Bo Chen
Xinyi Huang (Eds.)

LNICST
386

Applied Cryptography in Computer and Communications

First EAI International Conference, AC3 2021
Virtual Event, May 15–16, 2021
Proceedings

EAI

Springer

Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 386

More information about this series at http://www.springer.com/series/8197

Bo Chen · Xinyi Huang (Eds.)

Applied Cryptography in Computer and Communications

First EAI International Conference, AC3 2021
Virtual Event, May 15–16, 2021
Proceedings

Springer

Editors
Bo Chen
Michigan Technological University
Houghton, MI, USA

Xinyi Huang
Fujian Normal University
Fuzhou, Fujian, China

ISSN 1867-8211 ISSN 1867-822X (electronic)
Lecture Notes of the Institute for Computer Sciences, Social Informatics
and Telecommunications Engineering
ISBN 978-3-030-80850-1 ISBN 978-3-030-80851-8 (eBook)
https://doi.org/10.1007/978-3-030-80851-8

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, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

Preface

We are delighted to introduce the proceedings of the 2021 European Alliance for Innovation (EAI) International Conference on Applied Cryptography in Computer and Communications (AC3 2021). This conference brought together researchers, developers, and practitioners from around the world who focus on the area of applied cryptography in computer and communication systems. It included all aspects of applied cryptography, including symmetric cryptography, public-key cryptography, cryptographic protocols, cryptographic implementations, and cryptographic standards and practices, as well as using cryptography to solve real-world problems.

The technical program of AC3 2021 consisted of 11 papers in oral presentation sessions at 4 main conference tracks: Track 1 – Blockchain; Track 2 – Authentication; Track 3 – Secure Computation; and Track 4 – Practical Crypto Application. Aside from the high-quality technical paper presentations, the technical program also featured two keynote speeches and one technical workshop. The two keynote speeches were delivered by Prof. Kui Ren from Zhejiang University, China, and Prof. Robert Deng from Singapore Management University, Singapore. The organized workshop, the First International Workshop on Security for Internet of Things (IOTS 2021), included four technical papers, which aim to develop cryptographic techniques for ensuring IoT security.

Coordination with the steering committee, Dr. Imrich Chlamtac (chair), Dr. Jingqiang Lin, and Dr. Bo Luo, was essential for the success of the conference. We sincerely appreciate their constant support and guidance. It was also a great pleasure to work with such an excellent Organizing Committee team for their hard work in organizing and supporting the conference. We are grateful to the Technical Program Committee (TPC), led by our TPC Co-chairs, Dr. Jian Shen and Dr. Joseph K. Liu, who completed the peer-review process of technical papers and put together a high-quality technical program. We are also grateful to the Conference Managers, Aleksandra Sledziejowska and Lucia Sedlarova, for their support and all the authors who submitted their papers to the AC3 2021 conference and workshops.

We strongly believe that the AC3 conference provides a good forum for all researchers, developers, and practitioners to discuss all science and technology aspects that are relevant to applied cryptography. We also expect that the future editions of the AC3 conference will be as successful and stimulating as AC3 2021, as indicated by the contributions presented in this volume.

May 2021

Bo Chen
Xinyi Huang

Organization

Steering Committee

Imrich Chlamtac (Chair)	University of Trento, Italy
Bo Chen	Michigan Technological University, USA
Jingqiang Lin	Chinese Academy of Sciences, China
Bo Luo	University of Kansas, USA

Organizing Committee

General Chair

Bo Chen	Michigan Technological University, USA
Xinyi Huang	Fujian Normal University, China

Technical Program Committee Chair and Co-chair

Jian Shen	Nanjing University of Information Science and Technology, China
Joseph K. Liu	Monash University, Australia

Sponsorship and Exhibit Chairs

Qian Wang	Wuhan University, China
Liang Xiao	Xiamen University, China

Local Chairs

Zhixiong Chen	Putian University, China
Pinhui Ke	Fujian Normal University, China

Workshops Chair

Le Guan	University of Georgia, USA

Publicity and Social Media Chairs

Arcangelo Castiglione	University of Salerno, Italy
Chenhuang Wu	Putian University, China

Publications Chairs

Yuan Hong	Illinois Institute of Technology, USA
Shucheng Yu	Stevens Institute of Technology, USA

Web Chair

Xiaoyong Yuan	Michigan Technological University, USA

Posters and PhD Track Chair

Qi Li	Tsinghua University, China

Panels Chair

Jun Shao	Zhejiang GongShang University, China

Demos Chair

Jianting Ning	Singapore Management University, Singapore

Tutorials Chair

Kaitai Liang	University of Surrey, UK

Technical Program Committee

Hamid Ali Abed Alasadi	Basra University, Iraq
Man Ho Au	The Hong Kong Polytechnic University, Hong Kong
Joonsang Baek	University of Wollongong, Australia
Aniello Castiglione	University of Salerno, Italy
Quanwei Cai	Chinese Academy of Sciences, China
David Chadwick	University of Kent, UK
Jiageng Chen	Central China Normal University, China
Xiaofeng Chen	Xidian University, China
Yueqiang Cheng	Baidu, USA
Kim-Kwang Raymond Choo	The University of Texas at San Antonio, USA
Sherman S. M. Chow	The Chinese University of Hong Kong, Hong Kong
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Catalonia
Jose Maria de Fuentes	Universidad Carlos III de Madrid, Spain
Carmen Fernández-Gago	University of Malaga, Spain
José M. Fernandez	Ecole Polytechnique de Montreal, Canada

Dieter Gollmann	Hamburg University of Technology, Germany and National University of Singapore, Singapore
Stefanos Gritzalis	University of the Aegean, Greece
Gerhard Hancke	City University of Hong Kong, Hong Kong
Debiao He	Wuhan University, China
Shoichi Hirose	University of Fukui, Japan
Xinyi Huang	Fujian Normal University, China
Julian Jang-Jaccard	Massey University, New Zealand
Hiroaki Kikuchi	Meiji University, Japan
Kwangjo Kim	Korea Advanced Institute of Science and Technology, South Korea
Noboru Kunihiro	The University of Tokyo, Japan
Costas Lambrinoudakis	University of Piraeus, Greece
Albert Levi	Sabanci University, Turkey
Shujun Li	University of Kent, UK
Tieyan Li	Huawei International Pte Ltd, Singapore
Yingjiu Li	Singapore Management University, Singapore
Kaitai Liang	University of Surrey, UK
Joseph Liu	Monash University, Australia
Zhe Liu	Nanjing University of Aeronautics and Astronautics, China
Giovanni Livraga	University of Milan, Italy
Jiqiang Lu	Beihang University, China
Rongxing Lu	University of New Brunswick, Canada
Tzu-Chuen Lu	Chaoyang University of Technology, Taiwan
Di Ma	University of Michigan, USA
Weizhi Meng	Technical University of Denmark, Denmark
Chris Mitchell	Royal Holloway, University of London, UK
David Naccache	École Normale Supérieure, France
Takeshi Okamoto	Tsukuba University of Technology, Japan
Kazumasa Omote	University of Tsukuba, Japan
Pedro Peris-Lopez	Carlos III University of Madrid, Spain
Günther Pernul	Universität Regensburg, Germany
Josef Pieprzyk	Queensland University of Technology, Australia
Geong Sen Poh	Singtel, Singapore
Na Ruan	Shanghai Jiaotong University, China
Sushmita Ruj	Indian Statistical Institute, India
Jun Shao	Zhejiang Gongshang University, China
Jian Shen	Nanjing University of Information Science and Technology, China
Miguel Soriano	Universitat Politècnica de Catalunya, Spain
Chunhua Su	University of Aizu, Japan
Willy Susilo	University of Wollongong, Australia
Syh-Yuan Tan	Newcastle University, UK
Qiang Tang	New Jersey Institute of Technology, USA
Jaideep Vaidya	Rutgers University, USA

Cong Wang	City University of Hong Kong, Hong Kong
Ding Wang	Peking University, China
Chen Wang	Nanjing University of Information Science and Technology, China
Guilin Wang	Huawei, Singapore
Wei Wang	Chinese Academy of Sciences, China
Qianhong Wu	Beihang University, China
Shouhuai Xu	University of Texas at San Antonio, USA
Toshihiro Yamauchi	Okayama University, Japan
Huijie Yang	Nanjing University of Information Science and Technology, China
Kuo-Hui Yeh	National Dong Hwa University, Taiwan
Xun Yi	RMIT University, Australia
Yong Yu	Shaanxi Normal University, China
Tsz Hon Yuen	The University of Hong Kong, Hong Kong
Yuexin Zhang	Swinburne University of Technology, Australia
Yongjun Zhao	Nanyang Technological University, Singapore
Jianying Zhou	Singapore University of Technology and Design, Singapore
Tianqi Zhou	Nanjing University of Information Science and Technology, China
Antonis Michalas	Tampere University, Finland

Contents

Blockchain

Authentication

Secure Computation

Practical Crypto Application

The First International Workshop on Security for Internet of Things (IOTS 2021)

Blockchain

Anchor: An NDN-Based Blockchain Network

Shucheng Yu[1](✉), Noor Ahmed[2], and Ruiran Wang[1]

[1] ECE Department, Stevens Institute of Technology, Hoboken, NJ 07030, USA
{shucheng.yu,ruiran.wang}@stevens.edu
[2] Air Force Research Laboratory, Rome, NY 13441, USA
norman.ahmed@us.af.mil

Abstract. Efficient broadcasting is critical for timely propagation of transactions and blocks to the network in blockchain systems. Existing blockchain networks are usually characterized by a high communication overhead and redundant traffics which adversely impact the transaction/block latency and security. In the networking community, remarkable progresses have been made in past decades on network efficiency. In particular, information-centric networking (ICN) has been considered a promising approach for network efficiency and robustness by shifting from traditional host-centric paradigm to the content-centric paradigm. However, direct marriage between blockchain and ICN does not yield a satisfying solution due to various security gaps. In this paper, we make the first attempt to integrate the two by introducing a new blockchain network technique called *Anchor* based on an ICN technique - named-data networking (NDN). By tailoring NDN and cascading two NDN systems, we demonstrate an efficient yet secure solution for data propagation in blockchain systems. Simulation shows that Anchor consumes less bandwidth and has a lower transaction/block latency as compared to the Bitcoin flooding network and a recent technique Erlay.

Keywords: Blockchain · Information-centric network · Named-data network · Efficiency · Security · Privacy

1 Introduction

In blockchain systems it is crucial to timely propagate data (transactions or blocks) so as to achieve a consistent view and consensus among distributed participants (e.g., miners in Bitcoin). An efficient network layer is not only indispensable to blockchain performance including throughput, robustness, scalability and bandwidth efficiency. It is also closely relevant to the security of blockchain. Slow data propagation results in long synchronization delay, which can lead to

Effort sponsored by the Air Force under MOU FA8750-15-3-6000. The U.S. Government is authorized to reproduce and distribute copies for Governmental purposes notwithstanding any copyright or other restrictive legends.

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 3–9, 2021.
https://doi.org/10.1007/978-3-030-80851-8_1

security vulnerabilities [1,2] including blockchain forks [3]. Inappropriate network layer design may also lead to the eclipse attack [4] wherein attackers dominate the in- and out-bound communications of a victim.

In the popular permissionless blockchain Bitcoin, the gossip protocol is used for data propagation and consensus is achieved by "proof of work" (PoW). In the gossip protocol, data (transactions/blocks) is relayed by mining nodes in a greedy way - each node receiving the data immediately propagates to its neighbors by exchanging three messages - *inv* (announcement), *getdata* or *getblocks* (request) and *tx* or *block* (delivery of data). While the three-way message exchange can reduce redundant transmissions of actual data (transactions/blocks), it incurs a significant management overhead because of the announcement and request messages. Existing study [6] shows that announcement messages can count for around 30–50% of overall traffic of which 88% are redundant. Moreover, the per-hop three-way message exchange also causes additional delays as compared to direct unsolicited "push" of the data. Permissioned blockchain systems [5] are mostly based on the classical Byzantine fault tolerant (BFT) protocol or its variants for consensus. The BFT-based consensus usually involves multi-round broadcastings which share similar properties as the gossip protocol.

Existing techniques for blockchain network efficiency mainly resort to three approaches - deploying relay nodes to suppress redundant traffics [7], compressing transactions to reduce the message sizes [8], or limiting the fanout of forwarding links at each node to control the degree of redundancy [6]. However, these techniques either assumes centralized relay nodes are deployed [7], which is incompatible with the decentralized nature of blockchain, or limits the fanout of forward links [6], which inevitably prolongs the message propagation time and cause a larger delay to each transaction. Compressing transactions is an orthogonal technique that can be integrated to other designs including this work.

In the network community, remarkable progresses have been made in past decades on network efficiency. In particular, information-centric networking (ICN) [9] has been considered a promising approach for network efficiency and robustness by shifting from traditional host-centric paradigm to the content-centric paradigm. Essentially, the gossip protocol can be considered a special instance of ICN in the sense that data is published and requested by content. One difference is that ICN supports both asynchronous and synchronous production and consumption of data while the gossip protocol works in the synchronous manner. As compared to the synchronous counterpart, asynchronous production and consumption provides more flexibilities. For example, consumers can subscribe data in advance before it is produced. Once available, data can be directly pushed to consumers, saving the real-time announcements and requests. However, direct application of the asynchronous mode in blockchain faces nontrivial challenges. Specifically, as the content of data and even its producer are unknown before the data is produced, consumers do not know what and where to prescribe. One solution is to use designated relay nodes as in existing work [7]. However, this will require the relay nodes to be trusted. Yet another problem is that the "prior-subscription" through the designated relay nodes may reveal traffic pattern and lead to security vulnerabilities.

In this paper, we introduce a new blockchain data propagation protocol namely *Anchor* by tailoring an ICN technique - named-data networking (NDN) [10]. Specifically, we design a cascaded two-layer tailored NDN for asynchronous transaction/block production and consumption. Instead of using designated trusted relay nodes, we allow periodical election of a set of random relay nodes which we call *anchor nodes* (or *anchors*). While multiple random anchors improve system robustness under traffic manipulation attacks, redundant traffics from multiple anchors can be aggregated in our design. We protect traffic privacy by randomizing data propagation routes of each anchor node. Simulation with NDNSim shows that our design achieves a lower communication overhead and a shorter propagation latency as compared to Bitcoin gossip protocol and Erlay.

