<sup>4</sup> See footnote 2.

**Table 1.** Evaluation results of 200 manually reviewed PRDs.

Classification	#	%
Exact	167	83.5 %
Correct	16	8.0 %
Incorrect	17	8.5 %

**Table 2.** Inaccuracies in correct PRDs. More than one inaccuracy may occur in one PRD.

Inaccuracy	#
Lower maximum length	4
Different minimum length	8
Fewer allowed characters	4
Different min/max character occurrences	14

character as well as a maximum password length of 50 characters. The PRC only extracted the maximum password length. On the one hand, it is very likely that a random password of 50 characters will contain at least one uppercase and one lowercase character. On the other hand, the PRD does not enforce it and therefore a generated password might not be accepted. Another example is gamefaqs.com. The PRC extracted a maximum password length of 64 characters while the actual maximum length is 40, because the website has two password fields, one which allows 40 and one which allows 64 characters. A password generator that uses the maximum length is not able to generate a valid password for this service.

In summary, despite the very rigorous validation of the PRDs, the evaluation shows that the PRC correctly created PRDs for 91,5 % (183 out of 200) of the services.

## 5 Large-Scale PRD Generation

To be able to provide PRDs for a broad variety of services, we analyzed 1,000,000 websites<sup>5</sup>. Because the PRC needs the URL of the sign up page as input instead of the root URL of a service, we developed a Sign Up Page Finder (SUPF) to facilitate a large-scale PRD generation.

Instead of a time and resource consuming search over the entire website of a service, the SUPF uses search engines, such as Google or Bing, to effectively locate a service's sign up page. We use the service's URL in conjunction with the words *sign up* as the search term<sup>6</sup>. From the search results we investigate the first three<sup>7</sup>. The SUPF browses each result sequentially and checks if it contains a sign up page. This is done by looking for HTML `<input>` elements. A sign

<sup>5</sup> We took the Quantcast Top Million U.S. Web Sites list [18], which lists the top ranked domains based on the number of people visiting the websites from the US.

<sup>6</sup> The terms *create account*, *register* or *join* lead to comparable results.

<sup>7</sup> We evaluated this approach based on 200 services. For 91 % of the services the sign up page can be found through the first three search results. For 1.5 % the sign up page was listed between the fourth and the thirtieth result. For the remaining 7.5 % no sign up page was found within the first thirty results. The main reason is that these services do not allow indexing their sign up pages by search engines (e.g. wikipedia.org). For performance reasons we only considered the first three results.

up page usually contains at least an input field for the password and additional ones for user related data, such as name, email, or address. The SUPF requires a minimum of three input fields (one password and two additional ones) to accept a page as a sign up page. This filters out login and newsletter registration pages, where in general no password requirements are contained. The sign up pages detected by the SUPF can then be used as input for the PRC.

**Table 3.** Overview of websites processed during the large-scale PRD generation.

Analyzed websites	1,000,000
No results by the search engine	593,512
Website not available or download issues	169,651
No sign up page detected	164,713
PRDs created	<b>72,124</b>

The outcome of our large-scale PRD generation is shown in Table 3. For 593,512 of the 1,000,000 analyzed websites the search engines did not provide any results. This means that the websites have no user accounts, do not allow to create user accounts, or do not allow indexing their sign up pages by search engines<sup>8</sup>. From the 406,488 websites for which search results were delivered, 169,651 websites were not available or the sign up page could not be downloaded, even with multiple connection attempts to eliminate temporary connection or server problems. For 164,713 out of the remaining 236,837 services the SUPF did not detect a sign up page, e.g., for banking websites that allow logins but no registrations. In summary, the PPC in conjunction with the SUPF was able to create PRDs for 72,124 web services<sup>9</sup>. The processing of the 1,000,000 websites took approximately 4 weeks running 4 computers in parallel and produced more than 1 TB of traffic. The size of a single PRD is on average 1 KB.

## 6 PRD Distribution

The availability of PRDs is a crucial requirement for their use during password generation. In order to distribute PRDs, we realize a Password Requirements Description Distribution Service (PRDDS). It is developed as a Java Web Service and provides an interface through which applications can request a PRD for a service identified by its URL. This allows an immediate deployment of PRDs for a large variety of services without any efforts by the services. Furthermore, the PRDDS provides a feedback system for users to report erroneous PRDs.

<sup>8</sup> In comparison to the manual analysis of the Alexa Top 500 websites, where 60% have no user accounts, 593,512 websites (59%) is in the same range and as expected.

<sup>9</sup> The list is available at <http://www.passwordpolicy.info/prds.txt> as well as some examples at <http://www.passwordpolicy.info/prds.zip>.

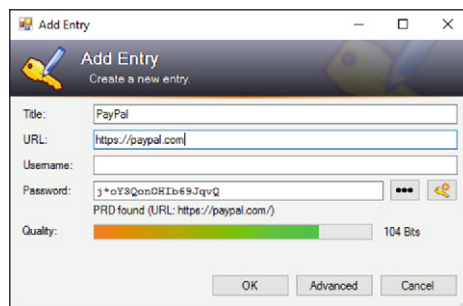
The connections to the PRDDS are TLS secured in order to prevent manipulation of the PRDs or eavesdropping on the requests. This prevents user profiling by third party attackers.

The disadvantage of the central PRDDS is, that the PRDDS itself in principle learns the services accessed by users. Well-known mitigations to this problem are connections over web proxies or VPN networks (cf. [1,4]) to anonymize the communication. While the TLS connection prevents the proxy or VPN provider to learn the contents of the requests, the named services hide the identity of users from the PRDDS. An even stronger guarantee for anonymity can be achieved by routing the communication through the TOR network [3].

An alternative to the PRDDS is the provision of PRDs by the services themselves using the *well-known location* scheme as specified in the RFC 5785 [17]. The PRD of a service would then be available at <https://www.example.org/well-known/prd.xml>. This enables applications to retrieve the PRD directly from each service in a unified way and removes the privacy issue and the requirement to trust in the correctness of the PRDs provided by the PRDDS. On the negative side, such a decentralized distribution depends on the cooperation of services and would entail a presumably long transition time. This gap is bridged by the PRDDS.

## 7 PRD-based Password Generation

To complete our tool set and demonstrate the use of PRDs by password generators we implemented an extension for KeePass<sup>10</sup>. Our extension allows users to enter the URL of a service in order to generate a password which complies with its password requirements (cf. Fig. 3). After entering the URL our extension retrieves the corresponding PRD from the PRDDS, configures the KeePass'



**Fig. 3.** KeePass Extension. After entering “<https://paypal.com>” in the URL field, the PRD for PayPal is retrieved from the PRDDS, KeePass’ password generator is configured, and a password is generated.

<sup>10</sup> KeePass [20] is an open-source password manager and provides an extension framework that allows third-party developers to enhance it with additional functionality.

password generator, and generates a password. Users neither need to find out the password requirement nor configure the KeePass' password generator manually.

## 8 Conclusion and Future Work

In this paper, we presented PRML which is intended to specify PRDs for Internet services. A PRD uniformly describes services' password requirements and makes them thereby available to password generators. This solves the two main challenges of using password generators today. First, users do not face the problem of rejected passwords anymore. Second, users do not need to look up the password requirements of a service and manually configure the password generator accordingly. The practicality of our solution and its capability to solve these problems have been demonstrated with the implementation of the PRC along with the according distribution service and an extension for KeePass which employs the PRDs for the password generator configuration. Our evaluation shows, that the PRC with the use of Natural Language Processing achieves a rate of 91,5% for the correct generation of PRDs. The PRC was used to generate PRDs for 72,124 services which are available through our distribution service. In essence, our contribution makes the usage of password generators as easy and comfortable as possible and helps users to employ strong passwords for their user accounts.

The generation of PRDs for further services is part of ongoing work. The database of PRDs forms the basis for a comprehensive security analysis of password requirements on the Internet.

## A Fictitious Example PRD

The PRD in Listing 1.1 defines the character sets lowercase letter, uppercase letters, numbers, special characters, and spaces. It requires that a password contains at least one special character and one number. Furthermore, the first character must be a lowercase letter. The password should not have more than 3 consecutive identical characters. The minimum password length is 10 and the maximum is 20 characters.

**Listing 1.1.** Fictitious Example PRD.

```
<prd url="http://www.example.com"version="1.0">
  <characterSets>
    <characterSet name="LowercaseLetters">
      <characters>abcdefghijklmnopqrstuvwxy</characters>
    </characterSet>
    <characterSet name="UppercaseLetters">
      <characters>ABCDEFGHIJKLMNQPQRSTUVWXYZ</characters>
    </characterSet>
    <characterSet name="Numbers">
      <characters>0123456789</characters>
    </characterSet>
    <characterSet name="Special">
      <characters>.,:;_-</characters>
```

```

    </characterSet >
    <characterSet name="Space">
        <characters > </characters >
    </characterSet >
</characterSets >
<properties >
    <characterSettings >
        <characterSet name="Special">
            <minOccurs>1</minOccurs >
        </characterSet >
        <characterSet name="Numbers">
            <minOccurs>1</minOccurs >
        </characterSet >
        <positionRestriction characterSet="LowercaseLetters">
            <positions>1</positions >
            <minOccurs>1</minOccurs >
        </positionRestriction >
    </characterSettings >
    <maxConsecutive>3</maxConsecutive >
    <minLength>10</minLength >
    <maxLength>20</maxLength >
</properties >
</prd >

```

## B Keywords

In the following appendix, we list the keywords that are used by the Keyword Annotator (cf. Sect. 4.1). The keywords are separated in three categories. First, keywords that are intended to find requirements with respect to the character sets. Second, usage verbs to interpret negation such as “do not use”. Third, keywords that focus on the minimum and maximum password length. We took the keywords from the password requirements of 20 websites and extended them by linguistically related words and terms (e.g. characters → character → chars → char).

### *Character Sets*

- alphanumeric, alphanumeric characters
- characters, character, chars, char
- number, numbers, numeral, numerals, numeric character, numeric characters
- english characters, non-blank characters, letter, letters, alphabetic, alphabetic character, alphabetic characters
- lowercase, lowercase letter, lowercase letters, lowercase character, lowercase characters, lower case, lower case letter, lower case letters, lower case character, lower case characters
- uppercase, uppercase letter, uppercase letters, uppercase character, uppercase characters, upper case, upper case letter, upper case letters, upper case character, upper case characters, capital letter, capital letters, capital character, capital characters

- space, space character, space character, spaces, blanks, blank spaces, blank space
- special character, special characters, special char, special chars, symbol, symbols
- consecutive, repeat, repeated, sequences, same

### *Usage Verbs*

- contain, include, use, have, exceed

### *Lengths*

- minimum, min, at least, more than, longer than, or longer, or more
- maximum, max, less than, fewer than, at most, up to, or less
- exactly

## References

1. Anonymizer. <https://www.anonymizer.com/>
2. Password requirements. <http://passrequirements.com>
3. TOR Project: Anonymity Online. <https://www.torproject.org>
4. TorGuard : online privacy protection services. <https://torguard.net>
5. Adams, A., Sasse, M.A., Lunt, P.: Making Passwords Secure and Usable. In: Thimbleby, H., Conaill, B., Thomas, P.J. (eds.) *People and Computers XII*, pp. 1–19. Springer, London (1997)
6. Alexa Internet. The top 500 sites on the web. <http://www.alexa.com/topsites>
7. AlFayyadh, B et al.: Improving usability of password management with standardized password policies, p. 8. Australia (2011)
8. Apache Software Foundation. Apache UIMA (2015). <https://uima.apache.org/>
9. Bishop, M., Klein, D.V.: Improving system security via proactive password checking. *Comput. Secur.* **14**(3), 233–249 (1995)
10. Bonneau, J.: The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In: *IEEE Symposium on Security and Privacy, SP 2012*, pp. 538–552. IEEE Computer Society, San Francisco, California, USA, 21–23 May 2012
11. Cambria, E., White, B.: Jumping NLP curves: A review of natural language processing research [review article]. *IEEE Comput. Int. Mag.* **9**(2), 48–57 (2014)
12. Castelluccia, C., Abdelber, C., Dürmuth, M., Perito, D.: When privacy meets security: leveraging personal information for password cracking. *CoRR*, abs/1304.6584 (2013)
13. Dell’Amico, M., Michiardi, P., Roudier, Y., Password strength: an empirical analysis. In: *INFOCOM*, pp. 983–991. IEEE (2010)
14. Ferrucci, D.A., Lally, A.: UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.* **10**(3–4), 327–348 (2004)
15. Florêncio, D. A. F., Herley, C.: A large-scale study of web password habits. In: Williamson, C. L., Zurko, M. E., Patel-Schneider, P. F., Shenoy, P. J. (eds.) *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pp. 657–666. ACM, Banff, Alberta, Canada, 8–12 May, 2007



16. Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., Lopez, J.: Guess again (and again, again): measuring password strength by simulating password-cracking algorithms. In: IEEE Symposium on Security and Privacy, SP 2012, pp. 523–537. IEEE Computer Society, San Francisco, California, USA, 21–23 May, 2012
17. Nottingham, M., Hammer-Lahav, E.: Defining Well-Known Uniform Resource Identifiers (URIs), RFC 5785 (2010). <https://tools.ietf.org/html/rfc5785>
18. Quantcast. Quantcast Top Million U.S. Web Sites (2015). <https://www.quantcast.com>
19. RANDOM.ORG Ltd. RANDOM.ORG Password Generator. <https://www.random.org/passwords/>
20. Reichl, D.: KeePass Password Safe (2015). <http://keepass.info>
21. Shay, R., Bertino, E.: A comprehensive simulation tool for the analysis of password policies. *Int. J. Inf. Sec.* **8**(4), 275–289 (2009)
22. Shay, R., Bhargav-Spantzel, A., Bertino, E.: Password policy simulation and analysis. In: *Digital Identity Management*, pp. 1–10. ACM (2007)
23. Squicciarini, A.C., Bhargav-Spantzel, A., Bertino, E., Czeksis, A.B.: Auth-SL - a system for the specification and enforcement of quality-based authentication policies. In: Qing, S., Imai, H., Wang, G. (eds.) *ICICS 2007*. LNCS, vol. 4861, pp. 386–397. Springer, Heidelberg (2007)
24. Stajano, F., Spencer, M., Jenkinson, G., Stafford-Fraser, Q.: Password-Manager Friendly (PMF): semantic annotations to improve the effectiveness of password managers. In: Mjølsnes, S.F., Forler, C., List, E., Lucks, S., Wenzel, J., Dürmuth, M., Kranz, T., Chang, D., Jati, A., Mishra, S., Sanadhya, S.K., Stajano, F., Spencer, M., Jenkinson, G., Stafford-Fraser, Q., Bicakci, K., Satiev, T., Tihanyi, N., Kovács, A., Vargha, G., Lénárt, Á., Jaeger, D., Graupner, H., Sapegin, A., Cheng, F., Meinel, C., Sandvoll, M., Boyd, C., Larsen, B.B., Kumar, A., Lauradoux, C., Millican, J. (eds.) *PASSWORD 2014*. LNCS, vol. 9393, pp. 61–73. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24192-0\_4
25. Thoeing, C.: PWGen (2015). <http://pwgen-win.sourceforge.net>
26. Wang, D., Wang, P.: The emperor’s new password creation policies: an evaluation of leading web services and the effect of role in resisting against online guessing. In: Pernul, G., et al. (eds.) *ESORICS 2015, Part II*. LNCS, vol. 9327, pp. 456–477. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24177-7\_23
27. Weir, M., Aggarwal, S., Collins, M. P., Stern, H.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: Al-Shaer, E., Keromytis, A. D., Shmatikov, V. (eds.). In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010*, pp. 162–175. ACM, Chicago, Illinois, USA, 4–8 October, 2010
28. Weir, M., Aggarwal, S., de Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: *IEEE Symposium on Security and Privacy*, pp. 391–405. IEEE Computer Society (2009)
29. Zviran, M., Haga, W.J.: Password security: an empirical study. *J. Manage. Inf. Syst.* **15**(4), 161–186 (1999)

# **Functional Encryption and Attribute-Based Cryptosystem**

# Leakage-Resilient Functional Encryption via Pair Encodings

Zuoxia Yu<sup>1</sup>, Man Ho Au<sup>1(✉)</sup>, Qiuliang Xu<sup>2</sup>, Rupeng Yang<sup>2</sup>,  
and Jinguang Han<sup>3,4</sup>

<sup>1</sup> Department of Computing, The Hong Kong Polytechnic University,  
Hung Hom, Hong Kong

zuoxia.yu@polyu.edu.hk, csallen@comp.polyu.edu.hk

<sup>2</sup> School of Computer Science and Technology, Shandong University,  
Jinan 250101, China

xql@sdu.edu.cn, orbyrp@gmail.com

<sup>3</sup> Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance  
and Economics, Nanjing 210003, China

jghan22@gmail.com

<sup>4</sup> State Key Laboratory of Information Security, Institute of Information  
Engineering, Chinese Academy of Sciences, Beijing 100093, China

**Abstract.** Leakage-resilient cryptography is proposed to address physical attacks on real world crypto-systems. Dual system encryption methodology is developed to guide design and analysis of various functional encryption schemes (FEs) with adaptive security. Observing the compatibility of dual system methodology and leakage-resilience, Lewko et al. present constructions of a number of strong leakage-resilient functional encryptions. In particular, they present fully secure identity-based encryption (IBE), hierarchical IBE (HIBE) and attribute-based encryption (ABE) satisfying the continual memory leakage (CML) model, one of the strongest models that allows continuous leakage on both user and master secret keys.

Inspired by the recent work from Attrapadung on pair encodings which greatly simplifies the design and analysis of FE, we propose a generic framework for constructing fully secure FEs in the CML model (LR-FEs). Specifically, our framework “compiles” predicate encodings into fully secure LR-FEs in a two-step process. Firstly, we propose a generic transformation of pair encoding schemes into their leakage-resilient forms. Next, we present another conversion that turns leakage-resilient pair encodings into fully secure LR-FEs. Our framework is highly compatible with Attrapadung’s, meaning that it is applicable to many existing pair encoding schemes.

The contribution of this paper is threefold. Firstly, our framework simplifies the design and analysis of LR-FEs into the design and analysis of predicate encodings. Secondly, our framework allows us to improve the security of some existing LR-FEs, such as LR-IBE with a tighter reduction. Thirdly, we discover new adaptively secure LR-FEs, including FE for regular languages, ABE for large universe and ABE with short ciphertext.

## 1 Introduction

**Leakage-Resilient Cryptography.** Traditionally, security of a cryptographic scheme relies on the secrecy of its secret states. In practice, however, state information could be revealed from measurements of the physical attributes of the device on which the cryptographic system is deployed. Attacks based on this extra information, such as timing attacks [19], power attacks [18], cold-boot attacks [16], etc., are grouped under the umbrella term of side-channel attacks.

Leakage-resilient cryptography was developed to address side-channel attacks. To model the additional capability in side-channel attacks, an attacker is allowed to submit an efficiently computable leakage function  $f$  to obtain the output of  $f$  on the current secret states of the cryptographic system. Many leakage models, differ in the restrictions imposed on  $f$ , have been proposed. Among them, the continual memory leakage (CML) model [11, 13] is believed to best describe those real world attacks. In this model, the entire lifetime of a scheme is divided into periods. At the end of each period, the secret state of the scheme is updated. The amount of leakage information in each time period is bounded, but the total amount of leakage during the lifetime of the scheme is unbounded. Since then, many cryptographic primitives have been designed in this model, include signatures [25], public-key encryption [14, 34], identity-based encryption (IBE) [22, 35], attribute-based encryption (ABE) [22], multiparty computation [10], etc.

**Functional Encryption (FE).** We recall the notion of FEs following the terminology of [1]. FE is a new paradigm of public-key encryption that supports fine-grained access control policy. In FEs, secret keys are associated with attributes  $X \in \mathcal{X}$ , ciphertexts are associated with attributes  $Y \in \mathcal{Y}$ , and a secret key can decrypt a ciphertext if and only if  $R(X, Y) = 1$ , where  $R$  is a predicate (access control policy) for attribute sets  $\mathcal{X}$  and  $\mathcal{Y}$ . For example, IBE [8, 30] can be viewed as a kind of FEs where  $R$  checks for equality. Achieving fully secure FEs even in the leakage-free setting is difficult, since the attacker can choose the target after obtaining many decryption keys for arbitrary attributes of its choice. In other words, a successful security proof requires the construction of a simulator that can generate decryption keys for arbitrary attributes and at the same time, the ability to obtain information from the challenge ciphertext (by the attacker) should be useful for the simulator. This seems to pose a dilemma: if the simulator can generate all decryption keys, it can generate a key to decrypt the challenge ciphertext by itself and thus the attacker will be of no use. Indeed, many FEs [5–7, 15, 29] can be proven secure when the attacker has to declare the target, say  $Y^*$ , before seeing the public parameters. This avoids the dilemma since the simulator can be constructed in a way that it can only generate decryption keys associated with  $X$  as long as  $R(X, Y^*) = 0$ .

**Dual System Methodology.** The dilemma was solved in 2009 by the seminal work of Waters [31]. The main idea is to utilise two types of keys and ciphertexts, namely, normal and semi-functional, in the security proof. Both semi-functional keys and ciphertext behave normally except that a semi-functional key cannot decrypt a semi-functional ciphertext. In the security proof, the simulator can only

generate semi-functional keys and ciphertexts. Now the simulator cannot create a key to decrypt the challenge ciphertext itself. The remaining part of the dual system methodology is, roughly speaking, to prove that the behavior of any attacker is the same regardless whether or not it is given normal or semi-functional keys and ciphertext. This strategy, now commonly known as dual system methodology, allows the constructions of many FEs [3, 20, 21, 23, 24, 26, 27].

**Dual System Methodology and Leakage Resilient Cryptography.** Lewko et al. [22] observed that dual system methodology supports security proof of cryptographic systems in the leakage model readily. The main idea is that in a dual system security proof, the simulator is capable of generating arbitrary decryption keys (in semi-functional form). If it can generate a key, it can naturally allow leakage of these keys. Based on this idea, they are able to introduce leakage-resilient IBE, ABE and HIBE in CML.

**Unifying Frameworks.** Based on the observation that many existing FEs in the dual system paradigm exhibit very similar properties, Attrapadung [1] and Wee [33] independently proposed unifying frameworks which support modular design and analysis of FEs. Both frameworks reduce the study of FEs to the study of a simpler object called a pair encoding in [1] or predicate encoding in [33]. Specifically, given encoding scheme  $P$ , one could convert  $P$  generically into a fully secure FE following the framework. Recently, there are a number of enhancement to these frameworks, including one that works in prime order groups [2, 12].

**The Problem Statement.** If FEs can be unified as in [1, 33], can LR-FEs be unified in a similar manner? Furthermore, can the new framework be built upon existing one so that one can transform existing encoding schemes into LR-FEs?

## 1.1 Our Contributions

In this paper, we give positive answers to the above questions. Inspired by the techniques of [22] and the framework of [1], we propose a framework for leakage-resilient adaptively secure FEs. The main component of our framework is the definition of a new primitive that we called leakage-resilient pair encodings. Our framework involves two transformations which are illustrated in Fig. 1:

- **Transformation from Pair Encodings to Leakage-Resilient Pair Encodings.** First, we define the notion of leakage-resilient pair encodings. Then, we present a transformation technique that turns a large class of pair encodings into leakage-resilient pair encodings.
- **Transformation from Leakage-Resilient Pair Encodings to Leakage-Resilient FEs.** We show how fully secure FEs in the CML model can be obtained generically from leakage-resilient pair encodings.

Concrete FEs as a result of our framework will be presented in Sect. 6 in detail. Looking ahead, our framework leads to improved schemes such as LR-IBE with tighter reduction and new schemes such as LR-FE for regular languages, LR-ABE for large universe and LR-ABE with short-ciphertext, all with adaptive security in the standard model and the CML model.

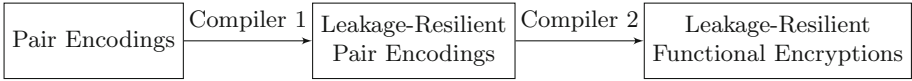


Fig. 1. Main results of our paper

### 1.2 Organization of Our Paper

The rest of the paper is organized as follows. In Sect. 2, we give some basic notations and definitions. Then we formalize the definition of leakage-resilient pair encoding schemes in Sect. 3. In Sect. 4, we show how to transform leakage-resilient pair encoding schemes to leakage-resilient adaptively secure FEs in the CML model. In Sect. 5, we introduce the transformation from pair encoding schemes to leakage-resilient pair encoding schemes. In Sect. 6, we describe several concrete constructions of leakage-resilient FEs in the CML model based on existing pair encoding schemes.

## 2 Preliminaries

**Notation.** We denote by  $r \xleftarrow{\$} R$  the fact that  $r$  is picked randomly and uniformly from finite set  $R$ . We use  $[n]$  to denote the set  $\{1, 2, \dots, n\}$ . We use lowercase (resp. uppercase) letters in boldface to denote vectors (resp. matrixes).  $\mathbf{0}^m$  denotes a zero vector whose length is  $m$ ,  $\mathbf{0}^{m \times n}$  denotes a zero matrix with  $m$  rows and  $n$  columns. Note that we treat all vectors in this paper as row vectors. Let  $\mathbb{G}$  be a group, and  $g \in \mathbb{G}$  is a group element. For  $g \in \mathbb{G}$  and  $\mathbf{k} = (k_1, k_2, \dots, k_n) \in \mathbb{Z}_N^n$ ,  $g^{\mathbf{k}}$  denotes  $(g^{k_1}, g^{k_2}, \dots, g^{k_n})$ . We use  $\cdot$  to denote dot product operation of vectors, and  $*$  to denote component-wise multiplication of vectors.

### 2.1 Composite Order Bilinear Groups and Cryptographic Assumptions

Our framework works in composite order group introduced in [9]. Let  $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$  denote bilinear groups of composite order  $N = p_1 p_2 p_3$ , where  $p_1, p_2$  and  $p_3$  are distinct primes, with a non-degenerate and efficiently computable bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ . The non-degenerate property of  $e$  means that  $e(g, h) \neq 1 \in \mathbb{G}_{\mathbb{T}}$  if  $g, h \neq 1 \in \mathbb{G}$ , and the bilinear property of  $e$  means  $e(g^a, g^b) = e(g, g)^{ab}$  for any

$g \in \mathbb{G}$  and  $a, b \in Z_N$ . Let  $\mathcal{G}(\lambda)$  be a bilinear group generator algorithm that takes as input a security parameter and outputs  $(\mathbb{G}, \mathbb{G}_T, e, N, p_1, p_2, p_3)$ , where  $(\mathbb{G}, \mathbb{G}_T)$  are bilinear groups with composite order  $N$ . If  $p|N$ , there exists a subgroup  $\mathbb{G}_p$  of  $\mathbb{G}$  with order  $p$ . We also have orthogonality for  $e$ : for any  $g \in \mathbb{G}_{p_i}$  and  $h \in \mathbb{G}_{p_j}$ , if  $p_i \neq p_j$ , then  $e(g, h) = 1$ .

**Subgroup Decision Assumptions (SD).** Our construction relies on the following three assumptions which are presented in [23, 31]. Each of the SD assumptions starts with  $\mathcal{G}(\lambda) \rightarrow (\mathbb{G}, \mathbb{G}_T, e, N, p_1, p_2, p_3)$ .

- SD1:** Given  $D = (g_1, g_3)$  and  $T \in \mathbb{G}$ , decides if  $T = T_1 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1 p_2}$  or  $T = T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1}$ , where  $g_1 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1}$ ,  $g_3 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_3}$ .
- SD2:** Given  $D = (g_1, Z_1 Z_2, g_3, W_2 W_3)$  and  $T \in \mathbb{G}$ , decides if  $T = T_1 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1 p_2 p_3}$  or  $T = T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1 p_3}$ , where  $g_1, Z_1 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1}$ ,  $W_2, Z_2 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_2}$ ,  $g_3, W_3 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_3}$ .
- SD3:** Given  $D = (g_1, g_2, g_3, g_1^\alpha Y_2, g_1^s W_2)$  and  $T \in \mathbb{G}$ , decides if  $T = T_1 = e(g_1, g_1)^{\alpha s}$  or  $T = T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_T$ , where  $g_1 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_1}$ ,  $g_2, W_2, Y_2 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_2}$ ,  $g_3 \stackrel{\$}{\leftarrow} \mathbb{G}_{p_3}$  and  $\alpha, s \stackrel{\$}{\leftarrow} Z_N$ .

The advantage of adversary  $\mathcal{A}$  for problem instance  $SD_i$  is defined as follows:

$$Adv_{\mathcal{A}}^{SD_i}(\lambda) = |Pr[\mathcal{A}(D, T_1)] - Pr[\mathcal{A}(D, T_2)]|.$$

Then assumptions **SD1**, **SD2** and **SD3** for  $\mathcal{G}$  assert that  $Adv_{\mathcal{A}}^{SD_i}(\lambda)$  is negligible for all probabilistic polynomial time adversary  $\mathcal{A}$ .

## 2.2 Functional Encryption

In this paper, we use the notion of predicate family in [1]. A **predicate family**  $\mathcal{R} = \{R_k\}_{k \in N^c}$  for some constant  $c \in N$ , where  $R_k : \mathcal{X}_k \times \mathcal{Y}_k \rightarrow \{0, 1\}$  is a predicate function that maps a pair of attributes (one in key space  $\mathcal{X}_k$ , the other one in ciphertext space  $\mathcal{Y}_k$ ) to  $\{0, 1\}$ .  $k$  is the index which specifies a description of predicate  $R_k \in \mathcal{R}$ . We require that the first entry of  $k$  specifies the domain of predicate function  $R_k$ , and write simply  $R_N$  to denote  $R_k$  whose domain is  $Z_N$ .  $R_N$  is *domain-transferable* if for  $p$  that divides  $N$ , then there exists projection maps  $f_1 : \mathcal{X}_N \rightarrow \mathcal{X}_p$ ,  $f_2 : \mathcal{Y}_N \rightarrow \mathcal{Y}_p$ , such that for all  $X \in \mathcal{X}_N$ ,  $Y \in \mathcal{Y}_N$ , if  $R_N(X, Y) = 1$ , then  $R_p(f_1(X), f_2(Y)) = 1$ , and if  $R_N(X, Y) = 0$ , then  $R_p(f_1(X), f_2(Y)) = 0$ .

**Functional Encryption (FE)** [1]. A functional encryption for predicate family  $\mathcal{R}$  is a tuple of four algorithms (**Setup**, **KeyGen**, **Encrypt**, **Decrypt**) whose description is given below.

**Setup** $(\lambda, k) \rightarrow (\text{PK}, \text{MSK})$ . The setup algorithm takes as input security parameter  $\lambda$ , family index  $k$  for predicate family  $\mathcal{R}$ , and outputs master secret key MSK and master public key PK.

**KeyGen**( $X, MSK, PK$ )  $\rightarrow SK$ . The key generation algorithm takes input  $MSK$ ,  $PK$ , an attribute  $X$  from  $\mathcal{X}_k$ , and outputs the secret key  $SK$  for  $X$ .

**Encrypt**( $Y, M, PK$ )  $\rightarrow CT$ . The encryption algorithm gets input an attribute  $Y \in \mathcal{Y}_k$ , message  $M$  and  $PK$ , then outputs a ciphertext  $CT$ .

**Decrypt**( $SK, CT$ )  $\rightarrow M$ . The decryption algorithm gets input  $SK$  for attribute  $X$  and  $CT$  for attribute  $Y$ , then outputs  $M$  or  $\perp$ .

**Correctness.** For all index  $k$ ,  $M \in \mathcal{M}$ ,  $X \in \mathcal{X}_k$  and  $Y \in \mathcal{Y}_k$ , such that  $R_k(X, Y) = 1$ , if **Encrypt**( $Y, M, PK$ )  $\rightarrow CT$ , **KeyGen**( $X, MSK, PK$ )  $\rightarrow SK$ , and **Decrypt**( $SK, CT$ )  $\rightarrow M'$ , then  $M = M'$ .

**Adaptive Security.** For any stateful adversary  $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ , we define advantage of  $\mathcal{A}$  as follows:

$$Adv_{\mathcal{A}}^{FE}(\lambda) = Pr \left[ \begin{array}{l} \mathbf{Setup} : (MSK, PK) \leftarrow \mathbf{Setup}(\lambda, k); \\ \mathbf{Phase1} : (st, M_0, M_1) \leftarrow \mathcal{A}_1^{\mathbf{KeyGen}(MSK, \cdot)}(PK); \\ b = b' : \mathbf{Challenge} : \begin{cases} b \xleftarrow{\$} \{0, 1\}; \\ CT \leftarrow \mathbf{Encrypt}(Y, M_b, PK); \end{cases} \\ \mathbf{Phase2} : \mathcal{A}_2^{\mathbf{KeyGen}(MSK, \cdot)}(CT, st); \\ \mathbf{Guess} : b' \leftarrow \mathcal{A}_2; \end{array} \right] - \frac{1}{2},$$

with the restriction that all key queries  $X$  that  $\mathcal{A}$  made to **KeyGen**( $MSK, X$ ) satisfies  $R_k(X, Y) = 0$ . A functional encryption scheme is **adaptively secure** if for all PPT adversary  $\mathcal{A}$ ,  $Adv_{\mathcal{A}}^{FE}(\lambda)$  is negligible.

### 2.3 A Lemma for Leakage-Resilient Analysis

The analysis of leakage-resilient property of our pair encoding scheme relies on the following lemma from [22]. Let  $\Delta(X_1, X_2)$  denote the statistical distance of two random variables.

**Lemma 1.** *Let  $m \in N$ ,  $m \geq 3$ , and  $p$  be a prime. Let  $\delta \xleftarrow{\$} Z_p^m$ ,  $\tau \xleftarrow{\$} Z_p^m$ , and  $\tau'$  be chosen uniformly and randomly from the set of vectors in  $Z_p^m$  which are orthogonal to  $\delta$  under dot product modulo  $p$ . Let  $f : Z_p^m \rightarrow W$  be some function. Then:*

$$\Delta((\delta, f(\tau)), (\delta, f(\tau'))) \leq \epsilon,$$

as long as

$$|W| \leq 4 \cdot \left(1 - \frac{1}{p}\right) \cdot p^{m-2} \cdot \epsilon^2.$$

## 3 Leakage-Resilient Pair Encoding Scheme

We first define a new notion called leakage-resilient pair encodings. Then we present a few definitions regarding its security requirements. Our definitions can be thought of as extending the definition of pair encodings [1] to capture leakage-resilience.



### 3.1 Syntax

A leakage-resilient pair encoding scheme for predicate family  $\mathcal{R}$  is a tuple of four deterministic algorithms (**Param**, **Enc1**, **Enc2**, **Pair**) whose description is given below.

**Param**( $\lambda$ )  $\rightarrow (n_1, n_2)$ .  $n_1$  denotes the number of the public variables  $\mathbf{h} = (h_1, h_2, \dots, h_{n_1})$  used in encoding algorithms,  $n_2$  denotes the number of another public variables  $\mathbf{x} = (x_1, x_2, \dots, x_{n_2})$  used in encoding algorithms.

**Enc1**( $X, N$ )  $\rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , where  $X \in \mathcal{X}$ ,  $N$ ,  $m_1$ ,  $m_2$  and  $m_3 \in \mathbb{N}$ .  $\mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})$  is a sequence of polynomials  $\{k_{1,\ell}\}_{\ell \in [m_1]}$ , each polynomial  $k_{1,\ell}$  is in  $\{\alpha\} \cup \{h_j\}_{j \in [n_1]} \cup \{x_k\}_{k \in [n_2]} \cup \{r_i\}_{i \in [m_3]}$ . Meanwhile,  $\mathbf{k}_2(\mathbf{h}, \mathbf{r})$  is a sequence of polynomials  $\{k_{2,\ell}\}_{\ell \in [m_2]}$  and each  $k_{2,\ell}$  is a polynomial in  $\{h_j\}_{j \in [n_1]} \cup \{r_i\}_{i \in [m_3]}$ . More precisely, this algorithm outputs two sets of coefficients, namely,  $\{C_\ell, C_{\ell,i}, C_{\ell,i,j}, C_{\ell,i,k}\}$  and  $\{C'_{\ell',i}, C'_{\ell',i,j}\}$ , defining two sequences of polynomials

$$\mathbf{k}_{1,\ell}(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r}) = C_\ell \alpha + \left( \sum C_{\ell,i} \cdot r_i \right) + \left( \sum C_{\ell,i,j} \cdot r_i h_j \right) + \sum C_{\ell,i,k} \cdot r_i x_k, \quad (1)$$

$$\mathbf{k}_{2,\ell'}(\mathbf{h}, \mathbf{r}) = \sum C'_{\ell',i} \cdot r_i + \sum C'_{\ell',i,j} \cdot r_i h_j, \quad (2)$$

where  $\ell \in [m_1], \ell' \in [m_2], i \in [m_3], j \in [n_1], k \in [n_2]$ .

The input attribute  $X$  of **Enc1** could be empty string  $\varepsilon$ . In this case, algorithm **Enc1** also works in the same way as mentioned above. For any set of predicate attribute  $\mathcal{X}$ , we require that the output of *Enc1* satisfies the regeneration property, meaning that for all **Enc1**( $X, N$ )  $\rightarrow (\mathbf{k}_1, \mathbf{k}_2)$ ( $X \in \mathcal{X}$ ) and **Enc1**( $\varepsilon, N$ )  $\rightarrow (\mathbf{k}_1^\varepsilon, \mathbf{k}_2^\varepsilon)$ , the following property holds:

Given  $(\mathbf{k}_1^\varepsilon(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r}), \mathbf{k}_2^\varepsilon(\mathbf{h}, \mathbf{r}))$  and non-empty attribute  $X \in \mathcal{X}$ , there exists an efficient algorithm that can compute  $(\mathbf{k}'_1(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r}'), \mathbf{k}'_2(\mathbf{h}, \mathbf{r}'))$  whose distribution is the same as  $(\mathbf{k}_1, \mathbf{k}_2)$ .

**Enc2**( $Y, N$ )  $\rightarrow (\mathbf{c} = (c_1, c_2, \dots, c_{w_1}); w_2)$ .  $\mathbf{c} = (c_1, c_2, \dots, c_{w_1})$  is a set of polynomials output by this encoding scheme with coefficients in  $Z_N$ . We require that each  $c_i$  is a polynomial in  $\{h_j\}_{j \in [n_1]} \cup \{s_i\}_{i \in [w_2]} \cup \{x_k\}_{k \in [n_2]} \cup \{s\}$ .

$$c_i(\mathbf{s}, \mathbf{x}, \mathbf{h}) = a_i s + \sum a_{i,j} s_j + \sum a'_{i,k} x_k s + \sum a_{i,j} h_j s + \sum a_{i,i,j} h_j s_i, \quad (3)$$

where  $\mathbf{s} = (s, s_1, s_2, \dots, s_{w_2}) \xleftarrow{\$} Z_N^{w_2+1}$ .

**Pair**( $X, Y, N$ )  $\rightarrow E$ . This algorithm takes as input  $X, Y, N$ , and outputs  $E \in Z_N^{(m_1+m_2) \times w_1}$ .

**Correctness.** For all valid attribute  $X$  ( $X$  is not empty), **Enc1**( $X, N$ )  $\rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , **Enc2**( $Y, N$ )  $\rightarrow (\mathbf{c}; w_2)$ , and **Pair**( $X, Y, N$ )  $\rightarrow E$ , we require that If  $R(X, Y) = 1$ , then

$$(\mathbf{k}_1, \mathbf{k}_2) \mathbf{E} \mathbf{c}^\top = \alpha s.$$

**$(\ell_1, \ell_2)$ -Leakage-Resilient Property (LR).**

We say a pair encoding scheme  $P$  defined above is  $(\ell_1, \ell_2)$ -leakage-resilient if for all probabilistic polynomial time adversary  $\mathcal{A}$ , advantage of  $\mathcal{A}$  in game  $\mathbf{Exp}_{\mathcal{G},b,\mathcal{A},LR}(\lambda)$  is negligible. We denote it by:

$$Adv_{\mathcal{A}}^{LR}(\lambda) = |Pr[\mathbf{Exp}_{\mathcal{G},0,\mathcal{A},LR}(\lambda) = 1] - Pr[\mathbf{Exp}_{\mathcal{G},1,\mathcal{A},LR}(\lambda) = 1]| \leq \text{negl}(\lambda).$$

$\mathbf{Exp}_{\mathcal{G},b,\mathcal{A},LR}(\lambda)$  :

1.  $(\mathbb{G}, \mathbb{G}_T, e, N, p_1, p_2, p_3) \leftarrow \mathcal{G}(\lambda)$ ;
2.  $g_1 \xleftarrow{\$} \mathbb{G}_{p_1}, g_2 \xleftarrow{\$} \mathbb{G}_{p_2}, g_3 \xleftarrow{\$} \mathbb{G}_{p_3}$ ;
3.  $\alpha \xleftarrow{\$} Z_N, (n_1, n_2) \leftarrow \mathbf{Param}(\lambda), \mathbf{h} \leftarrow Z_N^{n_1}, \mathbf{x} \leftarrow Z_N^{n_2}$ ;
4.  $st \leftarrow A_1^{\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^L(\cdot)}(g_1, g_2, g_3)$ ;
5.  $b' \leftarrow A_2^{\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^2(\cdot)}(st)$ ;

Here we require that both  $\mathcal{O}^L$  and  $\mathcal{O}^2$  are queried once only. Let  $f$  be any polynomial time computable leakage function whose output size is no longer than  $\ell_1$  or  $\ell_2$ .

- $\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^L(X, f)$ : run  $\mathbf{Enc1}(X, p_2) \rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , choose  $r \xleftarrow{\$} Z_{p_2}^{m_3}$ , return  $f(k)$ , where

$$k \leftarrow \begin{cases} (g_2^{\mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{x}, r)}, g_2^{\mathbf{k}_2(\mathbf{h}, r)}) & b = 0 \\ (g_2^{\mathbf{k}_1(0, \mathbf{h}, \mathbf{x}, r)}, g_2^{\mathbf{k}_2(\mathbf{h}, r)}) & b = 1 \end{cases}$$

*Notably*, the query  $X$  input to  $\mathcal{O}^L$  could be empty, in this case  $|f(k)| \leq \ell_1$ . If  $X \in \mathcal{X}$ , then  $|f(k)| \leq \ell_2$ .

- $\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^2(Y)$ : run  $\mathbf{Enc2}(Y, p_2) \rightarrow (\mathbf{c}; w_2)$ , choose  $\mathbf{s} \xleftarrow{\$} Z_{p_2}^{w_2+1}$ , return  $\mathbf{C} \leftarrow g_2^{\mathbf{c}(\mathbf{s}, \mathbf{h}, \mathbf{x})}$ .

**3.2 Security Definitions**

Similar to the definitions of pair encodings in [1], we define perfectly master-key hiding security and computationally master-key hiding security for leakage-resilient pair encodings.