2 Our Design

The overall idea of our design is to periodically elect a set of random anchor nodes to relay data to the network. The asynchronous data production and consumption is achieved through a cascaded two-layer NDN: in layer 1, anchors act as consumers and each other node as a potential producer; in layer 2, each anchor node acts as a producer and all other nodes as consumers. Newly produced data will first propagate through the layer-one NDN to anchors, which relay to other nodes over layer-two NDN. To allow prior-prescription before data is actually produced, we use time slots t_i, $i = 1, 2, \cdots$, as the "names" for data. In other words, all data produced within the same time slot t_i will be propagated through pre-determined routes. To protect traffic privacy, each node subscribes from a random up-steam node for each time slot t_i. Therefore, the actual routes in t_i is randomly and distributedly selected by all nodes en route. The level of privacy can be further adjusted by selecting an appropriate duration for time slots t_i. In our design, redundant transmissions of data relayed by different anchor nodes are detected and aggregated at each node en route they first reach.

2.1 Random Anchor Selection

Anchors are randomly selected at every interval T $(T \geq t_i)$. As there is no central party in the system, the selection shall be in fully distributed manner. For this purpose, we utilize the recently confirmed blocks as common seeds. Assume $\{B_{t-k}, B_{t-k+1}, \cdots, B_{t-1}, B_t\}$ are the $k+1$ recently confirmed blocks at time t. The anchor node is selected by a selector $i = h(h(B_{t-i})|h(B_t)|tstmp)\ mod\ N$, where N is the approximate total number of nodes in the system, $h()$ a one-way hash function, and *tstmp* the timestamp associated with B_t. Due to the unpredictability of blocks, the number i is difficult to predict before the latest block is confirmed. To automate the node selection in the distributed way, a node j is selected as an anchor node only if $i == Encrypt_{sk_j}(h(B_t)|tstmp)\ mod\ N$, where sk_j is the private key of node j. Given the randomness of the encryption function $Encrypt()$, the probability that the equality holds is $1/N$ for a random node j and on average one node is selected by the selector i. And k selectors will identify k random anchors on average. Note that, in case more than one node are selected by the same selector, we accept all of them as anchor nodes. The anchor nodes can be updated at every interval T by changing the selector i as $h(h(B_{t-i})|h(B_t)|tstmp + T)\ mod\ N$.

Once a new block is confirmed, each node first checks if it is an anchor node by locally calculating the k selectors and then compare if $Encrypt_{sk_j}(h(B_t)|tstmp)$ $mod\ N$ matches with any selector. If not, this node is not an anchor. Otherwise, it broadcasts the tuple $ancmt = (Encrypt_{sk_j}(h(B_t)|tstmp), i, pk_j, idx_j)$ to its neighbors as the announcement. Each node receiving the announcement verifies whether 1) $i == h(Decrypt_{pk_j}(Encrypt_{sk_j}(h(B_t)|tstmp)))\ mod\ N$ and 2) the decrypted $h(B_t)||tstmp$ matches with the block B_t in its local blockchain, where pk_j is the sender j's public key and i the selector included in $ancmt$. To thwart anchor manipulation, we restrict pk_j to those previously appeared in a confirmed block (e.g., as an address in a transaction) indexed by idx_j. This prevent attackers from temporarily generating a private key (e.g., by exhaustive search) in order to be selected as the anchor. The random distribution and unpredictability of anchors also help thwart traffic manipulation related attacks.

One potential problem may occur when there are forks with the blockchain. This may cause some nodes refuse to accept some valid anchors because they do not share the same longest chain of blocks. This can be addressed by accepting all $ancmt$ messages with valid selector i (via step 1) but letting anchors to include the hash values and nonces of p previous blocks in $ancmt$. Each node can verify the authenticity of $ancmt$ messages by checking their respective proofs of work (without payload).

2.2 Two-Layer Cascaded NDN Design

To facilitate efficient propagation of transactions and blocks by aggregating announcement and request messages in the gossip protocol, we design a two-layer cascaded NDN network as shown in Fig. 1. The broadcasted tuple $ancmt$ will serve as the announcement for both NDN networks within current interval T.

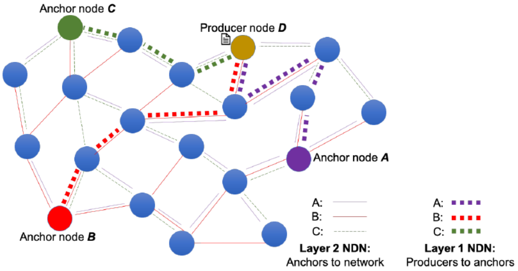


Fig. 1. Two-layer cascaded NDN

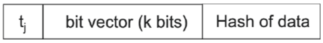
At layer 1, the data producers and the anchors form an NDN network. In a blockchain system, any node can be a potential data producer. Therefore, each anchor shall subscribe to every other node for all new transactions or blocks. As the subscription in layer 1 NDN is opposite to the direction of the propagation of *ancmt*, we let each node create layer-1 pending interest table (PIT) after layer 2 PIT has been created as discussed below.

In layer 2 NDN network, each anchor node acts as a producer and other nodes are consumers. The construction of the PIT table of layer 2 NDN network is straightforward: on receiving the announcement message (i.e., the verified *ancmt*), each node n_i randomly selects a sender n_j from all up-steam neighbors who relayed the *ancmt* message and sends an interest packet for time slot t_j to it for each future time slot $t_j \in T$. On receiving the interest packet, n_j creates a PIT entry with n_i as the outcoming interface for t_j. Meanwhile, n_i creates a PIT entry for layer 1 NDN for t_j with n_j as the outcoming interface. This means that n_i will forward all new data generated in t_j to n_j. In the layer 2 PIT tables, each entry also includes the index of the anchor, meaning that PIT entry is for that particular anchor node at time t_j.

Please note that in our design, the routes (i.e., PIT entries) of the nodes form a random broadcast tree rooted at each anchor for each time slot t_j. Layer 1 NDN also forms a random broadcast tree rooted at each node with all anchors being leaf nodes. A branch in a layer 1 NDN broadcast tree overlaps with a branch in a layer 2 NDN broadcast tree but with an opposite data propagation direction. Traffic pattern privacy is also well protected in our design. This is because for each time slot t_j, broadcast trees of both layer 1 and layer 2 are randomly determined by each node hop by hop.

2.3 Elimination of Duplicated Transmissions from Multiple Anchors

Another problem we need to address is the duplicated data traffics relayed by multiple anchors. This is because when a new data is generated, the producer broadcasts it to all anchors, each of whom relays the data to the entire network.



t_j	bit vector (k bits)	Hash of data

Fig. 2. Short message of duplicated data

To reduce duplicated transmissions, we design a short message for duplicated data as shown in Fig. 2. The bit vector assigns one bit for each anchor and is initialized as all 0's. Bit 1 means the data has been received from that anchor. When a node receives a data, it first updates its local bit vector by ORing its local bit vector with the received one. Then it checks its PIT entries. For those already fulfilled (i.e., data has been forwarded to the outcoming interface), it will just send the short message as shown in Fig. 2 (the hash of data can be replaced with a shorter index defined by each anchor); for those not fulfilled, the node will send the full message, which includes t_j, the updated bit vector, and the original data. In this way, duplicated transmissions are aggregated into short messages. We can apply encoding techniques to further compress the bit vector.

3 Performance Evaluation

We evaluate our design using NDNSim with 60,000 nodes, each connected to an average of eight other nodes. Transaction generation rate is seven per second.

T is 4 min and t_j is 1 min. We compare Anchor with the Bitcoin flooding network and Erlay for transaction propagation regarding bandwidth consumption and per-transaction latency.

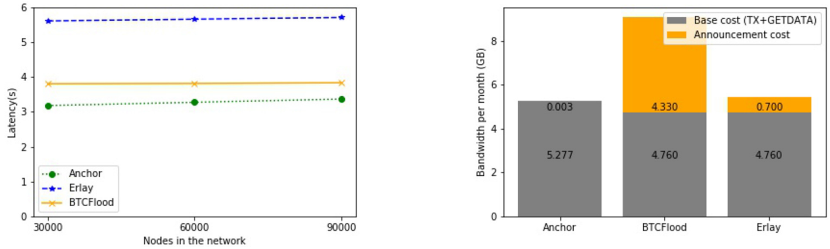


Fig. 3. Comparison of Anchor with BTCFlood and Erlay

As shown in Fig. 3, both the latency and the overall bandwidth consumption of Anchor are smaller than both Erlay and Bitcoin. The latency of Anchor increases slightly faster than the other two as the network size increases. This is because we used a fixed number of 40 anchor nodes, which become sparser when the network size increases. Although this increased latency can be flattened by deploying more anchor nodes, it will introduce more communication overhead because of redundant transmissions of anchors. As shown in the right figure, Anchor is able to aggregate almost all announcements (0.003 GB) within each time interval T. But it has a slightly higher overhead with the base cost (5.277 GB) mainly contributed by the bit vector.

4 Conclusion

In this paper, we introduce the first NDN-based efficient network layer for Bitcoin-like blockchain systems. With the new two-layer cascaded NDN design, we are able to significant aggregate the management messages of the gossip protocol with traffic pattern protected. Extensive simulation with a widely used simulator NDNSim shows that our design enjoys a lower communication overhead and a smaller latency as compared to the state of the art.

References

1. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
2. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 291–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_10

3. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE P2P, Italy, Trento, pp. 1–10 (2013)
4. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: USENIX Security 2015, Washington D.C., USA, pp. 129–144 (2015)
5. Xiao, Y., Zhang, N., Lou, W., Hou, Y.T.: A survey of distributed consensus protocols for blockchain networks. IEEE Commun. Surv. Tutor. **22**(2), 1432–1465 (2020)
6. Naumenko, G., Maxwell, G., Wuille, P.: Erlay: efficient transaction relay for bitcoin. In: ACM CCS 2019, London, UK, pp. 817–831 (2019)
7. Falcon. https://www.falcon-net.org/
8. Compact block relay. https://github.com/bitcoin/bips/
9. Information-Centric Networking. https://irtf.org/icnrg
10. Named-Data Networking. https://named-data.net/

An Identity-Based Blind Signature and Its Application for Privacy Preservation in Bitcoin

Yitao Chen[1], Qi Feng[2], Min Luo[2], Li Li[2], and Debiao He[2](✉)

[1] Wuhan Maritime Communication Research Institute, Wuhan 430205, China
[2] School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
{fengqi.whu,mluo,lli}@whu.edu.cn

Abstract. The privacy preservation in Bitcoin is increasingly important, partly due to its huge market capitalization and potential applications in distributed architectures. To protect the privacy of users in Bitcoin, a number of mechanisms have been proposed, where mixing service is a simple and frequently-used mechanism. The work, named Blindcoin, believes that an *unlinkable* blind signature scheme can help to guarantee the anonymity of users at the mixer side. Recently, Sarde and Banerjee presented an identity-based blind signature scheme. However, we found their scheme is vulnerable to a linkability attack. In this paper, we improve their scheme on this weakness and construct two *unlinkable* identity-based blind signature schemes, where one is in the standard setting and the other is in the proxy setting. Our approaches delinearize the two blinding factors so that malicious signer or proxy signer cannot find any helpful information from what she knows. The security, including unlinkability, of our schemes relies on the computational Diffie-Hellman assumption in the random oracle model as analyzed in this paper. We typically show that this is of great important to hide the relationship between message-signature pairs for the privacy-protecting in Bitcoin.

Keywords: Unlinkable blind signature · Privacy preservation · Bitcoin · Proxy blind signature · Identity-based cryptography

1 Introduction

The continued interest in Bitcoin is evident by its market capitalization, for example, it takes a market capitalization of $209,144,466,745 and has been topped the ranking of cryptocurrency since its publication in 2008[1]. However, Bitcoin provides a limited form of privacy preservation: static analysis attacks to de-anonymize an user are possible, even if she always creates pseudonyms when connecting to the Bitcoin system [4,20,25,28,30,31,35,44]. For example,

[1] See https://coinmarketcap.com/.

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 10–24, 2021.
https://doi.org/10.1007/978-3-030-80851-8_2

Androulaki et al. [4] conducted an experiment in an university, where students uses Bitcoin as the daily transaction currency. By utilizing cluster analysis based on the transaction fingerprints, they finally profiled approximately 40% of the participants, even some of them apply a fresh address for each transaction.

Some discussions of the importance and cryptographic mechanisms for privacy preservation in Bitcoin can be found in [10,11,18,24,27,41]. Mixing service, the frequently-used mechanism for protecting privacy since it was proposed by Chaum [8], allows users to mix the input/output relationship of their transactions, within some anonymity set, so that cannot be linked to the correct origin and destination [1–3,5,12,13,26,37]. Although many such mixing services exist, Valenta and Rowan [37] argued the *anonymity means that the users (including sender and receiver of a transaction) should be the only entities that know the mapping from their input address to their output address.* It imposes tight constraint on these services, i.e., the mixer (who performs the mixing step) has no information about the mapping from a transaction's input to output address.

To alleviate the risk of deanonymization on the mixer side, Valenta and Rowan [37] implemented a blind signature within Mixcoin [5]. They conceptually defined a blind signature scheme with three procedures, i.e., blinding (hiding the original message together with a random "blinding factor"), signing (signing the blinded message) and unblinding (removing the "blinding factor" to get a valid signature on the actual message). The core security assumption that the blind signature works well in Valenta and Rowan's protocol is that *the blind signature can be publicly verified while the signer has no information about the connection between the pair of message and signature.*

Up to now, a number of identity-based blind signature schemes [16,19,21,23,42] and proxy blind signature schemes [34,38–40,43] have been proposed. All of them have stated to be unlinkable, i.e., the original signer and proxy signer (who is authorized to sign on behalf of the original signer) can use their private keys to generate a valid signature on the blinded message, and cannot discover which messages were signed by them after the unblinding phase. More recently, Sarde et al. [32] also proposed a new identity-based blind signature scheme from bilinear pairings. Unfortunately, we find that their schemes cannot guarantee unlinkability and we will present this weakness more clearly later.

Therefore, in this paper, we firstly present an attack on the unlinkability of the blind signature scheme by Sarde et al. [32], and construct two unlinkable identity-based blind signature schemes, where one is in the standard setting and the other is in the proxy setting. Our standard unlinkable blind signature scheme improves the scheme presented in [32] by delinearizing two blinding factors such that malicious signer cannot find any helpful information from what she knows. This is of great important to *hide the relationship between message-signature pairs* for the privacy-protecting in Bitcoin. Such an approach also works well in our proxy blind signature, which maintains unlinkability between the message-signature pair both on the original signer and proxy signer side. We theoretically analyze the security and performance of our proposed schemes. While our schemes are slightly slower compared to blind signature schemes presented in [32], we demonstrate their great practical value by an example of potential application for privacy preservation in Bitcoin.

The rest of the paper is organized as follows. In the Sect. 2, we introduce some preliminaries. Section 3 reviews the blind signature scheme presented by Sarde et al. and Sect. 4 present a de-anonymizing attack on their scheme. Our improved unlinkable blind signature and unlinkable proxy blind signature schemes are described in (resp.) Sect. 5 and Sect. 6. Section 7 focuses on the security of our proposed schemes. In Sect. 8, we provide an theoretical evaluation of the proposed schemes. Finally, we give an example of potential application to privacy preservation in Bitcoin.

2 Preliminaries

In this section, we describe the relevant preliminaries required in the understanding of the proposed scheme.

2.1 Tate Bilinear Pairings

Assume that $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ denote three cycle groups with the same order of prime q. There exists a bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with following properties:

- *Bilinearity*: For any elements $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and any integers $x, y \in \mathbb{Z}_q^*$, the equation $e(xP, yQ) = e(P, Q)^{xy}$ holds.
- *Non-degeneracy*: For some elements $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, the inequation $e(P, Q) \neq \mathbf{1}_{\mathbb{G}_T}$ holds.
- *Computability*: Given two elements $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, there exist effective algorithms to compute $e(P, Q)$.

2.2 Computational Diffie-Hellman Assumption

Define $\mathbb{G}$ as a finite cycle group with the order of prime number $q = |\mathbb{G}|$ and generator of P. For unknown $x, y \in \mathbb{Z}_q$, the advantage to compute xyP from the tuple (P, xP, yP) for any probabilistic polynomial time (P.P.T) adversary $\mathcal{A}$ is negligible.

2.3 System Model

The architecture of this paper is shown in Fig. 1.

There are four (or five) types of participants with an (proxy) blind signature scheme: the key generation center, a user, a signer, a verifier, and a proxy signer just within proxy blind signature scheme.

- KGC: It is a trusted third party and its task is generating system parameters. Besides, it is also extract the private keys of the user, the signer and the proxy signer according to their identities.
- User: He/She is a client who intents to get signature on message m.

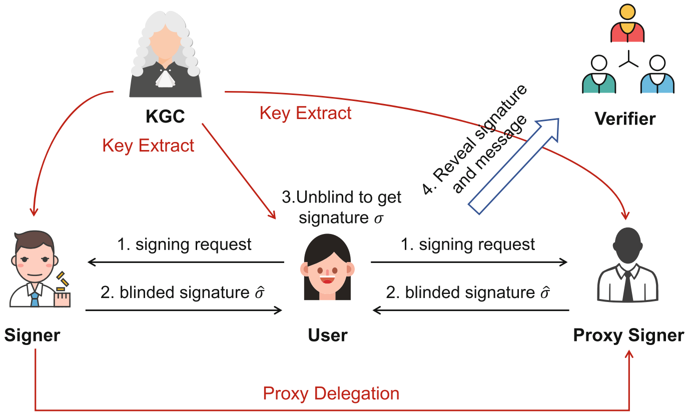

KGC
Key Extract
Key Extract
Verifier
4. Reveal signature and message
3.Unblind to get signature σ
1. signing request
2. blinded signature $\hat{\sigma}$
1. signing request
2. blinded signature $\hat{\sigma}$
Signer
User
Proxy Signer
Proxy Delegation

Fig. 1. The system model

- Signer: It is a server provider who gets his/her private key from the KGC and uses it to sign on blinded message provided from the User. After the signature process, he/she can not know any information about original message m.
- Proxy Signer: It is a proxy server provider being authorized to sign on behalf the Signer when the Signer is off-line. It is a specific character within the proxy blind signature scheme.
- Verifier: He/She can verify the signature on message m after the User publicly publishing the unblinded signature.

3 Review of Sarde et al.' Blind Signature Scheme

In [32], Sarde et al.' ID-based blind signature consists of the following algorithms:

- Setup: Let $\mathbb{G}_1$ denotes an additive group of generator P and order q and $\mathbb{G}_2$ denotes a multiplicative group of the same order, bilinear pairings e defines $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, three cryptographic hash functions are given by $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $h_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q$, $h_3 : \{0,1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q$. KGC randomly samples master private key $s \leftarrow_R \mathbb{Z}_q^*$ and computes master public key by $P_{pub} = s \cdot P$. Finally, system parameters are publicly set as $\texttt{params} = \{\mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_1, h_2, h_3\}$ and master private key s will be kept securely by KGC.
- Extract:
  1. On receiving signer's identity information ID, KGC computes $Q_{ID} = H_1(ID)$ as his or her public key.
  2. Output $S_{ID} = s \cdot Q_{ID}$ to the signer as private key.
- Blinding Phase:
  1. Signer randomly samples $r \leftarrow_R \mathbb{Z}_q^*$, computes and sends $R = r \cdot Q_{ID}$ to the user.

2. User samples two random values $k_1, k_2 \leftarrow_R \mathbb{Z}_q^*$, computes $u = h_2(R) \cdot k_1 \mod q$, $T = e(k_2 \cdot R + k_1 k_2 \cdot Q_{ID}, P_{pub})$ and $\hat{h} = h_3(m, T) + u \mod q$ and sends $\hat{h}$ to the signer.

- Signing Phase: Signer computes and returns the blinded signature $\hat{S} = (\hat{h} + r) \cdot S_{ID}$ to the user.
- Unblinding Phase:
  1. User unblinds the signature by $S = k_2 \cdot \hat{S}$, $h = \hat{h} - u \mod q$ and $d = k_2 \cdot (\hat{h} - k_1) \mod q$.
  2. Output the blind signature $\sigma = (S, h, d)$ of message m.
- Verify: On input a signature σ of message m, public key Q_{ID} and system parameters `params`, the verifier accepts the signature if and only if $h = h_3(m, e(S, P) \cdot e(Q_{ID}, P_{pub})^{-d})$ holds.

4 Attack on Sarde et al.' Blind Signature Scheme

In this section, we show that a curious signer in their blind signature scheme can link a signature with a signing requester:

1. Given a tuple of transcripts $\{r, R, S_{ID}, \hat{h}\}$ and candidate signature $\{S^*, h^*, d^*\}$ of message m^*, the signer firstly computes
$$k_1 = (\hat{h} - h^*) \cdot (h_2(R))^{-1}, k_2 = d^* \cdot (\hat{h} - k_1)^{-1}$$
2. Now the signer can check whether
$$e(k_2^{-1} \cdot S^*, P) = e(R + \hat{h} \cdot Q_{ID}, P_{pub})$$
to discover the signing requester.

This attack is workable because $\{r, R, S_{ID}, \hat{h}\}$ and $\{S^*, h^*, d^*\}$ of message m^* are all known to the signer and

$$\begin{aligned} e(k_2^{-1} \cdot S^*, P) &= e(k_2^{-1} \cdot (k_2^* \cdot (\hat{h}^* + r^*)) \cdot S_{ID}, P) \\ &= e((\hat{h}^* + r^*) \cdot S_{ID}, P)^{k_2^{-1} \cdot k_2^*} = e((\hat{h}^* + r^*) \cdot Q_{ID}, P_{pub})^{k_2^{-1} \cdot k_2^*} \end{aligned}$$

On the right side,

$$e(R + \hat{h} \cdot Q_{ID}, P_{pub}) = e((r + \hat{h}) \cdot Q_{ID}, P_{pub})$$

It is obliviously possible to discover the signing requester, because the probability of $k_2^{-1} \cdot k_2^* \cdot (\hat{h}^* + r^*) = r + \hat{h}$ is negligible on the conditions of 1) signer's random number $r^* \neq r$; 2) user's blinding factor $k_2 \neq k_2^*$; 3) the blinded message $\hat{h}^* \neq \hat{h}$ (defined jointly by signer's random number, user's blinding factor and one-way hash function).

Thus after the user publishes message-signature pair in public, a curious signer can link them to certain tuple that she keeps. Once we apply Sarde et al.' scheme for the privacy preservation in Bitcoin, a malicious mixer can trace the relationship between the transaction's sender and receiver without being authorized by the users. This attacks can work because the public u is multiplied by two integers. So it is easily factorizable if one integer is known. Our improvement is put the blind factor k_1 into the hash operation and guarantee security on the basis of one-way hash function.

5 Unlinkable ID-Based Blind Signature Scheme

In this section, we propose an improved version of Sarde et al.'s scheme, which satisfies untraceability besides the merits of original scheme.

- Setup: The proposed unlinkable proxy blind signature scheme are parameterized by $\texttt{params} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, P_1, P_2, P_{pub}, h_1, h_2\}$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two group of the same order q, Tate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ (an asymmetric pairing which is faster than the Weil bilinear pairing that was used in [32]), P_1 and P_2 denote the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $h_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ and $h_3 : \{0,1\}^* \times \mathbb{G}_T \rightarrow \mathbb{Z}_q^*$ define three cryptographic hash functions. The master private key $s \leftarrow_R \mathbb{Z}_q^*$ is chosen by KGC and $P_{pub} = s \cdot P_2$ is the master public key. Finally, KGC publishes the system parameter $\texttt{params}$ and keeps s securely.
- Extract: On receiving signer's identity string ID, KGC computes $Q_{ID} = H_1(ID)$ as his or her public key, and returns the corresponding private key $S_{ID} = s \cdot Q_{ID}$ to the signer.
- Blinding Phase:
  1. Signer randomly samples $r \leftarrow_R \mathbb{Z}_q^*$, computes and sends $R = r \cdot Q_{ID}$ to the user.
  2. User samples two random values $k_1, k_2 \leftarrow_R \mathbb{Z}_q^*$, computes
  $$u = h_2(k_1 \cdot R),$$
  $T = e(k_2 \cdot R + k_1 k_2 \cdot Q_{ID}, P_{pub})$, $\hat{h} = h_3(m, T) + u \mod q$ and sends $\hat{h}$ to the signer.
- Signing Phase: Signer computes and returns the blinded signature $\hat{S} = (\hat{h} + r) \cdot S_{ID}$ to the user.
- Unblinding Phase:
  1. User unblinds the signature by $S = k_2 \cdot \hat{S}$, $h = \hat{h} - u \mod q$, $d = k_2 \cdot (\hat{h} - k_1) \mod q$
  2. Output the blind signature $\sigma = (S, h, d)$ of message m.
- Verify: On input a signature σ of message m, public key Q_{ID} and system parameters $\texttt{params}$, the verifier accepts the signature if and only if $h = h_3(m, e(S, P) \cdot e(Q_{ID}, P_{pub})^{-d})$ holds.

Correctness. The correctness of our scheme can be verified following the prior work of Sarde et al.. Thus it is omitted here for simplicity.

6 Unlinkable ID-Based Proxy Blind Signature Scheme

Proxy signature is such a blind signature that a proxy signer is authorized to generate a blind signature on behalf of the original signer, and neither original signer nor proxy signer know the message. In this section, we further propose a unlinkable proxy blind signature scheme for convenience, economy and meet the cooperation demands of the modern companies, for example, multiple company's managers are assigned to answer the mixing services in turns.

Our construction is also built on the Sarde et al.'s proxy blind signature scheme [32]. We can see that their scheme fails to satisfy the unlinkability, as for a curious proxy signer, the hash value h received from user is exactly identical to that parsed from signature. Here, we improve their version and fill in gap of unlinkability.

Our unlinkable proxy blind signature scheme consists of following five phases:

- Setup: The proposed unlinkable proxy blind signature scheme has similar definition around the system parameters $\texttt{params} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, P_1, P_2, g, P_{pub}, H_1, h_2\}$ as standard one, except the cryptographic hash function $h_2 : \{0,1\}^* \times \mathbb{G}_T \rightarrow \mathbb{Z}_q^*$ and element $g = e(P_1, P_2)$. Finally, KGC publishes $\texttt{params}$ and keeps s securely itself.
- Extract: On receiving original signer's identity ID_s and proxy signer's identity ID_p, KGC computes $Q_{ID_s} = H_1(ID_s)$ and $Q_{ID_p} = H_1(ID_p)$ as their corresponding public keys, and returns the private keys $S_{ID_s} = s \cdot Q_{ID_s}$ and $S_{ID_p} = s \cdot Q_{ID_p}$ to the original signer and proxy signer respectively.
- Proxy Delegation:
  1. *Proxy Generation*: The original signer randomly samples $\alpha \leftarrow_R \mathbb{Z}_q^*$, computes $V = g^{\alpha}$, $A = S_{ID_s} \cdot V + \alpha \cdot P_1$, $\gamma = h_2(\omega, e(A, P_2))$ and $U = \gamma \cdot S_{ID_s} + \alpha \cdot P_1$, where ω is the proxy warrant which consists the identity information of the original signer and the proxy signer, message type to be signed by the proxy signer, the delegation limits of authority, valid period of delegation and so on.
  2. *Proxy Delivery*: The original signer sends (ω, V, U) to a proxy signer and publishes the warrant-voucher pair (ω, V) in public.
  3. *Proxy Verification*: Upon receiving secret values from the original signer, the proxy signer accepts it if the following equations holds:

$$e(U, P_2) = e(Q_{ID_s}, P_{pub})^{h_2(\omega, e(Q_{ID_s}, P_{pub})^V \cdot V)} \cdot V \tag{1}$$

- Proxy blind signature generation: When given the message m, user and proxy signer execute the following steps to generate a proxy blind signature:
  1. *Blinding*:
     (a) The proxy signer randomly samples $r \leftarrow_R \mathbb{Z}_q^*$, computes $R = g^r$, $K_p = U + R \cdot S_{ID_p}$ and sends R to the user via secure channel.
     (b) Now the user samples two random values $k_1, k_2 \leftarrow_R \mathbb{Z}_q^*$ and blind the message by $T = R \cdot g^{k_1} \cdot V^{k_2} \cdot e(Q_{ID_s}, P_{pub})^{\gamma k_2} \cdot e(Q_{ID_p}, P_{pub})^{Rk_2 + k_1 k_2}$, $\hat{h} = h_1(m, T) + k_1 \cdot k_2^{-1}$, and sends $\hat{h}$ to the signer.
  2. *Signing*: The proxy signer computes and returns the blinded signature $\hat{S} = \hat{h} \cdot K_p + r \cdot P_1$ to the user.
  3. *Unblinding*: Upon receiving $\hat{S}$, user unblinds it by computing $S = \hat{S} + k_1 \cdot P_1$, $h = \hat{h} - k_2$ and $d = R \cdot h - k_1 \cdot k_2$. The proxy blind signature on the message m is $\sigma = (S, h, d)$.