**Perfectly Master-Key Hiding Security.** Let  $P$  be a leakage-resilient pair encoding scheme. For  $N \in \mathbb{N}$ , if  $R(X, Y) = 0$ , let  $\mathbf{Param}(\lambda) \rightarrow (n_1, n_2)$ ,  $\mathbf{Enc1}(X, N) \rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ ,  $\mathbf{Enc2}(Y, N) \rightarrow (\mathbf{c}, w_2)$ . If the following two distributions are identical, we say  $P$  is perfectly master-key hiding (PMH).

$$\{c(\mathbf{s}, \mathbf{h}, \mathbf{x}), \mathbf{k}_1(0, \mathbf{h}, \mathbf{x}, r), \mathbf{k}_2(\mathbf{h}, r)\} \quad \{c(\mathbf{s}, \mathbf{h}, \mathbf{x}), \mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{x}, r), \mathbf{k}_2(\mathbf{h}, r)\}$$

where the probability is taken over  $\mathbf{h} \xleftarrow{\$} Z_N^{n_1}$ ,  $\alpha \xleftarrow{\$} Z_N$ ,  $\mathbf{r} \xleftarrow{\$} Z_N^{m_3}$ ,  $\mathbf{s} \xleftarrow{\$} Z_N^{w_2+1}$ ,  $\mathbf{x} \xleftarrow{\$} Z_N^{n_2}$ .

**Computational Security.** We define selectively master-key hiding security (SMH) and co-selectively master-key hiding security (CMH) on a bilinear group generator  $\mathcal{G}$  respectively as follows. We define a game template  $\mathbf{Exp}_{\mathcal{G},b,\mathcal{A},T}(\lambda)$  to encompass these two security notions by using different types of Oracles ( $\mathcal{O}^T = (\mathcal{O}^{T,1}, \mathcal{O}^{T,2})$ ,  $T \in \{\text{SMH}, \text{CMH}\}$ ) for adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  as follows:

$\mathbf{Exp}_{\mathcal{G},b,\mathcal{A},T}(\lambda)$  :

1.  $(\mathbb{G}, \mathbb{G}_T, e, N, p_1, p_2, p_3) \leftarrow \mathcal{G}(\lambda)$ ;
2.  $g_1 \xleftarrow{\$} \mathbb{G}_{p_1}, g_2 \xleftarrow{\$} \mathbb{G}_{p_2}, g_3 \xleftarrow{\$} \mathbb{G}_{p_3}$ ;
3.  $\alpha \xleftarrow{\$} Z_N, (n_1, n_2) \leftarrow \mathbf{Param}(\lambda), \mathbf{h} \leftarrow Z_N^{n_1}, \mathbf{x} \leftarrow Z_N^{n_2}$ ;
4.  $st \leftarrow A_1^{\mathcal{O}^{T,1}_{b,\alpha,\mathbf{h},\mathbf{x}}(\cdot)}(g_1, g_2, g_3)$ ;
5.  $b' \leftarrow A_2^{\mathcal{O}^{T,2}_{b,\alpha,\mathbf{h},\mathbf{x}}(\cdot)}(st)$ ;

The advantage of  $\mathcal{A}$  in the corresponding security game  $\mathbf{Exp}_{\mathcal{G},b,\mathcal{A},T}(\lambda)$  is defined as follows:

$$\mathit{Adv}_{\mathcal{A}}^T(\lambda) = |\Pr[\mathbf{Exp}_{\mathcal{G},0,\mathcal{A},T}(\lambda)] = 1| - |\Pr[\mathbf{Exp}_{\mathcal{G},1,\mathcal{A},T}(\lambda)] = 1|.$$

The different types of Oracle  $\mathcal{O}^T$  are defined as follows:

- **Selective Security (SMH).**  $\mathcal{O}^{SMH,1}$  can be queried once while  $\mathcal{O}^{SMH,2}$  can be queried polynomially many times.
  - $\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^{SMH,1}(Y)$ : Run  $\mathbf{Enc2}(Y, p_2) \rightarrow (\mathbf{c}; w_2)$ , choose  $\mathbf{s} \xleftarrow{\$} Z_{p_2}^{w_2+1}$ , return  $\mathbf{C} \leftarrow g_2^{\mathbf{c}(\mathbf{s}, \mathbf{h}, \mathbf{x})}$ .
  - $\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^{SMH,2}(X)$ : If  $R_{p_2}(X, Y) = 1$  or  $X$  is empty, return  $\perp$ . Run  $\mathbf{Enc1}(X, p_2) \rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , pick  $\mathbf{r} \xleftarrow{\$} Z_{p_2}^{m_3}$ , return  $k \leftarrow \begin{cases} (g_2^{\mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})}, g_2^{\mathbf{k}_2(\mathbf{h}, \mathbf{r})}) & b = 0 \\ (g_2^{\mathbf{k}_1(0, \mathbf{h}, \mathbf{x}, \mathbf{r})}, g_2^{\mathbf{k}_2(\mathbf{h}, \mathbf{r})}) & b = 1 \end{cases}$
- **Co-selective Security (CMH).** Both  $\mathcal{O}^{CMH,1}$  and  $\mathcal{O}^{CMH,2}$  can be queried only once.
  - $\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^{CMH,1}(X)$ : If  $X$  is empty, return  $\perp$ . Otherwise, run  $\mathbf{Enc1}(X, p_2) \rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , choose  $\mathbf{r} \xleftarrow{\$} Z_{p_2}^{m_3}$ , return  $k \leftarrow \begin{cases} (g_2^{\mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})}, g_2^{\mathbf{k}_2(\mathbf{h}, \mathbf{r})}) & b = 0 \\ (g_2^{\mathbf{k}_1(0, \mathbf{h}, \mathbf{x}, \mathbf{r})}, g_2^{\mathbf{k}_2(\mathbf{h}, \mathbf{r})}) & b = 1 \end{cases}$
  - $\mathcal{O}_{b,\alpha,\mathbf{h},\mathbf{x}}^{CMH,2}(Y)$ : If  $R_{p_2}(X, Y) = 1$ , then return  $\perp$ . Otherwise, run  $\mathbf{Enc2}(Y, p_2) \rightarrow (\mathbf{c}; w_2)$ , choose  $\mathbf{s} \xleftarrow{\$} Z_{p_2}^{w_2+1}$ , return  $\mathbf{C} \leftarrow g_2^{\mathbf{c}(\mathbf{s}, \mathbf{h}, \mathbf{x})}$ .

We would like to remark that perfect master-key hiding implies CMH but not SMH as in [1].

## 4 From Leakage-Resilient Pair Encoding to Leakage-Resilient Functional Encryption

In this section, we first present a generic construction of leakage-resilient functional encryptions using leakage-resilient pair encoding as a building block. Then, we present a security analysis of our generic construction. Our generic construction is based on [1].

### 4.1 Generic Construction

Let  $P = (Param, Enc1, Enc2, Pair)$  be a  $(\ell_{msk}, \ell_{sk})$ -leakage-resilient pair encoding scheme for predicate family  $\mathcal{R}$ . We construct a functional encryption FE for predicate family  $\mathcal{R}$  as follows:

- **Setup** $(\lambda, k)$ : Run  $(\mathbb{G}, \mathbb{G}_T, e, N, p_1, p_2, p_3) \leftarrow \mathcal{G}(\lambda)$ , pick generator  $g_1 \xleftarrow{\$} \mathbb{G}_{p_1}$ ,  $g_3 \xleftarrow{\$} \mathbb{G}_{p_3}$ . Run  $Param(\lambda) \rightarrow (n_1, n_2)$ , pick  $\mathbf{h} \xleftarrow{\$} Z_N^{n_1}$ ,  $\mathbf{x} \xleftarrow{\$} Z_N^{n_2}$ . Run  $Enc1(\varepsilon, N) \rightarrow (\mathbf{k}_1^\varepsilon, \mathbf{k}_2^\varepsilon; m_1, m_2, m_3)$ , where  $\alpha \xleftarrow{\$} Z_N$ ,  $\mathbf{r} \xleftarrow{\$} Z_N^{m_3}$ ,  $\mathbf{R}_1 \xleftarrow{\$} Z_N^{m_1}$ ,  $\mathbf{R}_2 \xleftarrow{\$} Z_N^{m_2}$ . The public key is  $PK = (g_1, g_3, e(g_1, g_1)^\alpha, g_1^{\mathbf{h}}, g_1^{\mathbf{x}})$ . The master secret key is  $MSK = (\mathbf{K}_1, \mathbf{K}_2) = (g_1^{\mathbf{k}_1^\varepsilon(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})} * g_3^{\mathbf{R}_1}, g_1^{\mathbf{k}_2^\varepsilon(\mathbf{h}, \mathbf{r})} * g_3^{\mathbf{R}_2})$ .
- **KeyGen** $(MSK, X, PK)$ : Run  $Enc1(X, N) \rightarrow (\mathbf{k}_{X,1}, \mathbf{k}_{X,2}; m_1, m_2, m_3)$ . Parse  $MSK = (g_1^{\mathbf{k}_1^\varepsilon(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})} * g_3^{\mathbf{R}_1}, g_1^{\mathbf{k}_2^\varepsilon(\mathbf{h}, \mathbf{r})} * g_3^{\mathbf{R}_2})$ . Choose  $\mathbf{r}' \xleftarrow{\$} Z_N^{m_3}$ . Then compute user key as follows:

$$SK = (\mathbf{K}_1, \mathbf{K}_2) = (g_1^{\mathbf{k}_{X,1}(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r} + \mathbf{r}')} * g_3^{\mathbf{R}_1}, g_1^{\mathbf{k}_{X,2}(\mathbf{h}, \mathbf{r} + \mathbf{r}')} * g_3^{\mathbf{R}_2}).$$

Notably,  $SK$  can be computed from  $MSK$  due to the regeneration property of  $P$ .

- **Encrypt** $(Y, M, PK)$ : Run  $Enc2(Y, N) \rightarrow (\mathbf{c}; w_2)$ . Pick  $\mathbf{s} = (s, s_1, s_2, \dots, s_{w_2}) \xleftarrow{\$} Z_N^{w_2+1}$ , and compute the ciphertext  $CT = (C_0, \mathbf{C}_1)$  as follows:

$$C_0 = M \cdot e(g_1, g_1)^{\alpha \mathbf{s}}, \quad \mathbf{C}_1 = g_1^{\mathbf{c}(\mathbf{s}, \mathbf{h}, \mathbf{x})}.$$

- **Decrypt** $(CT, SK)$ : Obtain  $X, Y$  from  $SK$  and  $CT$ . Check whether  $R(X, Y) = 1$ . If yes, run  $Pair(X, Y, N) \rightarrow E$ . Next compute  $e((\mathbf{K}_1, \mathbf{K}_2)^E, \mathbf{C}_1)$  to get the blinding factor  $e(g_1, g_1)^{\alpha \mathbf{s}}$  and get the plaintext  $M$ . That is,

$$M = \frac{C_0}{e((\mathbf{K}_1, \mathbf{K}_2)^E, \mathbf{C}_1)}.$$

**Semi-Functional Algorithms.** These algorithms will be used in the security proof of our generic construction.

**SFSetup** $(\lambda, k)$ : This algorithm is nearly the same as **Setup** $(\lambda, k)$  except that it additionally outputs a generator  $g_2 \xleftarrow{\$} \mathbb{G}_2$ ,  $\hat{\mathbf{h}} \xleftarrow{\$} Z_N^{n_1}$  and  $\hat{\mathbf{x}} \xleftarrow{\$} Z_N^{n_2}$ . We call  $\hat{\mathbf{h}}$  and  $\hat{\mathbf{x}}$  semi-functional parameters.

**SFEncrypt**( $Y, M, \text{PK}, g_2, \hat{\mathbf{h}}, \hat{\mathbf{x}}$ ): This algorithm first run  $\text{Enc2}(Y) \rightarrow (\mathbf{c}; w_2)$ . Then choose  $\mathbf{s} = (s, s_1, s_2, \dots, s_{w_2}) \xleftarrow{\$} Z_N^{w_2+1}$ ,  $\hat{\mathbf{s}} \xleftarrow{\$} Z_N^{w_2+1}$ , output the semi-functional ciphertext  $CT = (C_1, C_0)$ :

$$C_1 = g_1^{\mathbf{c}(\mathbf{s}, \mathbf{h}, \mathbf{x})} * g_2^{\mathbf{c}(\hat{\mathbf{s}}, \hat{\mathbf{h}}, \hat{\mathbf{x}})}, \quad C_0 = M \cdot e(g_1, g_1)^{\alpha s}$$

**SFKeyGen**( $X, \text{MSK}, \text{PK}, g_2, \text{type}, \hat{\alpha}, \hat{\mathbf{h}}, \hat{\mathbf{x}}$ ): Note that,  $X$  is a predicate attribute (or an empty string). In both case, run algorithm  $\text{Enc1}(X, N) \rightarrow (\mathbf{k}_{X,1}, \mathbf{k}_{X,2}; m_1, m_2, m_3)$ . Next pick and  $\hat{\mathbf{r}} \xleftarrow{\$} Z_N^{m_3}$ . Then output one type of semi-function secret key (or master key) depending on the input type  $t \in \{1, 2, 3\}$ .

$$\mathbf{K} = \begin{cases} (g_1^{\mathbf{k}_{X,1}(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})} * g_2^{\mathbf{k}_{X,1}(0, \hat{\mathbf{h}}, \hat{\mathbf{x}}, \hat{\mathbf{r}})} * g_3^{\mathbf{R}}, g_1^{\mathbf{k}_{X,2}(\mathbf{h}, \mathbf{r})} * g_2^{\mathbf{k}_{X,2}(\hat{\mathbf{h}}, \hat{\mathbf{r}})} * g_3^{\mathbf{R}}) & t = 1 \\ (g_1^{\mathbf{k}_{X,1}(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})} * g_2^{\mathbf{k}_{X,1}(\hat{\alpha}, \hat{\mathbf{h}}, \hat{\mathbf{x}}, \hat{\mathbf{r}})} * g_3^{\mathbf{R}}, g_1^{\mathbf{k}_{X,2}(\mathbf{h}, \mathbf{r})} * g_2^{\mathbf{k}_{X,2}(\hat{\mathbf{h}}, \hat{\mathbf{r}})} * g_3^{\mathbf{R}}) & t = 2 \\ (g_1^{\mathbf{k}_{X,1}(\alpha, \mathbf{h}, \mathbf{x}, \mathbf{r})} * g_2^{\mathbf{k}_{X,1}(\hat{\alpha}, 0, 0, 0)} * g_3^{\mathbf{R}}, g_1^{\mathbf{k}_{X,2}(\mathbf{h}, \mathbf{r})} * g_2^{\mathbf{k}_{X,2}(0, 0)} * g_3^{\mathbf{R}}) & t = 3 \end{cases}$$

## 4.2 Security Definition for Functional Encryption in the CML Model

A functional encryption scheme is adaptively secure in the continual memory leakage model [22] if there is no PPT adversary  $\mathcal{A}$  whose advantage is non-negligible in the following game.

**Game** $_{\text{real}}(\ell_{\text{msk}}, \ell_{\text{sk}})$ :

1. **Setup Phase:** Challenger  $\mathcal{C}$  runs **Setup**( $\lambda$ )  $\rightarrow$  (PK, MSK), gives PK to  $\mathcal{A}$ , and sets  $\mathcal{Q} = \emptyset$  and  $\mathcal{T} = \{(0, \varepsilon, \text{MSK}, 0)\}$ , where  $\mathcal{Q}$  denotes a subset of predicate attribute  $\mathcal{X}$ , and  $\mathcal{T}$  denotes a set of tuples  $(\mathcal{H}, \mathcal{X}, (\text{MSK or } SK), N)$ , where  $\mathcal{H}$  is the handles set,  $N$  denotes the number of leakage bit of key in this tuple.
2. **Phase 1:** In this phase,  $\mathcal{A}$  can make the following queries, namely, create query, leakage query and reveal query.
  - **Create query:**  $\mathcal{A}$  output a create query **Create**( $h, X$ ), where  $h$  is the handle of a tuple in  $\mathcal{T}$  whose key must be a master key,  $X$  can either be a predicate attribute or be the empty string  $\varepsilon$ .  $\mathcal{C}$  initially scans  $\mathcal{T}$  to check whether the tuple with handle  $h$  is of the form  $(h, \varepsilon, \text{MSK}', L)$ . If this tuple does not exist or is not in this form,  $\mathcal{C}$  returns  $\perp$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{C}$  runs **KeyGen**( $X, \text{MSK}', \text{PK}$ )  $\rightarrow K$  and adds tuple  $(H + 1, X, K, 0)$  to  $\mathcal{T}$ .  $K$  can be either a master key or a user key.  $\mathcal{C}$  also needs to update the current handel  $H \leftarrow H + 1$ .
  - **Leakage query:**  $\mathcal{A}$  makes a leakage query **Leak**( $h, f$ ) about a key with a handle  $h$  with a polynomial time computable function  $f$  with constant output.  $\mathcal{C}$  first scans  $\mathcal{T}$  to find this specified tuple with handle  $h$ , which is either with the form  $(h, X, SK_X, L)$  or  $(h, \varepsilon, \text{MSK}', L)$ .
    - If  $\mathcal{A}$  makes a **user key leakage query** for tuple  $(h, X, SK_X, L)$ .  $\mathcal{C}$  first checks whether  $f(SK_X) + L \leq \ell_{\text{sk}}$ . If yes, return  $f(SK_X)$  to  $\mathcal{A}$  and update  $L$  in this tuple with  $L + f(SK_X)$ . Else, return  $\perp$  to  $\mathcal{A}$ .

- If  $\mathcal{A}$  makes a **master key leakage query** for tuple  $(h, \varepsilon, \text{MSK}', L)$ ,  $\mathcal{C}$  first checks whether  $f(\text{MSK}') + L \leq \ell_{msk}$ . If yes, return  $f(\text{MSK}')$  to  $\mathcal{A}$  and update  $L$  in this tuple with  $L + f(\text{MSK}')$ . Else, return  $\perp$  to  $\mathcal{A}$ .
  - **Reveal Query**  $\mathcal{A}$  makes a reveal query **Reveal** $(h)$ .  $\mathcal{C}$  scans  $\mathcal{T}$  to find the corresponding tuple. If this handle refers to a master key,  $\mathcal{C}$  returns  $\perp$ . Else,  $\mathcal{C}$  returns the user secret key in tuple  $(h, X, SK, L)$  to adversary and add  $X$  to the set  $\mathcal{Q}$ .
3. **Challenge Phase:**  $\mathcal{A}$  outputs two challenge messages  $M_0$  and  $M_1$  for a challenge attribute  $Y^*$  with the restriction that for all  $X \in \mathcal{Q}$ ,  $R_N(X, Y^*) \neq 1$ . If no, return  $\perp$ , else  $\mathcal{C}$  chooses  $b \leftarrow^* \{0, 1\}$ , compute **Encrypt** $(Y^*, M_b, \text{PK}) \rightarrow CT$  and return  $CT$  to  $\mathcal{A}$ .
  4. **Phase 2:**  $\mathcal{A}$  can only make create query and reveal query for attribute  $X'$  with the restriction that  $R_N(X', Y^*) \neq 1$ .  $\mathcal{C}$  return the corresponding result to  $\mathcal{A}$ .
  5. **Guess Phase:**  $\mathcal{A}$  output  $b' \in \{0, 1\}$ . If  $b' = b$ , then  $\mathcal{A}$  succeeds.

The advantage of  $\mathcal{A}$  in **Game<sub>real</sub>** is defined as  $\text{Adv}_{\mathcal{A}}(\lambda) = |\text{Pr}[b' = b] - \frac{1}{2}|$ .

*Remarks:* Note that this security model is equivalent to the widely accepted continual leakage model if keys can be updated properly and no leakage is allowed during the update process. We refer the reader to [22] for more details. Here, it is clear that scheme in Sect. 4.1 possesses the ability to properly update keys.

### 4.3 Security Proof of Our Generic Construction

Here we describe two ways to prove the security of our generic construction. If the underlying leakage-resilient pair encoding scheme can achieve computational security (SMH and CMH), we give a tighter reduction with cost of  $\mathcal{O}(q_1)$ . On the other hand, if the underlying leakage-resilient pair encoding scheme is perfectly master-key hiding<sup>1</sup>, we achieve a reduction with cost of  $\mathcal{O}(q_1 + q_2)$ . Here  $q_1$  and  $q_2$  are the numbers of queries made in Phase 1 and Phase 2 respectively. Regarding the security of the construction, we have the following theorem whose proof can be found in the full version of this paper due to page limitation.

**Theorem 1.** *Assume that the  $(\ell_{msk}, \ell_{sk})$ -leakage-resilient pair encoding scheme  $P$  is selectively and co-selectively master-key hiding in  $\mathcal{G}$  and subgroup assumption SD1, SD2, SD3 hold in  $\mathcal{G}$ . Then FE constructed above from  $P$  is  $(\ell_{msk}, \ell_{sk})$ -leakage-resilient and adaptively secure in the continual memory leakage model. For any PPT adversary  $\mathcal{A}$ , there exists adversary  $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_6$  who run nearly the same time as  $\mathcal{A}$ , such that for any  $\lambda$ :*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}(\lambda) \leq & 2\text{Adv}_{\mathcal{B}_1}^{SD1}(\lambda) + (2q_1 + 3)\text{Adv}_{\mathcal{B}_2}^{SD2}(\lambda) + q_1\text{Adv}_{\mathcal{B}_3}^{CMH}(\lambda) \\ & + q_1\text{Adv}_{\mathcal{B}_4}^{LR}(\lambda) + \text{Adv}_{\mathcal{B}_5}^{SMH}(\lambda) + \text{Adv}_{\mathcal{B}_6}^{SD3}(\lambda) \quad (4) \end{aligned}$$

---

<sup>1</sup> Recall that perfectly master-key hiding only implies computational co-selective security but not computational selective security.

**Theorem 2.** *Assuming the  $(\ell_{msk}, \ell_{sk})$ -leakage-resilient pair encoding scheme  $P$  is perfectly master-key hiding and subgroup assumption  $SD1, SD2, SD3$  hold in  $\mathcal{G}$ . Then  $FE$  constructed above is  $(\ell_{msk}, \ell_{sk})$ -leakage-resilient and adaptively secure in the continual memory leakage model. For any PPT adversary  $\mathcal{A}$ , there exists adversary  $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$  who run nearly the same time as  $\mathcal{A}$ , such that for any  $\lambda$ :*

$$Adv_{\mathcal{A}}(\lambda) \leq 2Adv_{\mathcal{B}_1}^{SD1}(\lambda) + (2(q_1 + q_2) + 1)Adv_{\mathcal{B}_2}^{SD2}(\lambda) + q_1 Adv_{\mathcal{B}_3}^{LR}(\lambda) + Adv_{\mathcal{B}_4}^{SD3}(\lambda) \quad (5)$$

## 5 From Pair Encoding to Leakage-Resilient Encoding

In this section, we show how to construct leakage-resilient encodings from pair encodings that satisfy some additional constraints. We would like to remark that these conditions are not overly restrictive. In fact, a large number of pair encoding schemes in [1] satisfies the requirements.

### 5.1 Extending the Definition of Pair Encoding in [1] to Support Encoding of the Empty Attribute

We extend the definition of pair encoding schemes in [1] to support encoding for the empty attribute  $\varepsilon$ . Specifically, a pair encoding scheme  $\mathbf{P} = (\mathbf{Param}, \mathbf{Enc1}, \mathbf{Enc2}, \mathbf{Pair})$  for predicate family  $\mathcal{R}$  is defined as follows:

- $\mathbf{Param}(\lambda) \rightarrow n_1$ , pick  $\mathbf{h} = (h_1, h_2, \dots, h_{n_1}) \xleftarrow{\$} Z_N^{n_1}$ .
- $\mathbf{Enc1}(X, N)$ :
  - If  $X$  is empty,  $\mathbf{Enc1}(\varepsilon, N) \rightarrow (\mathbf{k}_1^\varepsilon, \mathbf{k}_2^\varepsilon; 1, 1, 0)$  and  $\mathbf{k}_1^\varepsilon(\alpha, \mathbf{h}, \mathbf{r}) = \alpha, \mathbf{k}_2^\varepsilon(\mathbf{h}, \mathbf{r}) = 0$ .
  - If  $X \in \mathcal{X}$ , then  $\mathbf{Enc1}(X, N) \rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , where  $X \in \mathcal{X}$ . Let  $\mathbf{r} = (r_1, r_2, \dots, r_{m_3}) \xleftarrow{\$} Z_N^{m_3}$ .  $\mathbf{k}_1 = \{k_{1,\ell}\}_{\ell \in [m_1]}$  is a set of polynomials, each one is in  $\{\alpha\} \cup \{h_i\}_{i \in [n_1]} \cup \{r_j\}_{j \in [m_3]}$ .  $\mathbf{k}_2 = \{k_{2,\ell'}\}_{\ell' \in [m_2]}$  is a set of polynomials and each polynomial is in  $\{h_i\}_{i \in [n_1]} \cup \{r_j\}_{j \in [m_3]}$ . That is,

$$k_{1,\ell} = b_\ell \alpha + \sum_{i \in [n_1], j \in [m_3]} b_{\ell,i,j} h_i r_j. \quad (6)$$

$$k_{2,\ell'} = \sum_{j \in [m_3]} b_{\ell',j} r_j. \quad (7)$$

*Remark 1:* The output of  $\mathbf{Enc1}(X)$  is divided into two sets  $\mathbf{k}_1$  and  $\mathbf{k}_2$ , and the concatenation of these two sets  $(\mathbf{k}_1, \mathbf{k}_2)$  is exactly the same as the output of  $\mathbf{Enc1}$  as defined in [1]. Moreover, it is obvious that given  $(\mathbf{k}_1^\varepsilon, \mathbf{k}_2^\varepsilon)$  and any attribute  $X \in \mathcal{X}$ , we can generate an encoding  $(\mathbf{k}_1, \mathbf{k}_2)$  for  $X$  which is properly distributed.





**Table 1.** New schemes/constructions from our framework.

Predicates	Adaptively secure?	Leakage-resilient?	Constructions	Underlying pair encoding in [1]
IBE	✓	×	[1, 23]	
	✓	✓ (CML)	[22], this paper (tighter reduction)	Scheme 1
FEs for regular language	×	×	[32]	
	✓	×	[1]	
KP-ABE for large universe	✓	✓ (CML)	this paper (new construction)	Scheme 3
	×	×	[28]	
KP-ABE with short ciphertext	✓	×	[1]	
	✓	✓ (CML)	this paper (new construction)	Scheme 4
KP-ABE with short ciphertext	×	×	[4]	
	✓	×	[1]	
	✓	✓ (CML)	this paper (new construction)	Scheme 5
KP-DSE	✓	✓ (CML)	this paper (new construction)	
	×	×	[17]	
	✓	×	[1]	
KP-DSE	✓	✓ (CML)	this paper (new construction)	Scheme 6
	✓	✓ (CML)	this paper (new construction)	

Regarding the security of this transformation, we have the following theorem whose proof can be found in the full version of this paper due to page limitation.

**Theorem 3.**  $P'$  is a  $(\ell_{msk} = (n - 1 - 2c)\log(p_2), \ell_{sk} = (n - 1 - 2c)\log(p_2))$ -leakage-resilient pair encoding scheme, for some positive constant  $c$  such that  $p_2^{-c}$  is negligible, if  $P$  satisfies the following conditions:

1. The first row of the reconstruction matrix  $\mathbf{E}$  of  $P$  is  $(1, 0, 0, \dots, 0)$ .
2. For any attribute  $X$  and  $\mathbf{Enc1}(X, N) \rightarrow (\mathbf{k}_1, \mathbf{k}_2; m_1, m_2, m_3)$ , the two distributions  $(\mathbf{k}_1(0, \mathbf{h}, \mathbf{r}), \mathbf{k}_2(\mathbf{h}, \mathbf{r}))$  and  $(\mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{r}), \mathbf{k}_2(\mathbf{h}, \mathbf{r}))$  are identical except that

$$\mathbf{k}_{11}^\alpha = \mathbf{k}_{11}^0 + \alpha,$$

where  $\mathbf{k}_{11}^\alpha$  is the first polynomial of  $\mathbf{k}_1(\alpha, \mathbf{h}, \mathbf{r})$ ,  $\mathbf{k}_{11}^0$  is the first polynomial of  $\mathbf{k}_1(0, \mathbf{h}, \mathbf{r})$ , and  $\alpha \xleftarrow{\$} Z_N, \mathbf{h} \xleftarrow{\$} Z_N^{n_1}, \mathbf{r} \xleftarrow{\$} Z_N^{m_3}$ .

## 6 Instantiations of Our Framework

Based on existing pair encodings, we achieve a number of new schemes and schemes with better properties. This is summarised in Table 1.

**Acknowledgement.** We appreciate the anonymous reviewers for their valuable suggestions. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61572294, 61300213).

## References

1. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014)
2. Attrapadung, N.: Dual system encryption framework in prime-order groups. IACR Cryptology ePrint Archive 2015:390 (2015)
3. Attrapadung, N., Libert, B.: Functional encryption for inner product: achieving constant-size ciphertexts with adaptive security or support for negation. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 384–402. Springer, Heidelberg (2010)
4. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011)
5. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
6. 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)
7. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
10. Boyle, E., Goldwasser, S., Jain, A., Kalai, Y.T.: Multiparty computation secure against continual memory leakage. In: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, pp. 1235–1254. ACM (2012)
11. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage. In: 2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 501–510. IEEE (2010)
12. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015)

13. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: 2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 511–520. IEEE (2010)
14. Dodis, Y., Lewko, A., Waters, B., Wichs, D.: Storing secrets on continually leaky devices. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS), pp. 688–697. IEEE (2011)
15. 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, pp. 89–98. ACM (2006)
16. Halderman, J.A., Schoen, S.D., Hening, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM* **52**(5), 91–98 (2009)
17. Hamburg, M.: Spatial Encryption. Ph.D. thesis, Stanford University (2011)
18. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
19. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
20. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
21. 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)
22. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Heidelberg (2011)
23. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
24. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
25. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures resilient to continual leakage on memory and computation. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 89–106. Springer, Heidelberg (2011)
26. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
27. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
28. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 463–474. ACM (2013)
29. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

31. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
32. 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)
33. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014)
34. Yang, R., Xu, Q., Zhou, Y., Zhang, R., Hu, C., Yu, Z.: Updatable hash proof system and its applications. In: Pernul, G., et al. (eds.) ESORICS 2015, Part I. LNCS, vol. 9326, pp. 266–285. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-24174-6\\_14](https://doi.org/10.1007/978-3-319-24174-6_14)
35. Yuen, T.H., Chow, S.S.M., Zhang, Y., Yiu, S.M.: Identity-based encryption resilient to continual auxiliary leakage. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 117–134. Springer, Heidelberg (2012)

# Secret Handshakes with Dynamic Expressive Matching Policy

Lin Hou<sup>1</sup>, Junzuo Lai<sup>1,2(✉)</sup>, and Lixian Liu<sup>1</sup>

<sup>1</sup> Department of Computer Science, Jinan University,  
Guangzhou, China

linhou19@gmail.com, 944427149@qq.com

<sup>2</sup> The State Key Laboratory of Integrated Services Networks,  
Xidian University, Xi'an, China

laijunzuo@gmail.com

**Abstract.** Secret handshake is an important building block of private communication over public networks, which allows two members of the same group to secretly authenticate each other and agree on a shared key for further communication. Ateniese et al. [1] introduced attribute-based secret handshake, in which a group member Alice can complete the handshake protocol with another group member Bob by specifying the attributes Bob must have. In this paper, we propose the first efficient attribute-based secret handshake scheme which supports arbitrary matching policies with unlinkable and reusable credentials. Specifically, we first present a generic construction of attribute-based secret handshakes from *centralized* ciphertext-policy attribute-based encryption (CP-ABE). Based on the construction, we present a highly efficient attribute-based secret handshake scheme employing the CP-ABE scheme in [18].

## 1 Introduction

Given the pervasiveness and public nature of today's Internet, communication privacy is becoming a grave concern. Many techniques have been proposed in literatures for achieving communication privacy over public networks. Among them, privacy-preserving authentication protocols such as *secret handshake* schemes are important building blocks.

Secret handshakes, first introduced by Balfanz et al. [2], allow two members of the same group to secretly and privately authenticate each other and agree on a shared key for subsequent communication. Such a handshake is privacy-preserving in the sense that someone who is not in the group cannot perform the handshake. On the other hand, any two parties who are members of the same group could authenticate to each other. A common cited application of such interactions is mutual authentication of two CIA agents, in which they should be able to successfully complete the handshake while others should not be able to recognize the handshake.

Ateniese et al. [1] extended the framework of secret handshakes to include roles and support dynamic matching of attributes associated with the role in

a threshold way, which is also called attribute-based secret handshakes. This dynamic matching allows users to specify the group of person with whom they would like to perform a secret handshake, rather than a static group setting. Moreover, each member has a number of attributes (say  $n$ ) associated with her membership. For instance, a depressed patient Alice wants to authenticate herself and reveal her illness only to others authenticated as psychologist. When setting up a handshake with some psychologist, Alice can specify what attributes the psychologist must have, such as (`psychologist|female|...|inLosAngeles`). The handshake succeeds iff the credentials of the psychologist match  $d$  or more of the attributes specified by Alice for some threshold  $d \leq n$ .

Traditional secret handshake has many appealing applications, such as digital content protection and anonymous routing in ad-hoc networks. However, attribute-based secret handshake is used more broadly. Such as in online dating system, employing attribute-based secret handshake allows any two users to check whether each of them meets the expectations of the other without revealing any additional personal information beforehand.

Unfortunately, previous attribute-based secret handshakes in literatures only support simple matching policy, i.e. threshold favor, and there is no scheme that supports expressive matching policies before this paper. The difficulty in constructing such a scheme comes from its security requirements. A secure secret handshake must provide three basic properties. The first one is *impersonator resistance*, which means any adversary not satisfying the matching policy is unable to authenticate himself to an honest member. And *detector resistance* requires the adversary above cannot decide whether some honest party satisfies the rules or not. The last one, *unlinkability*, demands that it should be infeasible to tell whether two (successful) execution of handshake protocol were performed by the same party or not. Most attribute-based schemes may not possess detector resistance, thus making it a challenge to construct secret handshakes with a dynamic expressive matching policy.

## 1.1 Our Contributions

In this paper, we investigate the construction of attribute-based secret handshakes from ciphertext-policy attribute-based encryption (CP-ABE) schemes, and propose an efficient secret handshake protocol which supports attribute matching with more *flexible* or *expressive* access structures than the existing ones in literatures. Specifically, our contributions are as follows:

1. We first introduce a generic construction of attribute-based secret handshakes employing *centralized* CP-ABE with *partially hidden access structures*. Centralized CP-ABE is slightly modified from traditional CP-ABE with an extra **Init** algorithm, which runs by the System Administrator (SA). In centralized CP-ABE, SA publishes the global public parameter to all Private Key Generators (PKG) before the setup procedure. The formal definition is described in Sect. 2.3. In *partially hidden access structure* model [18], each attribute includes two parts, i.e. attribute name and attribute value. If the set of

attributes associated with a user's private key does not satisfy the access structure associated with a ciphertext, attribute values in the access structure are hidden while the attribute names are still public. Since in many applications, specific attribute values carry much more sensitive information than the generic attribute names, this model is sufficient and plausible in practice.

2. Based on the generic construction of attribute-based secret handshakes from centralized CP-ABE, we present a concrete instantiation employing the partial hidden policy CP-ABE scheme proposed in [18]. Specifically, We first modify the scheme in [18] to get a centralized CP-ABE scheme, and then use it to construct an efficient attribute-based secret handshake. The result handshake scheme not only supports dynamic expressive matching policy but also provides all the security properties in standard model.

## 1.2 Related Work

We only focus on closely related works, and refer the reader to [2, 19, 31] for discussions on some loosely related ones.

**Secret Handshake.** The concept of secret handshakes was first introduced by Balfanz et al. [2]. Several proposals on secret handshake schemes followed, based on bilinear maps [2], computational Diffie-Hellman [10, 32], and RSA [28]. However, users in these schemes are *linkable*. Namely, an attacker can recognize two instances of a protocol executed by the same party. In order to achieve unlinkability, the scheme in [2] resorts to use one-time credentials.

Xu and Yung [31] presented secret handshake schemes that achieve unlinkability with reusable credentials instead of one-time credentials, but only offer a weak notion of privacy called  $k$ -anonymity. Unlinkable secret handshake schemes with strong notion of privacy were proposed later in [1, 15].

Tsudik and Xu [27] introduced the first scheme for group secret handshakes, which achieves unlinkability with reusable credentials; however, their scheme ensures successful authentication among group members only if every member holds the same most recently distributed group key, a condition which results in high real-time communication overhead between the group manager and group members. Jarecki et al. [13] presented another scheme for group secret handshakes which fits into the standard PKI setting and avoids having the group manager broadcasting key-update messages to group members; however, as in [2], the scheme uses one-time credentials to achieve unlinkability.

Jarecki et al. [14] considered a very strong notion of secret handshakes, referred to as affiliation-hiding authenticated key exchange, which guarantees security under arbitrary composition of protocol sessions.

Ateniese et al. [1] presented a secret handshake scheme with dynamic matching, in which each party can specify both the group and the role the other must have in order for the handshake to succeed. They also gave a novel extension of secret handshakes to include attributes, allowing the handshake to support

threshold-based matching on attributes, as we discussed at the beginning of this section.

As an independent research interest, revocation of credentials in secret handshakes was investigated in [15,26].

A related topic is oblivious signature-based envelope (OSBE), introduced by Li et al. [20]. Informally, an OSBE enables a sender to send an envelope (encrypted message) to a recipient, with the assurance that the latter will be able to open it only if he holds the signature on a prior agreed-upon message. Nasserian and Tsudik [21] observed that two symmetric instances of OSBE may yield a secret handshake.

**Attribute-Based Encryption.** The notion of ABE was first introduced by Sahai and Waters as an application of their fuzzy identity-based encryption (IBE) scheme [25], where both ciphertexts and secret keys are associated with sets of attributes. Decryption is enabled if and only if the ciphertext and secret key attribute sets overlap by at least a fixed threshold value  $d$ . There are two kinds of ABE schemes, key-policy and ciphertext-policy ABE schemes.

In a key-policy ABE scheme [12,24], every ciphertext is associated with a set of attributes, and every user's secret key is associated with an access structure on attributes. Decryption is enabled if and only if the ciphertext attribute set satisfies the access structure associated with the user's secret key. The notion of predicate encryption (PE) [16] is related to key-policy ABE. In a PE scheme, secret keys correspond to predicates and ciphertexts are associated with a set of attributes; the secret key  $SK_f$  corresponding to a predicate  $f$  can be used to decrypt a ciphertext associated with an attribute set  $I$  if and only if  $f(I) = 1$ . Katz, Sahai, and Waters [16] also introduced the idea of *attribute-hiding*, a security notion for PE that is stronger than the basic security requirement of *payload-hiding*. Roughly speaking, *attribute-hiding* requires that a ciphertext conceal the associated attributes as well as the plaintext, while *payload-hiding* only requires that a ciphertext conceal the plaintext.

In a ciphertext-policy ABE (CP-ABE) scheme [4,11,29], the situation is reversed. That is, attributes are associated with user's secret keys and access structures (also called ciphertext policies) with ciphertexts. Nishide et al. [22] proposed CP-ABE schemes where encryptor-specified access structures are hidden. Access structures in their schemes support AND operation, and the security of the schemes were only proved in a weak model, which can be considered to be analogous to the selective-ID model [5,9] used in IBE schemes. Lai et al. [18] proposed a partial hidden CP-ABE scheme which supports a wide range of access structures in standard model.

### 1.3 Organization

The rest of the paper is organized as follows. In Sect. 2, we review some standard notations and cryptographic definitions. We also formally define the notion of centralized CP-ABE and fully security notion. We then present the generic



construction of secure attribute-based secret handshake from fully secure centralized CP-ABE in Sect. 3. This generic construction ensures that the handshake scheme supports the same access structures on attributes as those supported by the underlying fully secure CP-ABE scheme. In Sect. 4, we present a concrete secret handshake with dynamic expressive matching policy. Finally, we state our conclusion in Sect. 5.

## 2 Preliminaries

If  $S$  is a set, then  $s \stackrel{\$}{\leftarrow} S$  denotes the operation of picking an element  $s$  uniformly at random from  $S$ . Let  $\mathbb{N}$  denote the set of natural numbers. If  $\lambda \in \mathbb{N}$  then  $1^\lambda$  denotes the string of  $\lambda$  ones. Let  $z \leftarrow A(x, y, \dots)$  denote the operation of running an algorithm  $A$  with inputs  $(x, y, \dots)$  and output  $z$ . A function  $f(\lambda)$  is *negligible* if for every  $c > 0$  there exists a  $\lambda_c$  such that  $f(\lambda) < 1/\lambda^c$  for all  $\lambda > \lambda_c$ .

### 2.1 Composite Order Bilinear Groups

Composite order bilinear groups were first introduced in [7]. We use bilinear groups whose order is the product of three distinct primes.

Let  $\mathcal{G}$  be an algorithm that takes as input a security parameter  $1^\lambda$  and outputs a tuple  $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ , where  $p, q, r$  are distinct primes,  $\mathbb{G}$  and  $\mathbb{G}_T$  are cyclic groups of order  $N = pqr$ , and  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  is a map such that

1. (Bilinear)  $\forall g, h \in \mathbb{G}, a, b \in \mathbb{Z}_N, \hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$ ;
2. (Non-degenerate)  $\exists g \in \mathbb{G}$  such that  $\hat{e}(g, g)$  has order  $N$  in  $\mathbb{G}_T$ .

We further require that multiplication in  $\mathbb{G}$  and  $\mathbb{G}_T$ , as well as the bilinear map  $\hat{e}$ , are computable in time polynomial in  $\lambda$ . We use  $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$  to denote the subgroups of  $\mathbb{G}$  having order  $p, q$ , and  $r$ , respectively. Observe that  $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$ . Note also that if  $h_p \in \mathbb{G}_p$  and  $h_q \in \mathbb{G}_q$  then  $\hat{e}(h_p, h_q) = 1$ . A similar rule holds whenever  $\hat{e}$  is applied to elements in distinct subgroups.

We now state the complexity assumptions we use. The first assumption is just the subgroup decision problem in the case where the group order is a product of three primes. We justify these assumptions in Appendix A by proving that they hold in the generic group model assuming finding a non-trivial factor of the group order  $N$  is hard. Note that our assumptions are non-interactive (in contrast to, e.g., the LRSW assumption [8]) and of fixed size (in contrast to, e.g., the  $q$ -SDH assumption [6]).

**Assumption 1.** *Let  $\mathcal{G}$  be as above. We define the following distribution:*

$$\begin{aligned} (p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e}) &\leftarrow \mathcal{G}(1^\lambda), \quad N = pqr, \quad g_p \stackrel{\$}{\leftarrow} \mathbb{G}_p, \quad g_r \stackrel{\$}{\leftarrow} \mathbb{G}_r, \\ D &= (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, g_p, g_r), \\ T_1 &\stackrel{\$}{\leftarrow} \mathbb{G}_p \times \mathbb{G}_q, \quad T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_p. \end{aligned}$$

The advantage of an algorithm  $\mathcal{A}$  in breaking Assumption 1 is defined as

$$Adv_{\mathcal{A}}^1 = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

**Definition 1.** we say  $\mathcal{G}$  satisfies Assumption 1 if for any polynomial time algorithm  $\mathcal{A}$ ,  $Adv_{\mathcal{A}}^1$  is negligible.