– Verify: On input a signature σ of message m, public keys Q_{ID_s}, Q_{ID_p}, proxy warrant-voucher pair (ω, V), the verifier firstly recovers $\gamma = h_2(\omega, e(Q_{ID_s}, P_{pub})^V \cdot V)$, then accepts the signature if

$$h = h_1(m, e(S, P_2) \cdot e(Q_{ID_s}, P_{pub})^{-h\gamma} \cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}) \qquad (2)$$

holds.

Correctness
The correctness of Eq. (1), i.e., the proxy verification can be proved as follows:

$$\begin{aligned}
e(Q_{ID_s},&P_{pub})^{h_2(\omega,e(Q_{ID_s},P_{pub})^V\cdot V)} \cdot V = e(Q_{ID_s}, P_{pub})^{h_2(\omega,e(S_{ID_s}\cdot V,P_2)\cdot V)} \cdot V\\
&= e(Q_{ID_s}, P_{pub})^{h_2(\omega,e(S_{ID_s}\cdot V,P_2)\cdot g^\alpha)} \cdot g^\alpha\\
&= e(Q_{ID_s}, P_{pub})^{h_2(\omega,e(S_{ID_s}\cdot V+\alpha\cdot P_1,P_2))} \cdot g^\alpha\\
&= e(Q_{ID_s}, P_{pub})^{h_2(\omega,e(A,P_2))} \cdot g^\alpha = e(Q_{ID_s}, P_{pub})^\gamma \cdot g^\alpha\\
&= e(\gamma \cdot S_{ID_s} + \alpha \cdot P_1, P_2) = e(U, P_2)
\end{aligned}$$

Furthermore, the correctness of Eq. (2), i.e., the proxy blind signature verification can be proved as follows.

$$\begin{aligned}
&e(S, P_2) \cdot e(Q_{ID_s}, P_{pub})^{-h\gamma} \cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}\\
&\;= e(\hat{S} + k_1 \cdot P_1, P_2) \cdot e(Q_{ID_s}, P_{pub})^{-h\gamma} \cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}\\
&\;= e(\hat{h} \cdot K_p + r \cdot P_1 + k_1 \cdot P_1, P_2) \cdot e(Q_{ID_s}, P_{pub})^{-h\gamma} \cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}\\
&\;= e(U + R \cdot S_{ID_p}, P_1)^{h+k_2} \cdot g^r \cdot g^{k_1} \cdot e(Q_{ID_s}, P_{pub})^{-h\gamma} \cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}\\
&\;= e(\gamma \cdot S_{ID_s} + \alpha \cdot P_1 + R \cdot S_{ID_p}, P_2)^{h+k_2} \cdot R \cdot g^{k_1} \cdot e(Q_{ID_s}, P_{pub})^{-h\gamma}\\
&\;\cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}\\
&\;= e(S_{ID_s}, P_2)^{\gamma\cdot(h+k_2)} \cdot g^{\alpha\cdot(h+k_2)} \cdot e(S_{ID_p}, P_2)^{R\cdot(h+k_2)} \cdot R \cdot g^{k_1}\\
&\;\cdot e(Q_{ID_s}, P_{pub})^{-h\gamma} \cdot e(Q_{ID_p}, P_{pub})^{-d} \cdot V^{-h}\\
&\;= e(Q_{ID_s}, P_{pub})^{\gamma\cdot k_2} \cdot V^{k_2} \cdot e(Q_{ID_p}, P_{pub})^{R\cdot(h+k_2)-d} \cdot R \cdot g^{k_1}\\
&\;= T
\end{aligned}$$

7 Security Analysis

In this section, we will show that the proposed improved schemes meet the security requirements, especially the *unlinkability* that plays a critical role for the privacy preservation in Bitcoin.

7.1 Analysis of Blind Signature

A blind signature scheme should meet three security properties: blindness, unforgeability, and unlinkability [7,9,15,17,33]. Now we examine the security of our scheme described in Sect. 5 according to the property:

- *Blindness*: To protect the privacy of signed message, the signer should not know the content of message when she signs. In our scheme, user securely samples two random values k_1, k_2 and calculates T and u to randomize the message m using one-way hash function. We can see that, at this point, the signer has no information about T and u, thus she cannot obtain the content of message.
- *Unforgeability*: To guarantee authenticity and non-repudiation of signature, no one, except the signer, can produce a valid blind signature without permission. The unforgeability of our scheme is lying on signer's private key $S_{ID} = s \cdot Q_{ID}$ (calculated by KGC's master private key s) and random value r. It is easy to prove that our scheme can be reduced to CDH assumption in the random oracle model using Forking Lemma [29]. Thus, no one can forge a valid signature without private key.
- *Unlinkability*: To provide a fully preservation of message's privacy, the signer should not trace the connection between revealed signature and the blinded message she signed before. In our scheme, before the message m and its signature $\sigma = \{S, h, d\}$ are published, the user will break the linear relationship of blinding factors k_1, k_2 among S and d, which means that the signer cannot find any helpful information from the tuple of $\{r, Q_{ID}, S_{ID}, \hat{h}\}$ she knows. Thus, it is hard for signer to link the blinded signature $\{\hat{S}, \hat{h}\}$ with the public message-signature pair.

7.2 Analysis of Proxy Blind Signature

A proxy blind signature scheme should meet seven security properties: distinguishable, identifiablity, prevention of misuse, non-repudiation, unforgeability, verifiability and unlinkability [6,14,22,36]. Here, we analyze that our scheme in Sect. 6 satisfies these properties:

- *Distinguishability*: To maintain clear boundaries of responsibility, the proxy blind signature generated by proxy signer should be distinguishable from the normal one by original signer. In our scheme, the original signer's private key is S_{ID_s} calculated from ID_s while the proxy signer's private key is S_{ID_p} calculated from different ID_p. Furthermore, the warrant ω, who consists the detail proxy information, is one of the components of the proxy key W and finally embedded into the proxy blind signature. Thus, one can distinguish the proxy blind signature from a normal one easily by verifying the validity of the proxy blind signature.
- *Identifiablity*: For publicly verifiable accountability, the proxy signer, original signer and their agency relationships should be efficiently identified. Using the proxy warrant-voucher pair (ω, V) and both signer's identities, one can verify the validity of blind signature $\sigma = \{S, h, d\}$. However, original signer's identity ID_s and proxy signer's identity ID_p appear in different location of the verification equation (2), it will be unacceptable if any one of them is mismatched. Thus, the proxy signer can be efficiently identified from the proxy signature.

- *Prevention of misuse*: To protect the interests of the original signer, any misuse of proxy key pair deviated from producing proxy signature could be detected publicly. In our improved scheme, the original signer issues $U = h_2(\omega, e(A, P)) \cdot S_{ID_s} + \alpha \cdot P$ during the proxy generation, where the warrant ω consists the detail proxy information, such as identities of both parties, message type to be signed by the proxy signer, the delegation limits of authority, valid period of delegation and so on. Based on the security of original signer's private key S_{ID_s} and random value α, the proxy signer cannot sign any messages deviated from the warrant ω.
- *Non-repudiation*: The proxy blind signature is a proof of both proxy signer and original, therefore, a proxy blind scheme should guarantee neither of them can later deny their signatures. During the blinding phase of our scheme, the proxy key K_p is created by original signer's private key S_{ID_s} and proxy signer's private key S_{ID_p}. Cooperating with the identifiablity analyzed before, we can say that neither of them can sign in place of the other party nor both of them can deny having signed the message.
- *Unforgeability*: To guarantee authenticity of signature, no one, except the proxy signer, can produce a valid proxy blind signature without permission. The unforgeability of our scheme is lying on original signer's private key $S_{ID_s} = s \cdot Q_{ID_s}$, proxy signer's private key $S_{ID_p} = s \cdot Q_{ID_p}$ (both are calculated by KGC's master private key s) and random values α, r. Similarly, based on CDH assumption, it is easy to prove that an P.P.T adversary (even a original signer or signature receiver) cannot forge in our improved scheme without private key.
- *Verifiability*: The proxy blind signature should be verified by anyone. Our improved blind signature can satisfy verifiability as the verifier can check the validity of the proxy blind signature using Eq. (2). The correctness of it is already shown by Eq. (1).
- *Unlinkability*: To protect the privacy of message, the signer (*including proxy signer and original signer*) should not trace the connection between revealed signature and the blinded message she signed before. In the proposed scheme, on one side, the original signer or a verifier has no information about random factor r and blinding factors k_1, k_2, so it's difficult for them to find the relationship between the blinded signature $\{\hat{h}, \hat{S}\}$ and proxy blind signature $\sigma = \{S, h, d\}$. On the other side, given all the signature transcript $\{\hat{h}_i, R_i, \hat{S}_i\}$, it is still unable for proxy signer to link a published proxy blind signature $\sigma = \{S, h, d\}$ with one she signed before because she still has no helpful information of blinding factors k_1, k_2 (including $e(Q_{ID_s}, P_{pub})^{k_2}$, V^{k_s}, $e(Q_{ID_p}, P_{pub})^{k_1 \cdot k_2}$, $e(Q_{ID_p}, P_{pub})^{k_2}$, $e(P, P)^{k_1}$ and so on). Thus, our improved proxy blind signature scheme can achieve perfectly unlinkability.

8 Performance Analysis and Comparison

To show the practicality of our protocols proposed in Sect. 5 and 6, we analyze their performance and compare them with Sarde et al.' schemes [32]. The notations used in this section are as follows:

- T_{bp}: the execution time of the bilinear pairing.
- T_{sm}: the execution time of scalar multiplication over group $\mathbb{G}_1$.
- T_{gm}: the execution time of multiplication of two elements over group $\mathbb{G}_T$ (and $\mathbb{G}_2$ in Sarde et al.).
- T_{ga}: the execution time of addition of two elements over group $\mathbb{G}_1$.
- T_{inv}: the execution time of inversion of an integer under modulo q.
- T_{im}: the execution time of integer multiplication modulo q.
- T_h: the execution time of hashing.

Table 1. Comparison of computational cost of blind signature

Phases	Sarde et al. [32]	This paper
Extract	$T_{sm} + T_h$	$T_{sm} + T_h$
Blinding	$T_{bp} + 3 \times T_{sm} + 2 \times T_{im} + 2 \times T_h$	$T_{bp} + 4 \times T_{sm} + T_{im} + 2 \times T_h$
Signing	T_{sm}	T_{sm}
Unblinding	$T_{sm} + T_{im}$	$T_{sm} + T_{im}$
Verification	$2 \times T_{bp} + T_{sm} + T_{inv} + T_h$	$2 \times T_{bp} + T_{sm} + T_{inv} + T_h$

The comparison of algebraic operations required for different phases are summarized in Table 1 and Table 2. We mark that the computational cost for unlinkability of blind signature is T_{sm} in the binding phase. The computational cost for unlinkability of proxy blind signature is $T_{bp} + 2 \times T_{sm} + 2 \times T_{im}$ in the blinding phase and T_{im} in the unblinding phase of proxy blind signature generation. However, our proposed schemes still benefit from the higher-performance of Type-3 bilinear pairing, which has been optimized many years and can be executed faster than the symmetrical pairing used by Sarde et al.

9 Application for Privacy Preservation in Bitcoin

We review the Blindcoin project [37] in this section, which utilizes a blind signature scheme to hide the mapping between a user's input and output addresses from mix.

According to the transaction architecture (i.e., UTXO or unspent transaction outputs) of Bitcoin, it is linkable between senders and receivers within a transaction, therefore, by analyzing the public content (e.g., analytical attack), one can infer some privacy information. Some analysis attacks have succeed to extract users' identities [4,20,25,28,30,31,35,44]. The simplest solution to mitigate this attack is to obfuscate the transaction's relationships with the help of mixer. Senders wrap transactions with the output addresses of mixer, and then mixer wrap other irrelevant transactions with the output addresses of receivers. When a massive of transactions engage in this mixing task, the relationships

Table 2. Comparison of computational cost of proxy blind signature

Phases		**Sarde et al. [32]**	**This paper**
Extract		$2 \times (T_{sm} + T_h)$	$2 \times (T_{sm} + T_h)$
Proxy delegation	**Proxy generation**	$T_{bp} + 4 \times T_{sm} + T_{gm} + T_{ga} + T_h$	$T_{bp} + 4 \times T_{sm} + T_{gm} + T_{ga} + T_h$
	Proxy verification	$3 \times T_{bp} + T_{sm} + 3 \times T_{gm} + T_h$	$3 \times T_{bp} + T_{sm} + 3 \times T_{gm} + T_h$
Proxy blind signature generation	**Blinding**	$3 \times T_{bp} + 4 \times T_{sm} + 4 \times T_{gm} + T_{ga} + T_h$	$4 \times T_{bp} + 6 \times T_{sm} + 4 \times T_{gm} + T_{ga} + 2 \times T_{im} + T_h$
	Signing	$2 \times T_{sm}$	$2 \times T_{sm}$
	Unblinding	$2 \times T_{sm} + 2 \times T_{ga}$	$2 \times T_{sm} + 2 \times T_{ga} + T_{im}$
Verification		$4 \times T_{bp} + 4 \times T_{gm} + 4 \times T_{sm} + 3 \times T_{inv} + 2 \times T_h$	$4 \times T_{bp} + 4 \times T_{gm} + 4 \times T_{sm} + 3 \times T_{inv} + 2 \times T_h$

between each transaction's origin and destination are hided well. Blindcoin, presented by Valenta et al., followed the mixing mechanism and combine the blind signature scheme to hide the user's privacy at the mixer side. There are three kind of participants in Blindcoin, i.e., sender $\mathcal{S}$, mixer $\mathcal{M}$ and receiver $\mathcal{R}$. Sender $\mathcal{S}$ anonymously transfers a mount of Bitcoin to receiver $\mathcal{R}$ with the assistance of mixer $\mathcal{M}$ by executing following steps:

- Setup: The mixer $\mathcal{M}$ publishes the mix parameters $\mathsf{mixparams}$ into the public ledger including the expiry date t_1, t_2, t_3, t_4, the value v of Bitcoin put into a transaction, the mixing fee ρ, block chunk ω and so on.
- Sender Submits Offer: The sender $\mathcal{S}$ sends its offer to $\mathcal{M}$ including $\mathsf{mixparams}$ and blinded token $[T = \{\mathsf{addr}_{\mathsf{out}}, n\}]_{\mathsf{Blind}}$. This token T contains the output address $\mathsf{addr}_{\mathsf{out}}$ and secure random number n, and being masked using the blinding phase of blind signature scheme.
- Mixer Answers Partial Warranty: If $\mathcal{M}$ accepts this offer, then it wrap partial warranty using blinded token T, an escrow address $\mathsf{addr}_{\mathsf{esc}}$ for the sender to pay to, and $\mathsf{mixparams}$. This partial warranty will be signed using signing algorithm of blind signature scheme, i.e., forming $\mathsf{PriW} = \{[T]_{\mathsf{Blind}}, \mathsf{addr}_{\mathsf{esc}}, \mathsf{mixparams}\}_{\mathsf{sig}}$.
- Sender Pays: Sender $\mathcal{S}$ then transfer v amount of Bitcoin from any input address $\mathsf{addr}_{\mathsf{in}}$ to $\mathsf{addr}_{\mathsf{esc}}$ by time t_1.
- Mixer Completes Warranty: Once the sender pays the funds, mixer $\mathcal{M}$ must complete the warranty by signing the blinded token, i.e., forming $\mathsf{PubW} = \{[T]_{\mathsf{Blind}}\}_{\mathsf{sig}}$ and publishes it to the public ledger by time t_2. This public information allows any third party to verify that the sender did indeed transfer their funds to the escrow address on time and the mixer has completed the warranty by time t_2.
- Sender Anonymously Unblinds Output Addresses: After seeing PubW in the public ledger, $\mathcal{S}$ can apply unblinding phase to recover the signed token,

formed as $\{T = \{\mathsf{addr}_{\mathsf{out}}, n\}\}_{\mathsf{sig}}$. The sender (anonymously connects with another identity $\mathcal{S}'$) posts the signed token to the public log by time t_3.

- Mixer Pays to Output Address: Once the signed output address being published in public ledger by time t_3, $\mathcal{M}$ computes a beacon function with the inputs of t_3, n and block chunk ω for each token $T = \{\mathsf{addr}_{\mathsf{out}}, n\}$. The chunk destined for that output address will be kept by $\mathcal{M}$ if $\mathsf{Beacon}(t_3, \omega, n) \leq \rho$; else, $\mathcal{M}$ pays v amount of Bitcoin to all unblinded output addresses before time t_4.
- One Party Cheats: If $\mathcal{M}$ fails to pay a chunk to each of the output addresses that are passed the beacon function by time t_4, then $\mathcal{S}$ can publish the partial warranty $\mathsf{PriW} = \{[T]_{\mathsf{Blind}}, \mathsf{addr}_{\mathsf{esc}}, \mathsf{mixparams}\}_{\mathsf{sig}}$, and all the public information (e.g., transaction $\mathsf{tx}(v, \mathsf{addr}_{\mathsf{in}}, \mathsf{addr}_{\mathsf{esc}})$ presented by time t_1, the signed token $\{T = \{\mathsf{addr}_{\mathsf{out}}, n\}\}_{\mathsf{sig}}$) to incriminate $\mathcal{M}$. Every verifier can check the public log and blockchain to see if both parties followed the protocol honestly.

Discussion and Conclusion. Based on the *unlinkability* of blind signature scheme, $\mathcal{M}$ can check the validity of output address but cannot link the output address $\mathsf{addr}_{\mathsf{out}}$ to the corresponding input address $\mathsf{addr}_{\mathsf{in}}$. This is why unlinkability is a core feature to guarantee privacy of Bitcoin even a mixer (who knows many information) is curious or malicious. Therefore, our proposed protocols are extremely valuable for the privacy preservation in Bitcoin.

Acknowledgments. We would like to thank the anonymous reviewers. The work was supported by the National Natural Science Foundation of China (Nos. 61972294, 61932016), the Special Project on Science and Technology Program of Hubei Province (No. 2020AEA013), the Natural Science Foundation of Hubei Province (No. 2020CFA052) and the Wuhan Municipal Science and Technology Project (No. 2020010601012187).

References

1. Bitcoin fog. http://bitcoinfog.com. Accessed 2020
2. Bitmixer. https://bitcointalk.org/index.php?topic=415396.160. Accessed 2020
3. Onionbc. http://6fgd4togcynxyclb.onion/. Accessed 2020
4. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 34–51. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_4
5. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: anonymity for bitcoin with accountable mixes. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 486–504. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_31
6. Chande, M.K., Lee, C.C., Li, C.T.: Cryptanalysis and improvement of a ECDLP based proxy blind signature scheme. J. Discrete Math. Sci. Cryptogr. **21**(1), 23–34 (2018)
7. Chaum, D.: Blind signature system. In: Chaum, D. (eds.) Advances in Cryptology, pp. 153–153. Springer, Boston (1984). https://doi.org/10.1007/978-1-4684-4730-9_14

8. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
9. Fan, C.I., Chen, W.K., Yeh, Y.S.: Randomization enhanced Chaum's blind signature scheme. Comput. Commun. **23**(17), 1677–1680 (2000)
10. Feng, Q., He, D., Zeadally, S., Khan, M.K., Kumar, N.: A survey on privacy protection in blockchain system. J. Network Comput. Appl. **126**, 45–58 (2019)
11. Genkin, D., Papadopoulos, D., Papamanthou, C.: Privacy in decentralized cryptocurrencies. Commun. ACM **61**(6), 78–88 (2018)
12. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: TumbleBit: an untrusted bitcoin-compatible anonymous payment hub. In: Network and Distributed System Security Symposium (2017)
13. Heilman, E., Baldimtsi, F., Goldberg, S.: Blindly signed contracts: anonymous on-blockchain and off-blockchain bitcoin transactions. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 43–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_4
14. Hu, L., Zheng, K., Hu, Z., Yang, Y.: A secure proxy blind signature scheme based on ECDLP. In: 2009 International Conference on Multimedia Information Networking and Security, vol. 1, pp. 454–457. IEEE (2009)
15. Hwang, M.S., Lee, C.C., Lai, Y.C.: An untraceable blind signature scheme. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **86**(7), 1902–1906 (2003)
16. James, S., Gowri, T., Babu, G., Reddy, P.V.: Identity-based blind signature scheme with message recovery. Int. J. Electri. Comput. Eng. (2088–8708) **7**(5) (2017)
17. Juang, W.S., Lei, C.L.: Partially blind threshold signatures based on discrete logarithm. Comput. Commun. **22**(1), 73–86 (1999)
18. Khalilov, M.C.K., Levi, A.: A survey on anonymity and privacy in bitcoin-like digital cash systems. IEEE Commun. Surv. Tutor. **20**(3), 2543–2585 (2018)
19. Kong, W., Shen, J., Vijayakumar, P., Cho, Y., Chang, V.: A practical group blind signature scheme for privacy protection in smart grid. J. Parallel Distrib. Comput. **136**, 29–39 (2020)
20. Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in bitcoin using P2P network traffic. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 469–485. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_30
21. Kumar, M., Katti, C.P., Saxena, P.C.: A secure anonymous e-voting system using identity-based blind signature scheme. In: Shyamasundar, R.K., Singh, V., Vaidya, J. (eds.) ICISS 2017. LNCS, vol. 10717, pp. 29–49. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72598-7_3
22. Lal, S., Awasthi, A.K.: Proxy blind signature scheme. J. Inf. Sci. Eng. Cryptology ePrint Archive, Report 72 (2003)
23. Li, J., Zhang, Y., Yang, S.: Cryptanalysis of new proxy blind signature scheme with warrant. In: International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2005) (2005)
24. Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q.: A survey on the security of blockchain systems. Future Gener. Comput. Syst. **107**, 841–853 (2020)
25. Liao, K., Zhao, Z., Doupé, A., Ahn, G.J.: Behind closed doors: measurement and analysis of Cryptolocker ransoms in bitcoin. In: 2016 APWG Symposium on Electronic Crime Research (eCrime), pp. 1–13. IEEE (2016)
26. Maxwell, G.: CoinSwap: transaction graph disjoint trustless trading, October 2013

27. Meiklejohn, S., Orlandi, C.: Privacy-enhancing overlays in bitcoin. In: Brenner, M., Christin, N., Johnson, B., Rohloff, K. (eds.) FC 2015. LNCS, vol. 8976, pp. 127–141. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48051-9_10
28. Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. Future Internet **5**(2), 237–250 (2013)
29. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_33
30. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Altshuler, Y., Elovici, Y., Cremers, A., Aharony, N., Pentland, A. (eds.) Security and Privacy in Social Networks, pp. 197–223. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-4139-7_10
31. Reynolds, P., Irwin, A.S.: Tracking digital footprints: anonymity within the bitcoin system. J. Money Laundering Control (2017)
32. Sarde, P., Banerjee, A.: A secure ID-based blind and proxy blind signature scheme from bilinear pairings. J. Appl. Secur. Res. **12**(2), 276–286 (2017)
33. Shao, Z.: Improved user efficient blind signatures. Electron. Lett. **36**(16), 1372–1374 (2000)
34. Shaobin, W., Fan, H., Guohua, C.: Secure efficient proxy blind signature schemes based DLP. In: Seventh IEEE International Conference on E-Commerce Technology (CEC 2005), pp. 452–455. IEEE (2005)
35. Spagnuolo, M., Maggi, F., Zanero, S.: BitIodine: extracting intelligence from the bitcoin network. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 457–468. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_29
36. Tan, Z., Liu, Z., Tang, C.: Digital proxy blind signature schemes based on DLP and ECDLP. MM Res. Preprints **21**(7), 212–217 (2002)
37. Valenta, L., Rowan, B.: Blindcoin: blinded, accountable mixes for bitcoin. In: Brenner, M., Christin, N., Johnson, B., Rohloff, K. (eds.) FC 2015. LNCS, vol. 8976, pp. 112–126. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48051-9_9
38. Verma, G.K., Singh, B.: Efficient message recovery proxy blind signature scheme from pairings. Trans. Emerg. Telecommun. Technol. **28**(11), e3167 (2017)
39. Verma, G.K., Singh, B., Singh, H.: Provably secure certificate-based proxy blind signature scheme from pairings. Inf. Sci. **468**, 1–13 (2018)
40. Xue, Q., Cao, Z.: A new proxy blind signature scheme with warrant. In: IEEE Conference on Cybernetics and Intelligent Systems, 2004. vol. 2, pp. 1386–1391. IEEE (2004)
41. Zheng, Z., Xie, S., Dai, H.N., Chen, X., Wang, H.: Blockchain challenges and opportunities: a survey. Int. J. Web Grid Serv. **14**(4), 352–375 (2018)
42. Zhu, H., Tan, Y.a., Zhang, X., Zhu, L., Zhang, C., Zheng, J.: A round-optimal lattice-based blind signature scheme for cloud services. Future Gener. Comput. Syst. **73**, 106–114 (2017)
43. Zhu, H., Tan, Y.a., Zhu, L., Zhang, Q., Li, Y.: An efficient identity-based proxy blind signature for semioffline services. Wireless Commun. Mobile Comput. **2018** (2018)
44. Zola, F., Eguimendia, M., Bruse, J.L., Urrutia, R.O.: Cascading machine learning to attack bitcoin anonymity. In: 2019 IEEE International Conference on Blockchain (Blockchain), pp. 10–17. IEEE (2019)

Blockchain-Based Sealed-Bid Domain Name Auction Protocol

Genhua Lu[1], Yi Zhang[1], Zhongxiang Lu[2], Jun Shao[1](✉), and Guiyi Wei[2]

[1] School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China

[2] School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China

weigy@zjgsu.edu.cn

Abstract. Domain name system (DNS), mapping domain names to IP addresses, is critical to the Internet's running. However, the centralized architecture is one of the major criticisms of the current DNS. Many works suggest introducing the blockchain into DNS, but the existing blockchain-based DNSs do not support the domain name auction that is important for the domain name transfer. To solve this problem, we in this paper propose a blockchain-based sealed-bid domain name auction protocol by combining the smart contract, the Pedersen commitment, and zero-knowledge proof. Compared with the previous blockchain-based auction protocols, our proposal is the first one holding the tx-fairness, bidding-fairness, bid-guarantee, and fund-privacy at the same time. For showing its effectiveness, we also give an illustration of our proposal based on the smart contract system in Ethereum.

Keywords: DNS · Blockchain · Sealed-bid auction

1 Introduction

Domain Name System (DNS) is one of the crucial infrastructures of the Internet. With its help, people can use the human-readable domain name to visit websites instead of the hard-to-remember digital IP address [1]. The fundamental part of DNS is the management of root names and top-level domain names, which is principally maintained by the Internet Corporation for Assigned Names and Numbers (ICANN) [2]. This centralized architecture of the current DNS suffers from many attacks [3], such as the single-point failure and power abuse. One of the famous incidents of single-point failure on the current DNS is the 2016 Dyn cyberattack [4]. The failure of DNS provided by Dyn caused major websites, including Amazon.com, GitHub, Twitter, and Reddit, to convert unreachable

This work was supported by National Key Research and Development Program of China under grant number: 2019YFB1804500.

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 25–38, 2021.
https://doi.org/10.1007/978-3-030-80851-8_3

via their corresponding domain names. To mitigate the above problems, it is natural to introduce the decentralized architecture into the current DNS.