**Assumption 2.** Let  $\mathcal{G}$  be as above. We define the following distribution:

$$\begin{aligned} (p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e}) &\leftarrow \mathcal{G}(1^\lambda), N = pqr, \\ g_p, X_1 &\stackrel{\$}{\leftarrow} \mathbb{G}_p, X_2 \stackrel{\$}{\leftarrow} \mathbb{G}_q, g_r \stackrel{\$}{\leftarrow} \mathbb{G}_r, \\ D &= (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, g_p, X_1 X_2, g_r), \\ T_1 &\stackrel{\$}{\leftarrow} \mathbb{G}_p \times \mathbb{G}_q, T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_p. \end{aligned}$$

The advantage of an algorithm  $\mathcal{A}$  in breaking Assumption 2 is defined as

$$Adv_{\mathcal{A}}^2 = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

**Definition 2.** we say  $\mathcal{G}$  satisfies Assumption 2 if for any polynomial time algorithm  $\mathcal{A}$ ,  $Adv_{\mathcal{A}}^2$  is negligible.

**Assumption 3.** Let  $\mathcal{G}$  be as above. We define the following distribution:

$$\begin{aligned} (p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e}) &\leftarrow \mathcal{G}(1^\lambda), N = pqr, \\ \omega, s \in \mathbb{Z}_N, g_p, Z_1 &\stackrel{\$}{\leftarrow} \mathbb{G}_p, X_2, Y_2, Z_2 \stackrel{\$}{\leftarrow} \mathbb{G}_q, g_r \stackrel{\$}{\leftarrow} \mathbb{G}_r, \\ D &= (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, g_p, g_p^\omega X_2, g_p^s Y_2, Z_1 Z_2, g_r), \\ T_1 &= \hat{e}(g_p, g_p)^{\omega s}, T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_T. \end{aligned}$$

The advantage of an algorithm  $\mathcal{A}$  in breaking Assumption 3 is defined as

$$Adv_{\mathcal{A}}^3 = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

**Definition 3.** we say  $\mathcal{G}$  satisfies Assumption 3 if for any polynomial time algorithm  $\mathcal{A}$ ,  $Adv_{\mathcal{A}}^3$  is negligible.

**Assumption 4.** Let  $\mathcal{G}$  be as above. We define the following distribution:

$$\begin{aligned} (p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e}) &\leftarrow \mathcal{G}(1^\lambda), N = pqr, \\ a \in \mathbb{Z}_N, g_p &\stackrel{\$}{\leftarrow} \mathbb{G}_p, g_q, Q_1, Q_2, Q \stackrel{\$}{\leftarrow} \mathbb{G}_q, g_r, R_0, R_1, R \stackrel{\$}{\leftarrow} \mathbb{G}_r, \\ D &= (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, g_p R_0, g_p^a R_1, g_p Q_1, g_p^{1/a} Q_2, g_q, g_r), \\ T_1 &= g_p^a Q R, T_2 \stackrel{\$}{\leftarrow} \mathbb{G}_T. \end{aligned}$$

The advantage of an algorithm  $\mathcal{A}$  in breaking Assumption 4 is defined as

$$Adv_{\mathcal{A}}^4 = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

**Definition 4.** we say  $\mathcal{G}$  satisfies Assumption 4 if for any polynomial time algorithm  $\mathcal{A}$ ,  $Adv_{\mathcal{A}}^4$  is negligible.

## 2.2 Attribute-Based Secret Handshake Scheme

An attribute-based secret handshake scheme (denoted by ABSH) consists of the following algorithms:

- Setup.** Given a security parameter  $1^\lambda$ , the algorithm generates the public parameters **params** common to all subsequently generated groups.
- CreateGroup.** The group administrator  $GA$  in group  $G$  runs the algorithm on input of **params**, and outputs the group public information  $GPK_G$  and group secret key  $GSK_G$ .
- AddUser.** This is a protocol between  $GA$  and a user. On input of a set of attributes  $S_U$  of user  $U$ ,  $GA$  outputs the user's group credential  $cred_U$  using  $GA$ 's key pair  $(GPK_G, GSK_G)$ .
- HandShake.** This is an authentication protocol executed between users  $A$  and  $B$ , who may belong to different groups. At the end of the protocol, if  $A$ 's target requirements are matched by  $B$  and vice versa,  $A$  and  $B$  will authenticate each other by sharing a common secret key for subsequent secure communication.

We consider the following core security properties for secret handshake schemes:

- Impersonator resistance:** An adversary not satisfying the requirements of the handshake protocol can not authenticate to an honest user.
- Detector resistance:** An adversary not satisfying the requirements of the handshake protocol can not decide whether an honest user satisfies the requirements or not.
- Unlinkability:** It is not feasible to tell whether two executions of the handshake protocol were performed by the same users or not.

## 2.3 Centralized CP-ABE with Partially Hidden Access Structures

Centralized CP-ABE is slightly modified from traditional CP-ABE which consists of the following five algorithms:

- Init**( $1^\lambda$ ): It takes as input a security parameter  $\lambda$ , and output a global public parameter **PP**.
- Setup**(**PP**): It takes as input the global public parameter **PP**, and outputs a public key **MPK** and a master secret key **MSK**.
- KeyGen**(**MPK**, **MSK**,  $S$ ): It takes as input the public key **MPK**, the master secret key **MSK** and a set of attributes  $S$ . It outputs a secret key  $SK_S$ .
- Encrypt**(**MPK**,  $m$ ,  $\mathbb{A}$ ): It takes as input the public key **MPK**, a message  $m$  and an access structure  $\mathbb{A}$ . It outputs a ciphertext  $c$ .
- Decrypt**(**MPK**,  $SK_S$ ,  $c$ ): It takes as input the public key **MPK**, a secret key  $SK_S$  and a ciphertext  $c$ . It outputs a message  $m$ .

Let  $(\mathbf{MPK}, \mathbf{MSK}) \leftarrow \text{Setup}(1^\lambda)$ ,  $SK_S \leftarrow \text{KeyGen}(\mathbf{MPK}, \mathbf{MSK}, S)$ , and  $c$  is the output of the algorithm  $\text{Encrypt}(\mathbf{MPK}, m, \mathbb{A})$ . For correctness, we require the following to hold:

1. If the set  $S$  of attributes in a private key satisfies the access structure  $\mathbb{A}$  in a ciphertext, then  $m \leftarrow \text{Decrypt}(\text{MPK}, \text{SK}_S, c)$ ;
2. Otherwise, with overwhelming probability,  $\text{Decrypt}(\text{MPK}, \text{SK}_S, c)$  outputs a random message.

**Partially Hidden Access Structures.** In the construction of CP-ABE with partially hidden access structures [18], each attribute includes two parts, i.e. *attribute name* and *attribute value*. It is assumed that there are  $n$  categories of attributes and every user has  $n$  attributes with each attribute belonging to a different category. Let  $i$  denote the attribute name of the  $i^{\text{th}}$  category attribute. A user's attribute set  $\mathcal{S}$  is parsed as  $(s_1, \dots, s_n)$ , where  $s_i \in \mathbb{Z}_N$  is the value of attribute  $i$ . We express an access formula by  $(\mathcal{A}, \rho, \mathcal{T})$ , where  $\mathcal{A}$  is  $\ell \times n$  share-generating matrix,  $\rho$  is a map from each row of  $\mathcal{A}$  to an attribute name, i.e.  $\rho : \{1, \dots, \ell\} \rightarrow \{1, \dots, n\}$ , and  $\mathcal{T}$  can be parsed as  $(t_{\rho(1)}, \dots, t_{\rho(\ell)})$  and  $t_{\rho(i)}$  is the value of attribute  $\rho(i)$  specified by the access formula. A user's attribute set  $\mathcal{S}=(s_1, \dots, s_n)$  satisfies an access formula  $(\mathcal{A}, \rho, \mathcal{T})$  if and only if there exist  $\mathcal{I} \subseteq \{1, \dots, \ell\}$  and constants  $\{\omega_i\}_{i \in \mathcal{I}}$  such that

$$\sum_{i \in \mathcal{I}} \omega_i A_i = (1, 0, \dots, 0) \text{ and } s_{\rho(i)} = t_{\rho(i)} \text{ for } \forall i \in \mathcal{I},$$

where  $A_i$  denotes the  $i^{\text{th}}$  row of  $A$ .

**Security Model.** We now give the security model for centralized CP-ABE with partially hidden access structures, described as a game between a challenger and an adversary  $\mathcal{A}$ . The game proceeds as follows:

**Setup.** The challenger runs  $\text{Init}(1^\lambda)$  and  $\text{Setup}(\text{PP})$  to obtain the public parameters MPK and a master secret key MSK. It gives the public parameters MPK to the adversary  $\mathcal{A}$  and keeps MSK to itself.

**Query phase 1.** The adversary  $\mathcal{A}$  adaptively queries the challenger for secret keys corresponding to sets of attributes  $\mathcal{S}_1, \dots, \mathcal{S}_q$ . In response, the challenger runs  $\text{SK}_{\mathcal{S}_i} \leftarrow \text{KeyGen}(\text{MPK}, \text{MSK}, \mathcal{S}_i)$  and gives the secret key  $\text{SK}_{\mathcal{S}_i}$  to  $\mathcal{A}$ , for  $1 \leq i \leq q$ .

**Challenge.** The adversary  $\mathcal{A}$  submits two (equal length) messages  $M_0, M_1$  and two access structures  $(\mathbf{A}, \rho, \mathcal{T}_0), (\mathbf{A}, \rho, \mathcal{T}_1)$ , subject to the restriction that,  $(\mathbf{A}, \rho, \mathcal{T}_0)$  and  $(\mathbf{A}, \rho, \mathcal{T}_1)$  cannot be satisfied by any of the queried attribute sets. The challenger selects a random bit  $\beta \in \{0, 1\}$  and encryptes  $M_\beta$  to get the challenge ciphertext  $C = \text{Encrypt}(\text{MPK}, M_\beta, (\mathbf{A}, \rho, \mathcal{T}_\beta))$  and sends  $C$  to the adversary as its challenge ciphertext.

*Note that, the LSSS matrix  $\mathbf{A}$  and  $\rho$  are the same in the two access structures provided by the adversary. In a CP-ABE scheme with partially hidden access structures, one can distinguish the ciphertexts if the associated access structures have different  $(\mathbf{A}, \rho)$ , since  $(\mathbf{A}, \rho)$  is sent along with the ciphertext explicitly.*

**Query phase 2.** The adversary continues to adaptively query the challenger for secret keys corresponding to sets of attributes with the added restriction that none of these satisfies  $(\mathbf{A}, \rho, \mathcal{T}_0)$  and  $(\mathbf{A}, \rho, \mathcal{T}_1)$ .

**Guess.** The adversary  $\mathcal{A}$  outputs its guess  $\beta' \in \{0, 1\}$  for  $\beta$  and wins the game if  $\beta = \beta'$ .

The advantage of the adversary in this game is defined as  $|\Pr[\beta = \beta'] - \frac{1}{2}|$  where the probability is taken over the random bits used by the challenger and the adversary.

**Definition 5.** *The centralized CP-ABE scheme with partially hidden access structures is CPA secure if all polynomial time adversaries have at most a negligible advantage in this security game.*

Note that another stronger notion is *fully security* [19], which means that the ciphertext reveals no information about the underlying plaintext and completely hides the associated policy. However, the only known construction of fully secure CP-ABE schemes comes from Inner-product Predicate Encryption (IPE) [16], which causes a superpolynomial blowup in size for arbitrary access structures and is extremely impractical.

### 3 Attribute-Based Secret Handshake from Centralized CP-ABE

In this section, based on the secure centralized CP-ABE with partially hidden access structures, we propose a generic construction of attribute-based secret handshakes. Compared with the scheme proposed by Ateniese et al. [1], which only supports threshold-based access structures, our construction is more expressive thus the resulting handshake schemes support the same access structures on attributes as those supported by the underlying CP-ABE.

Suppose that  $\Pi$  is a centralized CP-ABE scheme with partially hidden access structures which contains algorithms `Init`, `Setup`, `KeyGen`, `Encrypt` and `Decrypt`. We can construct a attribute-based secret handshake scheme by defining its corresponding algorithms in the following way.

**ABSH.Setup**( $1^\lambda$ ): Given a security parameter  $\lambda$ , the algorithm runs

$$\Pi.PP \leftarrow \Pi.Init(1^\lambda)$$

and sets the public parameter

$$\text{params} = \Pi.PP$$

**ABSH.CreateGroup**(params): Given the public parameter `params`, the group administrator `GA` first runs

$$(\Pi.MPK, \Pi.MSK) \leftarrow \Pi.Setup(1^\lambda)$$

and then sets the group  $G$ 's public information  $\text{GPK}_G$  and secret key  $\text{GSK}_G$  as

$$(\text{GPK}_G, \text{GSK}_G) = (\Pi.MPK, \Pi.MSK).$$

**ABSH.AddUser**( $\text{GSK}_G, S_U$ ): To add a user  $U$  with a set of attributes  $S_U$  to the group  $G$ , the group administrator GA runs

$$\Pi.\text{SK}_{S_U} \leftarrow \Pi.\text{KeyGen}(\text{GPK}_G, \text{GSK}_G, S_U),$$

and gives user  $U$  the secret credential  $\text{cred}_U = \Pi.\text{SK}_{S_U}$ .

**ABSH.HandShake**( $A, B$ ): Let  $A$  be a member of group  $G_A$  and  $B$  be a member of group  $G_B$  for generality. Suppose  $A$  with a secret credential  $\text{cred}_A$ , which is a private key on a set of attributes  $S_A$ , and  $B$  with a secret credential  $\text{cred}_B$ , which is a private key on a set of attributes  $S_B$ , engage in a handshake protocol. The protocol proceeds as follows:

1.  $A$  chooses a random  $k_1$  and sends a ciphertext  $c_B$  to  $B$ , where

$$c_B \leftarrow \Pi.\text{Encrypt}(\text{GPK}_{G'_B}, k_1, \mathbb{A}_B),$$

and  $G'_B$  is the group that  $B$  must be in and  $\mathbb{A}_B$  is the access structure on attributes that  $B$  must satisfy in order to complete the handshake.

2. Similarly,  $B$  chooses a random  $k_2$  and sends a ciphertext  $c_A$  to  $A$ , where

$$c_A \leftarrow \Pi.\text{Encrypt}(\text{GPK}_{G'_A}, k_2, \mathbb{A}_A),$$

and  $G'_A$  is the group that  $A$  must be in and  $\mathbb{A}_A$  is the access structure on attributes that  $A$  must satisfy in order to complete the handshake.

3. Upon receiving the ciphertext  $c_A$ ,  $A$  runs

$$k_2 \leftarrow \Pi.\text{Decrypt}(\text{GPK}_{G_A}, \text{cred}_A, c_A).$$

4. Upon receiving the ciphertext  $c_B$ ,  $B$  runs

$$k_1 \leftarrow \Pi.\text{Decrypt}(\text{GPK}_{G_B}, \text{cred}_B, c_B).$$

If  $G_A = G'_A$ ,  $G_B = G'_B$ ,  $S_A$  satisfies  $\mathbb{A}_A$  and  $S_B$  satisfies  $\mathbb{A}_B$ , then at the end of the handshake, both  $A$  and  $B$  share the key  $k = (k_1, k_2)$ .

**Theorem 1.** *If the centralized CP-ABE scheme  $\Pi$  is secure in the model defined in Sect. 2.3, then the resulting secret handshake scheme is impersonator resistant, detector resistant and unlinkable.*

To keep the paper compact, we just give the core idea of the proof here.

*Impersonator resistance:* Let  $\mathcal{A}$  be an adversary who attacks impersonator resistance of the secret handshake scheme. When  $\mathcal{A}$  wants to authenticate to an honest user  $U$ ,  $U$  chooses a random  $k$  and sends a ciphertext  $c$  to  $\mathcal{A}$ ,

$$c \leftarrow \Pi.\text{Encrypt}(\text{GPK}_G, k, \mathbb{A}),$$

where  $G$  is the group that  $\mathcal{A}$  must be in and  $\mathbb{A}$  is the access structure on attributes that  $\mathcal{A}$  must satisfy. If  $\mathcal{A}$  is not a member of the group or  $\mathcal{A}$  does not satisfy  $\mathbb{A}$ , because the CP-ABE scheme  $\Pi$  has plaintext privacy, then  $\mathcal{A}$  cannot achieve any information about  $k$  and the handshake will fail.

*Detector resistance:* Let  $\mathcal{A}$  be an activate adversary. When  $\mathcal{A}$  engages in the secret handshake protocol with an honest user  $U$ , Since  $\mathcal{A}$  does not satisfy the access structure specified by  $U$  and the underlying CP-ABE scheme  $\Pi$  has plaintext privacy, the handshake will fail and  $\mathcal{A}$  can not decide whether  $U$  satisfies the access structure or not.

In the case that  $\mathcal{A}$  is a passive adversary, due to hidden policy privacy of the CP-ABE scheme  $\Pi$ , the ciphertexts sent during the handshake protocol do not reveal the attribute value information in the access structure and  $\mathcal{A}$  also cannot decide whether an honest user satisfies the access structure or not.

*Unlinkability:* In our secret handshake scheme, the messages exchanged during the handshake protocol are the ciphertexts of the CP-ABE scheme  $\Pi$ . Because  $\Pi$  has ciphertext-policy privacy, protocol messages do not reveal any information about the access structures on attributes; therefore, it is impossible to distinguish whether two different executions of the protocol were performed by the same user or not.

It is apparent that the secret handshake scheme obtained from our generic construction preserves the access structures of the underlying CP-ABE scheme, which will support dynamic and expressive matching policies.

## 4 An Efficient Instantiation

Based on the construction above, we describe a concrete instantiation of attribute-based secret handshake employing the CP-ABE scheme proposed in [18]. We first modify the scheme to a centralized CP-ABE, and then obtain the attribute-based secret handshake as follows.

**Setup**( $1^\lambda$ ): The setup algorithm first runs  $\mathcal{G}(1^\lambda)$  to obtain  $(p_1, p_2, p_3, p_4, \mathbb{G}, \mathbb{G}_T, e)$  with  $\mathbb{G} = \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3} \times \mathbb{G}_{p_4}$ , where  $\mathbb{G}$  and  $\mathbb{G}_T$  are cyclic groups of order  $N = p_1 p_2 p_3 p_4$ . Next it chooses  $g \in \mathbb{G}_{p_1}$ ,  $X_3 \in \mathbb{G}_{p_3}$ ,  $X_4 \in \mathbb{G}_{p_4}$  uniformly at random. The public parameters are published as

$$\text{params} = (N, g, X_3, X_4)$$

**CreateGroup**(params): The group administrator GA of a group  $G$  takes the public parameter **params** as input and chooses  $h, u_1, \dots, u_n \in \mathbb{G}_{p_1}$ ,  $Z \in \mathbb{G}_{p_4}$ ,  $\alpha, a \in \mathbb{Z}_N$  uniformly at random. Then outputs group  $G$ 's public information

$$GPK_G = (g^a, e(g, g)^\alpha, u_1, \dots, u_n, H = h \cdot Z).$$

The master secret key is  $GSK_G = (h, \alpha)$ .

**AddUser**( $GSK_G, S_U$ ): To add a member  $U$  with a set of attributes  $S_U$  to the group  $G$ , the group administrator GA takes input  $GPK_G$ ,  $GSK_G$  and  $S_U$ , and chooses  $t \in \mathbb{Z}_N$ ,  $R, R', R_1, \dots, R_n \in \mathbb{G}_{p_3}$  uniformly at random. Then the credential  $\text{cred}_U = (S_U, K_U, K'_U, \{K_U^i\}_{1 \leq i \leq n})$  is computed as

$$K_U = g^\alpha g^{at} R, K'_U = g^t R', K_U^i = (u_i^{s_i} h)^t R_i.$$

**HandShake**( $A, B$ ): Let  $A$  be a member of group  $G_A$  and  $B$  be a member of  $G_B$ . Suppose  $A$  with a secret credential  $\text{cred}_A$  of attributes  $S_A$ , and  $B$  with a secret credential  $\text{cred}_B$  corresponding to attributes  $S_B$ , are engaging in a handshake protocol. The protocol proceeds as follows:

1.  $A$  sets a policy  $\mathbb{A} = (\mathbf{M}, \rho, \mathcal{T}_B)$  that can be satisfied by  $S_B$ , in which  $\mathbf{M}$  is an  $\ell \times n$  matrix,  $\rho$  is a map from each row  $M_x$  of  $\mathbf{M}$  to an attribute name and  $\mathcal{T}_B = (t_{\rho(1)}, \dots, t_{\rho(\ell)}) \in \mathbb{Z}_N^\ell$ . Then  $A$  randomly chooses  $k_1 \in \mathbb{G}_T$ ,  $r_{1x}, r'_{1x} \in \mathbb{Z}_N$ ,  $v_1, v'_1 \in \mathbb{Z}_N^n$ , where  $v_1 = (s_1, v_{10}, \dots, v_{1n})$  and  $v'_1 = (s'_1, v'_{10}, \dots, v'_{1n})$ . For  $1 \leq x \leq \ell$ , it picks  $Z_{1,x}, Z'_{1,x}, Z_{2,x}, Z'_{2,x} \in \mathbb{G}_{p_4}$ . Finally,  $A$  utilizes  $\text{GPK}_{G_{B'}} = (g^{\alpha_1}, e(g, g)^{\alpha_1}, u_1, \dots, u_n, H = h \cdot Z)$  to compute

$$\begin{aligned} \tilde{C}_1 &= k_1 \cdot e(g, g)^{\alpha_1 s_1}, \quad C'_1 = g^{s_1}, \\ C_{1,x} &= g^{\alpha_1 M_x \cdot v_1} (u_{\rho(x)}^{t_{\rho(x)}} H)^{-r_{1x}} \cdot Z_{1,x}, \quad D_{1,x} = g^{r_{1x}} \cdot Z'_{1,x}, \\ \tilde{C}_2 &= e(g, g)^{\alpha_1 s'_1}, \quad C'_2 = g^{s'_1}, \\ C_{2,x} &= g^{\alpha_1 M_x \cdot v'_1} (u_{\rho(x)}^{t_{\rho(x)}} H)^{-r'_{1x}} \cdot Z_{2,x}, \quad D_{2,x} = g^{r'_{1x}} \cdot Z'_{2,x}. \end{aligned}$$

and sends the ciphertext  $C_B = ((\mathbf{M}, \rho), \tilde{C}_1, C'_1, \{C_{1,x}, D_{1,x}\}_{1 \leq x \leq \ell}, \tilde{C}_2, C'_2, \{C_{2,x}, D_{2,x}\}_{1 \leq x \leq \ell})$  to  $B$ . Note that  $G_{B'}$  is the group that  $B$  must be in order to complete the handshake.

2.  $B$  also chooses a policy  $\mathbb{A}' = (\mathbf{M}', \rho', \mathcal{T}_A)$  that can be satisfied by  $S_A$ , in which  $\mathbf{M}'$  is an  $\ell \times n$  matrix,  $\rho'$  is a map from each row  $M'_x$  of  $\mathbf{M}'$  to an attribute name and  $\mathcal{T}_A = (t_{\rho'(1)}, \dots, t_{\rho'(\ell)}) \in \mathbb{Z}_N^\ell$ . Then  $B$  randomly chooses  $k_2 \in \mathbb{G}_T$ ,  $r_{2y}, r'_{2y} \in \mathbb{Z}_N$ ,  $v_2, v'_2 \in \mathbb{Z}_N^n$ , where  $v_2 = (s_2, v_{20}, \dots, v_{2n})$  and  $v'_2 = (s'_2, v'_{20}, \dots, v'_{2n})$ . For  $1 \leq y \leq \ell$ , it picks  $Z_{3,y}, Z'_{3,y}, Z_{4,y}, Z'_{4,y} \in \mathbb{G}_{p_4}$ . Finally,  $B$  uses  $\text{GPK}_{G_{A'}} = (g^{\alpha_2}, e(g, g)^{\alpha_2}, u'_1, \dots, u'_n, H' = h' \cdot Z')$  to compute

$$\begin{aligned} \tilde{C}_3 &= k_2 \cdot e(g, g)^{\alpha_2 s_2}, \quad C'_3 = g^{s_2}, \\ C_{3,y} &= g^{\alpha_2 M'_y \cdot v_2} (u_{\rho'(y)}^{t_{\rho'(y)}} H')^{-r_{2y}} \cdot Z_{3,y}, \quad D_{3,y} = g^{r_{2y}} \cdot Z'_{3,y}, \\ \tilde{C}_4 &= e(g, g)^{\alpha_2 s'_2}, \quad C'_4 = g^{s'_2}, \\ C_{4,y} &= g^{\alpha_2 M'_y \cdot v'_2} (u_{\rho'(y)}^{t_{\rho'(y)}} H')^{-r'_{2y}} \cdot Z_{4,y}, \quad D_{4,y} = g^{r'_{2y}} \cdot Z'_{4,y}. \end{aligned}$$

and sends  $C_A = ((\mathbf{M}', \rho'), \tilde{C}_3, C'_3, \{C_{3,y}, D_{3,y}\}_{1 \leq y \leq \ell}, \tilde{C}_4, C'_4, \{C_{4,y}, D_{4,y}\}_{1 \leq y \leq \ell})$  to  $A$ . Note that  $G_{A'}$  is the group that  $A$  must belong to in order to complete the handshake.

3. Upon receiving the ciphertext  $C_A$ ,  $A$  parses  $C_A$  as  $((\mathbf{M}', \rho'), \tilde{C}_3, C'_3, \{C_{3,y}, D_{3,y}\}_{1 \leq y \leq \ell}, \tilde{C}_4, C'_4, \{C_{4,y}, D_{4,y}\}_{1 \leq y \leq \ell})$ , and uses  $(\text{GPK}_{G_A}, \text{cred}_A)$  to calculate  $\text{I}_{\mathbf{M}', \rho'}$  from  $(\mathbf{M}', \rho')$ , where  $\text{I}_{\mathbf{M}', \rho'}$  denotes the set of minimum subsets of  $\{1, \dots, \ell\}$  that satisfies  $(\mathbf{M}', \rho')$ . It then checks if there exists a  $I' \in \text{I}_{\mathbf{M}', \rho'}$  that satisfies

$$\tilde{C}_4 = e(C'_4, K_A) / \left( \prod_{i \in I'} (e(C_{4,i}, K'_A) \cdot e(D_{4,i}, K_A^{\rho'(i)}))^{\omega'_i} \right),$$



where  $\sum_{i \in \mathcal{I}'} \omega'_i A'_i = (1, 0, \dots, 0)$ . If no element in  $\mathbf{I}_{\mathbf{M}', \rho'}$  satisfies the above equation, it outputs  $\perp$ . When  $G_A = G_{A'}$ ,  $A$  can recover

$$e(C'_3, K_A) / \left( \prod_{i \in \mathcal{I}'} (e(C_{3,i}, K'_A) \cdot e(D_{3,i}, K_A^{\rho'(i)}))^{\omega'_i} \right) = e(g, g)^{\alpha_2 s_2}.$$

and compute  $k_2$  as  $\tilde{C}_3 / e(g, g)^{\alpha_2 s_2}$ .

4. Upon receiving the ciphertext  $C_B$ ,  $B$  parses  $C_B$  as  $((\mathbf{M}, \rho), \tilde{C}_1, C'_1, \{C_{1,x}, D_{1,x}\}_{1 \leq x \leq \ell}, \tilde{C}_2, C'_2, \{C_{2,x}, D_{2,x}\}_{1 \leq x \leq \ell})$ , and uses  $(GPK_{G_B}, \text{cred}_B)$  to calculate  $\mathbf{I}_{\mathbf{M}, \rho}$  from  $(\mathbf{M}, \rho)$ , where  $\mathbf{I}_{\mathbf{M}, \rho}$  denotes the set of minimum subsets of  $\{1, \dots, \ell\}$  that satisfies  $(\mathbf{M}, \rho)$ . It then checks if there exists a  $\mathcal{I} \in \mathbf{I}_{\mathbf{M}, \rho}$  that satisfies

$$\tilde{C}_2 = e(C'_2, K_B) / \left( \prod_{i \in \mathcal{I}} (e(C_{2,i}, K'_B) \cdot e(D_{2,i}, K_B^{\rho(i)}))^{\omega_i} \right),$$

where  $\sum_{i \in \mathcal{I}} \omega_i M_i = (1, 0, \dots, 0)$ . If no element in  $\mathbf{I}_{\mathbf{M}, \rho}$  satisfies the above equation, it outputs  $\perp$ . When  $G_B = G_{B'}$ ,  $B$  can recover

$$e(C'_1, K_B) / \left( \prod_{i \in \mathcal{I}} (e(C_{1,i}, K'_B) \cdot e(D_{1,i}, K_B^{\rho(i)}))^{\omega_i} \right) = e(g, g)^{\alpha_1 s_1}.$$

and compute  $k_1$  as  $\tilde{C}_1 / e(g, g)^{\alpha_1 s_1}$ .

At the end of the handshake, both  $A$  and  $B$  share the key  $k = (k_1, k_2)$ .

According to Theorem 1, the secret handshake protocol is secure as long as the underlying modified CP-ABE scheme with partially hidden access structures is CPA secure. We now state the security theorem of the modified CP-ABE scheme.

**Theorem 2.** *If Assumptions 1, 2, 3 and 4 hold, then the modified centralized CP-ABE is CPA secure and the access structures is partially hidden.*

The proof employs the dual system technology proposed in [30] which is similar with the proof in [18]. In the underlying CP-ABE scheme, the encryption algorithm encrypts both the sharing key  $k_i$  and a constant message ‘1’ to get the ciphertext without the information of attribute-values in the access structure. When decrypting the ciphertext, the decryption algorithm first decrypts the second part of ciphertext to check whether the result equals to ‘1’, if so, the first part ciphertext could be decrypted correctly, otherwise, it means the access structure cannot be satisfied by the attributes associated with the key. We note that, the modification of the global parameters is intended to guarantee that users running the secret handshake protocol are using parameters in the same group, thus the adversary cannot tell whether say Alice is shaking with Bob or Carol.

## 5 Conclusions

In this paper, we studied attribute-based secret handshakes which support dynamic flexible or expressive matching policies on attributes compared to the threshold policy in previous schemes.

We first introduced a notion of fully secure centralized CP-ABE and then proposed a generic construction of attribute-based secret handshakes based on the primitive. Our handshake schemes support the same access structures on attributes as those supported by the underlying CP-ABE and achieves unlinkability with reusable credentials.

Then we proposed an efficient attribute-based secret handshake scheme employing CP-ABE scheme with partial hidden access structure. Our construction supports dynamic flexible matching policy and can provide impersonator resistance, detector resistance and unlinkability secure properties.

**Acknowledgement.** This work was supported by National Natural Science Foundation of China (Nos. 61572235, 61300226), Research Fund for the Doctoral Program of Higher Education of China (No. 20134401120017), Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2015A030306045), ISN Research Fund (No. ISN15-04) and Pearl River S&T Nova Program of Guangzhou.

## References

1. Ateniese, G., Kirsch, J., Blanton, M.: Secret handshakes with dynamic and fuzzy matching. In: Proceedings of the 14th Annual Network and Distributed System Security Symposium, NDSS (2007)
2. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.C.: Secret handshakes from pairing-based key agreements. In: IEEE Symposium on Security and Privacy 2003, pp. 180–196. IEEE Computer Society (2003)
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy 2007, pp. 321–334. IEEE Computer Society (2007)
5. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Crypt.* **21**(2), 149–177 (2008)
7. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
8. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)

9. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) *Advances in Cryptology — EUROCRYPT 2003*. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
10. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret handshakes from CA-Oblivious encryption. In: Lee, P.J. (ed.) *ASIACRYPT 2004*. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
11. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: *Proceedings of ACM CCS 2007*, pp. 456–465. ACM Press, New York (2007)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: *Proceedings of ACM CCS 2006*, pp. 89–98. ACM Press, New York (2006)
13. Jarecki, S., Kim, J.H., Tsudik, G.: Authentication for paranoids: multi-party secret handshakes. In: Zhou, J., Yung, M., Bao, F. (eds.) *ACNS 2006*. LNCS, vol. 3989, pp. 325–339. Springer, Heidelberg (2006)
14. Jarecki, S., Kim, J.H., Tsudik, G.: Beyond secret handshakes: affiliation-hiding authenticated key exchange. In: Malkin, T. (ed.) *CT-RSA 2008*. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
15. Jarecki, S., Liu, X.: Unlinkable secret handshakes and key-private group key management schemes. In: Katz, J., Yung, M. (eds.) *ACNS 2007*. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
16. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Cryptology ePrint Archive, Report 2007/404* (2007)
18. Lai, J.Z., RH. D, YJ. Li.: Expressive CP-ABE with partially hidden access structures. In: *7th ACM Symposium on Information, Computer and Communications Security* (2012)
19. 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)
20. Li, N., Du, W., Boneh, D.: Oblivious signature-based envelope. In: *ACM Symposium on Principles of Distributed Computing (PODC 2003)*, pp. 182–189. ACM Press, New York (2003)
21. Nasserian, S., Tsudik, G.: Revisiting oblivious signature-based envelopes. In: Di Crescenzo, G., Rubin, A. (eds.) *FC 2006*. LNCS, vol. 4107, pp. 221–235. Springer, Heidelberg (2006)
22. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellare, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) *ACNS 2008*. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008)
23. 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)
24. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: *Proceedings of ACM CCS 2007*, pp. 195–203. ACM Press, New York (2007)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

26. Sorniotti, A., Molva, R.: Secret handshakes with revocation support. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 274–299. Springer, Heidelberg (2010)
27. Tsudik, G., Xu, S.: Brief announcement: a flexible framework for secret handshakes. In: PODC 2005, p. 39. ACM Press, New York (2005)
28. Vergnaud, D.: RSA-based secret handshakes. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 252–274. Springer, Heidelberg (2006)
29. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. Cryptology ePrint Archive, Report 2008/290 (2008). <http://eprint.iacr.org/>
30. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
31. Xu, S., Yung, M.: k-anonymous secret handshakes with reusable credentials. In Proceedings of ACM CCS 2004, pp. 158–167. ACM Press, New York (2004)
32. Zhou, L., Susilo, W., Mu, Y.: Three-round secret handshakes based on ElGamal and DSA. In: Chen, K., Deng, R., Lai, X., Zhou, J. (eds.) ISPEC 2006. LNCS, vol. 3903, pp. 332–342. Springer, Heidelberg (2006)

# Ciphertext-Policy Attribute-Based Encryption with Key-Delegation Abuse Resistance

Yinhao Jiang<sup>(✉)</sup>, Willy Susilo, Yi Mu, and Fuchun Guo

Centre for Computer and Information Security Research,  
School of Computing and Information Technology, University of Wollongong,  
Wollongong, Australia

{yj971,wsusilo,ymu,fuchun}@uow.edu.au

**Abstract.** Attribute-based encryption (ABE) is a promising cryptographic primitive that allows one-to-many encryption. In such a system, users' private keys are linked to their access rights. We note that if a user can generate a new private key for a portion of his/her access right, this could potentially lead to some undesirable situations, which violate the access control policy. Interestingly, to date, there is no work that looks into this matter in detail nor addresses it. We point out that this is a "property" that exists in ABE systems, which we refer to "key-delegation abuse". ABE systems that suffer from key-delegation abuse will hinder the adoption of these systems in practice. In this work, *for the first time in the literature*, we address the "key-delegation abuse" problem in Ciphertext-policy Attribute-based Encryption (CP-ABE) systems. We introduce a new mechanism to enhance CP-ABE schemes that provide protections against this key-delegation abuse issue. We formalize the security requirements for such a property, and subsequently construct a CP-ABE scheme that satisfies the new security requirements. We also present an application of our scheme to a traceable CP-ABE, where the "traitors", i.e. the users who have leaked their keys, can be traced.

**Keywords:** Attribute-based encryption · Key-delegation abuse · Ciphertext-policy

## 1 Introduction

In modern cryptography, one of the most promising cryptographic primitive is the notion of Attribute-based Encryption (ABE), which allows one-to-many encryption. In this notion, there are two variants, namely Ciphertext-policy Attribute-based Encryption (CP-ABE) and Key-policy Attribute-based Encryption (KP-ABE), which essentially denotes the location of the embedded access policy, whether it is in the ciphertext or in the key. In this work, we mainly discuss CP-ABE. In CP-ABE, a ciphertext on a message is encrypted with an access policy while a private key for a user associated with a set of attributes corresponding to the private key satisfying the access policy.

The basic security in CP-ABE requires that a user cannot generate a *new* private key for an attribute set  $\omega'$  from a private key set for  $\omega$ , if  $\omega \subset \omega'$ . It is an interesting question whether the reverse is also true. That is, whether the key generation for the reversed subset relationship also holds this property. Interestingly, this important issue receives a very limited attention in the literature. Specifically, the question is: given a private key for attribute set  $\omega$ , can the user generate a new private key for any subset  $\omega' \subset \omega$ ? We note that if the answer is positive, this can lead to some undesirable situation. To illustrate this situation, consider the following scenario. A media broadcaster (who is the trusted authority in the cryptographic setting) controls the contents to its subscribers by encrypting the contents with a CP-ABE system. Without losing generality, the contents will be encrypted with an attribute set as follows:  $\{Sport, Biography, Drama, Comedy, Action, Thriller, Fantasy, Sci-Fi, Documentary, War\}$ . Note that there are ten attributes in the possible set in this example. Each possible channel is sold for \$10/month, and hence, it will cost \$100/month to subscribe to all channels. To make the package deal more attractive, the media broadcaster introduces a premium user package. For a premium user package, the user needs to subscribe to *all* channels, and hence the ten attributes, and the premium user will be granted two additional channels, namely  $\{HD, Hollywood-movies\}$ , and the premium price is \$100/month for the whole package. Consider the case where a malicious user, Malva, purchases the premium package. If the CP-ABE scheme that is adopted allows Malva to create a new private key for any attribute, which is a subset to the original attribute set that he has, then Malva can make money from this case. He will then construct a private key for the attribute *Sport* for example, and sells this for \$9/month, and for the ten possible attributes, he will accrue \$90/month. Additionally, he can sell any combinations of the attribute sets (such as  $\{Sport, Fantasy\}$ ) and again sell it at a cheaper price than \$20/month. Note that in total, he will make more than \$100/month by simply re-selling a combination of these channels. We should point out that in this case, it is clear that Malva will be able to manage his own groups of customers with private keys of different sets of attributes and in fact, Malva has functioned as an illegal “trusted authority”, who will compete with the original media broadcaster. Throughout this paper, we shall call this “property” in ABE as the *key-delegation abuse*, if the adversary can generate a private key for any subset without revealing his/her entire access rights. It is clear that this property is undesirable in some scenarios, as outlined above.

Interestingly, the property of *key-delegation abuse* exists in majority of CP-ABE schemes since private keys are usually designed with flexibility in order to meet the requirement of complex and variable access policies in the ciphertexts. As a result, different components of a private key are used for different access policies, which makes it possible for a user to split his/her private key to different parts and construct new private keys from these parts or parts from other users. To the best of our knowledge, the *key-delegation abuse* problem in ABE systems is still not yet well explored in the literature, and hence, it becomes an inherent problem in ABE. For the *key-delegation abuse* problem, existing solutions com-

bine users' private information with their private keys so that malicious users are wary of constructing new private keys based on theirs and introduce an extra trace device/algorithm to pinpoint malicious users from constructed new private keys. However, these approaches have two limitations: (1) they gave a deterrent solution, while users are still capable to issue new private keys; and (2) they need the constructed new key to trace who the malicious user is.

*Our goal in this work.* The aim of this work is to address the *key-delegation abuse* problem. In particular, we aim to make the notion of ABE to be more adoptable in practice, once the problem with key-delegation abuse is removed.

*Our contribution.* In this paper, *for the first time* we propose a ciphertext-policy attribute-based encryption scheme in which users cannot illegally generate new private keys of a subset of the users' original sets of attributes. The access structure used in our CP-ABE is constructed by an AND-gate. This is a subset of the access structures used in [3, 15]. In our scheme, a ciphertext with the access structure  $W$ , which consists of a single AND gate whose input are attributes described by an access policy attribute set  $W$ , can only be decrypted by a private key of a set of attributes  $\omega$  when  $W \subseteq \omega$ . Our technique can be summarized as follows. We utilize the property of bilinear groups. Then, we construct private key components for all attributes but based on two different sets of group elements as if it is contained in the set of attributes of the private key or not. Subsequently, we apply the secret sharing scheme on all attributes, and enforce the bilinear map of key components and ciphertext components for all attributes so that the key cannot be split nor combined with other private keys. We prove the security properties of the scheme in standard selective model. We also introduce a new security game based on [5] for the *key-delegation abuse* problem and prove the new feature of our scheme in generic group model. Additionally, we present an application of our scheme to achieve a traceable CP-ABE scheme, where traitors can be traced efficiently.

*Organization.* The paper is organized as follows: In Sect. 2, we discuss related work. Section 3 provides some background definitions and main properties of ABE system. In Sect. 4 our CP-ABE construction is presented, and the security proof is presented in Sect. 5. In Sect. 6, we discuss an application of our scheme to achieve a traceable CP-ABE scheme. Finally, in Sect. 7, we conclude with some discussions and future work.

## 2 Related Work

The concept of Identity-based Encryption (IBE) was put forth by Shamir [19] in order to ease public-key encryption and certificate management. Specifically, there is no necessity to verify the validity of users' certificates, as users' public key is in fact their identities, such as e-mail addresses or phone numbers. An encryptor can create a ciphertext under the receiver's identity without any prior information. Subsequently, Sahai and Waters [18] proposed the notion of Attribute-based Encryption (ABE) by replacing the identity in IBE with an attribute set.

ABE has been found very useful in providing fine-grained access control systems [6]. Subsequently, many flexible and efficient ABE schemes have been proposed in the literature. As classified by Goyal et al. [6], there are two variants of ABE, namely KP-ABE and CP-ABE. In a KP-ABE, ciphertext is associated with attribute sets, and each user private key is associated with an access structure that specifies which type of ciphertexts the key is able to decrypt. In contrast, in a CP-ABE system, each user's key is associated with an attribute sets and ciphertext specify an access policy over attributes. Therefore, the difference between CP-ABE and KP-ABE relies on who determines the access control policy in the encryption system. In a CP-ABE, when a message is being encrypted, it will be associated with an access structure over a predefined attribute sets. The user will only be able to decrypt a given ciphertext if his/her attributes satisfy the access structure specified in the ciphertext. The first KP-ABE construction [6] realized the monotonic access structures for key policies. To enable more flexible access polices, Ostrovsky et al. [17] developed a KP-ABE scheme to support the expression of non-monotone formulas in key policies. However, KP-ABE is less flexible than CP-ABE because in KP-ABE once a user's private key is issued the access policy is also determined, which makes the encryption more difficult as the encryptor needs to compare recipients' access policies to all other users' to choose a proper set of attributes for the ciphertext. Later, Bethencourt et al. [9] proposed the first CP-ABE construction. However, the construction [9] was only proved secure under the generic group model. To overcome this weakness, Cheung and Newport [3] presented another construction that is proved to be secure under the standard model. Then, Goyal et al. [4] presented another construction for more advanced access structures based on number theoretic assumption. Katz, Sahai, and Waters [10] proposed a novel predicate encryption scheme supporting inner product predicates. Their scheme is very general and can achieve both KP-ABE and hidden CP-ABE schemes. However, the constructions of [2, 10] are very inefficient compared to [16].