Meanwhile, with over ten years of development, the blockchain has become the most famous and successful decentralized architecture. Recently, many researchers have suggested introducing blockchain into the DNS [5–10]. On the other hand, the domain name auction is one of the primary ways to obtain the DNS domain name [11]; however, the existing blockchain-based DNS solutions cannot provide the functionality of domain name auction as well as expected.

- Namecoin [5], ConsortiumDNS [6], EmerDNS [7], and Blockstack [8] mainly focus on domain name registration but ignoring the domain name auction.
- Ethereum Name Service (ENS) [9] provides the auction functionality, but it is only for cryptocurrency addresses, not IP addresses.
- Handshake [10] realizes a sealed-bid domain name auction, where the bid will not be revealed until all the bids are committed. The main advantage of a sealed-bid domain name auction is that bidders are more willing to bid according to the real value of the auction item [12]. However, the last bidder in Handshake can always win the auction with a reasonable bid since the current possible highest bid is predictable.

In this paper, aiming at the above challenges, we propose a sealed-bid blockchain-based domain name auction protocol to realize the domain name transfer in a fair, secure, and privacy-preserving way. The main contributions of this paper can be summarized as follows.

- We compile the security requirements for the blockchain-based sealed-bid domain name auction for the first time, including the tx-fairness, bidding-fairness, bid-guarantee, and fund-privacy.
- By integrating the account-based consortium blockchain with the anonymous fund, smart contract, the Pedersen commitment, and zero-knowledge proof, we propose the first blockchain-based sealed-bid domain name auction protocol.
- We also give a security analysis to show our proposal satisfies the tx-fairness, bidding-fairness, bid-guarantee, and fund-privacy simultaneously.
- At last, we give an illustration of our proposal by using the smart contract system in Ethereum to show the feasibility of our proposal.

The remainder of this paper is organized as follows. In Sect. 2, we give the system and security models, as well as the design goals in this paper. In what follows, we give some basic knowledge related to our proposal. After that, we present our domain name auction protocol and its analysis in Sect. 4 and Sect. 5, respectively. Section 6 reviews the related works. At last, we conclude this paper in Sect. 7.

2 Models and Design Goals

This section will present the system and security models for our proposed blockchain-based domain name auction protocol and identify the properties our proposal should have.

2.1 System Model

In our system model, we mainly consider a typical domain name auction scenario, where we have a domain name owner, multiple domain name bidders, and a blockchain system, as shown in Fig. 1.

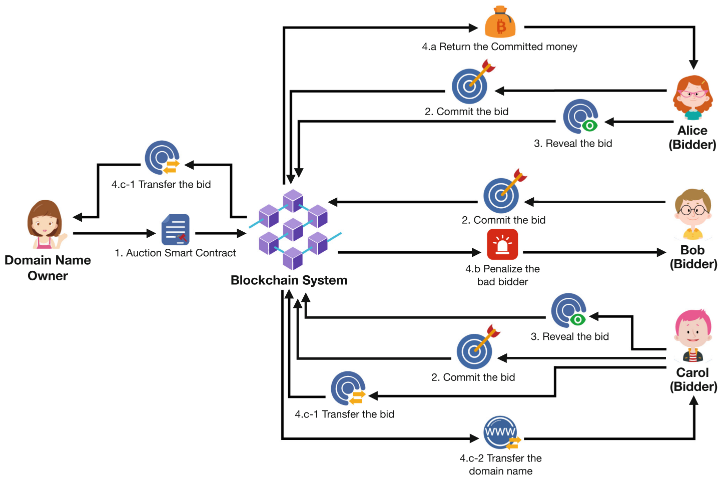


Fig. 1. The system model considered in this paper.

Our system is heavily based on the blockchain system that is an account-based consortium blockchain. The consensus nodes in the blockchain system could be the network authority in each government around the world. In other words, all the countries together, instead of the single entity ICANN alone, maintain the mapping from domain names to IP addresses. Hence, our system overcomes the problems caused by the centralized architecture.

All the participants in the blockchain system have at least one account. Every account has its corresponding fund of an encrypted format, such as $\mathsf{fund} = g^c h^r$, where c is the amount of the fund, r is a random number, and g and h are random elements from the underlying finite cyclic group. Furthermore, the blockchain also records the relationships between the domain name and the account.

When the domain name owner wants to sell his/her domain name, he/she deploys a smart contract in the blockchain. The domain name bidder can bid the domain name in a sealed-bid way, particularly the Vickrey auction. The Vickrey auction allows the bidder to submit the bid and keep the bid secret before all bids are submitted, and the winning bidder pays the second-highest bid instead of his/her bid. At the end of the auction, the domain name is transferred from the domain name owner to the winning bidder.

More detailed information about the underlying blockchain system and auction types can be found in Sect. 3.1 and Sect. 3.2, respectively.

2.2 Security Model

As with other blockchain-based systems [5–10], the blockchain in our system is assumed to be honest-but-curious. In particular, the blockchain system will faithfully execute the smart contract deployed by the domain name owner. However, it is also curious about others' secret information, including bids' values before the corresponding bidders open them and the values of funds the bidders have.

The domain name owner could be malicious. He/she will try his/her best to transfer to others the domain names not belonging to him/her or obtain the money from the bidder without transferring the domain name. The domain name bidders could also be malicious. They would launch active attacks, such as manipulating the communication data, to win the auction deviating from the rules of Vickrey auction or to obtain the domain name without payment.

2.3 Design Goal

This paper aims to design a blockchain-based domain name auction protocol with the following security properties.

TX-Fairness. The proposal should guarantee that once the domain name owner transfers the domain name to the winning bidder, he/she can obtain the corresponding money; vice versa.

Bidding-Fairness. The winning probability of the auction is not related to the time of bidding. In particular, the late bidding will not bring any advantage over the early bidding.

Bid-Guarantee. Anyone can verify that the bidder has enough money to cover his/her bid or not.

Fund-Privacy. No one can deduce how much money the bidder has in the blockchain system from the auction process.

3 Preliminaries

In this section, we will review some basic knowledge that will be used in our proposal, including the account-based consortium blockchain, auction types, commitment scheme, and zero-knowledge proof for an inequality.

3.1 Account-Based Consortium Blockchain

Blockchain is a technology enabling a group of users to build trust relationships in a decentralized way. Since the invention of the first blockchain system—Bitcoin [13], many blockchain systems have been proposed with different properties [14–16]. Generally speaking, the existing blockchain systems can be classified into three categories: public blockchain, consortium blockchain, and private blockchain, where anyone, only a specific group of users, and only a particular user can be the consensus nodes appending data to the blockchain, respectively. Unlike the existing blockchain-based DNS solutions [8,10,12], we adopt the consortium blockchain in our system due to its effectiveness and efficiency. Furthermore, we assume that the consensus nodes in our system are the network authorities in governments, which may be more realistic.

The consortium blockchain used in this paper belongs to the account-based style. Every user in this blockchain system has one or more accounts, and each account is recorded with its possessed fund and domain names. The fund recorded in the blockchain is with an encrypted format, such as $\mathsf{fund} = g^c h^r$, where c is the value of the fund, r is a random number, g and h are random elements in the underlying finite cyclic group, and no one knows $\log_h g$. In contrast, the domain names are recorded in plaintext format since they will be transferred from one account to another account through the sealed-bid auction.

At last, the consortium blockchain also supports smart contracts, which has been a fundamental functionality of a blockchain system since the invention of Ethereum [17]. Particularly, our consortium blockchain's consensus nodes can run the Turing-complete program instead of the program's creator.

Since we in this paper focus on the design of blockchain-based sealed-bid domain name auction, we omit the details of the underlying consortium blockchain. One may get such a blockchain from Ethereum with the anonymous fund and a consensus algorithm for consortium blockchains.

3.2 Auction Types

An auction is a process where people can buy items via the bidding method, and it usually involves an auctioneer and many bidders. The corresponding seller typically not appears but delegates the auctioneer to sell the items. Traditionally, there exist two main types of auctions [18]: open auction and sealed-bid auction. Every bidder can see others' bids in the former type and submit the bid to the auctioneer many times. In contrast, bidders in the latter type can submit their bids to the auctioneer only once in a private way. After all bids are submitted in a preset period, the auctioneer will open the bids and decide who the winner is. Since the sealed-bid auction can lead to a reasonable bid for the auction item with a higher probability [19], we adopt it in our system. Especially, we use the special sealed-bid auction named Vickrey auction in our system, where the bidder with the highest bid wins while he/she only needs to pay with the second-highest bid. Furthermore, we replace the auctioneer with the blockchain and assume the blockchain as an honest-but-curious party.

3.3 Pedersen Commitment

As we mentioned before, the auction we apply in this paper is the Vickrey auction, where the bid should be kept secret before the corresponding bidder opens it. To realize this process, we make use of the Pedersen commitment [20] that is widely used in many anonymous cryptocurrencies [21–23].

A commitment scheme usually contains two stages: `commit` and `reveal`. In the former stage, the sender commits a value to the receiver and reveals it in the latter stage. It requires that the receiver cannot reveal the committed value before the latter stage, and the sender cannot change the committed value after the former stage.

The Pedersen commitment goes as follows. In the `commit` stage, the sender sends $\mathsf{fund} = g^x h^r$ to the receiver, where x is the committed value, r is a random number, g and h are public parameters from a finite cyclic group with a big prime order, and no one knows $\log_h g$. Later on, the sender in the `reveal` stage sends (x', r') to the receiver. If $\mathsf{fund} = g^{x'} h^{r'}$ holds, the receiver accepts the committed value; otherwise, he/she rejects it. One of the properties of the Pedersen commitment is homomorphic addition. In particular, given $\mathsf{fund}_1 = g^{x_1} h^{r_1}$ and $\mathsf{fund}_2 = g^{x_2} h^{r_2}$, we have that $\mathsf{fund}_1 \cdot \mathsf{fund}_2 = g^{x_1} h^{r_1} \cdot g^{x_2} h^{r_2} = g^{x_1+x_2} h^{r_1+r_2}$. The value of $\mathsf{fund}_1 \cdot \mathsf{fund}_2$ is essentially a commitment to $x_1 + x_2$.

3.4 Zero-Knowledge Proof

The bid-guarantee and fund-privacy are the two security properties that our proposal should satisfy. In particular, the bidder in our proposal has the capability to prove that he/she has enough money to cover the committed bid without revealing how much fund he/she has. In this paper, we make use of the zero-knowledge proof for the inequality between two positive integers, which is original from [23].

Given $\mathsf{fund}_1 = g^{x_1} h^{r_1}$, $\{\mathsf{fund}'_{2i} = g^{a_i \cdot 2^i} h^{r'_{2i}}\}_{i=0}^{\ell}$, the prover proves that $x_1 \geq \sum_{i=0}^{\ell} a_i \cdot 2^i$ with the knowledge of x_1, r_1, and $\{a_i \in \{0,1\}, r'_{2i}\}_{i=0}^{\ell}$, where g and h are the same as that in the Pedersen commitment, and ℓ is a positive integer larger than the bit-length of any possible value of the fund in our system. The whole process contains the following two parts.

In the first part, the prover proves that $x_1 = \sum_{i=0}^{\ell} a_i \cdot 2^i + \sum_{i=0}^{\ell} b_i \cdot 2^i$ with the conditions $\mathsf{fund}_1 = g^{x_1} h^{r_1}$, $\{\mathsf{fund}'_{2i} = g^{a_i \cdot 2^i} h^{r'_{2i}}\}_{i=0}^{\ell}$, and $\{\mathsf{fund}'_{3i} = g^{b_i \cdot 2^i} h^{r'_{3i}}\}_{i=0}^{\ell}$ by providing a signature corresponding to the public key $(h, \mathsf{fund}_1 / (\prod_{i=0}^{\ell} \mathsf{fund}'_{2i} \cdot \mathsf{fund}'_{3i}))$. Note that if $x_1 = \sum_{i=0}^{\ell} a_i \cdot 2^i + \sum_{i=0}^{\ell} b_i \cdot 2^i$, the prover knows the value of $\log_h \mathsf{fund}_1 / (\prod_{i=0}^{\ell} \mathsf{fund}'_{2i} \cdot \mathsf{fund}'_{3i}) = r_1 - \sum_{i=0}^{\ell} (r'_{2i} + r'_{3i})$ and can generate the signature; otherwise, the prover does not know it and cannot generate the signature since $\log_h g$ is kept secret from everyone.

In the second part, the prover proves that all a_i's and b_i's belong to $\{0,1\}$ as follows. Let us take a_i as an example. The prover only needs to provide a ring signature corresponding to the public keys (h, fund'_{2i}) and $(h, \mathsf{fund}'_{2i}/g^{2^i})$. The ring signature scheme [24] allows the verifier to check the signature's validity without revealing the public key corresponding to the real signing key. Note that if $a_i = 0$, the prover knows $\log_h \mathsf{fund}'_{2i} = r'_{2i}$; and if $a_i = 1$, the prover knows $\log_h \mathsf{fund}'_{2i}/g^{2^i} = r'_{2i}$ instead; for other cases, the prover does not know either $\log_h \mathsf{fund}'_{2i}$ or $\log_h \mathsf{fund}'_{2i}/g^{2^i}$ since $\log_h g$ is kept secret from everyone. As a result, the verifier can check whether $a_i \in \{0,1\}$ via the ring signature.

4 Our Proposal

We are now ready to present our blockchain-based sealed-bid auction protocol, where we have the following four phases: Create the Auction, Commit the Bid, Reveal the Bid, and Close the Auction. In the first phase, the domain name owner deploys a smart contract in the blockchain to start the sealed-bid domain name auction. The bidder would submit his/her bid on the domain name via the smart contract in the second phase. After that, the bidder would open the bid in a predefined period. At last, the blockchain would finalize the auction according to the smart contract. Note that we have four functions in the auction smart contract, namely CREATE (Fig. 2), COMMIT (Fig. 3), REVEAL (Fig. 4), and FINALIZE (Fig. 5), which are invoked in the four phases, respectively.