In [7], Hinek et al. mentioned the problem of *key cloning*, and another third party should be involved in each users decryption in their scheme, which makes it impractical. Then, the problem of building a secure CP-ABE supporting traceability has recently been studied in [8, 12–14]. The ciphertext access policies in [8, 12] only support a single AND gate with wild-card. The traceable CP-ABE proposed in [14] is as fully secure, highly expressive and efficient as a conventional CP-ABE such as the one in [11], but it only supports tracing 'well-formed' illegally constructed private keys. Later, [13] proposed a new CP-ABE scheme proved fully secure which can trace not 'well-formed' illegally constructed private keys. However, traceability cannot prevent "key-delegation abuse" issue – malicious users can still illegally generate keys in private.

## 2.1 Violating Access Control Policy with “Key-Abuse” Property

The *key-delegation abuse* property is that a user who owns a private key for attribute set  $\omega$  can generate a new private for a subset  $\omega' \subset \omega$ . This property exists in majority of CP-ABE schemes. In the following, we shall demonstrate



that this key-delegation abuse property can lead to some undesirable situation where the access control policy is violated.

Without losing generality, we shall consider the Cheung and Newport scheme proposed in [3]. Cheung and Newport proposed a CP-ABE scheme [3] (which is referred to as the CN scheme throughout this paper), in which access structure is restricted to an AND gate, but attributes  $i$  are allowed to be either positive  $i$ , negative  $\neg i$  or “don’t care”. In their system, let the attribute universe be  $\mathcal{N} = \{1, 2, 3, 4, 5\}$ , then the public key is  $PK = (G = \langle g \rangle, e : G \times G \rightarrow G_T, Y = e(g, g)^y \in G_T, \{T_k = g^{t_k}\}_{k=1, \dots, 15})$ , and the master secret key is  $MK = (y, t_1, \dots, t_{15} \in \mathbb{Z}_p)$ . A private key for attribute set  $\omega = \{1, 2, 3\}$  is

$$sk_\omega = \left( \hat{D} = g^{y-r}, \{D_i = g^{\frac{r_i}{t_i}}\}_{i=1,2,3}, \{D_i = g^{\frac{r_i}{t_{5+i}}}\}_{i=4,5}, \{F_i = g^{\frac{r_i}{t_{10+i}}}\}_{i=1, \dots, 5} \right)$$

where  $r = \sum_{i=1}^5 r_i$ . To encrypt a message  $M$  with AND gate  $W = (1 \wedge 2)$  we have

$$C = \left( W, \tilde{C} = MY^s, \hat{C} = g^s, \{C_i = T_i^s\}_{i=1,2}, \{C_i = T_{10+i}^s\}_{i=3,4,5} \right).$$

To decrypt ciphertext  $C$ , only part of the private key  $(\hat{D}, D_1, D_2, F_3, F_4, F_5)$  is used during decryption since

$$M = \frac{\tilde{C}}{e(\hat{C}, \hat{D}) \cdot \prod_{i=1,2} e(C_i, D_i) \prod_{i=3,4,5} e(C_i, F_i)} = \frac{\tilde{C}}{e(g^s, g^{y-r}) \prod_{i=1}^5 e(g, g)^{r_i \cdot s}} = \frac{\tilde{C}}{e(g, g)^{y \cdot s}} = \frac{\tilde{C}}{Y^s}.$$

Thus, the user who owns  $sk_\omega$  can generate a new key

$$sk' = \left( \hat{D}' = \hat{D}, \{D'_i = D_i\}_{i=1,2}, \{F'_i = F_i\}_{i=1, \dots, 5} \right)$$

to decrypt ciphertexts with AND gate (1),  $(1 \wedge 2)$ , (2) or  $\emptyset$  since  $sk'$  satisfies the decryption algorithm.

From the example, it can be seen that to decrypt ciphertexts with different access policies, different parts of a private key are used during the decryption, which makes it plausible to illegally generate new keys. This property of key-delegation abuse does not break the security of encryption schemes and sometimes is adopted for applications like key delegation. However, unauthorized key generation can lead to violation of access control policy.

### 3 Background

We first give formal definitions for the security of Ciphertext-policy Attribute Based Encryption (CP-ABE). Then we give background information on pairings and complexity assumptions.

### 3.1 Access Structure [3]

Generally speaking, an access structure on attributes is a rule  $\mathbb{A}$  that returns either 0 or 1 given an attribute set  $\omega$ . We say that  $\omega$  satisfies  $\mathbb{A}$  iff  $\mathbb{A}$  answers 1 on  $\omega$ . Access structures may be Boolean expressions, threshold trees, etc.

In this paper, we focus on access structures that consist of a single AND gate whose inputs are attributes. This is denoted  $\mathbb{A} = \bigwedge_{i \in W} i$ , where  $W$  is a subset of the universal attribute set and every  $i$  is an attribute. Given an attribute set  $\omega$ ,  $\mathbb{A}$  answers 1 iff for all  $i \in W$ ,  $i \in \omega$ . Thus,  $\omega$  satisfies  $\mathbb{A}$  iff  $W \subseteq \omega$ .

### 3.2 CP-ABE Definition

A ciphertext-policy attribute-based encryption system consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

**Setup.** The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters  $PK$  and a master secret key  $MK$ .

**Encrypt**( $PK, M, \mathbb{A}$ ). The encryption algorithm takes in the public parameters  $PK$ , the message  $M$ , and an access structure  $\mathbb{A}$  over the universe of attributes.

It will output a ciphertext  $CT$  such that only users whose private keys associated with attribute sets which satisfy the access structure  $\mathbb{A}$  can decrypt  $M$ . We assume that the ciphertext implicitly contains  $\mathbb{A}$ .

**KeyGen**( $MK, \omega$ ). The key generation algorithm takes as input the master secret  $MK$  and a set of attributes  $\omega$ . It outputs a private key  $sk$  associated with  $\omega$ .

**Decrypt**( $PK, CT, sk$ ). The decryption algorithm takes as input the public parameters  $PK$ , a ciphertext  $CT$ , which contains an access structure  $\mathbb{A}$ , and a private key  $sk$ , which is a private key for a set of attributes  $\omega$ . If the attribute set  $\omega$  satisfies the access structure  $\mathbb{A}$  then the algorithm will decrypt the ciphertext and return a message  $M$ .

**Selective CPA Security Model.** We now give the security definition for CP-ABE system. This is described by a security game between a challenger and an adversary. The game proceeds as follows:

**Init.** The adversary outputs a challenge access structure  $\mathbb{A}^*$  to the challenger.

**Setup.** The challenger runs the Setup algorithm and gives the public parameters  $PK$  to the adversary.

**Phase 1.** The adversary queries the challenger for private keys corresponding to sets of attributes  $\omega_1, \dots, \omega_{q_1}$  without any satisfying the access policy  $\mathbb{A}^*$ .

**Challenge.** The adversary declares two equal length messages  $M_0$  and  $M_1$ . The challenger flips a random coin  $\beta \in \{0, 1\}$ , and encrypts  $M_\beta$  with  $\mathbb{A}^*$ , producing  $CT^*$ . It gives  $CT^*$  to the adversary.

**Phase 2.** The adversary queries the challenger for private keys corresponding to sets of attributes  $\omega_{q_1+1}, \dots, \omega_q$  with the same restriction in **Phase 1**.

**Guess.** The adversary outputs a guess  $\beta'$  for  $\beta$ .

The advantage of an adversary in winning this game is defined to be  $\Pr[\beta' = \beta] - \frac{1}{2}$ .

**Definition 1.** A ciphertext-policy attribute-based encryption system is selective chosen-plaintext attack secure if all polynomial time adversaries have at most a negligible advantage in this security game.

**Security Model Against Key-Abuse Attack.** We now give the security definition against Key-Abuse Attack in CP-ABE system. This is described by a security game between a challenger and an adversary. The game is formalized based on [5] and proceeds as follows:

**Setup.** The challenger runs the Setup algorithm and gives the public parameters  $PK$  to the adversary. The attribute universe  $\mathcal{U}$  and message space  $\mathcal{M}$  are also defined during this step.

**Queries.** The adversary queries the challenger for private keys corresponding to different sets of attributes  $\omega_1, \dots, \omega_q \subseteq \mathcal{U}$ . In response, for each query  $\omega_j$  for  $1 \leq j \leq q$  the challenger runs  $\text{KeyGen}(MK, \omega_j)$  to compute the private key  $sk_j$ , and send it back to the adversary  $\mathcal{A}$ .  $\mathcal{A}$  can query the challenger adaptively.

**Output.** The adversary chooses a new attribute set  $\omega^* \neq \omega_j$  for  $1 \leq j \leq q$ , generates a new private key  $sk^*$  for attribute set  $\omega^*$ , a new general decryption algorithm  $\text{Dec}^*(PK, CT, sk)$ , and send them to the challenger.

The adversary *wins* if

1.  $\text{Dec}^*(sk^*, \text{Enc}(PK, M, \mathbb{A})) = M$  for all  $\mathbb{A} = \bigwedge_{i \in W} i$ ,  $W \subseteq \omega^*$  and any message  $M \in \mathcal{M}$ .
2. For all possible decryption algorithm  $\text{Dec}'(sk^*, \text{Enc}(PK, M, \mathbb{A}))$  outputs  $\perp$  for all  $W \not\subseteq \omega^*$  and any message  $M \in \mathcal{M}$ .

The advantage of  $\mathcal{A}$  is defined to be the probability that  $\mathcal{A}$  wins the security game.

**Definition 2.** A ciphertext-policy attribute-based encryption system is secure against Key-Abuse Attack if all polynomial time adversaries have at most a negligible advantage in this security game.

### 3.3 Pairings and Complexity Assumption

**Bilinear Groups.** We briefly review the bilinear maps and bilinear map groups [1].

Let  $G, G_T$  be cyclic groups of prime order  $p$ , and let  $g$  be a generator of  $G$ . We say  $G$  has an admissible bilinear map,  $e : G \times G \rightarrow G_T$ , into  $G_T$  if the following three conditions hold:

1. *Bilinearity*  $e(g^a, g^b) = e(g, g)^{ab}$  for all  $a, b$ .
2. *Non-degeneracy*  $e(g, g) \neq 1$ .
3. *Computability*  $e(g^a, g^b)$  for all  $g^a, g^b \in G$  can be computed efficiently.

**Complexity Assumption.** We state our complexity assumption below.

**Definition 3** (*Decisional Bilinear Diffie-Hellman (BDH) Assumption*). Suppose a challenger chooses  $a, b, c, z \in \mathbb{Z}_p$  at random. The Decisional BDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple  $(A = g^a, B = g^b, C = g^c, Z = e(g, g)^{abc})$  from the tuple  $(A = g^a, B = g^b, C = g^c, Z = e(g, g)^z)$  with more than a negligible advantage.

### 4 CP-ABE Construction

In this section, we shall present our CP-ABE scheme. For simplicity, let the universe of attributes be  $\mathcal{U} := \{1, \dots, n\}$  for some natural number  $n$ .

In our construction the key generation algorithm will link the key components of one user with a specific set of group elements, and then apply the secret sharing technology to all attributes, so that the key cannot be split or combined to obtain other workable secret keys. Each private key will be generated including one key component per attribute: if the user owns this attribute the key component will be generated with the set of group elements of  $t_i$ ; otherwise, generated with the set of group elements of  $t_{n+i}$ . The encryption algorithm will take as input an AND gate and distribute a random exponent  $r \in \mathbb{Z}_p$  according to all attributes: if an attribute is included in the AND gate there will be only one ciphertext component for this attribute generated with the set of group elements  $h_i$  for decryption; otherwise, two ciphertext components for this attribute will be generated with  $h_i$  and  $h_{n+i}$ .

**Setup**( $\lambda, \mathcal{U}$ ): Given a security parameter  $\lambda$  and an attribute universe  $\mathcal{U}$  of size of  $n$ , the setup algorithm first chooses a bilinear group  $G$  of prime order  $p$ . It then chooses random numbers  $t_1, \dots, t_{2n}, \alpha \in \mathbb{Z}_p$ , random group generators  $g_0, h_0 \in G$ , and computes

$$Y = e(g_0, h_0)^\alpha, h_1 = h_0^{t_1}, \dots, h_n = h_0^{t_n}, h_{n+1} = h_0^{t_{n+1}}, \dots, h_{2n} = h_0^{t_{2n}}.$$

The public parameters PK are  $PK = (h_1, \dots, h_{2n}, Y, e, G, G_T, \mathcal{U})$ . The master secret key MK is  $MK = (g_0, t_1, \dots, t_{2n}, \alpha)$ .

**Enc**( $PK, M, \mathbb{A}$ ): To encrypt a message  $M \in G_T$  with an access structure  $\mathbb{A} = \bigwedge_{i \in W} i$  the following steps are taken. A random value  $r \in \mathbb{Z}_p$  is chosen uniformly. The ciphertext is then created as:

$$CT = (\mathbb{A}, E' = MY^r, \{E_i = h_i^r\}_{i \in W}, \{E_i = h_{n+i}^r, E'_i = h_i^r\}_{i \in \mathcal{U} \setminus W}).$$

**KeyGen**( $MK, \omega$ ): To generate a private key for attribute set  $\omega \subseteq \mathcal{U}$  the following steps are taken.  $n - 1$  random values  $x_1, \dots, x_{n-1}$  are randomly chosen in  $\mathbb{Z}_p$  and compute  $x_n = \alpha - x_1 - \dots - x_{n-1} \in \mathbb{Z}_p$ . The private key for the attribute set  $\omega$ :

$$sk = \left( \omega, \{D_i = g_0^{\frac{x_i}{t_i}}\}_{i \in \omega}, \{D_i = g_0^{\frac{x_i}{t_{n+i}}}\}_{i \in \mathcal{U} \setminus \omega} \right).$$

$\text{Dec}(PK, CT, sk)$ : Suppose that a ciphertext,  $CT$ , is encrypted with an access structure  $\mathbb{A} = \bigwedge_{i \in W} i$  and we have a private key for attribute set  $\omega$ , where  $W \subseteq \omega$ .

Then, the ciphertext can be decrypted by following steps:

$$\begin{aligned} \prod_{i \in W \cup \{\mathcal{U} \setminus \omega\}} e(D_i, E_i) \prod_{i \in \omega \setminus W} e(D_i, E'_i) &= \prod_{i \in \omega} e(g_0^{\frac{x_i}{t_i}}, h_i^r) \prod_{i \in \mathcal{U} \setminus \omega} e(g_0^{\frac{x_i}{t_{n+i}}}, h_{n+i}^r) \\ &= \prod_{i \in \omega} e(g_0^{\frac{x_i}{t_i}}, h_0^{t_i r}) \prod_{i \in \mathcal{U} \setminus \omega} e(g_0^{\frac{x_i}{t_{n+i}}}, h_0^{t_{n+i} r}) \\ &= e(g_0, h_0)^{r \sum_{i \in \mathcal{U}} x_i} = e(g_0, h_0)^{\alpha r}. \\ \frac{\prod_{i \in W \cup \{\mathcal{U} \setminus \omega\}} e(D_i, E_i) \prod_{i \in \omega \setminus W} e(D_i, E'_i)}{E'} &= \frac{MY^r}{e(g_0, h_0)^{\alpha r}} = M. \end{aligned}$$

## 5 Security Proof

We shall prove the following theorem.

**Theorem 1** *If the DBDH assumption holds, our CP-ABE scheme defined in Sect. 3 is secure in the sense of Definition 2.*

*Proof* To prove the theorem, let us assume that there is an adversary  $\mathcal{A}$  that can break our CP-ABE scheme with non-negligible probability. We show how to use this adversary to construct an algorithm  $\mathcal{B}$  which breaks the DBDH assumption.

For the algorithm  $\mathcal{B}$  breaking the DBDH assumption, we let the challenger set the groups  $G$  and  $G_T$  of prime  $p$  with an efficient bilinear map,  $e$  and generator  $g$ . The challenger then flips a fair binary coin  $\mu$  independent of  $\mathcal{B}$ 's view. If  $\mu = 0$  the challenger sets  $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^{abc})$ ; otherwise  $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^z)$ . At a high level, our simulation works as follows. We build a simulator that simulates the joint distribution consisting of adversary's view in its attack in the security game, and the hidden bit  $\beta$  which is not a part of the adversary's view.

We will show that if the input comes as  $\mu = 0$ , the simulation will be perfect, and so the adversary will launch its full ability breaking our CP-ABE. We will also show that if the input comes as  $\mu = 1$ , then the adversary's view is independent of  $\beta$ , and therefore the adversary's advantage is negligible. This immediately implies  $\mathcal{B}$  distinguishing the distribution of its input tuple: run the simulator and adversary together, and if the simulator outputs  $\beta$  and the adversary outputs  $\beta'$ ,  $\mathcal{B}$  outputs  $\mu = 0$  if  $\beta = \beta'$ , and 1 otherwise.

We now give the details of the simulator.

The input to the simulator is  $(p, G, G_T, e, g, A = g^a, B = g^b, C = g^c, Z)$ .

**Init.** During the *Init* phase, the simulator receives the challenge access structure  $\mathbb{A}^* = \bigwedge_{i \in W^*} i$ , where  $W^* \subseteq \mathcal{U}$ , from the adversary  $\mathcal{A}$ .

**Setup.** First simulator chooses random numbers  $v, \nu, \lambda_1, \dots, \lambda_n, \gamma_1, \dots, \gamma_n \in \mathbb{Z}_p$ . Next, the simulator computes

$$\begin{aligned} g_0 &= g^v, h_0 = g^\nu, h_i|_{i \in \mathcal{U}} = g^{\nu\lambda_i} = h_0^{\lambda_i}, \\ h_{n+i}|_{i \in \omega^*} &= B^{\nu\gamma_i} = h_0^{b\gamma_i}, h_{n+i}|_{i \in \mathcal{U} \setminus \omega^*} = g^{\nu\gamma_i} = h_0^{\gamma_i}, \\ Y &= e(A, B)^{v\nu} = e(g_0, h_0)^{ab}. \end{aligned}$$

Since  $h_i = h_0^{t_i}$  and  $h_{n+i} = h_0^{t_{n+i}}$  for each attribute  $i \in \mathcal{U}$ , the simulator sets  $t_i := \lambda_i \in \mathbb{Z}_p$  for each attribute  $i \in \mathcal{U}$ ,  $t_{n+i} := b\gamma_i \in \mathbb{Z}_p$  for each attribute  $i \in \omega^*$  and  $t_{n+i} := \gamma_i \in \mathbb{Z}_N$  for each attribute  $i \in \mathcal{U} \setminus \omega^*$ . Since  $Y = e(u_0, v_0)^\alpha$ , the simulator also sets  $\alpha := ab \in \mathbb{Z}_p$ .

The simulated public parameters are  $PK = (h_1, \dots, h_{2n}, Y, e, G, G_T, \mathcal{U})$ . The master secret key is  $MK = (g_0, t_1, \dots, t_{2n}, \alpha)$ .

**Phase 1.** The adversary  $\mathcal{A}$  makes private key queries. The simulator responds to a query on  $\omega$ , where  $W^* \not\subseteq \omega$ , as follows. Observe that there must exist an attribute  $k \in W^*$  such that  $k \notin \omega$ . The simulator first chooses such an attribute  $k$ . Next, the simulator chooses  $x'_1, \dots, x'_{n-1} \in \mathbb{Z}_N$  uniformly at random and computes  $x'_n = -\sum_i x'_i$ . Then the simulator sets  $x_i := bx'_i$  for each attribute  $i \neq k \in \mathcal{U}$  and  $x_k := ab + bx'_k$  for the attribute  $k$ . Finally, the simulator computes

$$\begin{aligned} \forall i \in \omega, D_i &= B^{\frac{vx'_i}{\lambda_i}} = (g^v)^{\frac{bx'_i}{\lambda_i}} = g_0^{\frac{x_i}{t_i}} \\ \forall i \notin \omega, i \in W^*, i \neq k, D_i &= g^{\frac{vx'_i}{\gamma_i}} = (g^v)^{\frac{bx'_i}{b\gamma_i}} = g_0^{\frac{x_i}{t_{n+i}}} \\ \forall i \notin \omega, i \in W^*, i = k, D_k &= A^{\frac{v}{\gamma_k}} \cdot g^{\frac{vx_k}{\gamma_k}} = g^{\frac{(ab+bx'_k)v}{b\gamma_k}} = g_0^{\frac{x_k}{t_{n+k}}} \\ \forall i \notin \omega, i \notin W^*, D_i &= B^{\frac{vx'_i}{\gamma_i}} = (g^v)^{\frac{bx'_i}{\gamma_i}} = g_0^{\frac{x_i}{t_{n+i}}} \end{aligned}$$

and passes  $sk = (\omega, \{D_i\}_{i \in \mathcal{U}})$  onto  $\mathcal{A}$ .

Here we check the correctness of the simulated private key.

$$\sum_{i \in \mathcal{U}} x_i = \sum_{i \neq k, i \in \mathcal{U}} x_i + x_k = b \sum_{i \neq k, i \in \mathcal{U}} x'_i + ab + bx'_k = ab.$$

**Challenge.** The adversary  $\mathcal{A}$  outputs messages  $M_0, M_1$ . The simulator generates a bit  $\beta \in \{0, 1\}$  and sends  $\mathcal{A}$  the challenge ciphertext:

$$\begin{aligned} CT^* &= \left( \mathbb{A}^*, E' = M_\beta \cdot Z^{v\nu}, \{E_i = C^{\nu\lambda_i} = h_i^c\}_{i \in W^*}, \right. \\ &\quad \left. \{E_i = C^{\nu\gamma_i} = h_{n+i}^c, E'_i = C^{\nu\lambda_i} = h_i^c\}_{i \in \mathcal{U} \setminus W^*} \right). \end{aligned}$$

**Phase 2.**  $\mathcal{A}$  makes key generation queries, and the simulator responds as in Phase 1.

**Guess.** Finally, the adversary outputs guesses  $\beta'$ . If  $\beta = \beta'$ ,  $\mathcal{B}$  outputs 0 indicating that  $Z = e(g, g)^{abc}$ ; otherwise, it outputs 1.

**Perfect Simulation:** When  $\mu = 1$  and  $Z = e(g, g)^{abc}$ , we have

$$E' = M_{\beta}e(g, g)^{abc\nu\nu} = M_{\beta}e(g_0, h_0)^{abc} = M \cdot Y^c.$$

Thus,  $CT^*$  is a valid ciphertext for  $\mathbb{A}^*$ , and the public key and challenge ciphertext issued by the simulator comes from a distribution identical to that in the actual construction; however, we still must show that the private keys issued by the simulator are appropriately distributed. To show that the keys issued by the simulator are appropriately distributed, it suffices to show that, from  $\mathcal{A}$ 's view, the value  $g^a, g^b$  is uniformly random and independent. But this follows from the fact that  $g^a, g^b$  is chosen uniformly random in  $G$  from the input.

**Probability Analysis:** We assume the adversary  $\mathcal{A}$  breaks our CP-ABE scheme with non-negligible probability  $\epsilon$ . If  $Z = e(g, g)^{abc}$ , then the simulation is perfect, and  $\mathcal{A}$  will guess the bit  $\beta$  correctly with probability  $1/2 + \epsilon$ . Else,  $Z = e(g, g)^z$  is uniformly random in  $G_T$ , and thus  $E'$  is uniformly random and independent element in  $G_T$ . In this case, with probability  $1 - 1/p$  the value of  $\beta$  is independent from  $\mathcal{A}$ 's view. Thus, we have that

$$\Pr[\mathcal{B}(A, B, C, Z) = 0] - \Pr[\mathcal{B}(A, B, C, Z) = 0] \geq \epsilon - 1/p.$$

This concludes the proof of Theorem. □

**Theorem 2.** *Our CP-ABE scheme is secure against Key-Abuse Attack (in the sense of Definition 3) in the generic group model.*

*Proof.* We briefly discuss the high level idea of the here. A full security proof is given in Appendix A.

**High Level Idea.** In the generic group model, the adversary can only manipulate group elements by using the canonical group operations, independent of the encoding for group elements. Thus if the adversary is given group elements  $g^{\delta_1}, \dots, g^{\delta_t} \in G$  as its only inputs, then each element of  $G$  output by the adversary must be of the form  $g^{\pi(\delta_1, \dots, \delta_t)}$ , where  $\pi$  is a fixed multilinear polynomial.

Suppose the adversary gives a new private key  $sk^*$  with a decryption algorithm  $Dec^*(\cdot)$  for an attribute set  $\omega^*$ , with which ciphertexts encrypted with  $\mathbb{A}^* = \bigwedge_{i \in \omega^*} i$  can be decrypted. Using a standard argument for the generic group model, we first show that if this is to happen with non-negligible probability, then the multi-linear polynomials as described above in the new private key must also satisfy corresponding constraints. Thus our approach is to assume that the multi-linear polynomials corresponding to the adversary's output satisfy the required constraints, and then obtain a contradiction. We proceed by arguing that in order to satisfy the constraints, the polynomials must have certain structure (i.e., they can only depend on certain given group elements).

First, for a ciphertext  $CT$  encrypted under  $\mathbb{A}^* = \bigwedge_{i \in \omega^*} i$  the new private key can decrypt  $M$  from  $E'$  if it can be used to compute  $Y^r = e(g_0, h_0)^{\alpha r}$ . Thus, it contains a group element in  $G$  for each attribute in  $\mathcal{U}$  to pair the corresponding  $E_i$  (or  $E'_i$ ) in the ciphertext in bilinear map. We denote these group elements by

$D_i^*$  for attribute  $i$  in  $\mathcal{U}$  and the necessary structure of the new private key can be presented as  $(\omega^*, \{D_i^*\}_{i \in \mathcal{U}})$ .

After narrowing down the necessary construction for  $sk^*$ , we note that  $D_i^*$  needs to be constructed based on key components  $D_i^{(j)}$  from  $j$ -th queried private key  $sk^{(j)}$  for attribute set  $\omega_j$  since there is no other given group elements related to the unknown master secret key  $\alpha$  for the adversary. Nevertheless, we also note that because of the difference of the queried attribute sets, for the same attribute  $i$  the key components  $D_i^{(j)}$  might be generated based on different sets of group elements, which makes them irreconcilable to be combined together. Thus, the new private key  $sk^*$  can only depend on one queried private key  $sk_j$  where  $\omega^* \subset \omega_j$ . But this will result in that  $sk^*$  can be used to decrypt ciphertexts encrypted with  $\omega_j$  that is an attribute set beyond the supposed  $\omega^*$ , which contradicts the second condition in the security game's definition.

## 6 Application: Traceable CP-ABE

In this section, we shall discuss an application of our CP-ABE scheme to realize a traceable CP-ABE scheme, which is a CP-ABE scheme that is equipped with a traitor tracing mechanism. The main purpose of traitor tracing in ABE system is to guarantee that any user who illegally shared his/her private key can be traced. Many works have explored traceability in ABE schemes [8, 12–14]. Most of them focused on tracing new keys generated in collusive way, but few can prevent one user generating new workable keys in private. Based on our “key-delegation abuse” resistant CP-ABE scheme, we can obtain a *Traceable CP-ABE* system that can trace privately generated illegal new keys with an extended attribute universe. Each user is given an attribute set that consists of attributes from the original attribute universe, which present his/her access right, as well as attributes from the extended attribute set, which indicate his/her identity. To be specific, we first let the original attribute universe be  $\mathcal{U} := \{1, \dots, n\}$  and a user identity space be  $\mathcal{I}$  of size of  $2^l$ , and we have the extended universe  $\mathcal{U}' := \{1, \dots, n + l\}$  in which attributes  $\{1, \dots, n\}$  are used for describing access right and attributes  $\{n + 1, \dots, n + l\}$  are used to indicate identities. Next, when a private key for an attribute set  $\omega$  (which only consists of attributes  $\{1, \dots, n\}$ ) and a user identity  $ID$  is queried, the user's identity  $ID$  is mapped to a distinct binary string  $L_{ID} \in \{0, 1\}^l$  by a collision-resistant hash function. According to the identity binary string  $L_{ID}$ , if the  $k$ -th digit is 1 the corresponding  $(n + k)$ -th attribute is added into a dummy attribute set  $\omega_{ID}$ . The private key is then generated based on attribute set  $\omega' = \omega \cup \omega_{ID}$ . Since the decryption algorithm of our CP-ABE scheme requires corresponding key components for all attributes in the extended universe and our CP-ABE scheme is key-delegation abuse resistant, a user who wants to share his/her private key needs to give away the whole key, which will also give away the unique dummy attribute set. Thus, if a private key is shared, then the user will be traceable.



Using this technique, we can now describe our traceable CP-ABE construction using our key-delegation abuse resistant CP-ABE scheme  $\Pi_{CP-ABE}^{KAR} = (\text{Setup}^{KAR}, \text{KeyGen}^{KAR}, \text{Enc}^{KAR}, \text{Dec}^{KAR})$  as follows.

**Setup**( $\lambda, \mathcal{I}, \mathcal{U}$ ): Given a security parameter  $\lambda$ , a user identity space  $\mathcal{I}$  of size of  $2^l$  and an attribute universe  $\mathcal{U}$  of size of  $n$ , the setup algorithm first sets the new universe  $\mathcal{U}' := \{1, \dots, n+l\}$ . Next, it runs  $\text{Setup}^{KAR}(\lambda, \mathcal{U}')$  to get the master secret key  $MK^{KAR}$  and the public parameters  $PK^{KAR}$  of  $\Pi_{CP-ABE}^{KAR}$ . Then it chooses a collision-resistant hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ . Finally, it sets the public parameters  $PK = (PK^{KAR}, H)$ , and the master secret key  $MK = MK^{KAR}$ .

**Enc**( $PK, M, \mathbb{A}$ ): To encrypt a message  $M \in G_T$  with an access structure  $\mathbb{A} = \bigwedge_{i \in W} i$  the encryption algorithm publishes  $CT = \text{Enc}^{KAR}(PK^{KAR}, W, M)$ .

**KeyGen**( $PK, MK, \omega, ID$ ): To generate a private key for attribute set  $\omega \subseteq \mathcal{U}$  and a user identity  $ID \in \mathcal{I}$  the following steps are taken. First, compute the identity binary string  $L_{ID} = H(ID)$  and store the tuple of  $\langle ID, L_{ID} \rangle$  into an internal list in the Trace algorithm. Then, a dummy attribute set  $\omega_{ID}$  is generated by adding  $(n+k)$ -th attribute if  $k$ -th digit of  $L_{ID}$  equals to 1. Finally, it outputs  $sk_{\omega, ID} = \text{KeyGen}^{KAR}(PK^{KAR}, MK^{KAR}, \omega \cup \omega_{ID})$ .

**Dec**( $PK, CT, sk_{\omega, ID}$ ): Suppose that a ciphertext,  $E$ , is encrypted with a set of attribute  $W$  and a private key has an access right of the attribute set  $\omega$ , where  $W \subseteq \omega$ .

Then, the message is recovered as  $M = \text{Dec}^{KAR}(PK^{KAR}, E_W, sk_{\omega, ID})$ .

**Trace**( $PK, sk'$ ) Let  $sk' = (\omega', \{D'_i\}_{i \in \omega'}, \{D'_i\}_{i \in \mathcal{U}' \setminus \omega'})$  be a valid decryption key. It means that  $\prod_{i \in \omega'} e(D'_i, h_i) \prod_{i \in \mathcal{U}' \setminus \omega'} e(D'_i, h_{n+i}) = Y$ . Then, reconstruct the user identity binary string  $L_{ID'} \in \{0, 1\}^l$  by setting  $k$ -th digit to 1 if  $(n+k) \in \omega'$ ; otherwise 0. Next, search the internal list for a tuple  $\langle ID, L_{ID} \rangle$  where  $L_{ID} = L_{ID'}$  and reveal the corresponding  $ID$  as the identity of the traitor.

**Theorem 3.** *If the DBDH assumption holds, our Traceable CP-ABE scheme defined in Sect. 3 is secure in the sense of Definition 2.*

*Proof.* The proof is similar to that for Theorem 1, and hence, we omit it.

**Theorem 4.** *Our Traceable CP-ABE scheme is secure against Key-Abuse Attack (in the sense of Definition 3) in the generic group model.*

*Proof.* The proof is similar to that for Theorem 2, and hence, we omit it.

## 7 Conclusion

In this paper, we investigated an important property in ABE schemes, which we call as the “key-delegation abuse”. When an ABE scheme is not key-delegation abuse resistant, it means that the private keys that the users have will allow those users to generate new set of private keys without the need of the trusted

authority’s involvement. To be more specific, the new *derivative keys* can be generated for attribute set  $\omega'$  from a private key set for  $\omega$ , if  $\omega' \subset \omega$ . We outlined some severeness of this situation in practice, and we also pointed out that the existing schemes in the literature suffer from this problem. It is indeed interesting that this issue has not been well studied in the literature despite its importance for the adoption of ABE in the real situation. In this work, we proposed a security notion for the key-delegation abuse property and presented a new CP-ABE scheme that is key-delegation abuse resistant. We also proved the security of the scheme in both of standard selective CPA model and the proposed model. Additionally, we also presented an extension of our CP-ABE scheme to a traceable CP-ABE scheme, which will allow the “traitors” to be traced in the system. Our scheme is *the first* CP-ABE scheme that is key-delegation abuse resistant. For the future work, it will be interested to construct a key-delegation abuse resistant ABE scheme that is based on standard hardness assumption.

**Acknowledgement.** This work is partially supported by ARC Project (DP130101383).

## Appendix A Proof of Theorem 2

*Proof.* We consider two random encodings  $\psi_0, \psi_1$  of the additive group  $\mathbb{Z}_p$  respectively, that is injective maps  $\psi_0, \psi_1 : \mathbb{Z}_p \rightarrow \{0, 1\}^L$ , where  $L > 3 \log(p)$ . We write  $G = \psi_0(x) : x \in \mathbb{Z}_p, G_T = \psi_1(x) : x \in \mathbb{Z}_p$ . We are given oracles to compute the induced group action on  $G, G_T$  and an oracle to compute a non-degenerate bilinear map  $e : G \times G \rightarrow G_T$ . We refer to  $G$  as a generic bilinear group.

We now proceed with the proof, following the standard approach for generic groups with  $\psi_0, \psi_1, G, G_T$  defined as above. Let  $g = \psi_0(1)$ (we will write  $g^x$  to denote  $\psi_0(x)$ ), and  $e(g, g)^x$  to denote  $\psi_1(x)$ .

For any generic-group adversary, the security game against *key-delegation abuse* is considered carried out by a simulator as follows. For each group element seen or created by the adversary, this simulator keeps track of its discrete logarithm by means of a multivariate rational functions in the following indeterminate formal variables:

$$\sum = \{v, \nu\} \cup \{t_i\}_{i \in \mathcal{U}} \cup \{x_i^{(j)}\}_{i \in \mathcal{U}, j \in [q]}.$$

The simulation also associates each group element with some rational function. For each distinct rational function in its collection, it inputs the value of the rational function to corresponding encoding  $\psi_0$  or  $\psi_1$  and gives the result to the adversary as the encoding of that particular group element. The functions are associated with the group elements in the simulation as follows:

First, we suppose  $g_0 = g^\nu, h_0 = g^\nu$ .

- Public parameters  $PK$  generated by Setup  $PK = (h_1, \dots, h_{2n}, Y)$ .
  1.  $\{\nu t_i\}_{i \in \mathcal{U}}$ , representing  $h_i = h_0^{t_i} = g^{\nu t_i}$ .

- 2.  $\{\nu t_{n+i}\}_{i \in \mathcal{U}}$ , representing  $h_i = h_0^{t_{n+i}} = g^{\nu t_i}$ .
- Private key components given by KeyGen. Let  $sk_j$  be the  $j$ -th queried private key for the attribute set  $\omega_j$ .

$$sk_j = \left( \omega_j, \{D_i^{(j)} = g_0^{\frac{x_i^{(j)}}{t_i}}\}_{i \in \omega}, \{D_i^{(j)} = g_0^{\frac{x_i^{(j)}}{t_{n+i}}}\}_{i \in \mathcal{U} \setminus \omega} \right)$$

- 1.  $\{\frac{\nu}{t_i} x_i^{(j)}\}_{j \in [q], i \in \omega_j}$ , representing  $D_i^{(j)} = g_0^{\frac{x_i^{(j)}}{t_i}}$ .
- 2.  $\{\frac{\nu}{t_{n+i}} x_i^{(j)}\}_{j \in [q], i \in \mathcal{U} \setminus \omega_j}$ , representing  $D_i^{(j)} = g_0^{\frac{x_i^{(j)}}{t_{n+i}}}$ .

We note that in the actual game, the values of the formal variables are chosen uniformly at random in  $\mathbb{Z}_p$ . Two distinct functions may in that case evaluate to the same value. The simulation is faithful to the standard interaction in a generic group, except in the event that two of the distinct functions evaluate to the same value on a random assignment to the formal variables. For any two distinct functions of the form listed above, the probability of this happening is at most  $O(q)/p$ , since the degree of distinct multivariate polynomials is at most  $O(q)$ . Since this probability is negligible, we ignore this case.

Now the adversary outputs a purported new private key  $sk^*$  for a new attribute set  $\omega^*$  with a suitable decryption algorithm  $Dec^*(\cdot)$ . We first observe that to decrypt a ciphertext  $CT$  encrypted with an access structure  $\mathbb{A} = \bigwedge_{i \in W} i$ , where  $W$  is equal to or a subset of  $\omega^*$ . The new private key  $sk^*$  should contain a group element for each attribute to pair the corresponding group element  $E_i$  (or  $E'_i$ ) in the ciphertext in bilinear map for  $Y^r = e(g_0, h_0)^{\alpha r}$ . We denote these group elements by  $D_i^*$  and the necessary structure of the new private key can be presented as  $(\omega^*, \{D_i^*\}_{i \in \mathcal{U}})$ . On the other hand, as long as the new private key satisfies the winning conditions the adversary can construct the new key  $sk^*$  the way it wants to make it look different, which means the adversary can construct the new private key component  $D_i^*$  using a linear combination of the functions listed above.

Here, we note that if the adversary tries to construct  $D_i^*$  using any functions other than  $D_i^{(j)}$ , then using this part of  $D_i^*$  in bilinear map will result in meaningless group element in  $G_T$  for decryption, which also needs to be eliminated by computing it separately; since it needs to be eliminated afterwards, we do not include it in following discussion.

WLOG, we assume the new private key  $sk^*$  contains the following least structure for each attribute  $i$ :

$$\begin{aligned} D_i^* &= \pi_i(D_i^{(1)}, \dots, D_i^{(q)}) := (D_i^{(1)})^{\beta_{i,1}} (D_i^{(2)})^{\beta_{i,2}} \dots (D_i^{(q)})^{\beta_{i,q}} \\ &= u_0^{\frac{1}{t_i} \sum_{i \in \omega_j} \beta_{i,j} x_i^{(j)} + \frac{1}{t_{n+i}} \sum_{i \notin \omega_j} \beta_{i,j} x_i^{(j)}} \end{aligned}$$

where  $\pi_i(D_i^{(1)}, \dots, D_i^{(q)}) := (D_i^{(1)})^{\beta_{i,1}}(D_i^{(2)})^{\beta_{i,2}} \dots (D_i^{(q)})^{\beta_{i,q}}$  represents a function in  $G$  using components  $D_i^{(j)}$  from queried private keys.

Then we can represent  $D_i^*$  as  $\frac{v}{t_i} \sum_{i \in \omega_j} \beta_{i,j} x_i^{(j)} + \frac{v}{t_{n+i}} \sum_{i \notin \omega_j} \beta_{i,j} x_i^{(j)}$ .

To win in the game,  $D_i^*$  needs to meet following conditions:

1.  $\sum_{i \in \mathcal{U}} \sum_{j \in [q]} \beta_{i,j} x_i^{(j)} = \alpha$ .
2.  $\forall i \in \omega^*, \sum_{i \notin \omega_j} \beta_{i,j} x_i^{(j)} = 0$ .
3.  $\forall i \notin \omega^*, \sum_{j \notin \omega_j} \beta_{i,j} x_i^{(j)} \neq 0$  and  $\sum_{i \in \omega_j} \beta_{i,j} x_i^{(j)} = 0$ .

The rest of our proof proceeds by assuming the new private key  $sk^*$  satisfies the conditions above, and obtaining a contradiction: that the new private key  $sk^*$  can be used to decrypt ciphertexts encrypted with a queried attribute set  $\omega_j$  which contradicts the second condition in the security game's definition.

Considering condition 1,  $\sum_{i \in \mathcal{U}} \sum_{j \in [q]} \beta_{i,j} x_i^{(j)} = \alpha$ . Since  $\sum_i x_i^{(j)} = \alpha$  for  $j \in [q]$  and  $x_i^{(j)}$  is uniformly random chosen in  $\mathbb{Z}_p$ , we have

$$\beta_{1,j} = \beta_{2,j} = \dots = \beta_{n,j}.$$

We denote them by  $\beta_j$ .

Considering condition 2, for all  $i \in \omega^*, \sum_{i \notin \omega_j} \beta_{i,j} x_i^{(j)} = \sum_{j \notin \omega_j} \beta_j x_i^{(j)} = 0$ . Since  $x_i^{(j)}$  is uniformly random chosen in  $\mathbb{Z}_p$ , it can be concluded that

$$\text{if } \exists i \in \omega^* \text{ and } i \notin \omega_j, \beta_j = 0$$

which is equivalent to

$$\text{if } \beta_j \neq 0, \omega^* \subseteq \omega_j.$$

Considering condition 3, for all  $i \notin \omega^*, \sum_{j \notin \omega_j} \beta_{i,j} x_i^{(j)} = \sum_{j \notin \omega_j} \beta_j x_i^{(j)} \neq 0$  and  $\sum_{i \in \omega_j} \beta_{i,j} x_i^{(j)} = \sum_{i \in \omega_j} \beta_j x_i^{(j)} = 0$ . Since  $x_i^{(j)}$  is uniformly random chosen in  $\mathbb{Z}_p$ , it can be concluded that

$$\text{if } \exists i \notin \omega^* \text{ and } i \in \omega_j, \beta_j = 0$$

which is equivalent to

$$\text{if } \beta_j \neq 0, \omega_j \subseteq \omega^*.$$

So  $\omega^*$  equals to a queried attribute set  $\omega_j$ , which results in either the adversary cannot generate a new key as  $\omega^* \neq \omega_j$  for  $j \in [q]$  or the new key will be able to decrypt ciphertexts encrypted with  $\omega_j$  as well since only one queried private key  $sk_j$  can be used. Therefore, our assumptions cannot be true. The adversary cannot successfully generate a new private key  $sk^*$  to win the game.  $\square$

## References

1. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
2. 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)
3. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. pp. 456–465. CCS 2007, NY, USA. ACM, New York (2007)
4. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
5. Goyal, V., Sahai, A., Lu, S., Waters, B.: Black box accountable authority identity-based encryption (2008)
6. Goyal, V., Sahai, A., Pandey, O., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of ACM CCS06, pp. 89–98 (2006)
7. Hinek, M.J., Jiang, S., Safavi-Naini, R., Shahandashti, S.F.: Attribute-based encryption with key cloning protection. Cryptology ePrint Archive, Report 2008/478 (2008). <http://eprint.iacr.org/>
8. Li, J., Kui Ren, K.K.: A2BE: Accountable attribute-based encryption for abuse free access control. IACR Cryptology ePrint Archive (2009)
9. Bethencourt, J., Amit Sahai, B.W.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334. IEEE (2007)
10. Katz, J., Sahai, A.B.W.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Crypt.* **26**(2), 191–224 (2013)
11. 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)
12. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, pp. 386–390. NY, USA. ACM, New York (2011)
13. Liu, Z., Cao, Z., Wong, D.S.: Blackbox traceable cp-abe: how to catch people leaking their keys by selling decryption devices on ebay, pp. 475–486, IACR Cryptology ePrint Archive (2013)
14. Liu, Z., Cao, Z., Wong, D.S.: White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. *Nformaon Forn and Ry Ranaon* on **8**(1), 76–88 (2013)
15. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellare, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008)
16. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden ciphertext policies. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **92**(1), 22–32 (2009)

17. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. *Cryptology ePrint Archive, Report 2007/323* (2007). <http://eprint.iacr.org/>
18. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
19. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

# Chosen Ciphertext Secure Attribute-Based Encryption with Outsourced Decryption

Cong Zuo, Jun Shao<sup>(✉)</sup>, Guiyi Wei, Mande Xie, and Min Ji

School of Computer and Information Engineering, Zhejiang Gongshang University,  
Hangzhou 310018, People's Republic of China  
zuocong10@gmail.com, chn.junshao@gmail.com,  
{weigy,xiemd,jimin}@zjgsu.edu.cn

**Abstract.** Although attribute-based encryption (ABE) is a useful cryptographic tool to realize expressive access policy on ciphertexts, it is not quite suitable for mobile devices. The root cause lies in that the size of the ciphertext and the decryption cost are usually proportional to the complexity of the access policy. To solve this problem, a variant of ABE, named attribute-based encryption with outsourced decryption (OD-ABE), was proposed by Green, Hohenberger and Waters. Especially, OD-ABE allows a proxy with a transformation key delegated from the user to simplify ABE ciphertexts satisfied by the user's attributes. On the other hand, this transformation also makes it tricky to design an OD-ABE scheme achieving the CCA security that is generally considered as the standard notion of security for a cryptosystem. However, the existing OD-ABE schemes only achieve the re-randomizable (replayable) CCA security. In this paper, we propose the CCA security model for OD-ABE and a concrete scheme secure in our proposed security model. We believe that this improvement on the security of OD-ABE will lead to a wider spectrum of applications.

**Keywords:** Attribute-based encryption · Chosen ciphertext security · Outsourced decryption · Random oracle model

## 1 Introduction

Cloud storage is becoming the most popular solution for the data sharing due to its cost-effective property. Since the cloud server is not always fully trusted, the fine-grained access control on the encrypted data is quite desired from the viewpoint of users. Attribute-based encryption (ABE), introduced by Sahai and Waters [18], is a promising solution for this requirement. There exist two types of ABE schemes: key-policy ABE (KP-ABE) [7] and ciphertext-policy ABE (CP-ABE) [1, 4, 11, 19]. In the former type, the access policy is embedded into the user's private key. While in the latter one, the access policy is embedded into the ciphertext. Only if the user's attributes satisfy the access policy, the ciphertext can be decrypted by the user's private key.

Although ABE is a very useful cryptographic tool to realize fine-grained access control, the inefficient performance is its Achilles' Heel. In particular, the size of ABE ciphertext and decryption cost are usually proportional to the complexity of the access policy. This impedes the use of ABE in mobile devices due to the limited resources [9]. However, it has been reported that mobile exceeded PC Internet usage in early 2014 [16]. To solve this conflict, Green et al. [8] proposed the concept of attribute-based encryption with outsourced decryption (OD-ABE), where it allows a proxy (such as the cloud) with a transformation key to transform ABE ciphertexts into simple and constant size ciphertexts, while the proxy cannot obtain the corresponding plaintext. By using OD-ABE, the most of the decryption cost on ABE ciphertexts can be moved from the user to the cloud.

It is well-known that the chosen ciphertext security is generally considered as the right security notion for a cryptosystem. However, none of the existing OD-ABE schemes achieve CCA security. The traditional CCA security of public key encryption guarantees that any ciphertext cannot be malleable. Nevertheless, the ciphertext transformation is expected as a regular functionality in OD-ABE, which makes the definition of CCA security tricky. In this paper, we will propose the CCA security for OD-ABE by following the spirit of the traditional CCA security as close as we can. Furthermore, we will propose a concrete CCA secure OD-ABE scheme by applying the CHK [3] and FO [5] techniques on the ABE scheme in [4]. As we can see later, the design approach used in this paper can be applied to other ABE schemes, such as [1, 6, 11, 15, 20–22].

## 1.1 Related Work

The history of OD-ABE is quite simple compared to that of ABE. The concept of OD-ABE is proposed by Green et al. [8]. The notions of CPA security and CCA security, along with several CPA-secure and RCCA-secure OD-ABE scheme, are also proposed in [8]. After that, many research efforts have been dedicated to design OD-ABE schemes [10, 12–14, 17]. However, none of them aims to promote the security from RCCA security to CCA security.

We note that OD-ABE has the similar situation with proxy re-encryption (PRE) [2], where a proxy with a re-encryption key can also do ciphertext transformation. However, the intending decryptor changes after the ciphertext transformation in PRE, which makes the CCA security of PRE cannot be used in OD-ABE.

## 1.2 Paper Organization

The remaining paper is organized as follows. In Sect. 2, the definition and CCA security models of attribute-based encryption with outsourced decryption are introduced. In Sect. 3, we present our proposal for CCA secure attribute-based encryption with outsourced decryption. In what follows, we give the security proof of our proposal in our proposed CCA security model. At last, we conclude the paper in Sect. 5.



## 2 Definitions

### 2.1 Attribute-Based Encryption with Outsourced Decryption

**Definition 1 (Attribute-Based Encryption with Outsourced Decryption (OD-ABE)).** An attribute-based encryption scheme with outsourced decryption OD-ABE is a tuple of probabilistic polynomial time (PPT) algorithms  $(\text{KeyGen}, \text{Ext}, \text{OKGen}, \text{Enc}, \text{TDec}, \text{Dec})$ .

- $\text{KeyGen}(\lambda) \rightarrow (\text{mpk}, \text{msk})$ . On input a security parameter  $\lambda$ , the key generation algorithm  $\text{KeyGen}$  outputs the master public/private key pair  $(\text{mpk}, \text{msk})$  of the private key generator (PKG). Note that the master public key  $\text{mpk}$  is included in all of the following algorithms implicitly.
- $\text{Ext}(\text{msk}, I_k) \rightarrow d$ . On input the master private key pair  $\text{msk}$  and an attribute set (resp. access structure)  $I_k$ , the key extraction algorithm  $\text{Ext}$  outputs a private key  $d$  corresponding to  $I_k$ .
- $\text{OKGen}(d) \rightarrow (\text{tk}, \text{rk})$ . On input a private key  $d$  corresponding to  $I_k$ , the outsourced decryption key generation algorithm  $\text{OKGen}$  outputs an outsourced decryption key  $ok$  containing a transformation/retrieving key pair  $(\text{tk}, \text{rk})$  corresponding to  $I_k$ .
- $\text{Enc}(I_e, m) \rightarrow C$ . On input an access structure (resp. attribute set)  $I_e$  and a message  $m$  from the message space  $\mathcal{M}$ , the encryption algorithm  $\text{Enc}$  outputs a ciphertext  $C$  corresponding to  $I_e$ .
- $\text{TDec}(\text{tk}, C) \rightarrow C_t$ . On input a transformation key  $\text{tk}$  corresponding to  $I_k$  and a ciphertext  $C$  encrypted under  $I_e$ , the outsourced decryption  $\text{TDec}$  outputs a transformed ciphertext  $C_t$  if  $f(I_e, I_k) = 1$ <sup>1</sup>;  $\perp$  otherwise.
- $\text{Dec}(\text{dk}, C) \rightarrow m$ . There are two cases in this algorithm, since there are two types of ciphertexts: One is from  $\text{Enc}$ , and the other is from  $\text{TDec}$ .
  - If the input ciphertext  $C$  is from  $\text{Enc}$  encrypted under  $I_e$ , the decryption key  $\text{dk}$  is the private key  $d$  corresponding to  $I_k$ . It outputs  $m$  if  $f(I_e, I_k) = 1$ ; or  $\perp$  otherwise.
  - If the input ciphertext  $C$  is from  $\text{TDec}$  encrypted under  $I_e$ , the decryption key  $\text{dk}$  is the retrieving key  $\text{rk}$  corresponding to  $I_k$ . It outputs  $m$  if  $f(I_e, I_k) = 1$ ; or  $\perp$  otherwise.

**Correctness.** For any message  $m$  in the message space  $\mathcal{M}$ ,  $(\text{mpk}, \text{msk}) \leftarrow \text{KeyGen}(\lambda)$ ,  $d \leftarrow \text{Ext}(\text{msk}, I_k)$ ,  $(\text{tk}, \text{rk}) \leftarrow \text{OKGen}(d)$ , the following conditions must hold:

$$\text{Dec}(d, \text{Enc}(I_e, m)) = m, \text{ if } f(I_e, I_k) = 1.$$

and

$$\text{Dec}(\text{rk}, \text{TDec}(\text{tk}, \text{Enc}(I_e, m))) = m, \text{ if } f(I_e, I_k) = 1.$$

*Remark 1* (Differences Between our Definition and that in [8]). The main difference between our definition and that in [8] is that we explicitly extract the key (retrieving key) used to decrypt the transformed ciphertexts from the private

<sup>1</sup> If  $I_e$  is satisfied by  $I_k$ , we have  $f(I_e, I_k) = 1$ ; otherwise, we have  $f(I_e, I_k) = 0$ .

key  $d$ . It is because that the retrieving key is usually much shorter than the private key, and the user may only store the retrieving key but not the private key in the mobile device for saving storage. This small difference makes the OD-ABE scheme closer to underlying mobile device application. We note that the definition given in [13] has the similar setting.

## 2.2 Chosen Ciphertext Security for Attribute-Based Encryption with Outsourced Decryption

As we noticed before, in attribute-based encryption with outsourced decryption, there are two formats of ciphertexts. One is the original ciphertext generated from algorithm **Enc**, and the other is the transformed ciphertext computed from algorithm **TDec**. Hence, there are two cases in this definition.

*The Challenge Ciphertext is an Original Ciphertext.*

**Setup:** The challenger  $\mathcal{C}$  runs **KeyGen**( $\lambda$ ) to get the master public/private key pair  $(mpk, msk)$ . After that,  $\mathcal{C}$  returns  $mpk$  to the adversary  $\mathcal{A}$ , while keeping  $msk$  secret.

**Phase 1:**  $\mathcal{A}$  issues queries  $q_1, \dots, q_{n_1}$  where query  $q_i$  is one of:

- Private key generation oracle  $\mathcal{O}_{sk}$ : On input  $I_k$ ,  $\mathcal{C}$  returns the corresponding private key  $d$ .
- Transformation key generation  $\mathcal{O}_{tk}$ : On input  $I_k$  and an index  $i$ ,  $\mathcal{C}$  returns the corresponding transformation key  $tk$ .
- Retrieving key generation  $\mathcal{O}_{rk}$ : On input  $I_k$  and an index  $i$ ,  $\mathcal{C}$  returns the corresponding retrieving key  $rk$ . Note that if  $tk$  and  $rk$  are corresponding to the same private key and the corresponding indexes are the same, then  $tk$  and  $rk$  are considered as a pair generated from **OKGen**.
- Outsourced decryption oracle  $\mathcal{O}_{od}$ : On input  $(C_i, I_k)$  and an index  $i$ ,  $\mathcal{C}$  returns **TDec**( $tk, C_i$ ), where  $C_i$  is an original ciphertext, and  $tk$  is a transformation key corresponding to  $I_k$  and index  $i$ .
- Decryption oracle  $\mathcal{O}_{dec}$ : On input  $(C_i, I_k)$  and the optional input index  $i$ ,  $\mathcal{C}$  returns **Dec**( $dk, C_i$ ), where  $dk$  is a decryption key corresponding to  $I_k$ , and it would be a private key  $d$  if the input ciphertext  $C_i$  is an original ciphertext, or  $rk$  corresponding to  $I_k$  and index  $i$  if  $C_i$  is a transformed ciphertext. Note that if the transformed ciphertext is not generated from  $\mathcal{O}_{od}$  or **TDec** with the transformation key  $tk$  from  $\mathcal{O}_{tk}$ , the challenger simply returns  $\perp$ .

These queries may be asked adaptively, that is, each query  $q_i$  may depend on the replies to  $q_1, \dots, q_{i-1}$ .

**Challenge:** Once  $\mathcal{A}$  decides that Phase 1 is over, it outputs two equal length plaintexts  $m_0^*, m_1^*$  from the message space  $\mathcal{M}$ , and an attribute set (resp. access structure)  $I_e^*$  on which it wishes to challenge. There are two restrictions on  $I_e^*$ , (i) For all  $I_k$ 's queried to  $\mathcal{O}_{sk}$ ,  $f(I_e^*, I_k) \neq 1$ . (ii) For all  $I_k$ 's queried to both  $\mathcal{O}_{tk}$  and  $\mathcal{O}_{rk}$ ,  $f(I_e^*, I_k) \neq 1$ .  $\mathcal{C}$  picks a random bit  $\mathbf{b} \in \{0, 1\}$  and sets  $C^* = \text{Enc}(I_e^*, m_{\mathbf{b}}^*)$ . It sends  $C^*$  as the challenge to  $\mathcal{A}$ .

**Phase 2:** Same as Phase 1 but the challenger will output  $\perp$  in the following cases, where the adversary can trivially obtain the message  $m$  corresponding to  $C^*$ .

- $\mathcal{O}_{sk}$ :  $f(I_e^*, I_k) = 1$ .
- $\mathcal{O}_{tk}$ :  $f(I_e^*, I_k) = 1$  and  $(I_k, \mathbf{i})$  has been queried to  $\mathcal{O}_{rk}$ .
- $\mathcal{O}_{rk}$ :  $f(I_e^*, I_k) = 1$  and  $(I_k, \mathbf{i})$  has been queried to  $\mathcal{O}_{tk}$ .
- $\mathcal{O}_{od}$ :  $C_i = C^*$ ,  $f(I_e^*, I_k) = 1$ , and  $(I_k, \mathbf{i})$  has been queried to  $\mathcal{O}_{rk}$ .
- $\mathcal{O}_{dec}$ : There are the following cases.
  - $C_i = C^*$  and  $f(I_e^*, I_k) = 1$ .
  - $C_i \leftarrow \text{TDec}(tk, C^*)$  and  $f(I_e^*, I_k) = 1$ , where  $tk \leftarrow \mathcal{O}_{tk}(I_k, \mathbf{i})$ .
  - $C_i \leftarrow \mathcal{O}_{od}(C^*, I_k, \mathbf{i})$  and  $f(I_e^*, I_k) = 1$ .

**Guess:** Finally, the adversary  $\mathcal{A}$  outputs a guess  $\mathbf{b}' \in \{0, 1\}$  and wins the game if  $\mathbf{b} = \mathbf{b}'$ .

The advantage  $\text{Adv}_{\text{OD-ABE}}^{\text{CCA-O}}(\lambda)$  is defined as  $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$ . The scheme OD-ABE is said to be *CCA-O* secure<sup>2</sup> if all efficient adversaries  $\mathcal{A}$  specified as above, the advantage  $\text{Adv}_{\text{OD-ABE}}^{\text{CCA-O}}(\lambda)$  is negligible.

*The Challenge Ciphertext is a Transformed Ciphertext.*

**Phase 1:** Identical to that in the challenge original ciphertext case.

**Challenge:** Once  $\mathcal{A}$  decides that Phase 1 is over, it outputs two equal length plaintexts  $m_0^*$ ,  $m_1^*$  from the message space, an attribute set (resp. access structure)  $I_e^*$ , an access structure (resp. attribute set)  $I_k^*$ , and an index  $\mathbf{i}^*$ . There is one restriction on  $(I_k^*, \mathbf{i}^*)$ :  $(I_k^*, \mathbf{i}^*)$  has never been queried to  $\mathcal{O}_{rk}$ .  $\mathcal{C}$  picks a random bit  $\mathbf{b} \in \{0, 1\}$  and sets  $C^* = \text{TDec}(tk^*, \text{Enc}(I_e^*, m_{\mathbf{b}}^*))$ , where  $tk^*$  is the output of  $\mathcal{O}_{tk}$  with input  $(I_k^*, \mathbf{i}^*)$ . It sends  $C^*$  as the challenge to  $\mathcal{A}$ .

**Phase 2:** Almost the same as that in Phase 1 but with the following constraints.

- $\mathcal{O}_{rk}$ :  $(I_k, \mathbf{i}) = (I_k^*, \mathbf{i}^*)$ .
- $\mathcal{O}_{dec}$ :  $C_i = C^*$  and  $(I_k, \mathbf{i}) = (I_k^*, \mathbf{i}^*)$ .

**Guess:** Identical to that in the challenge original ciphertext case.

The advantage  $\text{Adv}_{\text{OD-ABE}}^{\text{CCA-T}}(\lambda)$  is defined as  $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$ . The scheme OD-ABE is said to be *CCA-T* secure<sup>3</sup> if all efficient adversaries  $\mathcal{A}$  specified as above, the advantage  $\text{Adv}_{\text{OD-ABE}}^{\text{CCA-T}}(\lambda)$  is negligible.

*Remark 2* (Difference Between our Security Model and that in [8]). Compared to the RCCA security model in [8], the proposed security model has the following differences.

- We allow the adversary to query the decryption oracle with the ciphertext whose corresponding plaintext belongs to the challenge messages  $\{m_0^*, m_1^*\}$ , while we only disallow the adversary to issue the queries that can help it to win the game trivially. This difference makes our definition closer to the traditional CCA security definition.

<sup>2</sup> O stands for original ciphertexts.

<sup>3</sup> T stands for transformed ciphertexts.

- There two different security games in our definition corresponding to the two different formats of ciphertexts, which makes our definition closer to the nature of attribute-based encryption with outsourced decryption.
- As mentioned in Remark 1, the user may only store the retrieving key  $rk$  but not the private key  $d$  in the mobile device for some reasons (such as saving storage). In this case, the retrieving key might be obtained by the adversary for some incidents. For example, the mobile device is stolen or infected by computer virus. This kind of attack is captured by  $\mathcal{O}_{rk}$ , which makes the OD-ABE schemes proven secure in our security model have a wider spectrum of applications.

Due to the above differences, the scheme proposed in [8] cannot be proven secure in our security model.

*Remark 3 (CPA Security and Selective Security).* We can define CPA security as CCA security. In particular, we can obtain the CPA security game only if we remove the decryption oracle in the CCA security game.

Furthermore, if the adversary makes a decision on the challenge value  $I_e^*$  (and challenge index  $i^*$  for CCA-T security) before the **Setup** stage, we have the selective security.

### 2.3 Bilinear Groups

We say that a (multiplicative) cyclic group  $\mathbb{G}$  with a prime order  $q$  is a bilinear group if it satisfies the following properties.

- The group action in  $\mathbb{G}$  can be computed efficiently.
- There exists an admissible bilinear map  $\hat{e}$  as follows.
  - $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ , where  $\mathbb{G}_T$  is a (multiplicative) cyclic group with prime order  $q$ .
  - For all  $a, b \in \mathbb{Z}_q$ , the generator  $g$  of  $\mathbb{G}$ ,  $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ .

We denote **BSetup** as an algorithm that, on input the security parameter  $\lambda$ , outputs the parameters for a bilinear map as  $(q, g, \mathbb{G}, \mathbb{G}_T, \hat{e})$ , where  $q \in \Theta(2^\lambda)$ .

### 2.4 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Let  $(q, g, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \mathbf{BSetup}(\lambda)$ , and  $a, b, c, z$  be random elements from  $\mathbb{Z}_q^*$ . The decisional Bilinear Diffie-Hellman (DBDH) problem is to distinguish between the tuples of the form  $(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc})$  and  $(g, g^a, g^b, g^c, \hat{e}(g, g)^z)$ . An algorithm  $\mathcal{A}$  has advantage  $\epsilon$  in solving DBDH problem if

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^z) = 1]| \geq \epsilon.$$

We say the DBDH assumption holds in  $\mathbb{G}$  and  $\mathbb{G}_T$  if  $\epsilon$  is negligible.

### 2.5 Modified Decisional Diffie-Hellman (MDDH) Assumption

Let  $(q, g, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \text{BSetup}(\lambda)$ , and  $a, b, c$  be random elements from  $\mathbb{Z}_q^*$ . The modified decisional Diffie-Hellman (MDDH) problem is to distinguish between the tuples of the form  $(g, g^a, \hat{e}(g, g)^b, \hat{e}(g, g)^{ab})$  and  $(g, g^a, \hat{e}(g, g)^b, \hat{e}(g, g)^c)$ . An algorithm  $\mathcal{A}$  has advantage  $\epsilon$  in solving MDDH problem if

$$|\Pr[\mathcal{A}(g, g^a, \hat{e}(g, g)^b, \hat{e}(g, g)^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, \hat{e}(g, g)^b, \hat{e}(g, g)^c) = 1]| \geq \epsilon.$$

We say the MDDH assumption holds in  $\mathbb{G}$  and  $\mathbb{G}_T$  if  $\epsilon$  is negligible.

## 3 Our Proposal

Before giving our new OD-ABE scheme, we would like to present some intuitions of our construction. The main idea to realize CCA security for non-transformable public key encryption is to allow the decryptor to have the ability to check the validity of the ciphertext. In the OD-ABE scheme, the situation is a little bit more complex than the non-transformable public key encryption. It additionally needs two other decryptions. One is the ciphertext’s transformation by the proxy, and the other is the transformed ciphertext’s decryption by the decryptor. For the verifiability of the former, we will use the CHK transformation [3], which supports the public verifiability. Hence, the proxy can check the validity before doing transformation. For the latter one, the FO transformation [5] is applied instead of the CHK transformation. Although the ciphertext has been transformed, the decryptor can still check the validity of the transformed ciphertext by using the FO technique [5]. Combining the above ideas, we have our CCA-secure OD-ABE scheme. Our proposal is based on the ABE scheme due to Cheung and Newport [4], while the ideas used in this paper can be also applied to other ABE schemes, such as [1, 6, 11, 15, 20–22].

- **KeyGen:** We let the set of attributes be  $N = \{1, \dots, n\}$  for some nature number  $n$ . On input a security parameter  $\lambda$ , the PKG runs  $\text{BSetup}(\lambda)$  to obtain  $(q, g, \hat{e}, \mathbb{G}, \mathbb{G}_T)$ . It selects random elements  $y, t_1, \dots, t_{3n}$  from  $\mathbb{Z}_q$ . Let  $Y := \hat{e}(g, g)^y$  and  $T_i = g^{t_i}$  for each  $i \in \{1, \dots, 3n\}$ . It also selects hash functions  $H_1 : \{0, 1\}^{2\ell} \rightarrow \mathbb{Z}_q$ ,  $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^\ell$ ,  $H_3 : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ , and  $H_4 : \{0, 1\}^* \rightarrow \mathbb{G}$ , where  $\ell$  is the bit length of messages. The resulting master public key is

$$mpk := (\hat{e}, g, q, \mathbb{G}, \mathbb{G}_T, Y, T_1, \dots, T_{3n}, H_1, H_2, H_3, H_4),$$

and the corresponding master secret key is

$$msk := (y, t_1, \dots, t_{3n}).$$

- **Ext:** On input the master private key  $msk$  and an attribute set  $S \subseteq \{1, \dots, n\}$ , the PKG selects random  $r_i$  from  $\mathbb{Z}_q$  for  $i \in [1, n]$  and sets  $r := \sum_{i=1}^n r_i \bmod q$ , and  $\hat{D} = g^{y-r}$ . It also computes

$$D_i = \begin{cases} g^{r_i/t_i}, & \text{if } i \in S \\ g^{r_i/t_{n+i}}, & \text{if } i \notin S \end{cases} \quad \text{and} \quad F_i = g^{r_i/t_{2n+i}}, \quad i \in [1, n].$$

The resulting private key corresponding to the attribute set  $S$  is

$$d = (\hat{D}, \{(D_i, F_i) | i \in [1, n]\}).$$

- **OKGen**: On input a private key  $d = (\hat{D}, \{(D_i, F_i) | i \in [1, n]\})$  corresponding to  $S$ , the user selects a random element  $z$  from  $\mathbb{Z}_q$ . The resulting transformation key is

$$tk = (\hat{D}^z, \{(D_i^z, F_i^z) | i \in [1, n]\})$$

and the retrieving key

$$rk = z.$$

- **Enc**: On input an AND gate  $W = \bigwedge_{i \in I} \underline{i}$ , where  $I \subseteq \{1, \dots, n\}$ , and  $\underline{i}$  is the attribute  $i$  or its negation  $\neg i$ , the encryptor selects a random  $\mu$  from  $\{0, 1\}^\ell$  and computes  $s = H_1(\mu || m)$ ,  $\tilde{C}_1 = \mu \oplus H_2(Y^s)$ ,  $\tilde{C}_2 = m \oplus H_3(\mu)$ ,  $\hat{C}_1 = g^s$ , and

$$C_i = \begin{cases} T_i^s, & \text{if } i \in I \wedge \underline{i} = i \\ T_{n+i}^s, & \text{if } i \in I \wedge \underline{i} = \neg i \\ T_{2n+i}^s, & \text{if } i \in \{1, \dots, n\} \setminus I \end{cases}$$

At last, the encryptor computes  $\hat{C}_2 = H_4(W || \tilde{C}_1 || \tilde{C}_2 || \hat{C}_1 || C_1 || \dots || C_n)^s$ . The resulting ciphertext is  $CT = (W, \tilde{C}_1, \tilde{C}_2, \hat{C}_1, \hat{C}_2, \{C_i | i \in \{1, \dots, n\}\})$ .

- **TDec**: On input a transformation key  $tk = (\hat{D}^z, \{(D_i^z, F_i^z) | i \in [1, n]\})$  corresponding to  $S$  and a ciphertext  $C$  encrypted under  $W = \bigwedge_{i \in I} \underline{i}$ , the proxy first checks  $\hat{e}(\hat{C}_1, H_4(W || \tilde{C}_1 || \tilde{C}_2 || \hat{C}_1 || C_1 || \dots || C_n)) \stackrel{?}{=} \hat{e}(g, \hat{C}_2)$ , and  $\hat{e}(\hat{C}_1, T_i) \stackrel{?}{=} \hat{e}(g, C_i)$  for  $i \in I$  and  $\underline{i} = i$ ,  $\hat{e}(\hat{C}_1, T_{n+i}) \stackrel{?}{=} \hat{e}(g, C_i)$  for  $i \in I$  and  $\underline{i} = \neg i$ , and  $\hat{e}(\hat{C}_1, T_{2n+i}) \stackrel{?}{=} \hat{e}(g, C_i)$  for  $i \in \{1, \dots, n\} \setminus I$ . If one of them does not hold, the proxy aborts the algorithm; otherwise, it computes  $\tilde{C}_3 = \hat{e}(\hat{C}_1, \hat{D}^z) \cdot \prod_{i \in I} \hat{e}(C_i, D_i^z) \cdot \prod_{i \in \{1, \dots, n\} \setminus I} \hat{e}(C_i, F_i^z)$ . The resulting transformed ciphertext is  $CT' = (\tilde{C}_1, \tilde{C}_2, \tilde{C}_3)$ .
- **Dec**: There are two cases in this algorithm, since there are two types of ciphertexts: One is from **Enc**, and the other is from **TDec**.

- If the input ciphertext  $CT$  is from **Enc** encrypted under  $W = \bigwedge_{i \in I} \underline{i}$ , the decryption key  $dk$  is the private key  $d = (\hat{D}, \{(D_i, F_i) | i \in [1, n]\})$  corresponding to  $S$ . The decryptor first checks the validity of the ciphertext as in **TDec**. If it does not pass, the decryptor aborts the algorithm; otherwise, it computes  $\tilde{C}_3 = \hat{e}(\hat{C}_1, \hat{D}) \cdot \prod_{i \in I} \hat{e}(C_i, D_i) \cdot \prod_{i \in \{1, \dots, n\} \setminus I} \hat{e}(C_i, F_i)$ . Note that if  $S$  satisfies  $W$ , we have that  $\tilde{C}_3 = e(g, g)^{y \cdot s}$ . The decryptor can obtain  $m$  by computing  $\mu = \tilde{C}_1 \oplus H_2(\tilde{C}_3)$  and  $m = \tilde{C}_2 \oplus H_3(\mu)$ . If  $\tilde{C}_1 = \mu \oplus H_2(Y^{H_1(\mu || m)})$  holds, then the decryptor outputs  $m$ ; otherwise, it outputs  $\perp$ .
- If the input ciphertext  $CT'$  is from **TDec** encrypted under  $W$ , the decryption key  $dk$  is the retrieving key  $rk = z$  corresponding to  $S$ . Note that if  $S$  satisfies  $W$ , we have that  $\tilde{C}_3 = e(g, g)^{y \cdot s \cdot z}$ . The decryptor computes  $m$  by computing  $\mu = \tilde{C}_1 \oplus H_2(\tilde{C}_3^{1/z})$  and  $m = \tilde{C}_2 \oplus H_3(\mu)$ . If both of  $\tilde{C}_1 = \mu \oplus H_2(Y^{H_1(\mu || m)})$  and  $Y^{H_1(\mu || m)} = \tilde{C}_3^{1/z}$  hold, then the decryptor outputs  $m$ ; otherwise, it outputs  $\perp$ .

It is easy to see that with the help of outsourced decryption, the decryption cost is reduced from  $3n + 3$  pairings and 1 exponentiation in  $\mathbb{G}$  to only 2 exponentiations in  $\mathbb{G}$ .

**Correctness.** The correctness of the OD-ABE scheme can be easily obtained by the correctness of the scheme in [4] and the scheme in [8]. Hence, we omit it here.

### 4 Security Analysis

Now we will show that our OD-ABE scheme is CCA-secure in the sense of our proposed CCA security.

**Theorem 1.** *The proposed OD-ABE scheme is selective CCA-O secure in the random oracle model if the DBDH assumption holds in  $\mathbb{G}$  and  $\mathbb{G}_T$ .*

*Proof.* We show that if there exists an adversary  $\mathcal{A}$  that can break the selective CCA-O security of the proposed OD-ABE scheme, we can build an algorithm  $\mathcal{B}$  that can solve the DBDH problem, i.e., given the tuple  $(g, \mathbf{A} = g^a, \mathbf{B} = g^b, \mathbf{C} = g^c, \mathbf{Z}) \in \mathbb{G}^4 \times \mathbb{G}_T$ , it decides whether  $\mathbf{Z} = \hat{e}(g, g)^{abc}$ . In particular,  $\mathcal{B}$  will act as a challenger with  $\mathcal{A}$  to play the following selective CCA-O security game.

**Init:**  $\mathcal{A}$  sends the challenge gate  $W^* = \bigwedge_{i \in I^*} i$  to  $\mathcal{B}$ .

**Setup:**  $\mathcal{B}$  sets  $Y = \hat{e}(\mathbf{A}, \mathbf{B}) = \hat{e}(g, g)^{ab}$ , and chooses random  $\alpha_i, \beta_i, \gamma_i$  from  $\mathbb{Z}_q$  for each  $i \in [1, n]$ . The public elements  $T_i, T_{n+i}$  and  $T_{2n+i}$  are computed as follows.

$$T_i = \begin{cases} g^{\alpha_i}, & \text{if } i \in I^* \wedge i = i, \\ B^{\alpha_i}, & \text{otherwise;} \end{cases} \quad T_{n+i} = \begin{cases} g^{\beta_i}, & \text{if } i \in I^* \wedge i = -i, \\ B^{\beta_i}, & \text{otherwise;} \end{cases}$$

and

$$T_{2n+i} = \begin{cases} g^{\gamma_i}, & \text{if } i \notin I^*, \\ B^{\gamma_i}, & \text{otherwise.} \end{cases}$$

It also builds the following hash oracles.

- $H_1$ : On input  $R$  from  $\{0, 1\}^{2\ell}$ ,  $\mathcal{B}$  first finds the pair  $(R, h_1)$  in the list  $L_1$ . If it exists, it simply returns  $h_1$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{B}$  chooses a random element from  $\mathbb{Z}_q$  and sets as  $h_1$ . After that,  $\mathcal{B}$  records  $(R, h_1)$  in the list  $L_1$  and returns  $h_1$  to  $\mathcal{A}$ .
- $H_2$ : On input  $R$  from  $\mathbb{G}_T$ ,  $\mathcal{B}$  first finds the pair  $(R, h_2)$  in the list  $L_2$ . If it exists, it simply returns  $h_2$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{B}$  chooses a random element from  $\{0, 1\}^\ell$  and sets as  $h_2$ . After that,  $\mathcal{B}$  records  $(R, h_2)$  in the list  $L_2$  and returns  $h_2$  to  $\mathcal{A}$ .
- $H_3$ : On input  $R$  from  $\{0, 1\}^\ell$ ,  $\mathcal{B}$  first finds the pair  $(R, h_3)$  in the list  $L_3$ . If it exists, it simply returns  $h_3$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{B}$  chooses a random element from  $\{0, 1\}^\ell$  and sets as  $h_3$ . After that,  $\mathcal{B}$  records  $(R, h_1)$  in the list  $L_3$  and returns  $h_3$  to  $\mathcal{A}$ .

- $H_4$ : On input  $R$  from  $\{0, 1\}^*$ ,  $\mathcal{B}$  first finds the pair  $(R, h_4)$  in the list  $L_4$ . If it exists, it simply returns  $g^{h_4}$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{B}$  chooses a random element from  $\mathbb{Z}_q$  and sets as  $h_4$ . After that,  $\mathcal{B}$  records  $(R, h_4)$  in the list  $L_4$  and returns  $g^{h_4}$  to  $\mathcal{A}$ .

Other elements in the public key are generated as in the real execution.

**Phase 1:**  $\mathcal{A}$  issues queries  $q_1, \dots, q_{n_1}$  where query  $q_i$  is one of:

- $\mathcal{O}_{sk}$ :  $\mathcal{A}$  submits an attribute set  $S$  that does not satisfy  $W^*$ . There must exist  $j \in I^*$  such that: either  $j \in S$  and  $\underline{j} = \neg j$ , or  $j \notin S$  and  $\underline{j} = j$ .  $\mathcal{B}$  selects such a  $j$ . As in [4], we assume that  $j \notin S$  and  $\underline{j} = j$ .  $\mathcal{B}$  selects random  $r'_i$  from  $\mathbb{Z}_q$  for every  $i \in [1, n]$ , and implicitly sets  $r_j := ab + r'_j \cdot b \bmod q$  for each  $i \neq j$ ,  $r_i := r'_i \cdot b \bmod q$  for  $i = j$ , and  $r := \sum_{i=1}^n r_i \bmod q = ab + \sum_{i=1}^n r'_i \cdot b \bmod q$ . Hence, we can compute  $\hat{D} = \prod_{i=1}^n 1/B^{r'_i} = g^{ab-r}$ ,

$$D_i = \begin{cases} A^{1/\beta_j} \cdot g^{r'_j/\beta_j}, & i = j \\ B^{r'_i/\alpha_i}, & (i \neq j) \wedge (i \in S \wedge i \in I \wedge \underline{i} = i) \\ g^{r'_i/\alpha_i}, & (i \neq j) \wedge ((i \in S \wedge i \in I \wedge \underline{i} = \neg i) \vee (i \in S \wedge i \notin I)) \\ B^{r'_i/\alpha_i}, & (i \neq j) \wedge (i \notin S \wedge i \in I \wedge \underline{i} = \neg i) \\ g^{r'_i/\alpha_i}, & (i \neq j) \wedge ((i \notin S \wedge i \in I \wedge \underline{i} = i) \vee (i \notin S \wedge i \notin I)) \end{cases}$$

and

$$F_i = \begin{cases} A^{1/\gamma_j} \cdot g^{r'_j/\gamma_j}, & i = j \\ g^{r'_i/\gamma_i}, & (i \neq j) \wedge (i \in I) \\ B^{r'_i/\gamma_i}, & (i \neq j) \wedge (i \notin I) \end{cases}$$

The validity of  $(\hat{D}, \{(D_i, F_i) | i \in [1, n]\})$  can be verified as that in [4].

- $\mathcal{O}_{tk}$ : On input an attribute set  $S$  and an index  $i$ ,  $\mathcal{B}$  first checks if  $(i, ok, S)$  in the list  $L_{ok}$ . If it exists,  $\mathcal{B}$  returns the corresponding  $tk$ . Otherwise,  $\mathcal{B}$  does the following steps. If  $S$  does not satisfy  $W^*$ , then  $\mathcal{B}$  can obtain the corresponding private key  $d$  by querying  $\mathcal{O}_{sk}$  with  $S$ , and the corresponding outsourced decryption key  $ok = (tk, rk) = ((\hat{D}^z, \{(D_i^z, F_i^z) | i \in [1, n]\}), z)$  by running  $\text{OKGen}(d)$ . If  $S$  satisfies  $W^*$ , then  $\mathcal{B}$  chooses random elements  $z$  from  $\mathbb{Z}_q$ ,  $R, \{(R_i, R'_i) | i \in [1, n]\}$  from  $\mathbb{G}$ , and sets the outsourced decryption key  $ok = (tk, rk) := ((R^z, \{(R_i^z, R'_i^z) | i \in [1, n]\}), z)$ . At last,  $\mathcal{B}$  records  $(i, ok, S)$  into the list  $L_{ok}$ , and returns  $tk$  to  $\mathcal{A}$ .
- $\mathcal{O}_{rk}$ : It is almost the same as  $\mathcal{O}_{tk}$ . The only difference is that the returned value is the retrieving key  $rk$ .
- $\mathcal{O}_{od}$ : On input  $(C_i, S, i)$ ,  $\mathcal{B}$  first checks whether  $S$  satisfies  $W$  embedded in  $C_i$ . If it does not,  $\mathcal{B}$  outputs  $\perp$ ; otherwise, it returns  $\text{TDec}(tk, C_i)$ , where  $tk$  is the transformation key from  $\mathcal{O}_{tk}$  with input  $(S, i)$ .
- $\mathcal{O}_{dec}$ : On input  $(C_i, S, i)$ ,  $\mathcal{B}$  first checks whether  $S$  satisfies  $W$  embedded in  $C_i$ . If it does not,  $\mathcal{B}$  outputs  $\perp$ ; otherwise,  $\mathcal{B}$  does the following steps.
  - If the ciphertext is an original ciphertext,  $\mathcal{B}$  first checks the validity of the ciphertext as the real execution. If it does not pass, it outputs  $\perp$  and aborts. If it passes, it finds the pairs  $(R_1, h_1)$ ,  $(R_2, h_2)$ ,  $(R_3, h_3)$  in lists  $L_1, L_2, L_3$ , respectively. These pairs should satisfy  $R_1 = \mu || m$ ,



$\tilde{C}_1 = \mu \oplus h_2$ ,  $R_2 = Y^{h_1}$ ,  $\tilde{C}_2 = m \oplus h_3$ , and  $R_3 = \mu$ . If there does not exist such pairs, then  $\mathcal{B}$  outputs  $\perp$ ; otherwise, it outputs  $m$ .

- If the ciphertext is a transformed ciphertext,  $\mathcal{B}$  finds  $(i, ok, S)$  in the list  $L_{ok}$ . If it does not exist, it outputs  $\perp$ . Otherwise, it finds the pairs  $(R_1, h_1)$ ,  $(R_2, h_2)$ ,  $(R_3, h_3)$  in lists  $L_1, L_2, L_3$ , respectively. These pairs should satisfy  $R_1 = \mu || m$ ,  $\tilde{C}_1 = \mu \oplus h_2$ ,  $R_2 = Y^{h_1}$ ,  $\tilde{C}_2 = m \oplus h_3$ , and  $R_3 = \mu$ . If there does not exist such pairs, then  $\mathcal{B}$  outputs  $\perp$ . Otherwise, it checks whether  $\tilde{C}_3 = (\hat{e}(g, R) \cdot \prod_{i \in I} \hat{e}(g, R_i) \cdot \prod_{i \in \{1, \dots, n\} \setminus I} \hat{e}(g, R'_i))^{z \cdot h_1}$ , where  $R, R_i, R'_i$  are used to generate the corresponding transformation key recorded in the list  $L_{ok}$ , and  $W = \bigwedge_{i \in I} \hat{e}(g, R_i)$ . If it does not,  $\mathcal{B}$  outputs  $\perp$ ; otherwise, it outputs  $m$ .

**Challenge:** Once  $\mathcal{A}$  decides that Phase 1 is over, it outputs two equal length plaintexts  $m_0^*, m_1^*$  from the message space  $\{0, 1\}^\ell$ ,  $\mathcal{B}$  sets  $\tilde{C}_1 = \mu \oplus H_2(\mathbf{Z})$ ,  $\tilde{C}_2 = m_{\mathbf{b}} \oplus H_3(\mu)$ ,  $\hat{C}_1 = \mathbf{C}$ ,  $\hat{C}_2 = \mathbf{C}^{h_4}$ ,  $\{\mathbf{C}^{\alpha_i} | i \in I^* \wedge \underline{i} = i\}$ ,  $\{\mathbf{C}^{\beta_i} | i \in I^* \wedge \underline{i} = -i\}$ , and  $\{\mathbf{C}^{\gamma_i} | i \in \{1, \dots, n\} \setminus I^*\}$ , where  $\mathbf{b}$  is randomly chosen from  $\{0, 1\}$ ,  $\mu$  is a random element from  $\{0, 1\}^\ell$ , and  $h_4$  is the corresponding value in the list  $L_4$ . According to the analysis in [4] and [5], the resulting challenge ciphertext is valid if  $\mathbf{Z} = \hat{e}(g, g)^{abc}$ .