4.1 Create the Auction

After deploying the smart contract, the domain name owner invokes the function CREATE with $(T_1, T_2, \mathtt{name}, \sigma)$, where T_1 and T_2 are the respective deadlines for committing and revealing bids, `name` is the domain name for bidding, and the last value is a signature on the smart contract under the public key corresponding to the account possessing the domain name. When the consensus nodes in the blockchain system receive the function call, they check the validity of $(T_1, T_2, \mathtt{name}, \sigma)$ and whether there is no current active auction smart contract corresponding to `name`. If all of them are valid, the consensus nodes will run the rest of the function CREATE; otherwise, they stop running the function.

```
function CREATE($T_1$, $T_2$, name, $\sigma$)
    if $T_1$, $T_2$, name, and $\sigma$ are valid then
        if no active auction smart contract corresponding to name then
            initialize array bidder[ ]
            set state = 1
    return
```

Fig. 2. Pseudo-code for the function CREATE in the auction smart contract.

The pseudo-code of the function CREATE can be found in Fig. 2. The array `bidder`[ ] is used to record the information related to the bidder, including the public key `pk`, commitment $\{\mathtt{fund}'_i = g^{c'_i \cdot 2^i} h^{r'_i}\}_{i=0}^{\ell}$ for the bid, and the values $\{(c'_i, r'_i)\}_{i=0}^{\ell}$ for revealing the bid, where g, h, ℓ are the same as that explained in Sect. 3. The domain of `state` is $\{0, 1\}$, and 0 and 1 denote the inactive and active state of the smart contract, respectively.

4.2 Commit the Bid

When the bidder finds some auction smart contract on the domain name he/she is interested in, he/she invokes the function COMMIT with

$(\mathtt{pk}, \{\mathsf{fund}'_i\}_{i=0}^{\ell}, \sigma, \mathtt{ZKP})$, where $\mathtt{pk}$ is the public key corresponding to one of the bidder's accounts, $\{\mathsf{fund}'_i = g^{c'_i \cdot 2^i} h^{r_i}\}_{i=0}^{\ell}$ is the commitment for the bid $\sum_{i=0}^{\ell} c'_i \cdot 2^i$, σ is a signature on the smart contract under the public key $\mathtt{pk}$, and $\mathtt{ZKP}$ is the zero-knowledge proof for the statement $c \leq \sum_{i=0}^{\ell} c'_i \cdot 2^i \leq 0$. Here, c is the value of the fund corresponding to $\mathtt{pk}$, and it is recorded as $\mathsf{fund} = g^c h^r$ in the blockchain as mentioned in Sect. 3.1. It is easy to see that we can obtain $\mathtt{ZKP}$ as the steps in Sect. 3.4. When the consensus nodes receive the function call for Commit, they check whether the array $\mathtt{bidder}[\,]$ does not contain $\mathtt{pk}$, whether it is still in the valid period according to T_1, and the validity of $(\sigma, \mathtt{ZKP})$. If all of them are valid, the consensus nodes continue running the function Commit; otherwise, they stop running the function. The pseudo-code of the function Commit can be found in Fig. 3.

```
function Commit($\mathtt{pk}$, $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$, $\sigma$, $\mathtt{ZKP}$)
    if $\mathtt{pk}$ does not exist in $\mathtt{bidder}[\,]$ then
        if current time and $(\sigma, \mathtt{ZKP})$ are valid then
            add $\mathtt{pk}$ and $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$ into $\mathtt{bidder}[\,]$
    return
```

Fig. 3. Pseudo-code for the function Commit in the auction smart contract.

4.3 Reveal the Bid

When the bidding phase stops according to T_1, the bidder can start to reveal his/her bid by invoking the function Reveal with $(\mathtt{pk}, \{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell})$, where $\mathtt{pk}$ is the same public key the bidder used to invoke the function Commit, and $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$ are the values used to compute the commitment $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$. When the consensus nodes receive the function call for Reveal, they check whether it is still in the valid period according to T_2, whether there exist $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$ corresponding to $\mathtt{pk}$ in the array $\mathtt{bidder}[\,]$, and whether all $\mathsf{fund}'_i = g^{\bar{c}'_i \cdot 2^i} h^{\bar{r}'_i}$ for $i = 0, \cdots, \ell$ hold. If all of them are valid, the consensus nodes continue running the function Reveal; otherwise, they stop running the function. The pseudo-code of the function Reveal can be found in Fig. 4.

4.4 Close the Auction

After the third phase, anyone can deduce who is the winner according to $\{\bar{c}'_i\}$ recorded in the array $\mathtt{bidder}[\,]$. In this case, the auction winner will be the one who invokes the function Finalize. Note that anyone can invoke the function Finalize. In other words, the domain name owner can invoke the function for a quicker domain name transfer.

When the consensus nodes receive the function call, they check whether it is the time to finalize the auction according to T_2 and whether the state of the

```
function REVEAL(pk, $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$)
    if pk exists in bidder[] then
        if current time is valid then
            if $\mathsf{fund}'_i = g^{\bar{c}'_i \cdot 2^i} h^{\bar{r}'_i}$ for $i = 0, \cdots, \ell$ then
                add $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$ into bidder[] according to pk
    return
```

Fig. 4. Pseudo-code for the function Reveal in the auction smart contract.

smart contract is active. If both of them are valid, the consensus nodes continue running the function Finalize; otherwise, they stop running the function. The pseudo-code of the function Finalize can be found in Fig. 5, where we mainly deal with three cases for the bidders. The first one is that the bidder failed to open his/her bid in the third phase, and they will be punished by reducing the bid from the corresponding fund. The second one is for the winner whose money of the second-highest bid value will be transferred to the domain name owner, and the domain name is transferred to the winner from the domain name owner. The last case is for the ones following our protocol but losing this auction. We do not need to do anything in this case.

```
function FINALIZE( )
    if it's time for finalizing the smart contract then
        if state == 1 then
            for each item in bidder[] do
                if the bidder corresponding pk failed to reveal the bid then
                    punish the corresponding bidder by reducing his/her fund
                if the bidder winning the bidding then
                    reduce the winner's fund according to the second higher price
                    transfer name to the winner
                    set state = 0
    return
```

Fig. 5. Pseudo-code for the function Finalize in the auction smart contract.

5 Analysis of Our Proposal

In this section, we will analyze our proposal in terms of security and feasibility.

5.1 Security Analysis

This subsection shows our proposal satisfies the tx-fairness, bidding-fairness, bid-guarantee, and fund-privacy one by one as follows.

TX-Fairness. In this paper, we assume that the consensus nodes will faithfully execute the smart contract in the immutable blockchain. Hence, the codes in lines 8-11 in the function Finalize transferring the money and domain name are executed or not executed at the same time. In this case, we have the tx-fairness immediately.

Bidding-Fairness. To show the bidding-fairness of our proposal, we only need to show that no one can reveal the bid from the corresponding commitment. Fortunately, we can obtain this claim directly from the security of the Pedersen commitment.

Bid-Guarantee. It is easy to see that the zero-knowledge proof used in our system guarantees that the bidder has enough money to cover the bid.

Fund-Privacy. Firstly, $\mathsf{fund} = g^c h^r$ won't leak any information about c, since there are many pair (c', r')'s satisfying $\mathsf{fund} = g^{c'} h^{r'}$. Secondly, $\mathtt{ZKP}$ won't leak the concrete value of c due to the security property of the underlying zero-knowledge proof. Hence, we have the fund-privacy property.

5.2 Performance Evaluation

In this subsection, we evaluate our proposal's performance based on our experimental results in terms of off-chain cost and on-chain cost. The prototype we implement is based on the BigInteger class of Java language and the BigNumber library of Solidity language for the off-chain and on-chain parts. Furthermore, we run our prototype in a laptop with Intel(R) Core(TM) i5-1038NG7 CPU @2.00 GHz, 16 GB RAM, macOS 11.2 operating system, and Java 14, and the rinkeby test chain of Ethereum for obtaining the off-chain cost and on-chain cost, respectively. In our experiments, we set the bit-length of the order of g and h as 160, and all experiments are conducted 20 times and the average is recorded.

Off-Chain Cost. The main computational cost for the off-chain part of our proposal is due to the generation of commitments and zero-knowledge proofs. In particular, the bidder has to generate a commitment for the fund and a zero-knowledge proof for a statement. The statement shows that the bidder has enough money to cover his/her bidding. The experimental results can be found in Fig. 6. The abscissa represents the values of ℓ, and the ordinate denotes the computing time. From Fig. 6, "commitA" and "proofGen" denote the commitment commiting and the zero-knowledge proof of fund, respectively. And we can see that the computational cost is proportional to ℓ as expected, while the cost is acceptable.

On-Chain Cost. The on-chain cost is mainly due to the gas cost for the on-chain part of our proposal, especially, the verification of zero-knowledge proofs in the function Commit and the verification of commitments in the function Reveal. Here, we ignore the gas cost due to the function of Finalize. The experimental results can be found in Fig. 7, where "commit", "proof_x",

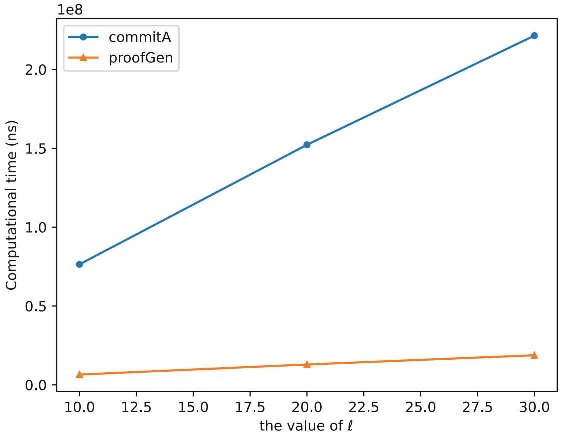


Fig. 6. The off-chain computational cost of our proposal.

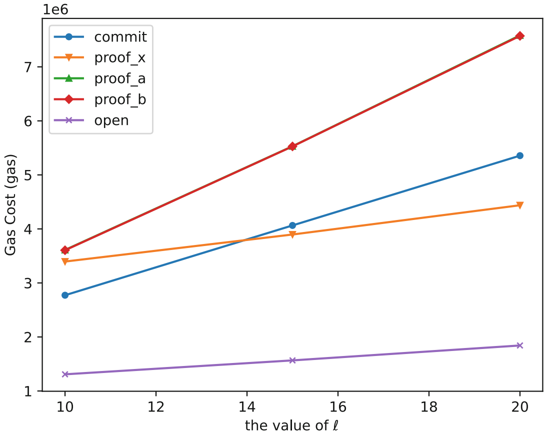


Fig. 7. The gas cost of our proposal.

"proof_a", "proof_b", and "open" denote the commitment commiting, zero-knowledge proofs of x, $a = \sum_0^{\ell} a_i \cdot 2^i$, and $b = \sum_0^{\ell} b_i \cdot 2^i$, and the commitment opening, respectively. From Fig. 6, we can see that the computational cost is also proportioanl to ℓ as the off-chain cost, while the cost is still acceptable. Note that the costs for "proof_a" and "proof_b" are the same since they require the same execution steps.

6 Related Work

As described in Sect. 4, our proposal is closed to blockchain-based DNS and blockchain-based auction. Hence, we give related works in two parts respectively.

6.1 Blockchain-Based DNS

To solve the traditional DNS' centralization problem, Namecoin [5], as the first solution, used blockchain's decentralized characteristics to realize the decentralized DNS. However, the resulting system is along with some side effects, such as cybersquatting. The Namecoin team proposed a new solution named Blockstack [8] by using the virtual chain, distributed hash table, cloud storage, and other techniques to tackle the above problems. Later on, ConsortiumDNS proposed another solution with better performance on data storage and domain name resolution. As we know, the domain name auction is one of the typical ways to obtain the domain name [11]. However, none of the above blockchain-based systems provide such functionality. In [9], Ethereum officials proposed a domain name system based on the smart contract to resolve domain names into Ethereum addresses but not the digital IP address. The Namebase team recently proposed a blockchain-based DNS, named Handshake, with a domain name auction protocol [10]. Their auction protocol follows the Vickrey auction, where the submitted bids are not revealed until all the bids are submitted, and the bidder with the highest bid wins the auction while paying with the second-highest bid. However, in their auction protocol, anyone can deduce the possible highest bid of the submitted bids, which may bring advantages over the latter bidders. In other words, the auction protocol cannot hold the bidding-fairness property. To the best of our knowledge, it is still challenging to design a fair, secure, and privacy-preserving blockchain-based sealed-bid domain name auction protocol.

6.2 Blockchain-Based Auction

Using the secure multi-party computing (MPC) technique, Kasba et al. [25] proposed the first blockchain-based auction protocol. However, the underlying secure MPC scheme is of a high level of interactivities, which leads to an inefficient solution, especially in the blockchain scenario. With the help of the trusted execution environment (TEE), Yuan et al. [26] proposed an efficient blockchain-based auction protocol without sacrificing the smart contract's confidentiality. Nevertheless, the TEE technique usually demands users to update their hardware, which not everyone can afford. What's worse, the security of implementation of TEE is still controversial [27,28]. Blass and Kerschbaum [29] proposed a new blockchain-based auction protocol still based on the secure MPC technique with a good performance. Nonetheless, the proposed protocol needs a pre-fixed bidding order and a semi-trusted auctioneer that is usually unnecessary in the blockchain scenario. Galal and Youssef [18] proposed an auction protocol over the Ethereum blockchain. However, it is not as efficient as expected due to the use of interactive zero-knowledge proof schemes. Furthermore, there is no mechanism to check whether the bidder has enough money to bid in the proposed auction protocol. Note that, although there is a deposit in their protocol, this deposit cannot be used as proof that the bidder has enough money to fulfill the bid. Recently, Nguyen and Thai [30] proposed a new efficient auction protocol based on multi-party state channels in terms of storage cost. However, the resulting protocol cannot support the sealed-bid auction. Qusa et al. [31] also proposed a new blockchain-based auction protocol, while the bidder colluding with some bidders can obtain some advantages over the rest of the bidders. It is fair to say that a fair, secure, and privacy-preserving blockchain-based auction protocol is still desired.