**Phase 2:** Almost the same as Phase 1 but with the specified restrictions.

**Guess:** Finally, the adversary  $\mathcal{A}$  outputs a guess  $\mathbf{b}' \in \{0, 1\}$  on  $\mathbf{b}$ . If  $\mathbf{b} = \mathbf{b}'$ ,  $\mathcal{B}$  decides  $\hat{e}(g, g)^{abc} = \mathbf{Z}$ ; otherwise, it decides  $\hat{e}(g, g)^{abc} \neq \mathbf{Z}$ .

According to the analysis in [5], the above simulation will abort with a negligible probability. Hence, we obtain the theorem.  $\square$

**Theorem 2.** *The proposed OD-ABE scheme is selective CCA-T secure in the random oracle model if the MDDH assumption holds in  $\mathbb{G}$  and  $\mathbb{G}_T$ .*

*Proof.* We show that if there exists an adversary  $\mathcal{A}$  that can break the selective CCA-T security of the proposed OD-ABE scheme, we can build an algorithm  $\mathcal{B}$  that can solve the MDDH problem, i.e., given the tuple  $(g, \mathbf{A} = g^{\mathbf{a}}, \mathbf{B} = \hat{e}(g, g)^{\mathbf{b}}, \mathbf{Z}) \in \mathbb{G}^2 \times \mathbb{G}_T^2$ , it decides whether  $\mathbf{Z} = \hat{e}(g, g)^{ab}$ . In particular,  $\mathcal{B}$  will act as a challenger with  $\mathcal{A}$  to play the following selective CCA-T security game.

**Init:**  $\mathcal{A}$  sends the challenge gate  $W^* = \bigwedge_{i \in I^*} \hat{e}(g, g)^i$  and challenge index  $i^*$  to  $\mathcal{B}$ .

**Setup:**  $\mathcal{B}$  selects random  $y, t_1, \dots, t_{3n}$  from  $\mathbb{Z}_q$ , and computes the public elements as follows:  $Y = \hat{e}(g, g)^y$  and  $T_i = g^{t_i}$  for every  $i \in [1, 3n]$ . It also builds hash oracles for  $H_1, H_2$ , and  $H_3$  as that in the proof of Theorem 1, while it chooses  $H_4$  as the real execution. It is easy to see that the resulting master secret key is  $msk = (y, t_1, \dots, t_{3n})$  that is known to  $\mathcal{B}$ .

**Phase 1:**  $\mathcal{A}$  issues queries  $q_1, \dots, q_{n_1}$  where query  $q_i$  is one of:

- $\mathcal{O}_{sk}$ :  $\mathcal{A}$  submits an attribute set  $S$ ,  $\mathcal{B}$  runs  $\text{Ext}(msk, S)$  to obtain  $d$  and returns it to  $\mathcal{A}$ .
- $\mathcal{O}_{tk}$ : On input an attribute set  $S$  and an index  $i$ ,  $\mathcal{B}$  first finds  $(i, ok, S)$  in the list  $L_{ok}$ . If it exists,  $\mathcal{B}$  returns the corresponding  $tk$  and aborts. Otherwise, if  $i \neq i^*$  or  $S$  does not satisfy  $W^*$ ,  $\mathcal{B}$  obtains the corresponding private key  $d$  by querying  $\mathcal{O}_{sk}$  with  $S$ , and the corresponding outsourced

decryption key  $ok = (tk, rk) = ((\hat{D}^z, \{(D_i^z, F_i^z) | i \in [1, n]\}), z)$  by running  $\text{OKGen}(d)$ ; if  $\mathbf{i} = \mathbf{i}^*$  and  $S$  satisfies  $W^*$ ,  $\mathcal{B}$  obtains the corresponding private key  $d$  by querying  $\mathcal{O}_{sk}$  with  $S$ , and the corresponding outsourced decryption key  $ok = (tk, rk) = ((\mathbf{A}^{y-r}, \{\mathbf{A}^{r_i/t_i} | i \in S\}, \{\mathbf{A}^{r_i/t_{n+i}} | i \notin S\}, \{\mathbf{A}^{r_i/t_{2n+i}} | i \in [1, n]\}), \blacksquare)$ , where  $r = \sum_{i=1}^n r_i, \{r_i \in Z_q | i \in [1, n]\}$  are the random elements used to generate the corresponding private key  $d$ , and  $\blacksquare$  means that the value of  $rk$  is unknown. Note that, in the case of that  $\mathbf{i} = \mathbf{i}^*$  and  $S$  satisfies  $W^*$ ,  $\mathcal{B}$  implicitly sets  $z = a$  that is unknown. At last,  $\mathcal{B}$  records  $(\mathbf{i}, ok, S)$  into the list  $L_{ok}$  and returns  $tk$  to  $\mathcal{A}$ .

- $\mathcal{O}_{rk}$ : It is almost the same as  $\mathcal{O}_{tk}$ . The only difference is that the returned value is the retrieving key  $rk$ .
- $\mathcal{O}_{od}$ : On input  $(C_i, S, \mathbf{i})$ ,  $\mathcal{B}$  first checks whether  $S$  satisfies  $W$  corresponding to  $C_i$ . If it does not,  $\mathcal{B}$  outputs  $\perp$ ; otherwise, it returns  $\text{TDec}(tk, C_i)$ , where  $tk$  is the transformation key from  $\mathcal{O}_{tk}$  with input  $(S, \mathbf{i})$ .
- $\mathcal{O}_{dec}$ : On input  $(C_i, S, \mathbf{i})$ , there are two cases.
  - If the ciphertext is an original ciphertext,  $\mathcal{B}$  runs  $d \leftarrow \text{KeyGen}(msk, S)$  and  $m \leftarrow \text{Dec}(d, C_i)$ , and returns  $m$  to  $\mathcal{A}$ .
  - If the ciphertext is a transformed ciphertext,  $\mathcal{B}$  finds  $(\mathbf{i}, ok, S)$  in the list  $L_{ok}$ . If it does not exist, it outputs  $\perp$  and aborts. Otherwise,  $\mathcal{B}$  does the following steps.
    - \*. If  $\mathbf{i} \neq \mathbf{i}^*$  or  $S$  does not satisfy  $W^*$ , it uses the corresponding  $rk$  to obtain the message  $m$  and returns it to  $\mathcal{A}$ .
    - \*. If  $\mathbf{i} \neq \mathbf{i}^*$  and  $S$  satisfies  $W^*$ , it finds the pairs  $(R_1, h_1), (R_2, h_2), (R_3, h_3)$  in lists  $L_1, L_2, L_3$ , respectively. These pairs should satisfy  $R_1 = \mu || m, \tilde{C}_1 = \mu \oplus h_2, R_2 = Y^{h_1}, \tilde{C}_1 = m \oplus h_3$ , and  $R_3 = \mu$ . If there does not exist such pairs, then  $\mathcal{B}$  outputs  $\perp$ . Otherwise, it checks whether  $\tilde{C}_3 = \hat{e}(A, g)^{y \cdot h_1}$ . If it does not,  $\mathcal{B}$  outputs  $\perp$ ; otherwise, it outputs  $m$ .

**Challenge:** Once  $\mathcal{A}$  decides that Phase 1 is over, it outputs two equal length plaintexts  $m_0^*, m_1^*$  from the message space  $\{0, 1\}^\ell$ ,  $\mathcal{B}$  sets  $\tilde{C}_1 = \mu \oplus H_2(\mathbf{B}^y)$ ,  $\tilde{C}_2 = m_{\mathbf{b}} \oplus H_3(\mu)$ , and  $\tilde{C}_3 = \mathbf{Z}^y$ , where  $\mu$  is chosen randomly from  $\{0, 1\}^*$ . According to the analysis in [5], if  $\mathbf{Z} = \hat{e}(g, g)^{ab}$ , then the resulting challenge transformed ciphertext is valid.

**Phase 2:** Almost the same as Phase 1 but with the specified restrictions.

**Guess:** Finally, the adversary  $\mathcal{A}$  outputs a guess  $\mathbf{b}' \in \{0, 1\}$  on  $\mathbf{b}$ . If  $\mathbf{b} = \mathbf{b}'$ ,  $\mathcal{B}$  decides  $\hat{e}(g, g)^{ab} = \mathbf{Z}$ ; otherwise, it decides  $\hat{e}(g, g)^{ab} \neq \mathbf{Z}$ .

According to the analysis in [5], the above simulation will abort with a negligible probability. Hence, we obtain the theorem. □

## 5 Conclusions

In this paper, we investigated the CCA security of attribute-based encryption with outsourced decryption (OD-ABE). In particular, we proposed the CCA security model for OD-ABE and the first OD-ABE scheme with CCA security.

However, the proposal is proven secure in the random oracle model. It is interesting design a new OD-ABE scheme that can be proven secure in the standard model.

**Acknowledgements.** We would like to thank all the anonymous reviewers for their helpful comments. This work was supported by the National Natural Science Foundation of China [grant numbers 61472364, 61472365, and 61572435]; and the Natural Science Foundation of Zhejiang Province [grant numbers LR13F020003, LZ16F020001, and LR15G010001].

## References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE S&P 2007, pp. 321–334 (2007)
2. Blaze, M., Bleumer, G., Strauss, M.J.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
3. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
4. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: ACM CCS 2007, pp. 456–465 (2007)
5. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
6. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98 (2006)
7. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
8. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: USENIX Security Symposium (2011)
9. He, S., Chen, J., Jiang, F., Yau, D.K.Y., Xing, G., Sun, Y.: Energy provisioning in wireless rechargeable sensor networks. *IEEE Trans. Mob. Comput.* **12**(10), 1931–1942 (2013)
10. Lai, C.G.J., Deng, R.H., Weng, J.: Attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Inf. Forensics Secur.* **8**(8), 1343–1354 (2013)
11. 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)
12. Li, J., Huang, X., Li, J., Chen, X., Xiang, Y.: Securely outsourcing attribute-based encryption with checkability. *IEEE Trans. Parallel Distrib. Syst.* **25**(8), 2201–2210 (2014)
13. Lin, S., Zhang, R., Ma, H., Wang, M.: Revisiting attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Inf. Forensics Secur.* **10**(10), 2119–2130 (2015)

14. Mao, X., Lai, J., Mei, Q., Chen, K., Weng, J.: Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Dependable Secur. Comput.* (99), 1 (2015)
15. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: *ACM CCS 2007*, pp. 195–203 (2007)
16. O’Toole, J.: Mobile apps overtake PC internet usage in US CNN Money. <http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/>, February 2014
17. Qin, B., Deng, R.H., Liu, S., Ma, S.: Attribute-based encryption with efficient verifiable outsourced decryption. *IEEE Trans. Inf. Forensics Secur.* **10**(7), 1384–1393 (2015)
18. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
19. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
20. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) *ASIACRYPT 2012*. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012)
21. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) *CRYPTO 2009*. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
22. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) *PKC 2011*. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

# Accountable Large-Universe Attribute-Based Encryption Supporting Any Monotone Access Structures

Yinghui Zhang<sup>1,2</sup>(✉), Jin Li<sup>3</sup>, Dong Zheng<sup>1</sup>, Xiaofeng Chen<sup>4</sup>, and Hui Li<sup>4</sup>

<sup>1</sup> National Engineering Laboratory for Wireless Security,  
Xi'an University of Posts and Telecommunications,  
Xi'an 710121, People's Republic of China  
yhzhaang@163.com

<sup>2</sup> State Key Laboratory of Cryptology,  
P.O. Box 5159, Beijing 100878, People's Republic of China

<sup>3</sup> School of Computer Science, Guangzhou University,  
Guangzhou 510006, People's Republic of China

<sup>4</sup> State Key Laboratory of Integrated Service Networks (ISN),  
Xidian University, Xi'an 710071, People's Republic of China

**Abstract.** Ciphertext-policy attribute-based encryption (CP-ABE) is a promising cryptographic primitive for fine-grained access control on data outsourced to clouds. However, there still exists one critical functionality missing in existing CP-ABE schemes, which is the prevention of key abuse. Specifically, two kinds of key abuse problems are considered in this paper: malicious key sharing among colluding users, and key escrow problem of the semi-trusted authority. For a user, any malicious behavior including illegal key sharing should be traced. For the semi-trusted authority, it should be accountable for its misbehavior including illegal key re-distribution. For better performance and security, it is also indispensable to support large universe and full security in CP-ABE. To the best of our knowledge, none of the existing traceable CP-ABE schemes simultaneously supports large universe and full security. In this paper, we construct a white-box traceable CP-ABE scheme with weak public user traceability, weak public authority accountability and weak public auditing in the sense that no additional secret keys are needed. The scheme supports large universe, and attributes do not need to be pre-specified during the system setup phase. Our scheme is proven fully-secure in the random oracle model and it can take any monotonic access structures as ciphertext policies.

**Keywords:** Attribute-based encryption · User traceability · Authority accountability · Large universe · Full security · Weak public traceability

## 1 Introduction

Attribute-based encryption (ABE) is very promising in implementing flexible access control on the data outsourced to clouds. The notion of ABE was

introduced by Sahai and Waters [25] as a fuzzy version of identity-based encryption (IBE). Goyal *et al.* [8] further extended this idea and defined two complementary notions of ABE: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE).

A series of ABE work has been done to achieve better performance and security. In particular, large universe and full security are two significant and practical properties of ABE, which have attracted much attention in the research community. Lewko *et al.* [11] took the large universe issue into account and classified ABE into two flavors: the small universe ABE (SU-ABE) and the large universe ABE (LU-ABE). In SU-ABE constructions, attributes are fixed at the system setup phase. Furthermore, the system public parameters often depend on the amount of attributes in the system, and hence the scale of the attribute universe is polynomially bounded in the security parameter. In the case of LU-ABE, the attribute universe scale can be exponentially large. Obviously, LU-ABE is more practical than SU-ABE in that the system designer needs not to choose a bound of attributes at system setup. However, many prior schemes have to rely on a weaker security model known as selective security, where the attacker must specify some challenge ciphertext information before seeing the system public parameters. As a more stronger security model, full security [10] allows the attacker to adaptively choose challenge targets based on system public parameters.

Nevertheless, as a major issue remain to be solved, user traceability and authority accountability have severely limited the applications of ABE [10, 11]. In a CP-ABE system, for example, the decryption keys are issued by an attribute authority based on users' attributes, which are usually shared by multiple users and hence are not uniquely linked to users' identification information. Obviously, as the foundation of one-to-many encryption mechanism, the characteristic of attribute sharing introduces the malicious user traceability issue: an explicitly leaked decryption key is non-traceable because the underlying attributes are shared by multiple users. On the other hand, the attribute authority is capable of re-distributing decryption keys for any user without any risk of being caught. So, if the attribute authority is not fully-trusted, it is indispensable to provide a method to make the authority accountable. In the above description, any user and the authority who exactly leak a decryption key to the third user intentionally or unintentionally will be identified, which is called white-box traceability. As a relatively stronger notion, black-box traceability can trace the malicious users and authority even if they only leak a decryption equipment instead of the decryption key.

There are some ABE solutions [7, 13, 15, 16, 18–21, 30] for the purpose of user traceability and authority accountability. In these schemes, white-box user traceability [13, 15, 20, 21, 30] and black-box user accountability [7, 16, 18, 19] are considered. Meanwhile, only schemes [15, 16, 18, 21] achieve full security and schemes [7, 19, 20, 30] support large universe. In particular, only the solutions [13, 21, 30] take one step further towards *authority accountability* although in the white-box model. As one of the latest work, the scheme [21] allows tracing and weak public

**Table 1.** Features comparison between authority accountable CP-ABE schemes

Schemes	Security	MAS	Large universe	Weak public UT	Weak public AA	Weak Public Auditing
[13]	selective (random)	×	×	✓	✓	–
[30]	selective (standard)	✓	×	×	×	–
[21]	full (standard)	✓	×	×	×	✓
Ours	full (random)	✓	✓	✓	✓	✓

auditing in the case of almost no storage. However, it is a small universe construction and has a security weakness shown later. In summary, it is necessary to efficiently add both user traceability and authority accountability to the original ABE while keeping the properties of large universe and full security.

**Our Contribution.** In this paper, we address the key abuse problems of CP-ABE while keeping desirable performance and security. The main contributions can be summarized as follows:

- We propose an authority accountable large universe CP-ABE scheme (AA-LU-CPABE) that simultaneously supports (1) weak public user traceability, (2) weak public authority accountability, (3) weak public auditing, (4) large universe, and (5) full security. The expression “weak public”<sup>1</sup> means that both traceability and auditing only involve decryption keys, which are indispensable in the white-box model, and no additional secret parameters such as master secret keys and identity tables are needed.
- For realizing user traceability, when a user queries for decryption key, his/her identity is inserted into a decryption key component, which is further implicitly signed as a fixed component such that the key owner is not able to re-randomize it. To achieve authority accountability, a user’s decryption key is generated by the user himself based on a primary decryption key, which is jointly determined by both the authority and the user.
- The AA-LU-CPABE scheme needs almost no storage for tracing in that it does not need to maintain an identity table of users for tracing. In addition, our scheme is proven secure in the random oracle model against adaptive adversaries, and is highly expressive and can take any monotonic access structures as ciphertext policies.

Note that only schemes [13, 21, 30] support *authority accountability*. We compare our work with schemes [13, 21, 30] in Table 1, where MAS, UT and AA mean monotonic access structures, user traceability and authority accountability, respectively. The symbol “–” represents the corresponding scheme does not need auditing. All the schemes are realized in the white-box model. It’s noted that only the proposed scheme simultaneously supports weak public traceability and auditing with large universe and full security.

<sup>1</sup> The expression “weak public” is similar to the term “partial public” in [20], in which only private user traceability is realized.

**Related Work.** Since the introduction of ABE [25], a plenty of researches have been done on flexible ABE schemes. The first CP-ABE scheme was proposed by Bethencourt *et al.* [4], which is proven secure in the generic group model. To improve the security proof, Cheung and Newport [6] proposed another CP-ABE construction and proved its security in the standard model. The construction supports the access structures of AND gate on different attributes. In order to further protect users' attribute privacy, anonymous CP-ABE has been studied [9, 22, 33]. A series of CP-ABE schemes have been proposed for better expressiveness, security and efficiency [1, 27, 29, 31, 32, 34]. In particular, large universe, full security and traceability are important aspects to be considered.

The first large universe KP-ABE construction was given in [11], which achieves selective security under static assumptions in the standard model. They utilized the dual system framework on composite order groups to prove security. The first large universe CP-ABE scheme was proposed in [24], which is selectively secure under two  $q$ -type assumptions in the standard model. By utilizing dual vector spaces, Okamoto and Takashima [23] proposed the first fully secure unbounded CP-ABE scheme in the standard model. Lewko *et al.* [10] constructed a fully-secure CP-ABE scheme in the standard model, however, it fails to support large universe. Li *et al.* [13, 14] first introduced the notion of accountable CP-ABE. The scheme [13] takes into account authority accountability which is achieved by embedding additional user-specific information into the attribute decryption key. Yu *et al.* [28] considered how to defend the key-abuse problem in KP-ABE. A user traceable multi-authority CP-ABE scheme was proposed in [12]. For the purpose of expressiveness, Liu *et al.* proposed white-box [15] and black-box [16–18] traceable CP-ABE schemes. These schemes cannot support large universe even if they are fully-secure in the standard model. Large universe CP-ABE schemes with user accountability were proposed in the white-box model in [20] and in the black-box model in [7, 19], which are proven selectively-secure. Most of the above schemes fail to realize authority accountability while keeping expressive policies. For the sake of practicality, the solutions [21, 30] take one step further towards authority accountability although in the white-box model. However, the scheme [30] is selectively-secure and the scheme [21] is a small universe construction. In general, if a system requires user traceability, authority accountability, (weak public) auditing and full security, only the scheme [21] may be adopted. Unfortunately, we found that the scheme [21] is not secure. In fact, after receiving from the authority  $c$  and the primary decryption key  $SK_{pri} = \langle \bar{K}, \bar{T}, \bar{L}, \bar{L}', \{\bar{K}_i\}_{i \in S} \rangle$ , a user just sets  $t_{id} = \frac{c}{\bar{t}}$ , where  $t$  is chosen by himself and  $R_u = g^t$ , and generates the final decryption key  $SK_{id,S} = \langle K = \bar{K}(g^\mu)^{t_{id}}, T = \bar{T}, L = \bar{L}, L' = \bar{L}', R_u, t_{id}, \{K_i = \bar{K}_i\}_{i \in S} \rangle$ , where  $g^\mu$  is a public parameter. Then, the user randomly chooses a value  $c_0$  and is able to re-randomize  $SK_{id,S}$  based on the idea of changing  $c$  to  $c \cdot c_0$ . In this paper, to avoid the above security weakness, the attribute authority chooses two secrets  $c_0$  and  $\bar{c}$  and only  $\bar{c}$  is sent to the user. Most importantly,  $c_0$  and  $\bar{c}$  are used to generate different components of a decryption key during the key generation phase. The details can be found in Sect. 4.1.



**Organization.** The remaining of this work is organized as follows. Some preliminaries are reviewed in Sect. 2. We then present the formal definition and security models for AA-LU-CPABE in Sect. 3. In Sect. 4, the proposed AA-LU-CPABE construction together with its security results are described. Finally, we conclude this paper in Sect. 5.

## 2 Preliminaries

Throughout this paper, for  $\ell \in \mathbb{N}$ , we denote by  $[\ell]$  the set  $\{1, 2, \dots, \ell\}$ . For a set  $S$ ,  $|S|$  represents its cardinality, and  $s \in_R S$  means the variable  $s$  is chosen uniformly at random from  $S$ .

### 2.1 Cryptographic Background

**Definition 1. (Composite Order Bilinear Groups).** *Composite order bilinear groups are widely used in IBE and ABE systems, which are first introduced in [5]. We denote by  $\mathcal{G}$  a group generator, which takes a security parameter  $\lambda$  as inputs and outputs a description of a bilinear group  $\mathbb{G}$ . We define the output of  $\mathcal{G}$  as  $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, \hat{e})$ , where  $p_1, p_2, p_3$  are distinct primes,  $\mathbb{G}$  and  $\mathbb{G}_T$  are two cyclic groups of order  $N = p_1 p_2 p_3$ , and  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  is a bilinear map satisfying: (1) Bilinear:  $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$  for all  $a, b \in \mathbb{Z}_N$  and  $g, h \in \mathbb{G}$ , (2) Non-degenerate: There exists  $g \in \mathbb{G}$  such that  $\hat{e}(g, g)$  has order  $N$  in  $\mathbb{G}_T$ .*

Assume that group operations in  $\mathbb{G}$  and  $\mathbb{G}_T$  as well as the bilinear map  $\hat{e}$  are computable in polynomial time with respect to  $\lambda$ . Let  $\mathbb{G}_{p_i}$  be the subgroup of order  $p_i$  in  $\mathbb{G}$  for  $1 \leq i \leq 3$ . Note that for any  $X_i \in \mathbb{G}_{p_i}$  and  $X_j \in \mathbb{G}_{p_j}$ ,  $\hat{e}(X_i, X_j) = 1$  holds for  $i \neq j$ .

**Assumption 1. (Subgroup Decision Problem for 3 Primes):** *Given a group generator  $\mathcal{G}$ , define the following distribution:*

$$\begin{aligned} (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) &\stackrel{R}{\leftarrow} \mathcal{G}, g \stackrel{R}{\leftarrow} \mathbb{G}_{p_1}, X_3 \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}, \\ D &= (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, X_3), T_1 \stackrel{R}{\leftarrow} \mathbb{G}_{p_1 p_2}, T_2 \stackrel{R}{\leftarrow} \mathbb{G}_{p_1}. \end{aligned}$$

The advantage of an algorithm  $\mathcal{A}$  in breaking this assumption is defined to be:  $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|$ .

**Definition 2.** *We say that  $\mathcal{G}$  satisfies Assumption 1 if  $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\lambda)$  is a negligible function of  $\lambda$  for any polynomial time algorithm  $\mathcal{A}$ .*

**Assumption 2.** *Given a group generator  $\mathcal{G}$ , define the following distribution:*

$$\begin{aligned} (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) &\stackrel{R}{\leftarrow} \mathcal{G}, g, X_1 \stackrel{R}{\leftarrow} \mathbb{G}_{p_1}, X_2, Y_2 \stackrel{R}{\leftarrow} \mathbb{G}_{p_2}, X_3, Y_3 \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}, \\ D &= (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, X_1 X_2, X_3, Y_2 Y_3), T_1 \stackrel{R}{\leftarrow} \mathbb{G}, T_2 \stackrel{R}{\leftarrow} \mathbb{G}_{p_1 p_3}. \end{aligned}$$

The advantage of an algorithm  $\mathcal{A}$  in breaking this assumption is defined to be:  $\text{Adv}_{\mathcal{G}, \mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|$ .

**Definition 3.** We say that  $\mathcal{G}$  satisfies Assumption 2 if  $\text{Adv}_{2\mathcal{G},\mathcal{A}}(\lambda)$  is a negligible function of  $\lambda$  for any polynomial time algorithm  $\mathcal{A}$ .

**Assumption 3.** Given a group generator  $\mathcal{G}$ , define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}, \alpha, s \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2, Z_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3 \xleftarrow{R} \mathbb{G}_{p_3},$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, g^\alpha X_2, X_3, g^s Y_2, Z_2), T_1 = \hat{e}(g, g)^{\alpha s}, T_2 \xleftarrow{R} \mathbb{G}_T.$$

The advantage of an algorithm  $\mathcal{A}$  in breaking this assumption is defined to be:  $\text{Adv}_{3\mathcal{G},\mathcal{A}}(\lambda) = |\text{Pr}[\mathcal{A}(D, T_1) = 1] - \text{Pr}[\mathcal{A}(D, T_2) = 1]|$ .

**Definition 4.** We say that  $\mathcal{G}$  satisfies Assumption 3 if  $\text{Adv}_{3\mathcal{G},\mathcal{A}}(\lambda)$  is a negligible function of  $\lambda$  for any polynomial time algorithm  $\mathcal{A}$ .

**$\ell$ -SDH assumption** Let  $\mathbb{G}$  be a bilinear group of prime order  $p$  and  $g$  be a generator of  $\mathbb{G}$ , the  $\ell$ -Strong Diffie-Hellman ( $\ell$ -SDH) problem in  $\mathbb{G}$  is defined as follows: given a  $\ell + 1$ -tuple  $(g, g^x, g^{x^2}, \dots, g^{x^\ell})$  as inputs, output a pair  $(c, g^{1/(c+x)}) \in \mathbb{Z}_p \times \mathbb{G}$ . An algorithm  $\mathcal{A}$  has advantage  $\epsilon$  in solving  $\ell$ -SDH problem in  $\mathbb{G}$  if  $\text{Pr}[\mathcal{A}(g, g^x, g^{x^2}, \dots, g^{x^\ell}) = (c, g^{1/(c+x)})] \geq \epsilon$ , where the probability is over the random choice of  $x$  in  $\mathbb{Z}_p^*$  and the random bits consumed by  $\mathcal{A}$ .

**Definition 5.** We say that  $(\ell, t, \epsilon)$ -SDH assumption holds in  $\mathbb{G}$  if no  $t$ -time algorithm has advantage at least  $\epsilon$  in solving the  $\ell$ -SDH problem in  $\mathbb{G}$ .

## 2.2 Zero-Knowledge Proof of Knowledge of Discrete Log

A zero-knowledge proof of knowledge (ZK-PoK) of discrete log protocol that enables a prover to prove to a verifier that it possesses the discrete log of a given group element in question. Efficient ZK-PoK of discrete log protocols can be found in [26]. A ZK-PoK protocol has the proof of knowledge property besides the zero-knowledge property. The property of zero-knowledge implies that there exists a simulator which is able to simulate the view of a verifier in the protocol without being given the witness as inputs. The proof of knowledge property implies the existence of a knowledge-extractor which interacts with the prover and extracts the witness using rewinding techniques [3].

## 2.3 Access Policy

**Definition 6 (Access Structures [2]).** Let  $\mathcal{U}$  be a set of parties. A collection  $\mathbb{A} \subseteq 2^{\mathcal{U}}$  is monotone if  $\forall B \in \mathbb{A}$  and  $C \in 2^{\mathcal{U}}$ : if  $B \subseteq C$  then  $C \in \mathbb{A}$ . An access structure (resp. monotone access structure) on  $\mathcal{U}$  is a collection (resp. monotone collection)  $\mathbb{A}$  of non-empty subsets of  $\mathcal{U}$ , i.e.,  $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\emptyset\}$ . The sets in  $\mathbb{A}$  are called the authorized sets, otherwise, the sets are called the unauthorized sets.

**Definition 7 (Linear Secret Sharing Schemes (LSSS) [2]).** Let  $\mathcal{U}$  be the attribute universe and  $\mathbb{A}$  an access structure on  $\mathcal{U}$ . An LSSS can be used to represent an access structure  $\mathbb{A} = (M, \rho)$ , where  $M$  is an  $\ell \times n$  matrix which is called the share-generating matrix and  $\rho$  maps a row of  $M$  into an attribute. An LSSS consists of two algorithms of secret sharing and reconstruction as below.

- **Share** $((M, \rho), s)$ : This algorithm is used to share a secret value  $s$  based on attributes. Considering a vector  $\mathbf{v} = (s, y_2, \dots, y_n)^T$ , where  $s \in \mathbb{Z}_p$  is the secret to be shared and  $y_2, \dots, y_n \in_R \mathbb{Z}_p$ , then  $\lambda_i = M_i \cdot \mathbf{v}$  is a share of the secret  $s$  which belongs to the attribute  $\rho(i)$ , where  $M_i$  is the  $i$ -th row of  $M$ .
- **Reconstruction** $(\lambda_1, \dots, \lambda_\ell, (M, \rho))$ : This algorithm is used to reconstruct  $s$  from secret shares. Let  $S \in \mathbb{A}$  be any authorized set and  $I = \{i | \rho(i) \in S\} \subseteq \{1, 2, \dots, \ell\}$ . Then there exists coefficients  $\{\omega_i\}_{i \in I}$  such that  $\sum_{i \in I} \omega_i M_i = (1, 0, \dots, 0)$ , thus we have  $\sum_{i \in I} \omega_i \lambda_i = s$ .

### 3 Formal Definition and Security Model

#### 3.1 Formal Definition of AA-LU-CPABE

An AA-LU-CPABE scheme consists of six algorithms **Setup**, **KeyGen**, **Encrypt**, **Decrypt**, **Trace<sub>wp</sub>** and **Judge<sub>wp</sub>**. They are detailed as follows:

- **Setup** $(1^\lambda) \rightarrow (PK, MK)$ : The setup algorithm is run by the attribute authority. On input a security parameter  $\lambda$ , it outputs the system public key  $PK$  and the master key  $MK$ .
- **KeyGen** $(PK, MK, ID, S) \rightarrow SK_{ID,S}$ : This is an interactive protocol between the authority and a user with an identity  $ID$  and a set of attributes  $S$ . The system public key  $PK$  and  $(ID, S)$  are the common inputs to the authority and the user. The master key  $MK$  is the private input to the authority. At the end of the protocol, a decryption key  $SK_{ID,S}$  corresponding to  $(ID, S)$  is finally generated by the user based on a primary decryption key which is determined jointly by the authority and the user. Note that only  $S$  is implicitly included in  $SK_{ID,S}$ .
- **Encrypt** $(PK, m, (M, \rho)) \rightarrow CT$ : On input the system public key  $PK$ , a message  $m$  and an access policy  $(M, \rho)$  specified by the encryptor, it generates a ciphertext  $CT$  as the encryption of  $m$  with respect to  $(M, \rho)$ . Note that  $(M, \rho)$  is implicitly included in  $CT$ .
- **Decrypt** $(PK, CT, SK_{ID,S}) \rightarrow m$  or  $\perp$ : On input the system public key  $PK$ , a ciphertext  $CT$  of a message  $m$  under  $(M, \rho)$ , and a decryption key  $SK_{ID,S}$  associated with  $(ID, S)$ , it outputs the message  $m$  if  $S$  satisfies  $(M, \rho)$ , and the error symbol  $\perp$  otherwise.
- **Trace<sub>wp</sub>** $(PK, SK_{ID,S}) \rightarrow ID$  or  $\top$ : On input the system public key  $PK$  and a decryption key  $SK_{ID,S}$ , the tracing algorithm first checks whether  $SK_{ID,S}$  is well-formed or not. If  $SK_{ID,S}$  is well-formed, it extracts the identity  $ID$  from  $SK_{ID,S}$  and outputs  $ID$  to indicate that  $SK_{ID,S}$  is linked to  $ID$ . Otherwise, it outputs a special symbol  $\top$  to indicate that  $SK_{ID,S}$  does not need to be traced. A decryption key is well-formed means that it passes a “key sanity check” which guarantees that the decryption key can be used in the well-formed decryption process.
- **Judge<sub>wp</sub>** $(PK, SK_{ID,S}, SK_{ID,S}^*) \rightarrow$  guilty or innocent: This is an interactive protocol between a user  $(ID, S)$  with a decryption key  $SK_{ID,S}$  and a public auditor. When the user is identified as a malicious user by the system based

on the traced key  $SK_{ID,S}^*$ , the auditor judges whether the user is guilty or innocent upon receiving  $SK_{ID,S}$  from the user.

*Remark 1.* The tracing and judge algorithms need the decryption key  $SK_{ID,S}$ , which means the white-box model. However, no additional secret parameters are needed in our scheme. Note that master secret keys are required in the white-box schemes [21], identity tables are required in the white-box schemes [15,20,30], and suspected users’ decryption keys are needed in the white-box scheme [30].

### 3.2 Security Models for AA-LU-CPABE

An AA-LU-CPABE scheme is secure if the following requirements are satisfied. First, it satisfies the standard semantic security for CP-ABE: ciphertext indistinguishability under chosen-plaintexts attacks (IND-CPA). Second, it is intractable for the authority to create a decryption key such that the  $\mathbf{Trace}_{wp}$  algorithm outputs a user and the  $\mathbf{Judge}_{wp}$  algorithm outputs **guilty**. Finally, it is infeasible for a user to create a decryption key such that the user is **innocent** based on the  $\mathbf{Judge}_{wp}$  algorithm. Security for AA-LU-CPABE schemes are modeled in following three games between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{B}$ .

**The IND-CPA game.** The IND-CPA game for AA-LU-CPABE scheme is defined as follows:

- **Setup:**  $\mathcal{B}$  chooses a security parameter  $\lambda$ , and runs the **Setup** algorithm and sends the system public key  $PK$  to  $\mathcal{A}$ .
- **Phase 1:** In addition to hash queries, the adversary  $\mathcal{A}$  issues a polynomially bounded number of key generation queries:
  - **KeyGen Oracle**  $\mathcal{O}_{KeyGen}$ :  $\mathcal{A}$  submits an identity  $ID$  and an attribute set  $S$ ,  $\mathcal{B}$  gives  $\mathcal{A}$  the decryption key  $SK_{ID,S}$ .
- **Challenge:** Once  $\mathcal{A}$  decides that **Phase 1** is over, it outputs two equal length messages  $m_0$  and  $m_1$  from the message space and an access structure  $(M^*, \rho^*)$ . It is noted that  $(M^*, \rho^*)$  cannot be satisfied by any of the queried attribute sets.  $\mathcal{B}$  chooses a bit  $b \in_R \{0, 1\}$ , computes  $CT^* = \mathbf{Encrypt}(PK, m_b, (M^*, \rho^*))$  and sends  $CT^*$  to  $\mathcal{A}$ .
- **Phase 2:** The same as **Phase 1** except that the queried attribute sets cannot match  $(M^*, \rho^*)$ .
- **Guess:**  $\mathcal{A}$  outputs a guess bit  $b' \in \{0, 1\}$  and wins the game if  $b' = b$ .

The advantage of  $\mathcal{A}$  in the IND-CPA game is defined as follows:

$$\text{Adv}_{\text{AA-LU-CPABE}}^{\text{IND-CPA}}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|.$$

**Definition 8.** An AA-LU-CPABE scheme is fully-secure if no probabilistic polynomial-time (PPT) attacker can break the IND-CPA game with non-negligible advantage.

**The DishonestUser game.** The DishonestUser game for AA-LU-CPABE scheme is defined as follows.

- **Setup:**  $\mathcal{B}$  chooses a security parameter  $\lambda$ , and runs the **Setup** algorithm and sends the system public key  $PK$  to  $\mathcal{A}$ .
- **Key Query Phase:** For  $i \in [t_q]$ , while  $t_q$  is the number of key queries,  $\mathcal{A}$  and  $\mathcal{B}$  engage in the key generation protocol **KeyGen** to generate corresponding decryption keys  $SK_{ID_i, S_i}$  with respect to  $(ID_i, S_i)$ .  $\mathcal{A}$  gets the decryption keys  $\{SK_{ID_i, S_i}\}_{i \in [t_q]}$  and runs key sanity checks to ensure that they are well-formed. It aborts if any check fails.
- **Key Forgery Phase:**  $\mathcal{A}$  submits a decryption key  $SK_{ID^*, S^*}^*$  corresponding to  $(ID^*, S^*)$  to  $\mathcal{B}$ , where  $S^* \in \{S_1, S_2, \dots, S_{t_q}\}$ . If either of the following two cases is true,  $\mathcal{A}$  wins the game.
  1.  $\text{Trace}_{\text{wp}}(PK, SK_{ID^*, S^*}^*) \notin \{\top, ID_1, ID_2, \dots, ID_{t_q}\}$ .
  2.  $\text{Trace}_{\text{wp}}(PK, SK_{ID^*, S^*}^*) = ID_j \in \{ID_1, ID_2, \dots, ID_{t_q}\}$  and  $\text{Judge}_{\text{wp}}(PK, SK_{ID_j, S_j}, SK_{ID^*, S^*}^*) \rightarrow \text{innocent}$ .

The advantage of  $\mathcal{A}$  in the DishonestUser game is defined as follows:

$$\text{Adv}_{\text{AA-LU-CPABE}}^{\text{DishonestUser}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}].$$

**Definition 9.** An AA-LU-CPABE scheme is DishonestUser secure if all PPT attackers have at most a negligible advantage in the above DishonestUser game.

**The DishonestAuthority Game.** The DishonestAuthority game for AA-LU-CPABE scheme is defined as follows.

- **Setup:**  $\mathcal{A}$  (as a malicious authority) generates the system public key  $PK$ , and sends  $PK$ , a user's  $(ID^*, S^*)$  to  $\mathcal{B}$ .  $\mathcal{B}$  performs a sanity check on  $PK$  and  $(ID^*, S^*)$ , and it aborts if the check fails.
- **Key Generation Phase:**  $\mathcal{A}$  and  $\mathcal{B}$  engage in the key generation protocol **KeyGen** to generate a decryption key  $SK_{ID^*, S^*}$  corresponding to  $(ID^*, S^*)$ .  $\mathcal{B}$  gets  $SK_{ID^*, S^*}$  and runs a key sanity check to ensure that it is well-formed. It aborts if the check fails.
- **Output:**  $\mathcal{A}$  outputs a decryption key  $SK_{ID^*, S^*}^*$  and succeeds if  $\text{Trace}_{\text{wp}}(PK, SK_{ID^*, S^*}^*) \rightarrow ID^*$  and  $\text{Judge}_{\text{wp}}(PK, SK_{ID^*, S^*}, SK_{ID^*, S^*}^*) \rightarrow \text{guilty}$ .

The advantage of  $\mathcal{A}$  in the DishonestAuthority game is defined as:

$$\text{Adv}_{\text{AA-LU-CPABE}}^{\text{DishonestAuthority}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}].$$

**Definition 10.** An AA-LU-CPABE scheme is DishonestAuthority secure if all PPT attackers have at most a negligible advantage in the above DishonestAuthority game.

## 4 AA-LU-CPABE Construction

### 4.1 Construction

- **Setup( $1^\lambda$ ):** The attribute authority takes a security parameter  $\lambda$  as inputs and runs the group generator  $\mathcal{G}$  to get  $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, \hat{e})$ , where  $p_1, p_2, p_3$

are distinct primes,  $\mathbb{G}$  and  $\mathbb{G}_T$  are two cyclic groups of order  $N = p_1 p_2 p_3$ , and  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  is a bilinear map. Let  $\mathbb{G}_{p_i}$  be the subgroup of order  $p_i$  in  $\mathbb{G}$ , and  $g \in \mathbb{G}_{p_1}$  and  $X_3 \in \mathbb{G}_{p_3}$  be the generator of  $\mathbb{G}_{p_1}$  and  $\mathbb{G}_{p_3}$ , respectively. The attribute authority defines three collision-resistant hash functions  $H_0 : \mathbb{G}_{p_1} \rightarrow \mathbb{G}_{p_1}$ ,  $H_1 : \mathbb{G}_{p_1 p_3} \rightarrow \mathbb{Z}_N$  and  $H : \{0, 1\}^* \rightarrow \mathbb{G}_{p_1}$ . It also chooses two distinct primes  $p$  and  $q$  of equal length such that  $p, q \notin \{p_1, p_2, p_3\}$ ,  $\gcd(pq, (p-1)(q-1)) = 1$  and sets  $n = pq$ ,  $\omega = \varphi(n)$ , where  $\varphi(\cdot)$  is Euler's totient function. Then the attribute authority chooses  $a, \alpha, \beta \in_R \mathbb{Z}_N$ ,  $X_1 \in \mathbb{G}_{p_1}$  and computes  $Y = \hat{e}(g, g)^\alpha$ . Finally, the system public key is published as  $PK = \langle N, n, \omega, g, g^\alpha, g^\beta, X_1, Y \rangle$ , and the master key is  $MK = \langle a, \alpha, \beta, X_3 \rangle$ .

– **KeyGen**( $PK, MK, ID, S$ ): Let  $S$  be the attribute set for the user  $ID$  who obtains the corresponding decryption key. The user and the attribute authority initiate the following key generation protocol.

1. The user chooses  $r \in_R \mathbb{Z}_N^*$  with  $\gcd(r, N) = 1$  and computes  $R_u = g^r$ . Then it sends  $ID$ , the attribute set  $S$  and  $R_u$  to the attribute authority. Besides, it runs an interactive ZK-POK of the discrete log of  $R_u$  with respect to  $g$  with the attribute authority.
2. If and only if the ZK-POK is valid, the attribute authority proceeds to do the following. It first chooses  $t \in_R \mathbb{Z}_n^*$  with  $t \notin \{p, q\}$ ,  $c_0, \bar{c} \in_R \mathbb{Z}_N$  with  $\gcd(\bar{c}, N) = 1$  and random elements  $R_0, R'_0, R''_0, \{R_i\}_{i \in S}$  in  $\mathbb{G}_{p_3}$  based on  $X_3$ . Then the attribute authority sets  $c = c_0 + \bar{c}$ ,  $h = 1 + n$  and computes  $\bar{T} = h^{ID} t^{\bar{c}n} \pmod{n^2}$ ,  $\bar{L}_1 = g^{ac} R'_0$ ,  $\bar{L}_2 = g^{c_0} R''_0$ ,  $\bar{K} = g^{\frac{\alpha}{a+\bar{T}+H_1(\bar{L}_2)}} H_0(R_u)^{\frac{\beta}{a+\bar{T}+H_1(\bar{L}_2)}} X_1^c R_0$ ,  $\{\bar{K}_i = H(i)^{(a+\bar{T}+H_1(\bar{L}_2))c} R_i\}_{i \in S}$ , and then sends  $\bar{SK} = \langle \bar{K}, \bar{T}, \bar{L}_1, \bar{L}_2, \bar{c}, \{\bar{K}_i\}_{i \in S} \rangle$  to the user.
3. The user checks whether

$$\hat{e}(\bar{K}, g^a g^{\bar{T}+H_1(\bar{L}_2)}) = \hat{e}(\bar{L}_1 (\bar{L}_2 g^{\bar{c}})^{\bar{T}+H_1(\bar{L}_2)}, X_1) \hat{e}(H_0(R_u), g^\beta) Y,$$

$\hat{e}(\bar{L}_1, g) = \hat{e}(\bar{L}_2 g^{\bar{c}}, g^a)$ , and  $\forall i \in S, \hat{e}(H(i), \bar{L}_1 (\bar{L}_2 g^{\bar{c}})^{\bar{T}+H_1(\bar{L}_2)}) = \hat{e}(\bar{K}_i, g)$ . The user aborts the interaction if one of the above checks fails. Otherwise, the users computes  $T_0 = \frac{\bar{c}}{r}$  and sets the decryption key as

$$SK_{ID,S} = \langle K = \bar{K} g^{T_0}, T = \bar{T}, L_1 = \bar{L}_1, L_2 = \bar{L}_2, R_u, T_0, \{K_i = \bar{K}_i\}_{i \in S} \rangle.$$

– **Encrypt**( $PK, m, (M, \rho)$ ): The encryptor first chooses  $s \in_R \mathbb{Z}_N$  and then sets a random vector  $\mathbf{y} = (s, y_2, \dots, y_n)^\top$ , where  $y_2, \dots, y_n$  are used to share the encryption exponent  $s$ . For  $i = 1, \dots, \ell$ , it calculates  $\lambda_i = M_i \cdot \mathbf{y}$ , where  $M_i$  is the vector corresponding to the  $i$ th row of  $M$ . Then it calculates  $C = m Y^s$ ,  $C_0 = g^s$ ,  $C_1 = (g^a)^s$ ,  $C_2 = (g^\beta)^s$ ,  $\{C_{j,1} = X_1^{\lambda_j} H(\rho(j))^{-r_j}, C_{j,2} = g^{r_j}\}_{j \in [l]}$ , where  $r_j$  is randomly chosen in  $\mathbb{Z}_N$ . Finally, the encryptor sets ciphertext as

$$CT = \langle C, C_0, C_1, C_2, \{C_{j,1}, C_{j,2}\}_{j \in [l]} \rangle.$$

– **Decrypt**( $PK, CT, SK_{ID,S}$ ):  $CT = \langle C, C_0, C_1, C_2, \{C_{j,1}, C_{j,2}\}_{j \in [l]} \rangle$  under  $(A, \rho)$  is decrypted by a user  $(ID, S)$  with a decryption key  $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i \in S} \rangle$  as follows. The decryptor first checks whether

$S$  satisfies  $(A, \rho)$ . If not, the algorithm returns  $\perp$ . Otherwise, there must exist coefficients  $\{\omega_j \in \mathbb{Z}_N \mid \rho(j) \in S\}$  such that  $\sum_{\rho(j) \in S} \omega_j M_j = (1, 0, \dots, 0)$ , so  $\sum_{\rho(j) \in S} \omega_j \lambda_j = s$ . Then, the decryptor computes  $m = \frac{C}{B}$ , where

$$B = \frac{\hat{e}(C_0^{T+H_1(L_2)} C_1, K) (\hat{e}(C_2, H_0(R_u)) \hat{e}(C_0, (g^a g^{T+H_1(L_2)})^{T_0}))^{-1}}{\prod_{\rho(j) \in S} (\hat{e}(C_{j,1}, L_1(L_2 R_u^{T_0})^{T+H_1(L_2)}) \hat{e}(C_{j,2}, K_{\rho(j)})) \omega_j}.$$

- **Trace<sub>wp</sub>**( $PK, SK_{ID,S}$ ): If  $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i \in S} \rangle$  satisfying all the following checks, it is a well-formed decryption key, otherwise it is not well-formed and the algorithm outputs  $\top$ . Key Sanity Check is:

1.  $T \in \mathbb{Z}_{n^2}, T_0 \in \mathbb{Z}_N, K, L_1, L_2, R_u, K_i \in \mathbb{G}$ .
2.  $\hat{e}(L_1, g) = \hat{e}(L_2 R_u^{T_0}, g^a)$ .
3.  $\hat{e}(g^{-T_0} K, g^a g^{T+H_1(L_2)}) = \hat{e}(L_1(L_2 R_u^{T_0})^{T+H_1(L_2)}, X_1) \hat{e}(H_0(R_u), g^\beta) Y$ .
4.  $\exists i \in S$ , such that  $\hat{e}(H(i), L_1(L_2 R_u^{T_0})^{T+H_1(L_2)}) = \hat{e}(K_i, g)$ .

If  $SK_{ID,S}$  is well-formed, the algorithm will extract the identity  $ID$  from  $T = h^{ID} t^{\bar{c}n} \pmod{n^2}$  in  $SK_{ID,S}$  as follows. Note that  $T^\omega = h^{\omega ID} t^{\bar{c}n\omega} \pmod{n^2} = (1+n)^{\omega ID} t^{\bar{c}n\omega(n)} \pmod{n^2} = 1+n\omega ID \pmod{n^2}$ , hence it outputs the identity  $ID = \frac{(T^\omega \pmod{n^2})^{-1}}{n\omega}$ , which is used to identify the possible malicious user.

- **Judge<sub>wp</sub>**( $PK, SK_{ID,S}, SK_{ID,S}^*$ ): Suppose a user  $(ID, S)$  with the decryption key  $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i \in S} \rangle$  is identified as a malicious user by the system based on the traced key

$$SK_{ID,S}^* = \langle K^*, T^*, L_1^*, L_2^*, R_u^*, T_0^*, \{K_i^*\}_{i \in S} \rangle,$$

but it claims to be innocent and framed by the system. The user and the judge interact in the following protocol.

1. The user sends the decryption key  $SK_{ID,S}$  to the judge. The judge checks if  $SK_{ID,S}$  passes the key sanity checks used in the tracing algorithm and aborts if the check fails.
2. Otherwise, the judge tests whether  $T_0 = T_0^*$  or not. If no, it outputs innocent to indicate that the user is innocent and is framed by the system. Otherwise, it outputs guilty to indicate that  $SK_{ID,S}^*$  is maliciously leaked by the user.

## 4.2 Security Analysis

**Theorem 1.** *If Assumptions 1, 2 and 3 hold, then the proposed AA-LU-CPABE scheme is semantically secure.*

*Proof.* It's noted that the proposed AA-LU-CPABE scheme  $\Pi$  is based on the CP-ABE scheme [10] denoted by  $\Pi_o$ . Because the scheme  $\Pi_o$  is adaptively chosen attribute sets and chosen plaintexts secure under the assumptions 1, 2 and 3, if we can reduce the security of  $\Pi$  to that of  $\Pi_o$ , then the proposed AA-LU-CPABE scheme is secure in the IND-CPA security model under the assumptions 1, 2 and 3. In the following, we will show that any PPT attacker  $\mathcal{A}$  with a non-negligible advantage  $\text{Adv}_{\text{AA-LU-CPABE}}^{\text{IND-CPA}}(\mathcal{A}) = \epsilon$  in the proposed security model against  $\Pi$  can be

used to design a PPT simulator  $\mathcal{B}$ , which can break the security of  $\Pi_o$  with an advantage  $\text{Adv}_{\text{CPABE}}^{\text{IND-CPA}}(\mathcal{B}) = \epsilon$ . The simulator  $\mathcal{B}$  acts as the challenger and interacts with  $\mathcal{A}$  in the IND-CPA security model. The simulation proceeds as follows:

**Setup.** The challenger  $\mathcal{B}$  receives public parameters  $\langle N, g, g^\gamma, Y = \hat{e}(g, g)^\alpha, \{U_i = g^{u_i}\}_{i \in \mathcal{U}_o} \rangle$  from the challenger  $\mathcal{B}_o$  of  $\Pi_o$ , where  $\mathcal{U}_o$  is the attribute universe of  $\Pi_o$  and it satisfies  $|\mathcal{U}_o| \geq q_H$ , where  $q_H$  is the number of hash queries to  $H$ .  $\mathcal{B}$  also chooses two distinct primes  $p$  and  $q$  of equal length such that  $\text{gcd}(pq, (p-1)(q-1)) = 1$  and sets  $n = pq$ ,  $\omega = \varphi(n)$ . Then  $\mathcal{B}$  chooses  $a, \alpha, \beta \in_R \mathbb{Z}_N$ , sets  $X_1 = g^\gamma$  and the system public key as  $PK = \langle N, n, \omega, g, g^\alpha, g^\beta, X_1, Y \rangle$ . During the game,  $\mathcal{A}$  will consult  $\mathcal{B}$  for answers to the random oracles  $H_0, H_1$  and  $H$ .  $\mathcal{B}$  keeps three tables  $\mathcal{L}_0, \mathcal{L}_1$  and  $\mathcal{L}$  to store the answers used in  $H_0, H_1$  and  $H$ , respectively. Finally  $\mathcal{B}$  sends  $PK$  to  $\mathcal{A}$ .

**Phase 1.** The adversary  $\mathcal{A}$  makes the following queries.

- **Hash Oracle**  $\mathcal{O}_{H_0}(x_0)$ : Whenever there is a query on  $H_0$  for input  $x_0$ ,  $\mathcal{B}$  first looks if there is an item containing  $x_0$  in  $\mathcal{L}_0$ . If it is, the previous defined value is returned. Otherwise, it chooses  $\gamma' \in_R \mathbb{Z}_N$  and sets  $H_0(x_0) = g^{\gamma'}$ , adds the entry  $\langle x_0, \gamma', H_0(x_0) = g^{\gamma'} \rangle$  to  $\mathcal{L}_0$  and returns  $g^{\gamma'}$ .
- **Hash Oracle**  $\mathcal{O}_{H_1}(x_1)$ : Whenever there is a query on  $H_1$  for input  $x_1$ ,  $\mathcal{B}$  first looks if there is an item containing  $x_1$  in  $\mathcal{L}_1$ . If it is, the previous defined value is returned. Otherwise, it chooses  $\gamma'' \in_R \mathbb{Z}_N$ , adds the entry  $\langle x_1, \gamma'', H_1(x_1) = \gamma'' \rangle$  to  $\mathcal{L}_1$  and returns  $\gamma''$ . Note that, during the process of answering key generation queries,  $\mathcal{B}$  will adaptively update  $\mathcal{L}_1$ .
- **Hash Oracle**  $\mathcal{O}_H(x)$ : Whenever there is a query on  $H$  for input  $x$ ,  $\mathcal{B}$  first looks if there is an item containing  $x$  in  $\mathcal{L}$ . If it is, the previous defined value is returned. Otherwise, it sets  $H(x) = U_x$ , adds the entry  $\langle x, H(x) = U_x \rangle$  to  $\mathcal{L}$  and returns  $U_x$ .
- **KeyGen Oracle**  $\mathcal{O}_{\text{KeyGen}}(ID, S)$ : Suppose  $\mathcal{A}$  submits an identity  $ID$  and an attribute set  $S$  in a secret key query.  $\mathcal{B}$  sends  $S$  to  $\mathcal{B}_o$  and obtains the corresponding decryption key  $SK_S = \langle \hat{K} = g^\alpha g^{\gamma^c} R_0, \hat{L} = g^c R'_0, \{\hat{K}_i = U_i^c R_i\}_{i \in S} \rangle$ . Recall that during the key generation protocol, the key applicant chooses  $r \in_R \mathbb{Z}_N^*$  and computes  $R_u = g^r$ . Besides, the key applicant gives to the authority a zero-knowledge proof of knowledge of the discrete log of  $R_u$  with respect to  $g$ . The authority can extract  $r$  with all but negligible probability by using Extractor on the key applicant during the proof of knowledge protocol.  $\mathcal{B}$  chooses  $t \in_R \mathbb{Z}_n^*$ ,  $\bar{c} \in_R \mathbb{Z}_N$ , sets  $h = 1 + n$  and computes  $T = \bar{T} = h^{ID} t^{\bar{c}n} \text{ mod } n^2$ . Also,  $\mathcal{B}$  chooses  $R''_0 \in_R \mathbb{G}_{p_3}$  by using  $X_3, \gamma'' \in_R \mathbb{Z}_N$  and implicitly sets  $c_0 = \hat{c}/(a + T + \gamma'') - \bar{c} \text{ mod } N$ , and hence  $c = c_0 + \bar{c} = \hat{c}/(a + T + \gamma'')$ . Then  $\mathcal{B}$  sets  $\bar{L}_2 = \hat{L}^{\frac{1}{a+T+\gamma''}} R''_0 = g^c R'_0{}^{\frac{1}{a+T+\gamma''}} R''_0$  and returns  $\gamma''$  for the hash query  $H_1(\bar{L}_2)$ , that is,  $\mathcal{B}$  will add the entry  $\langle \bar{L}_2, \gamma'', H_1(\bar{L}_2) = \gamma'' \rangle$  to  $\mathcal{L}_1$ . Therefore,  $c = \hat{c}/(a + T + H_1(\bar{L}_2))$ . Subsequently,  $\mathcal{B}$  computes  $\bar{L}_1 = \hat{L}^{\frac{\alpha}{a+T+H_1(\bar{L}_2)}} = g^{ac} R'_0{}^{\frac{\alpha}{a+T+H_1(\bar{L}_2)}}$ ,  $\bar{K} = (\hat{K})^{\frac{1}{a+T+H_1(\bar{L}_2)}} H_0(R_u)^{\frac{\beta}{a+T+H_1(\bar{L}_2)}} = g^{\frac{\alpha}{a+T+H_1(\bar{L}_2)}} H_0(R_u)^{\frac{\beta}{a+T+H_1(\bar{L}_2)}} X_1^c R_0^{\frac{1}{a+T+H_1(\bar{L}_2)}}$ , and  $\{\bar{K}_i = \hat{K}_i = U_i^c R_i = H(i)^{(a+T+H_1(\bar{L}_2))c} R_i\}_{i \in S}$ . Subsequently,  $\mathcal{B}$  computes  $T_0 = \frac{\bar{c}}{r}$  and sets  $K =$



$\overline{K}g^{T_0}, L_1 = \overline{L}_1, L_2 = \overline{L}_2, \{K_i = \overline{K}_i\}_{i \in S}$ . Finally,  $\mathcal{B}$  returns the decryption key  $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i \in S} \rangle$ .

**Challenge.** The adversary  $\mathcal{A}$  submits two messages  $m_0$  and  $m_1$  of equal length and an LSSS access structure  $(M^*, \rho^*)$  to  $\mathcal{B}$ . Note that  $(M^*, \rho^*)$  cannot be satisfied by any of the queried attribute sets. Then  $\mathcal{B}$  sends  $m_0, m_1$  and  $(M^*, \rho^*)$  to  $\mathcal{B}_o$  to obtain the challenge ciphertext of  $\Pi_o$  as follows.

$$\widehat{CT} = \langle \widehat{C} = m_b Y^s, \widehat{C}_0 = g^s, \{\widehat{C}_{j,1} = g^{\gamma \lambda_j} U_{\rho(j)}^{-r_j}, \widehat{C}_{j,2} = g^{r_j}\}_{j \in [\ell]} \rangle.$$

$\mathcal{B}$  sets  $C = \widehat{C}, C_0 = \widehat{C}_0, C_1 = (\widehat{C}_0)^a = g^{as}, C_2 = (\widehat{C}_0)^\beta = g^{\beta s}, C_{j,1} = \widehat{C}_{j,1} = X_1^{\lambda_j} H(\rho(j))^{-r_j}$  and  $C_{j,2} = \widehat{C}_{j,2}$ . Finally,  $\mathcal{B}$  gives to  $\mathcal{A}$  the challenge ciphertext

$$CT = \langle C, C_0, C_1, C_2, \{C_{j,1}, C_{j,2}\}_{j \in [\ell]} \rangle.$$

**Phase 2.** The same as **Phase 1** except that the queried attribute sets cannot match  $(M^*, \rho^*)$ .

**Guess.** The adversary  $\mathcal{A}$  outputs a guess bit  $b'$  of  $b$ .  $\mathcal{B}$  just sends  $b'$  to  $\mathcal{B}_o$ . It easily follows that  $\text{Adv}_{\text{CPABE}}^{\text{IND-CPA}}(\mathcal{B}) = \text{Adv}_{\text{AA-LU-CPABE}}^{\text{IND-CPA}}(\mathcal{A}) = \epsilon$ .  $\blacksquare$

**Theorem 2.** *If Assumption 2 and  $\ell$ -SDH assumption hold, then the proposed AA-LU-CPABE scheme is DishonestUser secure provided that  $t_q < \ell$ , where at most  $t_q$  key queries are issued during the DishonestUser security game.*

*Proof.* Suppose there is a PPT adversary  $\mathcal{A}$  that wins the traceability game with a non-negligible advantage  $\epsilon$  after making  $t_q$  key queries. Without loss of generality, assuming  $\ell = t_q + 1$ , we construct a PPT algorithm  $\mathcal{B}$  that has a non-negligible advantage in breaking Assumption 2 or  $\ell$ -SDH assumption.  $\mathcal{B}$  is given instances as follows.

- $\mathcal{B}$  is given an instance of Assumption 2 problem: Let  $\mathbb{G}$  be a bilinear group of order  $N = p_1 p_2 p_3$ , where  $p_1, p_2, p_3$  are distinct primes,  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  be a bilinear map, and  $\mathbb{G}_{p_i}$  be the subgroup of order  $p_i$  in  $\mathbb{G}$ ,  $\hat{g}, X_1 \in \mathbb{G}_{p_1}, X_2, Y_2 \in \mathbb{G}_{p_2}$  and  $X_3, Y_3 \in \mathbb{G}_{p_3}$ .  $b \in \{0, 1\}$ , and  $X \in \mathbb{G}$  if  $b = 0$ ,  $X \in \mathbb{G}_{p_1 p_3}$  if  $b = 1$ .  $\mathcal{B}$  is given an instance  $\text{IN}_{\mathcal{A}2} = (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, \hat{g}, X_1 X_2, X_3, Y_2 Y_3, X)$ .
- $\mathcal{B}$  is given an instance of  $\ell$ -SDH problem: Let  $\mathbb{G}$  be a bilinear group of order  $N = p_1 p_2 p_3$ , where  $p_1, p_2, p_3$  are distinct primes,  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  be a bilinear map,  $\mathbb{G}_{p_i}$  be the subgroup of order  $p_i$  in  $\mathbb{G}$ ,  $a \in \mathbb{Z}_{p_1}^*$  and  $\hat{g} \in \mathbb{G}_{p_1}$ .  $\mathcal{B}$  is given an instance  $\text{IN}_{\text{SDH}} = (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, \hat{g}, \hat{g}^a, \dots, \hat{g}^{a^\ell}, p_1, p_2, p_3)$ .

The goal of  $\mathcal{B}$  is to output a bit  $b' \in \{0, 1\}$  to determine  $X \in \mathbb{G}$  or  $X \in \mathbb{G}_{p_1 p_3}$  for solving the assumption 2 problem, and a pair  $(T_r, \omega_r)$  satisfying  $\omega_r = \hat{g}^{a+T_r}$  for solving the  $\ell$ -SDH problem.  $\mathcal{B}$  will make use of  $\mathcal{A}$  to break Assumption 2 or  $\ell$ -SDH assumption. After the setup phase,  $\mathcal{A}$  can get the system public key from  $\mathcal{B}$ . During the key query phase, where at most  $t_q$  key queries are issued,  $\mathcal{A}$  submits identity and attribute set pairs to  $\mathcal{B}$  to get decryption keys. Then, after the key forgery phase,  $\mathcal{A}$  submits a decryption key  $SK^*$  corresponding to

$(ID^*, S^*)$  to  $\mathcal{B}$ . The detailed queries will be given in the full version of the paper due to the space limitation.

Finally, based on the complete probability formula, we can know that  $\mathcal{B}$  can break Assumption 2 with advantage at least  $\frac{\epsilon}{8}$  or break  $\ell$ -SDH assumption with advantage at least  $\frac{\epsilon}{8}$ . ■

**Theorem 3.** *If the discrete log assumption holds in  $\mathbb{G}_{p_1}$ , then the proposed AA-LU-CPABE scheme is DishonestAuthority secure.*

*Proof.* Suppose there is a PPT adversary  $\mathcal{A}$  that has a non-negligible advantage in winning the DishonestAuthority game for our AA-LU-CPABE scheme, we construct a PPT algorithm  $\mathcal{B}$  that has a non-negligible advantage in breaking the discrete log assumption in  $\mathbb{G}_{p_1}$ .  $\mathcal{B}$  proceeds as follows.  $\mathcal{B}$  receives from  $\mathcal{A}$  the system public key  $PK = \langle N, n, \omega, g, g^a, g^\beta, X_1, Y \rangle$  and a user's identity and attribute set  $(ID^*, S^*)$ . Then  $\mathcal{B}$  sends  $g$  to the discrete log problem challenger and obtains an instance  $(g, R_u = g^r)$  of the discrete log assumption.

$\mathcal{B}$  engages in the key generation protocol with  $\mathcal{A}$  to get a decryption key for the user  $(ID^*, S^*)$ . It sends  $R_u$  to  $\mathcal{A}$  and provides a zero-knowledge proof of knowledge of the discrete log of  $R_u$ . On the other hand,  $\mathcal{B}$  receives from  $\mathcal{A}$  a primary decryption key  $\overline{SK} = \langle \overline{K}, \overline{T}, \overline{L}_1, \overline{L}_2, \overline{c}, \{K_i\}_{i \in S^*} \rangle$ . Then,  $\mathcal{B}$  performs the key sanity checks.  $\mathcal{B}$  aborts the interaction if the checks fail. Otherwise,  $\mathcal{B}$  chooses  $r' \in_R \mathbb{Z}_N^*$ , computes  $T_0 = \frac{\overline{c}}{r'}$  and sets the decryption key based on the algorithm. Now with a non-negligible advantage,  $\mathcal{A}$  outputs a decryption key  $SK_{ID^*, S^*}^*$  and it succeeds in framing the user  $(ID^*, S^*)$ . Hence,  $SK_{ID^*, S^*}^*$  has the form of  $SK_{ID^*, S^*}^* = \langle K = \overline{K}g^{T_0^*}, T, L_1, L_2, R_u, T_0^*, \{K_i\}_{i \in S^*} \rangle$ . Finally,  $\mathcal{B}$  calculates  $\frac{\overline{c}}{T_0^*}$  as the solution of the discrete log problem  $(g, R_u)$ . More details will be given in the full version of the paper due to the space limitation. ■

## 5 Conclusions and Future Work

In this paper, we propose an AA-LU-CPABE scheme that simultaneously supports user traceability, authority accountability and auditing in the white-box model. Our scheme needs almost no storage for tracing in that it does not need to maintain an identity table of users for tracing. The proposed AA-LU-CPABE scheme is proven secure in the random oracle model against adaptive adversaries, and it is highly expressive and can take any monotonic access structures as ciphertext policies. It would be interesting to construct AA-LU-CPABE schemes with desirable security and performance features in the black-box model.

**Acknowledgements.** We are grateful to the anonymous reviewers for their invaluable suggestions. This work is supported by National Natural Science Foundation of China (No. 61402366, 61472091, 61272037, 61472472, and 61272457), Program for New Century Excellent Talents in University (No. NCET-13-0946), Distinguished Young Scholars Fund of Department of Education, Guangdong Province (No. Yq2013126), Natural Science Basic Research Plan in Shaanxi Province (No. 2015JQ6236), and Scientific Research Program Funded by Shaanxi Provincial Education Department (No. 15JK1686).

## References

1. Balu, A., Kuppusamy, K.: An expressive and provably secure ciphertext-policy attribute-based encryption. *Inf. Sci.* **276**, 354–362 (2014)
2. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Technion-Israel Institute of technology, Faculty of computer science (1996)
3. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334. IEEE, Los Alamitos (2007)
5. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
6. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: CCS 2007, pp. 456–465. ACM, New York (2007)
7. Deng, H., Wu, Q., Qin, B., Mao, J., Liu, X., Zhang, L., Shi, W.: Who Is touching my cloud. In: Kutylowski, M., Vaidya, J. (eds.) ICAIS 2014, Part I. LNCS, vol. 8712, pp. 362–379. Springer, Heidelberg (2014)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006, pp. 89–98. ACM, New York (2006)
9. Lai, J., Deng, R.H., Li, Y.: Expressive CP-ABE with partially hidden access structures. In: ASIACCS 2012, pp. 18–19. ACM, New York (2012)
10. 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)
11. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
12. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: ASIACCS 2011, pp. 386–390. ACM, New York (2011)
13. Li, J., Ren, K., Kim, K.: A2be: Accountable attribute-based encryption for abuse free access control. *Cryptology ePrint Archive*, Report 2009/118 (2009)
14. Li, J., Ren, K., Zhu, B., Wan, Z.: Privacy-aware attribute-based encryption with user accountability. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 347–362. Springer, Heidelberg (2009)
15. Liu, Z., Cao, Z., Wong, D.S.: White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. *IEEE Trans. Inf. Forensics Secur.* **8**(1), 76–88 (2013)
16. Liu, Z., Cao, Z., Wong, D.S.: Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay. In: CCS 2013, pp. 475–486. ACM, New York (2013)
17. Liu, Z., Cao, Z., Wong, D.S.: Traceable CP-ABE: how to trace decryption devices found in the wild. *IEEE Trans. Inf. Forensics Secur.* **10**(1), 55–68 (2015)
18. Liu, Z., Wong, D.S.: Traceable CP-ABE on prime order groups: Fully secure and fully collusion-resistant blackbox traceable. *Cryptology ePrint Archive*, Report 2015/850 (2015)

19. Liu, Z., Wong, D.S.: Practical ciphertext-policy attribute-based encryption: traitor tracing, revocation, and large universe. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) *Applied Cryptography and Network Security*. LNCS, vol. 9092, pp. 127–146. Springer, Switzerland (2015)
20. Ning, J., Cao, Z., Dong, X., Wei, L., Lin, X.: Large universe ciphertext-policy attribute-based encryption with white-box traceability. In: Kutylowski, M., Vaidya, J. (eds.) *ICAIS 2014, Part II*. LNCS, vol. 8713, pp. 55–72. Springer, Heidelberg (2014)
21. Ning, J., Dong, X., Cao, Z., Wei, L.: Accountable Authority Ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) *ESORICS 2015, Part II*. LNCS, vol. 9327, pp. 270–289. Springer, Switzerland (2015)
22. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden cryptor-specified access structures. In: Bellare, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) *Applied Cryptography and Network Security*. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008)
23. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
24. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: *CCS 2013*, pp. 463–474. ACM, New York (2013)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
26. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) *CRYPTO 1989*. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
27. Shi, Y., Zheng, Q., Liu, J., Han, Z.: Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. *Inf. Sci.* **295**, 221–231 (2015)
28. Yu, S., Ren, K., Lou, W., Li, J.: Defending against key abuse attacks in KP-ABE enabled broadcast systems. In: Chen, Y., Dimitriou, T.D., Zhou, J. (eds.) *SecureComm 2009*. LNICST, vol. 19, pp. 311–329. Springer, Heidelberg (2009)
29. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: *ASIACCS 2010*, pp. 261–270. ACM, New York (2010)
30. Zhang, X., Jin, C., Li, C., Wen, Z., Shen, Q., Fang, Y., Wu, Z.: Ciphertext-policy attribute-based encryption with user and authority accountability. In: Thuraisingham, B., et al. (eds.) *SecureComm 2015*. LNICST, vol. 164, pp. 500–518. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-28865-9\\_27](https://doi.org/10.1007/978-3-319-28865-9_27)
31. Zhang, Y., Chen, X., Li, J., Li, H., Li, F.: Fdr-abe: Attribute-based encryption with flexible and direct revocation. In: *INCoS 2013*, pp. 38–45. IEEE, Los Alamitos (2013)
32. Zhang, Y., Chen, X., Li, J., Li, H., Li, F.: Attribute-based data sharing with flexible and direct revocation in cloud computing. *KSII Transactions on Internet & Inf. Syst.* **8**(11), 4028–4049 (2014)
33. Zhang, Y., Chen, X., Li, J., Wong, D.S., Li, H.: Anonymous attribute-based encryption supporting efficient decryption test. In: *ASIACCS 2013*, pp. 511–516. ACM, New York (2013)
34. Zhang, Y., Zheng, D., Chen, X., Li, J., Li, H.: Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) *ProvSec 2014*. LNCS, vol. 8782, pp. 259–273. Springer, Heidelberg (2014)

# A Cloud-Based Access Control Scheme with User Revocation and Attribute Update

Peng Zhang<sup>1(✉)</sup>, Zehong Chen<sup>1</sup>, Kaitai Liang<sup>2</sup>, Shulan Wang<sup>1</sup>,  
and Ting Wang<sup>1,3</sup>

<sup>1</sup> ATR Key Laboratory of National Defense Technology,  
College of Information Engineering, Shenzhen University, Shenzhen, China  
{zhangp,zhchen,wangt}@szu.edu.cn, wangshulan@email.szu.edu.cn

<sup>2</sup> Department of Computer Science, Aalto University, Espoo, Finland  
kaitai.liang@aalto.fi

<sup>3</sup> School of Computer Science and Engineering,  
South China University of Technology, Guangzhou, China

**Abstract.** Ciphertext-policy attribute-based encryption (CP-ABE) is a well-known cryptographic technology for guaranteeing data confidentiality but also fine-grained data access control. It enables data owners to define flexible access policy for cloud-based data sharing. However, the user revocation and attribute update problems existing in CP-ABE systems that are long-standing unsolved in the literature. In this paper, we propose the first access control (CP-ABE) scheme supporting user revocability and attribute update. Specifically, the user revocation is defined in the identity-based setting that does not conflict our attribute-based design. The cost brought by attribute update is efficient in the sense that we only concentrate on the update of the ciphertexts associated with the corresponding updated attribute. Moreover, the security analysis shows that the proposed scheme is secure under the decisional Bilinear Diffie-Hellman assumption.

**Keywords:** Cloud · Access control · Attribute-based encryption · User revocability · Attribute update

## 1 Introduction

In cloud setting, network users usually outsource their personal data to cloud service providers [16]. Accordingly, the service providers may have chances to fully control the personal data that may incur undesirable security and privacy issues. It is undeniable that the abuse of the data by the providers or unauthorized access by malicious adversaries may become potential security threats for the data owners. To achieve access control for the data, encryption technology can be seen as a security guarantee, so that only the users with qualified rights can gain access to the data.

Ciphertext policy attribute-based encryption (CP-ABE), one of the most fine-grained access control (encryption) techniques, was first introduced by

Bethencourt *et al.* [1] in 2007. In a CP-ABE scheme, a ciphertext is associated with an access policy, and a user's secret key is associated with an attribute set. A user can decrypt a given ciphertext if and only if his attribute set satisfies the access structure of the ciphertext. CP-ABE with flexible one-to-many encryption mode provides a fine-grained and non-interactive access control mechanism for users' encrypted data [14]. Over the past few years, many CP-ABE schemes [2–5, 11, 14, 15, 18] have been proposed to achieve various functional purposes. Nonetheless, user revocability is still an elusive problem in the context of attribute-based encryption. A revoked user must have the rights to access the data no more, even if his attributes satisfy the outdated access policy.

For instance, an employer can revoke employees' rights to access to some sensitive project data while the employees are no longer the project members. To do so, we must guarantee that a given ciphertext of the project data should be updated so that the revoked user cannot reuse his outdated secret key to compromise the data. Since different users may have the same functional secret key associated with the identical attribute set, the user revocability is difficult to achieve in the attribute-based encryption (ABE) setting. In recent years, some interesting CP-ABE schemes supporting user revocability that have been proposed in the literature. Yu *et al.* [17] leveraged proxy re-encryption technique to fulfill user revocation, but the scheme only supports AND policy, and the key authority should re-generate all the secret keys including the proxy key in order to revoke users. Sahai *et al.* [12] addressed the problem through piecewise key generation. Liu *et al.* [10] proposed a time-based proxy re-encryption scheme to allow users' access right to terminate after a predetermined period of time. Besides, an identity-based user revocation was introduced by Li *et al.* [6] to achieve the user revocation by giving the revocation rights to encryptor directly. Horvath [4] proposed a multi-authority CP-ABE scheme with identity-based revocation. A revocable key-policy ABE was proposed by Shi *et al.* [13] to revoke users by updating the revocation list.

Another interesting problem in the ABE setting is attribute update. In dynamic real-world applications, a user attribute set may need to be updated multiple times. For example, the attribute set of an employee may be updated when his working role is changed inside a company, e.g., set  $A = \text{"Team member, Programming"}$   $\rightarrow$  set  $B = \text{"Manager, Product Design"}$ . The key generation authority should be requested to issue an updated secret key to the employee, so that the access rights of the employee for new encrypted data should be modified accordingly. At the same time, we need to guarantee that the employee cannot reuse his old and outdated secret key corresponding to "Team member, Programming" to gain access to the ciphertexts with access policy matching the set  $A$ . Attribute update is not trivial and straightforward in ABE, since the update of a single attribute will affect a wide range of users obtaining the same attribute.

A growing number of companies and industries choose to outsource their data to cloud service providers that makes the attribute update become an important need. This is so because the attribute description of employees may

be changed frequently with various working and promotion policies in a company. However, the attribute update problem has not been taken into account in the existing cloud-based access control schemes. The previously introduced schemes [4, 6, 10, 12, 13, 17] only focus on the user revocability. Motivated by the issues of user revocability and attribute update in cloud storage, we propose a secure identity-based revocation approach to solve the user revocability problem and meanwhile, develop an efficient updating method in which only those components related to the updated attribute need to be updated. We note that there are some recent technologies that have been introduced in the literature for cloud-based data sharing and user revocability, such as [7–9]. But this paper will leverage a different technique to achieve data sharing and revocability.

Inspired by [4, 14], we propose a cloud-based access control scheme with user revocation and attribute update in the context of attribute-based cryptographic encryption. We design a secure identity-based revocation approach to solve the ABE-based user revocability problem. In particular, we embed the identities of the revoked users into the ciphertext during the encryption phase, so that the revocation functionality does not have any effect on attributes. In addition, we make use of an efficient updating method to address the issue of attribute update. Specifically, we assign distinct updated (key) information for each system user who can update the secret key and the ciphertext that related to the updated attribute using the information. We state that the both approaches introduced in this paper greatly save the cost of the ciphertext storage.

## 2 Preliminaries

### 2.1 Access Structure

**Definition 1 (Access Structure [1]).** Let  $\{A_1, A_2, \dots, A_n\}$  be a group of attributes. For  $\forall B, C$ , if  $B \in \mathbb{A}$  and  $B \subseteq C$ , then  $C \in \mathbb{A}$ , we say that  $\mathbb{A} \subseteq 2^{\{A_1, A_2, \dots, A_n\}}$  is monotone. An access structure (or monotone access structure) contains a set (or monotone set)  $\mathbb{A}$  of non-empty subsets of  $\{A_1, A_2, \dots, A_n\}$ . Elements in  $\mathbb{A}$  are referred to as authorized elements, otherwise, elements are referred to as unauthorized elements.

In our scheme,  $\mathbb{A}$  will include authorized elements. Unless stated in another way, access structure used in this paper is in monotone form.

### 2.2 Access Tree

As in [1], let  $\mathcal{T}$  represent an access tree and  $x$  be a node of  $\mathcal{T}$ . Every non-leaf node  $x$  of  $\mathcal{T}$  denotes a threshold gate, which is stated by the number of its children  $num_x$  and its threshold  $l_x$ , where  $l_x \in [1, num_x]$ . If  $l_x = 1$  or  $l_x = num_x$ , then the threshold gate is an OR gate or AND gate, respectively. Every leaf node  $x$  of  $\mathcal{T}$  is stated by an attribute and a threshold  $l_x = 1$ .

Next, we define some functions to facilitate working with  $\mathcal{T}$ . In  $\mathcal{T}$ , we use functions  $parent(x)$  and  $att(y)$  to denote the parent of a node  $x$  and the attribute

related to a leaf node  $y$ , respectively. 1 to  $num_x$  are used to number the children of a node  $x$ . The function  $index(x)$  outputs a number related to a node  $x$ , in which  $index$  values are uniquely and arbitrarily distributed to nodes.

**Satisfying an Access Tree.** We let  $\mathcal{T}$  represent an access tree with root  $\Gamma$ , and  $\mathcal{T}_x$  be the sub-tree rooted at the node  $x$  in  $\mathcal{T}$ . we denote  $\mathcal{T}_x(S) = 1$  if and only if an attribute set  $S$  satisfies  $\mathcal{T}_x$ .  $\mathcal{T}_x(S)$  is computed recursively in the following. If  $x$  is a leaf node and  $att(x) \in S$ ,  $\mathcal{T}_x(S)$  outputs 1. If  $x$  is a non-leaf node, then for all children  $\{z\}$  of the node  $x$ , we compute  $\mathcal{T}_z(S)$ . If the number of  $\mathcal{T}_z(S) = 1$  is more than  $k_x$ ,  $\mathcal{T}_x(S)$  outputs 1.

### 2.3 Bilinear Maps

Like [1], let  $\mathbb{G}_0$  and  $\mathbb{G}_T$  be two groups of prime order  $p$ .  $g$  is the generator of  $\mathbb{G}_0$ . A bilinear mapping  $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$  satisfies the following properties:

1. **Bilinearity:** For any  $u, v \in \mathbb{G}_0$  and  $a, b \in \mathbb{Z}_p$ , it has  $e(u^a, v^b) = e(u, v)^{ab}$ .
2. **Non-degeneracy:** There exists  $u, v \in \mathbb{G}_0$  such that  $e(u, v) \neq 1$ .
3. **Computability:** For all  $u, v \in \mathbb{G}_0$ , there is an efficient computation  $e(u, v)$ .

### 2.4 The Decisional Bilinear Diffie-Hellman Assumption

We recall the definition of the decisional Bilinear Diffie-Hellman (DBDH) problem in [15] as follows. A challenger selects a group  $\mathbb{G}_0$  of prime order  $p$  according to the security parameter. Let  $g$  be a generator of  $\mathbb{G}_0$  and  $a, b, s \in \mathbb{Z}_p$  be selected at random. If the challenger gives an adversary  $(g, g^a, g^b, g^s)$ , it must be difficult for the adversary to distinguish a valid tuple  $e(g, g)^{abs} \in \mathbb{G}_T$  from a random element  $R \in \mathbb{G}_T$ .

An algorithm  $\mathcal{B}$  that outputs  $\vartheta \in \{0, 1\}$  has advantage  $\varepsilon$  in solving DBDH in  $\mathbb{G}_0$  if

$$|Pr[\mathcal{B}(g, g^a, g^b, g^s, Z = e(g, g)^{abs}) = 0] - Pr[\mathcal{B}(g, g^a, g^b, g^s, Z = R) = 0]| \geq \varepsilon$$

**Definition 2.** *The DBDH assumption holds if all poly-time algorithms have at most a negligible advantage in solving the DBDH problem.*

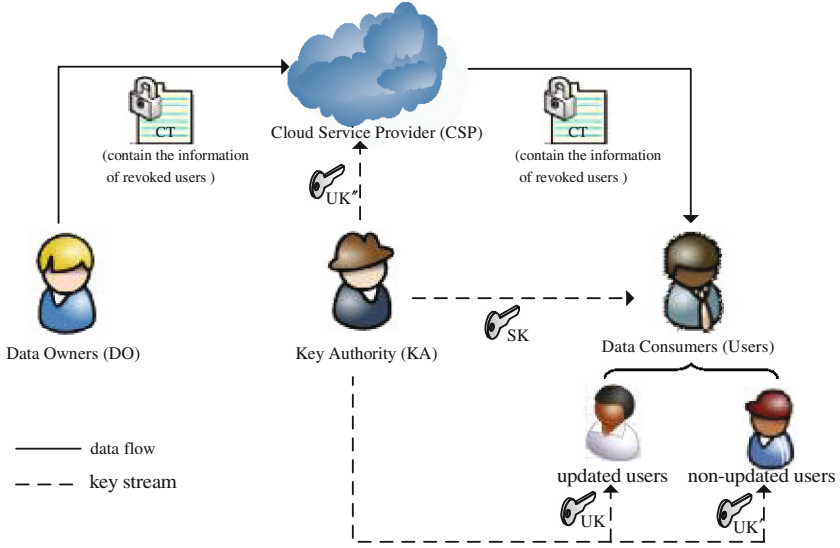
## 3 System Model and Security Model

### 3.1 System Description

A cloud-based access control scheme with user revocation and attribute update is shown in Fig. 1. There are four parties in the proposed scheme.

**Key authority (KA).** The KA is a fully trusted party following protocol specifications to perform any assigned tasks and next to output correct results. It is in charge of handling user registration, generating system parameters as well as a secret key  $SK$  for each system user, and update keys  $UK, UK', UK''$  for the updated users, non-updated users and the CSP, respectively.





**Fig. 1.** System model of the proposed scheme in the cloud computing system.

**Data owner (DO).** The DO is a party who has a great amount of files which to be uploaded to clouds. It is designed to define an access structure, and next to run an encryption operation under the structure. Finally, it uploads the resulting ciphertext to a cloud service provider.

**Cloud service provider (CSP).** The CSP is a trusted party who offers a variety of services, e.g., data storage and computing. It is also in charge of updating the ciphertext which related to the updated attribute.

**Data consumer (User).** A user is a party who would like to gain access to the ciphertext stored in CSP. When the user is not in the revoked list set by the DO and meanwhile, his attribute set satisfies the access structure embedded in the given ciphertexts, he is able to reveal the underlying files from the ciphertexts. In Fig. 1, updated users are those whose attributes (called the updated attributes) are updated to new attributes by the KA, non-updated users are those tagged with the updated attributes but not yet updating the attributes.

### 3.2 System Model

**Definition 3.** A cloud-based access control scheme with user revocation and attribute update includes the following five phases.

**Phase 1: System Initialization.**

$Setup(1^\kappa, L) \rightarrow (PK, MSK)$ . The setup algorithm intakes a security parameter  $\kappa$  and an attribute universe  $L$ , and outputs the public key  $PK$  and the master key  $MSK$ .

**Phase 2: Secret Key Generation.**

When a user requests to join the system, the KA assigns an attribute set  $S \subseteq L$  to the user in accordance with his identity  $ID$ .

$KeyGen(MSK, ID, S) \rightarrow SK$ . The key generation algorithm intakes  $MSK$ , an identity  $ID$  and an attribute set  $S$  that depicts the key, and outputs a secret key  $SK$ .

**Phase 3: File Encryption.**

To achieve high encryption efficiency, the DO first encrypts a given file with a content key  $ck$  by applying a symmetric encryption algorithm, and next to encrypt  $ck$  with the following algorithm.

$Encrypt(PK, ck, \mathbb{A}, RL) \rightarrow CT$ . The algorithm intakes  $PK$ , the content key  $ck$ , an access policy  $\mathbb{A}$  and a revoked user list  $RL$ , and outputs the ciphertext  $CT$ . Suppose that  $CT$  implicitly contains  $\mathbb{A}$ .

**Phase 4: File Decryption.**

A system user first downloads a ciphertext from the CSP. If his secret key satisfies the access policy  $\mathbb{A}$ , he will get  $ck$  by running the following algorithm.

$Decrypt(PK, SK, CT) \rightarrow ck$ . The algorithm intakes  $PK$ ,  $CT$  and  $SK$ , and outputs a  $ck$  or  $\perp$ . If  $SK$  matches the access policy  $\mathbb{A}$ , the user can recover the content key  $ck$ . Thus, the file will be revealed with  $ck$  by applying the symmetric decryption algorithm. Otherwise, decryption fails.

**Phase 5: Attribute Update.**

Assume an attribute  $j$  of a user is now updated to a new attribute  $w$  by the KA, where we refer to the attribute  $j$  as the updated attribute, and the user as the updated user, respectively. The users tagged with the updated attribute  $j$  but not yet updating the attribute is called non-updated users. The attribute update phase includes four steps: Update Key Generation ( $UKeyGen$ ) by the KA, Secret Key Update ( $SKUpdate1$ ) by the updated user, Secret Key Update ( $SKUpdate2$ ) by non-updated users, and Ciphertext Update ( $CTUpdate$ ) by the CSP.

$UKeyGen(PK, MSK, j, w, \cdot) \rightarrow (UK_{j \rightarrow w}, UK'_j, UK''_j)$ . The update key generation algorithm is run by the KA to update the attribute  $j$  of the updated user to the attribute  $w$ . It intakes  $PK$ ,  $MSK$ , the attributes  $j$  and  $w$ , and outputs three update keys  $UK_{j \rightarrow w}$ ,  $UK'_j$ , and  $UK''_j$ .

$SKUpdate1(SK, UK_{j \rightarrow w}) \rightarrow SK_u$ . The secret key update algorithm is run by the updated user. It intakes the current secret key  $SK$  of the updated user and the update key  $UK_{j \rightarrow w}$ , and outputs a new secret key  $SK_u$ .

$SKUpdate2(SK, UK'_j) \rightarrow SK_{nu}$ . The secret key update algorithm is run by non-updated user. It intakes the current secret key  $SK$  of non-updated user and the update key  $UK'_j$ , and outputs a new secret key  $SK_{nu}$ .

$CTUpdate(CT, UK''_j) \rightarrow \widetilde{CT}$ . The ciphertext update algorithm is run by the CSP. It intakes the ciphertext tagged with the updated attribute  $j$ , and the update key  $UK''_j$ , and outputs a new ciphertext  $\widetilde{CT}$ .

### 3.3 Security Model

We now define the chosen plaintext security of a cloud-based access control scheme with user revocation and attribute update, where a probabilistic polynomial time (PPT) adversary will commit to a challenge access structure  $\mathcal{T}^*$  at the outset of the security game.

- **Initialization.** A PPT adversary  $\mathcal{A}$  outputs a access structure  $\mathcal{T}^*$  to the challenger  $\mathcal{C}$ .
- **Setup.** The challenger  $\mathcal{C}$  runs the *Setup* algorithm and sends the public key  $PK$  to the adversary  $\mathcal{A}$ .
- **Phase 1.**  $\mathcal{A}$  can adaptively submit any attribute set  $S$  to  $\mathcal{C}$ , and query the secret key for  $S$  and identity  $ID_k$  which denotes the  $k$ -th  $ID$  query. The restriction is that all the queried attribute sets do not satisfy  $\mathcal{T}^*$ . Let  $UL$  denote the set of all queried  $ID_k$ . For the attribute set  $S$ ,  $\mathcal{C}$  runs the *KeyGen* algorithm and sends the corresponding secret key  $SK$  to  $\mathcal{A}$ .  $\mathcal{A}$  also makes update key queries when his attributes need to be updated, where the new attribute set also does not satisfy  $\mathcal{T}^*$ .  $\mathcal{C}$  performs the *UKeyGen* algorithm and sends the corresponding update keys to  $\mathcal{A}$ .
- **Challenge.**  $\mathcal{A}$  submits two equal length messages  $m_0, m_1$  and a set  $RL \subseteq UL$  of revoked identities to  $\mathcal{C}$ . After receiving the messages,  $\mathcal{C}$  randomly chooses one bit  $\vartheta \in \{0, 1\}$ , and encrypts  $m_{\vartheta}$  by running the *Encrypt* algorithm under the access structure  $\mathcal{T}^*$ . Finally,  $\mathcal{C}$  gives  $\mathcal{A}$  the challenge ciphertext  $CT^*$ .
- **Phase 2.**  $\mathcal{A}$  may query more update keys and secret keys for other attribute sets and identities. The restrictions are that these identities do not belong to  $RL$ , and none of the update keys and secret keys are able to decrypt the challenge ciphertext  $CT^*$ .
- **Guess.**  $\mathcal{A}$  outputs a guess  $\vartheta'$  of  $\vartheta$ .

In the above selective security model,  $\mathcal{A}$ 's advantage is

$$Adv(\mathcal{A}) = |Pr(\vartheta' = \vartheta) - \frac{1}{2}|$$

**Definition 4.** A cloud-based access control scheme with user revocation and attribute update is said to be selective CPA secure if all poly-time adversaries have at most a negligible advantage in the above model.

*Remarks.* We state that the above security game already implies the following two security notions - collusion resistance, and backward and forward security. Collusion resistance is one of the required security properties in the ABE setting. Even if multiple users (with respective secret keys) collude with each other, they are still not permitted to decrypt a given ciphertext if and only if each of the given secret key fails to match the description of the ciphertext. It is not difficult to see that the collusion resistance is directly implied from the above security notion, because the security game is defined in the context of ABE.

The backward security means that the updated user cannot decrypt the new ciphertext encrypted with the new public attribute key. The forward security

means that newly joined users who have enough attributes should be able to decrypt the previous ciphertext which is encrypted with the previous public key.

### 4 A Concrete Construction

Below we present a concrete construction for a cloud-based access control scheme with user revocation and attribute update. For simplicity, we suppose that there are  $m$  attributes in the system, denoted by  $L = \{1, 2, \dots, m\}$ . Let  $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$  be a bilinear map, and  $\mathbb{G}_0$  be a bilinear group of prime order  $p$  with generator  $g$ . Let  $H : \{0, 1\}^* \rightarrow \mathbb{G}_0$  be a hash function, which maps any attribute to a random element of  $\mathbb{G}_0$ . For any  $i \in \mathbb{Z}_p$ , the Lagrange coefficient  $\Delta_{i,L}(x) = \prod_{l \in L, l \neq i} \frac{x-l}{i-l}$ . Let  $RL = \{ID_1^*, ID_2^*, \dots, ID_k^*\}$  be the most recent list of  $k$  revoked users. The construction details are as follows.

**Phase 1: System Initialization.**

$Setup(1^\kappa, L) \rightarrow (PK, MSK)$ . The setup algorithm chooses a bilinear group  $\mathbb{G}_0$  of prime order  $p$  with generator  $g$  and two elements  $\alpha, \beta \in \mathbb{Z}_p$ . For each  $i \in L$ , the algorithm selects a random  $v_j$ . Finally, it sets the public key as  $PK = \{\mathbb{G}_0, g, g^\alpha, e(g, g)^\beta, \{PK_j = H(j)^{v_j} | j \in L\}\}$  and the master key as  $MSK = \{\alpha, g^\beta, \{v_j | j \in L\}\}$ .

**Phase 2: Secret Key Generation.**

When a user requests to join the system, the KA first handles the registration and identity authorization of the user. If the user passes the authorization, the KA assigns an attribute set  $S \subseteq L$  to the user in accordance with his identity  $ID$ .

$KeyGen(MSK, ID, S) \rightarrow SK$ . The key generation algorithm selects a random  $r \in \mathbb{Z}_p$ , which is a unique secret assigned to each system user. For each  $j \in S$ , it randomly chooses a number  $r_j \in \mathbb{Z}_p$  and stores it. Finally, it creates the secret key as  $SK = \{D = g^{\beta+\alpha r}, D' = g^{(\alpha+\beta+\alpha r)ID}, \forall j \in S : D_j = g^r \cdot H(j)^{v_j r_j}, D'_j = g^{\alpha r_j}\}$ .

**Phase 3: File Encryption.**

Before being uploaded to the CSP, a file  $M$  is processed with the following steps:

- (1) The DO picks a unique  $ID'$  for  $M$ .
- (2) The DO encrypts  $M$  with the content key  $ck$  by using the symmetric encryption algorithm, where  $ck$  is selected in a key space. Note that the resulting ciphertext is denoted as  $E_{ck}(M)$ .
- (3) The DO defines an access structure  $\mathcal{T}$ , and encrypts  $ck$  by running the following algorithm to output the content key ciphertext  $CT'$ .

$Encrypt(PK, ck, \mathcal{T}, RL) \rightarrow CT$ . For each node  $x$  in the access tree  $\mathcal{T}$ , the DO first selects two polynomials  $q_x$  and  $q'_x$ . Beginning from the root node  $R$ , these polynomials are selected in a top-down manner. For each node  $x$  in  $\mathcal{T}$ , the threshold value  $k_x$  needs only one more than the degree  $d_x$  of  $q_x$  and  $q'_x$ , that is,  $k_x = d_x + 1$ .

Starting from the root node  $R$ , the DO selects random numbers  $s, s^* \in \mathbb{Z}_p$ , and sets  $q_R(0) = s$  and  $q'_R(0) = s^*$ . Then the DO randomly chooses  $d_R$  and  $d'_R$  other points of  $q_R$  and  $q'_R$  to completely define them, respectively. For any other node  $x$ , it sets  $q_x(0) = q_{parent(x)}(index(x))$  and  $q'_x(0) = q'_{parent(x)}(index(x))$ , then selects  $d_x$  and  $d'_x$  other points randomly to completely define  $q_x$  and  $q'_x$ , respectively.

In  $\mathcal{T}$ , let  $X$  be a leaf node set and  $|RL| = k$ . The DO randomly selects  $s_i$  such that  $s^* = \sum_{i=1}^k s_i$ . Then  $CT'$  is constructed as

$$CT' = \left\{ \begin{array}{l} \mathcal{T}, \tilde{C} = ck \cdot e(g, g)^{\beta s}, C = g^s, \\ \forall x \in X : C_x = g^{\alpha q_x(0)}, C'_x = H(att(x))^{v_{att(x)} q_x(0)}, C''_x = g^{q'_x(0)}; \\ \forall i = 1, 2, \dots, k : C_{1,i}^* = g^{-s_i \cdot ID_i^*}, C_{2,i}^* = g^{s_i} \end{array} \right\}$$

Finally, the DO generates the integrated ciphertext  $CT = \{ID', E_{ck}(M), CT'\}$  and sends it to the CSP.

**Phase 4: File Decryption.**

A user first obtains the integrated ciphertext  $CT$  from the CSP. If the user is not in  $RL$ , and attributes he possesses satisfy the access structure, he can decrypt  $CT'$  successfully by running the following decryption algorithm and obtain the content key  $ck$ . Thus he can use  $ck$  to decrypt  $E_{ck}(M)$  and obtain the file  $M$ . The decryption algorithm is constructed as follows.

$Decrypt(PK, CT', SK) \rightarrow ck$ . We first computes  $D'' = g^{\beta + \alpha r} \cdot g^\alpha = g^{\alpha + \beta + \alpha r}$ . Note that the decryption procedure is defined as a recursive algorithm. Thus, we need to define the recursive algorithm  $DecryptNode(CT', SK, x)$  intaking  $CT', SK$ , and a node  $x$  from  $\mathcal{T}$ .

(1) If  $x$  is a leaf node. Let  $j = att(x)$ . If  $j \notin S$ , then  $DecryptNode(CT', SK, x) = null$ . If  $j \in S$ , then

$$\begin{aligned} DecryptNode(CT', SK, x) &= \frac{e(D_j, C_x) e(D'', C''_x)}{e(D'_j, C'_x)} \\ &= \frac{e(g^r \cdot H(j)^{v_j r_j}, g^{\alpha q_x(0)}) e(g^{\alpha + \beta + \alpha r}, g^{q'_x(0)})}{e(g^{\alpha r_j}, H(att(x))^{v_{att(x)} q_x(0)})} \\ &= \frac{e(g^r, g^{\alpha q_x(0)}) e(H(j)^{v_j r_j}, g^{\alpha q_x(0)}) e(g, g)^{(\alpha + \beta + \alpha r) q'_x(0)}}{e(g^{\alpha r_j}, H(j)^{v_j q_x(0)})} \\ &= e(g, g)^{\alpha r q_x(0)} e(g, g)^{(\alpha + \beta + \alpha r) q'_x(0)} \end{aligned}$$

(2) If  $x$  is a non-leaf node, the recursive algorithm  $DecryptNode(CT', SK, x)$  is defined as: for all nodes  $z$  that are children of  $x$ , it performs  $F_z = DecryptNode(CT', SK, z)$ . Let  $S_x$  be an arbitrary  $k_x - sized$  child nodes set  $\{z\}$ , then  $F_z \neq null$ . If the set does not exist,  $F_z = null$ . If not,  $F_x$  is calculated as

$$\begin{aligned}
 F_x &= \prod_{z \in S_x} F_z^{\Delta_{j, S'_x}(0)} \\
 &= \prod_{z \in S_x} (e(g, g)^{\alpha r q_z(0)} e(g, g)^{(\alpha + \beta + \alpha r) q'_z(0)})^{\Delta_{j, S'_x}(0)} \\
 &= \prod_{z \in S_x} (e(g, g)^{\alpha r q_{parent(z)}(index(z))} e(g, g)^{(\alpha + \beta + \alpha r) q'_{parent(z)}(index(z))})^{\Delta_{j, S'_x}(0)} \\
 &= \prod_{z \in S_x} (e(g, g)^{\alpha r q_x(j)} e(g, g)^{(\alpha + \beta + \alpha r) q'_x(j)})^{\Delta_{j, S'_x}(0)} \\
 &= e(g, g)^{\alpha r q_x(0)} e(g, g)^{(\alpha + \beta + \alpha r) q'_x(0)}
 \end{aligned}$$

where  $j = index(z)$  and  $S'_x = \{index(z) : z \in S_x\}$ .

Then the decryption algorithm can be defined by calling functions on the root node  $R$  of  $\mathcal{T}$ . If  $S$  satisfies  $\mathcal{T}$ , we get

$$\begin{aligned}
 F_R &= DecryptNode(CT', SK, R) \\
 &= e(g, g)^{\alpha r q_R(0)} e(g, g)^{(\alpha + \beta + \alpha r) q'_R(0)} \\
 &= e(g, g)^{\alpha r s} e(g, g)^{(\alpha + \beta + \alpha r) s^*}
 \end{aligned}$$

Thus, the content key  $ck$  can be derived as:

$$ck = \frac{\tilde{C} \cdot F_R}{e(D, C) \cdot \prod_{i=1}^k [e(D'', C_{1,i}^*) e(D', C_{2,i}^*)]^{1/(ID-ID_i^*)}}$$

The completeness of the decryption process is shown as follows.

$$\begin{aligned}
 \mathbf{A} &= \prod_{i=1}^k [e(D'', C_{1,i}^*) e(D', C_{2,i}^*)]^{1/(ID-ID_i^*)} \\
 &= \prod_{i=1}^k \left[ e(g^{\alpha + \beta + \alpha r}, g^{-s_i \cdot ID_i^*}) e(g^{(\alpha + \beta + \alpha r) ID}, g^{s_i}) \right]^{1/(ID-ID_i^*)} \\
 &= \prod_{i=1}^k \left[ e(g, g)^{-(\alpha + \beta + \alpha r) s_i \cdot ID_i^*} e(g, g)^{(\alpha + \beta + \alpha r) s_i \cdot ID} \right]^{1/(ID-ID_i^*)} \\
 &= \prod_{i=1}^k \left[ e(g, g)^{(\alpha + \beta + \alpha r) s_i (ID-ID_i^*)} \right]^{1/(ID-ID_i^*)} \\
 &= e(g, g)^{(\alpha + \beta + \alpha r) s^*}
 \end{aligned}$$

$$\mathbf{B} = \frac{F_R}{e(D, C) \cdot \mathbf{A}} = \frac{e(g, g)^{\alpha r s} e(g, g)^{(\alpha + \beta + \alpha r) s^*}}{e(g^{\beta + \alpha r}, g^s) e(g, g)^{(\alpha + \beta + \alpha r) s^*}} = \frac{1}{e(g, g)^{\beta s}}$$

$$ck = \tilde{C} \cdot \mathbf{B} = ck \cdot e(g, g)^{\beta s} \cdot \frac{1}{e(g, g)^{\beta s}}$$

$E_{ck}(M)$  can be decrypted with  $ck$  by applying the symmetric decryption algorithm.

### Phase 5: Attribute Update.

(1) *Update Key Generation* by the KA

$UKeyGen(PK, MSK, j, w) \rightarrow (UK_{j \rightarrow w}, UK'_j, UK''_j)$ . The KA takes as input  $PK, MSK$ , attributes  $j$  and  $w$ . It first selects a random  $r_w \in \mathbb{Z}_p$  ( $\tilde{v}_j \neq v_j$ ), and generates a update key  $UK_{j \rightarrow w}$  for the updated user to update his secret key as

$$UK_{j \rightarrow w} = \{UK'_{j \rightarrow w} = H(j)^{-r_j v_j} \cdot H(w)^{r_w v_w}, UK''_{j \rightarrow w} = r_w / r_j\}$$

then chooses a random  $\tilde{v}_j \in \mathbb{Z}_p$ , and generates  $UK'_j$  for each non-updated user to update his secret key, and also generates  $UK''_j$  for the CSP to update the ciphertext as

$$UK'_j = H(j)^{r_j (\tilde{v}_j - v_j)}, \quad UK''_j = \tilde{v}_j / v_j$$

The KA sends  $UK_{j \rightarrow w}, UK'_j$  and  $UK''_j$  to the updated user, those non-updated users and the CSP, respectively.

Then, the KA updates the public attribute key of the updated attribute  $j$  as

$$PK'_j = (PK_j)^{\tilde{v}_j / v_j} = (H(j)^{v_j})^{\tilde{v}_j / v_j} = H(j)^{\tilde{v}_j}$$

(2) *Secret Key Update* by the updated user

When the updated user receives  $UK_{j \rightarrow w}$ , he updates his secret key by performing the secret key update algorithm  $SKUpdate1$  as

$$SK_u = \left\{ \begin{array}{l} D = g^{\beta + \alpha r}, D' = g^{(\alpha + \beta + \alpha r) ID}, \\ \forall i \in S \setminus \{j\} : D_i = g^r \cdot H(i)^{v_i r_i}, D'_i = g^{\alpha r_i}; \\ D_w = g^r \cdot H(j)^{v_j r_j} \cdot UK'_{j \rightarrow w}, D'_w = g^{\alpha r_j} \cdot UK''_{j \rightarrow w} \end{array} \right\}$$

(3) *Secret Key Update* by non-updated users

Each non-updated user receives the corresponding update key  $UK'_j$ , and then runs the secret key update algorithm  $SKUpdate2$  to update his secret keys:

$$SK_{nu} = \left\{ \begin{array}{l} D = g^{\beta + \alpha r}, D' = g^{(\alpha + \beta + \alpha r) ID}, \\ \forall i \in S \setminus \{j\} : D_i = g^r \cdot H(i)^{v_i r_i}, D'_i = g^{\alpha r_i}; \\ D_j = g^r \cdot H(j)^{v_j r_j} \cdot UK'_j, D'_j = g^{\alpha r_j} \end{array} \right\}$$

(4) *Ciphertext Update* by the CSP

In order to ensure that the newly joined user is able to decrypt the previous ciphertext which is distributed before he joins the system, the ciphertext related to the updated attribute  $j$  should be updated. To reduce the burden on the data owner and improve the efficiency, we ask the CSP to update the ciphertext by performing the ciphertext update algorithm  $CTUpdate$  with  $UK_j''$  as

$$\widetilde{CT} = \left\{ \begin{array}{l} T, \tilde{C} = ck \cdot e(g, g)^{\beta s}, C = g^s, \\ \forall x \in X : C_x = g^{\alpha q_x(0)}, C_x'' = g^{q_x'(0)}, \\ \text{if } att(x) \neq j : C_x' = H(att(x))^{v_{att(x)} q_x(0)}, \\ \text{if } att(x) = j : C_x' = H(att(x))^{v_{att(x)} q_x(0) \cdot UK_{att(x)}''}, \\ \forall i = 1, 2, \dots, k : C_{1,i}^* = g^{-s_i \cdot ID_i^*}, C_{2,i}^* = g^{s_i} \end{array} \right\}$$

Note that only the components related to the updated attribute in the secret key and the ciphertext need to be updated.

### 5 Security Proof

In the following, we first show that the security of the proposed scheme can be reduced to the decisional bilinear Diffie-Hellamn (DBDH) assumption, then analyze that the proposed scheme is secure against collusion attacks.

**Theorem 1.** *If there exists a PPT adversary  $\mathcal{A}$  can win the proposed scheme with non-negligible advantage  $\varepsilon > 0$ , then there is a PPT algorithm  $\mathcal{B}$  that can distinguish a DBDH tuple from a random tuple with advantage  $\frac{\varepsilon}{2}$ .*

*Proof.* Let  $\mathbb{G}_0, \mathbb{G}_T$  be bilinear groups of prime order  $p$  with generator  $g$  and  $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$  be an efficiently computable bilinear map. First, the DBDH challenger  $\mathcal{C}$  chooses at random:  $a, b, s \in \mathbb{Z}_p, \vartheta \in \{0, 1\}$  and a random element  $R \in \mathbb{G}_T$ . We let  $Z$  to be  $e(g, g)^{abs}$  if  $\vartheta = 0$ , and  $R$  otherwise. Then  $\mathcal{C}$  sends  $\langle g, A, B, S, Z \rangle = \langle g, g^a, g^b, g^s, Z \rangle$  to  $\mathcal{B}$ . Now,  $\mathcal{B}$  plays the role of challenger in the following game.

- **Initialization.**  $\mathcal{A}$  selects a challenge access structure  $\mathbb{A}^*$  and sends it to  $\mathcal{B}$ .
- **Setup.** In order to provide  $PK$  to  $\mathcal{A}$ ,  $\mathcal{B}$  chooses a random  $\beta' \in \mathbb{Z}_p$  and computes  $\beta = \beta' + ab$ . Then it sets  $u = e(g, g)^\beta = e(g, g)^{\beta'} e(g, g)^{ab}$  and  $v = g^\alpha = g^a = A$ . For each  $j \in L$ ,  $\mathcal{B}$  selects a random  $v_j$  and computes  $h_j = H(j)^{v_j}$ . Finally,  $\mathcal{B}$  sends the public key  $PK = \{u, v, \{h_j | j \in L\}\}$  to  $\mathcal{A}$ .
- **Phase 1.** In this phase,  $\mathcal{B}$  responses secret key queries and update key queries from  $\mathcal{A}$ .  $\mathcal{A}$  can adaptively submits any attribute set  $S \subseteq L$  to  $\mathcal{B}$ , and query the secret key for  $S$  and identity  $ID_k$  which represents the  $k$ -th  $ID$  query. Let  $UL$  represent the set of all queried  $ID_k$ . First,  $\mathcal{B}$  selects a random  $r' \in \mathbb{Z}_p$  and computes  $r = r' - b$ . Then it sets  $D = g^\beta g^{\alpha r} = g^{(\beta' + ab) + a(r' - b)} = g^{\beta' + ar'} = g^{\beta'} \cdot A^{r'}$ , and  $D' = g^{(\alpha + \alpha r + \beta) ID_k} = g^{(a + ar' + \beta') ID_k} = A^{(1+r') ID_k} \cdot g^{\beta' ID_k}$ . For each  $j \in S$ ,  $\mathcal{B}$  selects  $r_j \in \mathbb{Z}_p$  at random and computes  $D_j = g^{r'} \cdot H(j)^{v_j r_j} =$



$g^{r'-b} \cdot H(j)^{v_j r_j} = \frac{g^{r'}}{B} \cdot H(j)^{v_j r_j}$ ,  $D'_j = g^{\alpha r_j} = g^{\alpha r_j} = A^{r_j}$ . At last,  $\mathcal{B}$  gives the secret key to  $\mathcal{A}$ .

For each update key query, suppose an attribute  $j$  needs to be updated to a new attribute  $w$  with a constraint that the new attribute set also does not satisfy  $\mathbb{A}^*$ .  $\mathcal{B}$  selects a random  $r_w \in \mathbb{Z}_p$  and sets the update key as  $UK_{j \rightarrow w} = \{UK'_{j \rightarrow w} = H(j)^{-v_j r_j} \cdot H(w)^{-v_w r_w}, UK''_{j \rightarrow w} = \frac{r_w}{r_j}\}$ .

- **Challenge.**  $\mathcal{A}$  submits two messages  $m_0, m_1$  and a set  $RL \subseteq UL$  of revoked identities to  $\mathcal{B}$ , where  $|RL| = k$ . For simplicity, let  $RL = \{ID_1^*, ID_2^*, \dots, ID_k^*\}$ . After receiving the messages,  $\mathcal{B}$  does the following. First,  $\mathcal{B}$  selects a random  $s^* \in \mathbb{Z}_p$ , then chooses  $s_1, s_2, \dots, s_{k-1} \in \mathbb{Z}_p$  at random and computes  $s_k = s^* - \sum_{i=1}^{k-1} s_i$ .  $\mathcal{B}$  also applies the linear secret sharing scheme related to  $\mathbb{A}^*$  to construct shares  $w_i$  of  $s^*$  for all related attributes  $i$ . Note that all of  $w_i$ , which are subjected to linear conditions imposed on them by applying the secret sharing scheme, are selected randomly from  $\mathbb{Z}_p$ . Then  $\mathcal{B}$  selects  $\vartheta \in \{0, 1\}$  at random, and constructs the challenge ciphertext  $CT^*$  as:  $\tilde{C} = m_\vartheta \cdot e(g, g)^{\beta s} = m_\vartheta \cdot e(g, g)^{(ab+\beta')s} = m_\vartheta Z e(g, g)^{\beta' s}$ ,  $C = g^s = S$ . For every related attribute  $i$ ,  $C'_i = g^{w_i}$ . For each  $j \in RL$ ,  $C_{1,j}^* = g^{-s_i \cdot ID_i^*}$  and  $C_{2,j}^* = g^{s_i}$ . Finally,  $\mathcal{B}$  gives  $\mathcal{A}$  the challenge ciphertext  $CT^*$ .
- **Phase 2.** The same as *Phase 1*, the additional restriction is that the queried identities do not belong to  $RL$ .
- **Guess.**  $\mathcal{A}$  outputs a guess  $\vartheta'$  of  $\vartheta$ .  $\mathcal{B}$  outputs 0 to indicate that  $T = e(g, g)^{abs}$  in the above game if  $\vartheta' = \vartheta$ ; otherwise,  $\mathcal{B}$  outputs 1 to guess that  $T = R$ .

If  $T = e(g, g)^{abs}$ , then  $CT^*$  is an available ciphertext. In this case,  $\mathcal{A}$ 's advantage is  $\varepsilon$ . Thus

$$Pr[\mathcal{B}(g, g^a, g^b, g^s, T = e(g, g)^{abs}) = 0] = \frac{1}{2} + \varepsilon$$

If  $T = R$ , then  $CT^*$  is completely random from  $\mathcal{A}$ 's view. Therefore,

$$Pr[\mathcal{B}(g, g^a, g^b, g^s, T = R) = 0] = \frac{1}{2}$$

At last,  $\mathcal{B}$ 's advantage in the security game is

$$\begin{aligned} \mathcal{B} &= \frac{1}{2}(Pr[\mathcal{B}(g, g^a, g^b, g^s, T = e(g, g)^{abs}) = 0] \\ &\quad + Pr[\mathcal{B}(g, g^a, g^b, g^s, T = R) = 0]) - \frac{1}{2} \\ &= \frac{1}{2}\left(\frac{1}{2} + \varepsilon + \frac{1}{2}\right) - \frac{1}{2} \\ &= \frac{\varepsilon}{2} \end{aligned}$$

**Table 1.** The summary of the new notations

Notation	Role
$\mathbb{G}_i(i = 0, T)$	the group or operations in group (exponentiation, multiplication)
$L_{\mathbb{G}_i}(i = 0, T)$	the length of an element in group $\mathbb{G}_i(i = 0, T)$
$C_e$	the $e$ operation ( $e$ denotes bilinear pairing)
$\mathbb{Z}_p$	the group $\{0, 1, \dots, p - 1\}$ with multiplication modulo $p$
$L_{\mathbb{Z}_p}$	the length of an element in finite field $\mathbb{Z}_p$
$l$	the number of the attribute universe
$t$	the number of attributes appeared in ciphertext
$k$	the number of attributes associated with a user's secret key
$r$	the number of revoked users
$n_{non,j}$	the number of non-revoked users hold the attribute $j$

**Table 2.** Comparisons of our scheme and the scheme in [4]

Schemes	Horvath [4]	Ours
<i>PK</i> size	$(l + 3)L_{\mathbb{G}_0} + lL_{\mathbb{G}_T}$	$(l + 2)L_{\mathbb{G}_0} + L_{\mathbb{G}_T}$
<i>MSK</i> size	$(2l + 2)L_{\mathbb{Z}_p}$	$(l + 1)L_{\mathbb{Z}_p} + L_{\mathbb{G}_0}$
<i>SK</i> size	$(k + 1)L_{\mathbb{G}_0}$	$2(k + 1)L_{\mathbb{G}_0}$
<i>CT</i> size	$(2t + 2r)L_{\mathbb{G}_0} + (t + 1)L_{\mathbb{G}_T}$	$(3t + 2r + 1)L_{\mathbb{G}_0} + L_{\mathbb{G}_T}$
OE	$(4t + 4r)\mathbb{G}_0 + (3t + 2)\mathbb{G}_T$	$(3t + 3r + 1)\mathbb{G}_0 + 2\mathbb{G}_T$
OD	$(4k + 3r - 1)\mathbb{G}_T + (2k + 2r)C_e$	$(4k + 3r + 1)\mathbb{G}_T + (3k + 2r + 1)C_e$
<i>Key</i> update	0	$(1 + n_{non,j})L_{\mathbb{G}_0} + L_{\mathbb{Z}_p}$
<i>CT</i> update	0	$L_{\mathbb{Z}_p}$

## 6 Efficiency Analysis

To make the analysis more understandable, we summarize new notations which we use in comparison in Table 1. Let OE and OD denote operations for encryption and decryption, respectively.

The comparison results are summarized in Table 2, we assume that the access structure and hash computation are not included in the calculation.

From Table 2, the *PK* size, *MSK* size and operations for encryption in our scheme are less than those in [4]. Moreover, our scheme requires additional computation  $(1 + n_{non,j})L_{\mathbb{G}_0} + 2L_{\mathbb{Z}_p}$  to achieve the attribute updating, which does not achieve in [4]. In the case of achieving user revocation, our scheme is more efficient than the scheme in [4], even if it is requires more  $e$  operations in the decryption process in our scheme.

## 7 Conclusion

A cloud-based access control scheme with user revocation and attribute update is proposed, where the revocation and updating methods are presented to address the user revocation and attribute update problems, respectively. The security analysis indicated that the proposed scheme is secure to the DBDH assumption. The efficiency analysis showed that the computational overhead of the proposed scheme is acceptable to achieve user revocation and attribute update. The proposed scheme is a promising technique, which can be used in cloud storage systems. The future work is to improve the efficiency of the access control scheme with user revocation and attribute update.

**Acknowledgments.** The work of this paper is supported by the National Natural Science Foundation of China (61171072), the Science & Technology Innovation Projects of Shenzhen, China (ZDSYS20140430164957660, JCYJ20140418095735596, JCYJ20150324141711562, JCYJ20150324141711665). Kaitai Liang is supported by privacy-aware retrieval and modelling of genomic data (PRIGENDA, No. 13283250), the Academy of Finland.

## References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: S&P, pp. 321–334. IEEE Computer Society (2007)
2. Chase, M., Chow, S.: Improving privacy and security in multi-authority attribute-based encryption. In: CCS, pp. 121–130. ACM (2009)
3. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: CCS, pp. 456–465. ACM (2007)
4. Horváth, M.: Attribute-based encryption optimized for cloud computing. In: Italiano, G.F., Margaria-Steffen, T., Pokorný, J., Quisquater, J.-J., Wattenhofer, R. (eds.) SOFSEM 2015-Testing. LNCS, vol. 8939, pp. 566–577. Springer, Heidelberg (2015)
5. Hur, J.: Improving security and efficiency in attribute-based data sharing. IEEE T. Knowl. Data En. **25**(10), 2271–2282 (2013)
6. Li, Y., Zhu, J., Wang, X., Chai, Y., Shao, S.: Optimized ciphertext-policy attribute-based encryption with efficient revocation. Int. J. Secur. Appl. **7**(6), 281–287 (2013)
7. Liang, K., Susilo, W.: Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **10**(9), 1981–1992 (2015)
8. Liang, K., Susilo, W., Liu, J.K.: Privacy-preserving ciphertext multi-sharing control for big data storage. IEEE Trans. Inf. Forensics Secur. **10**(8), 1578–1589 (2015)
9. Liang, K., Liu, J.K., Wong, D.S., Susilo, W.: An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014, Part I. LNCS, vol. 8712, pp. 257–272. Springer, Heidelberg (2014)
10. Liu, Q., Wang, G., Wu, J.: Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. Inf. Sci. **258**(3), 355–370 (2014)

11. Liu, X., Ma, J., Xiong, J., Li, Q., Ma, J.: Ciphertext-policy weighted attribute based encryption for fine-grained access control. In: INCoS, pp. 51–57. IEEE Computer Society (2013)
12. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012)
13. Shi, Y., Zheng, Q., Liu, J., Han, Z.: Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. *Inf. Sci.* **295**, 221–231 (2015)
14. Wang, S., Yu, J., Zhang, P., Wang, P.: A novel file hierarchy access control scheme using attribute-based encryption. *Appl. Mech. Mater.* **701**, 911–918 (2015)
15. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
16. Yang, Y., Liu, J.K., Liang, K., Choo, K.-K.R., Zhou, J.: Extended proxy-assisted approach: achieving revocable fine-grained encryption of cloud data. In: Pernul, G., Y A Ryan, P., Weippl, E. (eds.) ESORICS 2015, Part II. LNCS, vol. 9327, pp. 146–166. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-24177-7\\_8](https://doi.org/10.1007/978-3-319-24177-7_8)
17. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: CCS, pp. 261–270. ACM (2010)
18. Zhu, W., Yu, J., Wang, T., Zhang, P., Xie, W.: Efficient attribute-based encryption from R-LWE. *Chin. J. Electron.* **23**(4), 778–782 (2014)

# Author Index

- Au, Man Ho I-3, I-20, I-389, I-443, II-207  
Bagheri, Nasour II-301  
Bai, Bo I-3  
Barmawi, Ari Moesriami I-409  
Boyd, Colin I-161  
Braun, Johannes I-426  
Buchmann, Johannes I-426  
Bunder, Martin II-258  
  
Cai, Haibin I-265  
Carr, Christopher I-161  
Castiglione, Aniello I-231, II-37  
Castiglione, Arcangelo II-37  
Chan, Terence II-505  
Chen, Chao I-215  
Chen, Huaifeng II-409  
Chen, Jiageng II-333  
Chen, Rongmao II-3, II-223  
Chen, Wei I-20  
Chen, Xiaofeng I-361, I-509  
Chen, Zehong I-525  
Cheng, Yao II-481  
Choo, Kim-Kwang Raymond I-265, I-389  
Chow, Kam-Pui II-457  
Chow, Yang-Wai I-409  
Cornea, Tudor I-231  
Cui, Xingmin I-40  
  
Deng, Robert H. II-207, II-481  
Dobre, Ciprian I-231  
Duong, Dung Hoang II-427  
Dutta, Avijit I-343  
  
Erfani, Sarah M. II-493  
  
Fang, Junbin I-40, II-333  
Feng, Yong I-329  
Forler, Christian II-317  
  
Glennan, Timothy II-493  
Goh, Yong Kheng I-77  
Gosman, Catalin I-231  
  
Großschädl, Johann I-94  
Gu, Dawu II-134  
Guo, Fuchun I-477, II-3, II-170, II-223  
  
Han, Fengling I-329  
Han, Jinguang I-443  
Hanaoka, Goichiro II-269  
Hanzlik, Lucjan II-467  
Hao, Feng I-57  
He, Debiao II-73  
He, Kai II-207  
He, Xiaoyang II-285  
Ho, Siu-Wai II-505  
Horsch, Moritz I-426  
Hou, Lin I-461  
Hu, Jiankun II-87  
Hu, Wei I-310  
Huang, Xinyi I-310  
Huang, Zhangjie II-379  
Huh, Jun Ho I-128  
Hui, Lucas C.K. I-40  
  
Ji, Min I-495  
Jia, Chunfu II-153  
Jia, Keting II-363  
Jiang, Peng II-170  
Jiang, Tao I-361  
Jiang, Yin hao I-477  
Jiang, Zoe L. II-153  
  
Ke Wang, Eric I-40  
Khor, Ean Yee II-437  
Kim, Hyounghsick I-128  
Kluczniak, Kamil II-467  
Kunihiro, Noboru II-243, II-269  
Kutyłowski, Mirosław II-467  
  
Lai, An Chow I-77  
Lai, Jianchang II-223  
Lai, Junzuo I-461  
Lazaridis, Emmanuel I-77  
Leckie, Christopher II-493  
Lee, Min Cherng I-77

- Li, Bao II-285  
 Li, Hui I-509  
 Li, Jin I-509, II-153  
 Li, Lin I-94  
 Li, Ping II-153  
 Li, Tong II-153  
 Li, Xiangxue I-293  
 Li, Yannan I-389  
 Li, Yingjiu II-481  
 Li, Zhen I-198  
 Liang, Kaitai I-376, I-525, II-103  
 Lin, Xiaodong II-120  
 List, Eik II-317  
 Liu, Jianwei I-310  
 Liu, Jingwen I-310, II-87  
 Liu, Lixian I-461  
 Liu, Shigang I-215  
 Liu, Weiran II-87  
 Liu, Zhe I-94  
 Liu, Zheli II-153  
 Lou, Wenjing I-361  
 Lu, Haibing I-265  
 Lu, Jiqiang II-395  
 Lu, Rongxing II-120  
 Lucks, Stefan II-317  
 Luo, Xiapu I-3, I-20  
 Luo, Xiling II-87  
 Lv, Bo I-281
- Ma, Jianfeng I-361  
 Ma, Sha II-21  
 Ma, Xiaobo I-3  
 Mao, Yijun II-207  
 Masucci, Barbara II-37  
 McCorry, Patrick I-57  
 Mendel, Florian II-301  
 Minematsu, Kazuhiko II-347  
 Mitra, Robin I-77  
 Miyaji, Atsuko I-376  
 Morozov, Kirill I-181  
 Möser, Malte I-57  
 Mu, Yi I-477, II-3, II-21, II-57, II-170,  
 II-187, II-223
- Nandi, Mridul I-343  
 Nisbet, Alastair I-115  
 Nitaj, Abderrahmane II-258
- Ooi, Shih Yin II-437
- Palmieri, Francesco II-37  
 Pan, Shiran I-141, II-446  
 Pang, Ying Han II-437  
 Parampalli, Udaya II-134  
 Parra, Jhordany Rodriguez II-505  
 Paul, Goutam I-343  
 Petzoldt, Albrecht II-427  
 Phillips, James G. I-409  
 Phuong, Tran Viet Xuan II-103  
 Pop, Florin I-231  
 Pranata, Ilung I-409
- Qian, Haifeng I-293  
 Qin, Bo I-310  
 Qin, Lingyue II-409  
 Qiu, Xinliang I-3
- Ren, Yili II-87  
 Rudolph, Carsten I-249
- Samsudin, Azman II-333  
 Santis, Alfredo De II-37  
 Sasaki, Yu II-301  
 Schlipf, Mario I-426  
 Shahandasti, Siamak F. I-57  
 Shao, Jun I-495  
 Shen, Wuqiang I-281  
 Song, Ling II-379  
 Song, Youngbae I-128  
 Su, Chunhua I-376, II-333  
 Sun, Shi-Feng II-134  
 Sun, Yang I-310  
 Sun, Yujuan I-293  
 Susilo, Willy I-361, I-389, I-409, I-477, II-3,  
 II-103, II-223, II-258
- Takagi, Tsuyoshi I-181, II-427  
 Takayasu, Atsushi II-243  
 Tang, Shaohua I-281  
 Tang, Yajuan I-20  
 Teh, Je Sen II-333  
 Tian, Song II-285  
 Tonien, Joseph II-258  
 van Schyndel, Ron I-329
- Wang, Jingxuan I-40  
 Wang, Kunpeng II-285  
 Wang, Ning II-363  
 Wang, Shulan I-525  
 Wang, Ting I-525

- Wang, Xiaofen II-57  
 Wang, Xiaoyun II-409  
 Wang, Yu I-215  
 Wang, Zhen II-87  
 Wei, Guiyi I-495  
 Wei, Zhuo I-265  
 Wen, Qiaoyan II-170  
 Weng, Jian II-207  
 Wenzel, Jakob II-317  
 Woodward, Andrew I-115  
 Wu, Qianhong I-310, I-361, II-87  
 Wu, Wei I-198  
  
 Xia, Zhe II-73  
 Xiang, Yang I-215  
 Xiao, Bin I-20  
 Xiao, Min II-73  
 Xie, Mande I-495  
 Xu, Fei II-457  
 Xu, Qiuliang I-94, I-443  
 Xu, Rui I-181  
 Xu, Shengmin II-21  
 Xu, Wei II-87  
  
 Yamakawa, Takashi II-269  
 Yan, Shen I-141, II-446  
 Yang, Guomin I-409, II-3, II-21, II-103,  
 II-187  
 Yang, Min II-457  
  
 Yang, Qianqian II-379  
 Yang, Rupeng I-443  
 Yang, Xiaoyun II-73  
 Yang, Xuechao I-329  
 Yang, Yanjiang I-181, I-265  
 Yap, Wun-She I-77  
 Yi, Xun I-329  
 Yin, Chengyu I-20  
 Yiu, S.M. I-40  
 Yu, Wei II-285  
 Yu, Yong I-389  
 Yu, Yu I-293, II-134  
 Yu, Zuoxia I-443  
 Yuen, Tsz Hon II-134  
  
 Zhang, Kai I-293  
 Zhang, Mingwu II-3  
 Zhang, Peng I-525  
 Zhang, Shiwei II-187  
 Zhang, Xinpeng I-389  
 Zhang, Yinghui I-509  
 Zhao, Shuang I-3  
 Zhao, Yuhang I-141  
 Zheng, Dong I-509  
 Zhou, Jianying I-181  
 Zhou, Yuan I-293  
 Zhu, Wen-Tao I-141, II-446  
 Zou, Wei I-3  
 Zuo, Cong I-495