7 Conclusion

The blockchain-based DNS is considered one of the most promising decentralized DNSs. However, the current blockchain-based DNSs fail to provide a suitable domain name auction protocol essential for the domain name transfer. To fill this gap, we in this paper have proposed a new blockchain-based sealed-bid domain name auction protocol. Based on the security assumptions on the underlying blockchain system, the Pedersen commitment, and zero-knowledge proof, we have also shown that the proposed auction protocol can proceed in a fair, secure, and privacy-preserving way.

References

1. Mockapetris, P.V.: RFC1034: Domain Names-Concepts and Facilities (1987)
2. Mockapetris, P.V.: RFC1035: Domain Names-Implementation and Specification (1987)
3. Khormali, A., Park, J., Alasmary, H., Anwar, A., Saad, M., Mohaisen, D.A.: Domain name system security and privacy: a contemporary survey. Comput. Netw. **185**, 107699 (2021)
4. Verma, S., et al.: Stopping amplified DNS DDoS attacks through distributed query rate sharing. In: 11th International Conference on Availability, Reliability and Security, ARES 2016, IEEE Computer Society, pp. 69–78 (2016)
5. Loibl, A.: Namecoin (2014). https://www.namecoin.org/
6. Wang, X., Li, K., Li, H., Li, Y., Liang, Z.: ConsortiumDNS: a distributed domain name service based on consortium chain. In: 19th IEEE International Conference on High Performance Computing and Communications; 15th IEEE International Conference on Smart City; 3rd IEEE International Conference on Data Science and Systems, pp. 617–620. IEEE Computer Society (2017)
7. Emercoin NVS - Emercoin Community Documentation. https://emercoin.com/en/documentation/blockchain-services/emernvs
8. Ali, M., Nelson, J.C., Shea, R., Freedman, M.J., Blockstack: a global naming and storage system secured by blockchains. In: USENIX Annual Technical Conference, USENIX ATC 2016. USENIX Association 2016, pp. 181–194 (2016)
9. Johnson, N., Griffith, V.: Ethereum Name Service. https://ensuser.com/docs/readme.html
10. Roquerre, T.: Handshake project paper. https://handshake.org/files/handshake.txt
11. W. contributors: Domain Name Auction (2020). https://en.wikipedia.org/w/index.php?title=Domain_name_auction&oldid=975293048
12. Alvarez, R., Nojoumian, M.: Comprehensive survey on privacy-preserving protocols for sealed-bid auctions. Comput. Secur. **88** (2020)

13. Nakamoto, S., et al.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
14. Poon, J., Dryja, T.: The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf
15. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30. ACM (2016)
16. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68. ACM (2017)
17. Buterin, V., et al.: A Next-Generation Smart Contract and Decentralized Application Platform, white paper 3 (37) (2014)
18. Galal, H.S., Youssef, A.M.: Verifiable sealed-bid auction on the ethereum blockchain. IACR Cryptol. ePrint Arch. **2018**, 704 (2018)
19. Chandrashekar, T.S., Narahari, Y., Rosa, C.H., Kulkarni, D.M., Tew, J.D., Dayama, P.: Auction-based mechanisms for electronic procurement. IEEE Trans. Autom. Sci. Eng. **4**(3), 297–321 (2007)
20. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
21. Miers, I., Garman, C., Green, M., Rubin, A.D., Zerocoin: anonymous distributed e-cash from bitcoin. In: IEEE Symposium on Security and Privacy. SP 2013, pp. 397–411. IEEE Computer Society (2013)
22. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: IEEE Symposium on Security and Privacy. SP 2014, pp. 459–474. IEEE Computer Society (2014)
23. Noether, S.: Ring Signature Confidential Transactions for Monero. IACR Cryptology ePrint Archive 2015, 1098 (2015)
24. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
25. Kosba, A.E., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: IEEE Symposium on Security and Privacy, SP 2016, IEEE Computer Society, pp. 839–858 (2016)
26. Yuan, R., Xia, Y.-B., Chen, H.-B., Zang, B.-Y., Xie, J.: ShadowEth: private smart contract on public blockchain. J. Comput. Sci. Technol. **33**(3), 542–556 (2018). https://doi.org/10.1007/s11390-018-1839-y
27. Shaun Davenport, R.F.: SGX: The Good, The Bad and The Downright Ugly (2014). https://www.virusbulletin.com/virusbulletin/2014/01/sgx-good-bad-and-downright-ugly
28. Luis Merino, J.A.: SGX Secure Enclaves in Practice Security and Crypto Review (2016). https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf
29. Blass, E.-O., Kerschbaum, F.: Strain: a secure auction for blockchains. In: Lopez, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018. LNCS, vol. 11098, pp. 87–110. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99073-6_5
30. Nguyen, T., Thai, M.T.: A blockchain-based iterative double auction protocol using multiparty state channels. ACM Trans. Internet Technol. **21**, 1–22 (2021)
31. Qusa, H., Tarazi, J., Akre, V.: Secure e-auction system using blockchain: UAE case study. In: Advances in Science and Engineering Technology International Conferences (ASET). IEEE 2020, pp. 1–5 (2020)

Authentication

A Security Enhanced Key Management Service for ARM Pointer Authentication

Liqiang Zhang, Qingsong Chen, and Fei Yan(✉)

Key Laboratory of Aerospace Information Security and Trusted Computing,
Ministry of Education, School of Cyber Science and Engineering,
Wuhan University, Wuhan 430000, Hubei, China
yanfei@whu.edu.cn

Abstract. The memory-unsafe programming languages caused a pandemic of memory corruption bugs in ARM-based devices. To mitigate such threats, Control-Flow Integrity (CFI) is one of the most effective and popular solution, and integrated with the modish hardware makes it even more valuable, for instance, the ARM Pointer Authentication (PA), which can generate a message authentication code for a pointer and verify it to ensure the pointer is intact. However, according to some research, the QARMA algorithm, as a critical part of PA, is vulnerable to certain attacks, making it possible to recover the key.

In this paper, we present a key management service for PA. It utilizes the exception model of TrustZone to isolate the key generation process of PA securely, preventing the key from leaking to insecure memory; then takes advantage of a randomization scheme to dynamically derive separate keys for both kernel-space and user-space programs. Based on the scheme, we have implemented a prototype among the ARM Trusted Firmware, and also an enhanced backward-edge CFI solution. The evaluation shows that it introduces a reasonable and acceptable performance overhead, while provides better security guarantee.

Keywords: Pointer authentication · Key management · Control-flow integrity

1 Introduction

According to Google's report, memory corruption bugs generally accounted for more than 65% of High & Critical security bugs in Chrome and Android. These bugs exist in the "memory-unsafe" programming languages (e.g., C and C++), which allow developers to take fine-grained control of the memory through pointers. And security mechanisms such as Stack Canary, DEP and ASLR have indeed defeated the primitive attack patterns like code-injection, but in the meantime, code-reuse attacks gradually become the main pattern of hacking the software and system security.

Return-oriented programming (ROP) [1] is a well-known code-reuse technique which allows the attacker to reuse the original benign code snippets (called gadgets) in the memory and manipulate them to launch powerful attacks [2]. In order to defend against the ROP attack, multiple approaches have been proposed in the past decades,

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 41–55, 2021.
https://doi.org/10.1007/978-3-030-80851-8_4

such as control-flow integrity (CFI) [3]. However, software implementations of CFI are limited by high performance overhead or significant changes to system software architecture, and hardware-assisted ones are unlikely to ever be deployed because of requiring the underlying processor architecture changes.

In recent years, major processor vendors have directly embedded security primitives into their modern processors. ARM introduced Pointer Authentication (PA) in ARMv8.3-A to detect and reject crafted pointers through a set of instructions. PA calculates a cryptographic message authentication code for 64-bit pointers, and stores it at the high bits of the pointer, referred to Pointer Authentication Code (PAC). The pointer with PAC must be verified by the hardware before it is used as a return address or function pointer. Actually, Qualcomm has proposed a backward-edge CFI scheme based on PA to protect the return address [4], then Apple suggested to sign the virtual function pointers to protect forward-edge CFI [5].

However, the default QARMA block cipher algorithm used by PA is subjected to certain attacks: Li et al. [6] applied the meet-in-the-middle attack on reduced-round QARMA-64/128; Yang et al. [7] have utilized impossible differential attack to analyze QARMA with less time and space complexity; and related-tweak statistical saturation attack are also proved to be effective on QARMA [8]. Therefore, the ability of sophisticated attackers to perform cryptanalysis on the PA mechanism cannot be ignored. The existing PA-based schemes cannot be trustworthy enough to safeguard the CFI, for the following reasons: 1) the assigned PA keys for each process are constant, and 2) the attacker may attempt cryptanalysis attacks based on the collected PAC values over a long period of time.

In this paper, we design and implement Pointer Authentication Key Management Service (PAKMS), providing security enhanced PA key management service. We also implement a backward-edge CFI solution based on it to evaluate the security and performance overhead of PAKMS. Our main contributions are:

- Design: We propose a novel scheme, leveraging the privilege and exception levels of ARMv8-A to provide pointer authentication key management service, which could dynamically generate distinct keys for both the kernel and user-space applications (Sect. 4).
- Implementation: We have implemented a prototype of PAKMS based on the ARM Trusted Firmware (ATF). We also implement a LLVM-based backward-edge CFI solution for user-space applications, which proves the practicality of PAKMS (Sect. 5).
- Evaluation: We conducted systematic analysis of PAKMS on the ARM Fixed Virtual Platform (FVP) with nbench-byte benchmark, demonstrating that it has a reasonable performance overhead (Sect. 6).

2 Background

2.1 Privilege and Exception Levels

ARMv8-A architecture enables exception levels with a different privilege of accessing system resources. The exception levels are referred as EL<x>, with number x ranging from 0 to 3. EL0 is the least privileged one, typically used for applications; and EL1 is for the operating system. While the EL0 and EL1 are available in both the Non-Secure and Secure worlds introduced by TrustZone [9], the EL3 is the most privileged one that only resides in Secure World, reserved by low-level firmware or security code. The current exception level can only be switched when the processor takes or returns from an exception. For instance, SMC instruction can causes an exception to EL3, and SVC instruction to EL1.

There are two types of privilege relevant to the exception levels. One is memory privilege controlled by Memory Management Unit (MMU) and isolated by the TrustZone Address Space Controller (TZASC) in hardware. The other is privilege of accessing processor resources, i.e. System registers. Generally, the name of the System register indicates the lowest exception level required by a successful access. For example, the register *TTBR0_EL1* holds the base address of the translation table of applications at EL0, which requires privilege of EL1 at least.

2.2 Pointer Authentication

In 2016, ARM introduced an ISA extension in ARMv8.3-A, named Pointer Authentication (PA) [4]. The main purpose of PA is to provide a fast and security mechanism against exploitation of memory corruption bugs, by the pointer authentication codes (PACs). PAC is a short Message Authentication Code (MAC) which can be generated through a single instruction and embedded into the high bits of 64-bit pointers.

PA mainly introduces three types of instructions: (1) *PAC** instructions are used to generate and insert PAC into the pointer; (2) *AUT** instructions are designed to authenticate PAC, in other words, if PAC matches, the original pointer will be returned, or else it returns an invalid pointer that would trigger a fault upon use; (3) *XPAC** instructions also yield the original pointers, but without the authentication routine. As illustrated in Fig. 1, a *PAC** instruction calculates PAC over a 64-bit virtual address, a 64-bit modifier as a customized context and a 128-bit secret PA key stored in System registers, using Hash function (QARMA [10] by default). Since the PAC is embedded in unused bits of the pointer, its size is restricted both by the virtual address size and the memory tagging extension (MTE) introduced by ARMv8.5-A. For example, when the *VA_SIZE* is 48-bit and MTE is enabled, the PAC size is merely 7 bits.

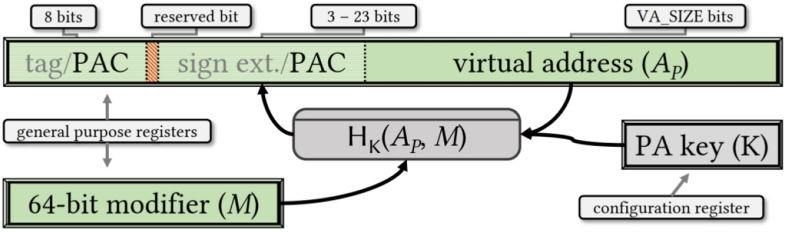


Fig. 1. PAC is computed on the pointer's address, a modifier and a key (from [17]).

Furthermore, PA also introduces a set of System registers to hold the PA keys used by the PAC cryptographic algorithm. There are five different keys in total, *IA* and *IB* keys are used for code pointers, *DA* and *DB* are for data pointers and *GA* is for general purpose. Take the *PACIA Xd, Xn* instruction as an example, it computes the corresponding authenticated pointer (returned by register *Xd*) over an original pointer kept in *Xd*, a 64-bit modifier kept in register *Xn* and the 128-bit IA key composed of the System register *APIAKeyHi_EL1* and *APIAKeyLo_EL1*.

3 Threat Model

3.1 Attacks

PA prevents the attacker from forging and utilizing pointers to hijack the control flows, relying on the signing and authenticating instructions. Hence, to obtain a legal PAC of the target address becomes attackers' primary goal. And PAC is calculated from a context-associated modifier and a confidential PA key, using QARMA algorithm by default, which consequently exists three possible attack channels as follows.

First of all, the modifier value is not confidential in principle, and can be either static (e.g., a hard-coded value) or dynamic (e.g., the stack pointer). On the one hand, reverse engineering could reveal the static value, and on the other hand, take the stack pointer as an example, it is not required to be unique for a specific function, therefore, attackers could reuse the signed pointer in one function when another vulnerable function executes with the same stack pointer.

Secondly, the management of PA keys could under attack. PA keys used by EL0 applications are generated and switched by the kernel. And the kernel tracks them in the *thread_struct* for each application, therefore, the keys remain constant throughout its lifetime and are shared by all threads in a single address space. A constant PA key is fragile to cryptanalysis attack based on known PAC values, and shared keys expand the search space for attackers to find a usable signing or authenticating gadget.

Lastly, PA also depends on the security of the underlying cryptographic primitives, i.e., the QARMA algorithm. For QARMA, researchers have proposed several cryptanalysis methods, such as meet-in-the-middle and impossible differential attacks. Thus, a sophisticated attacker could observe and collect adequate pairs of the original pointer and the signed one to attempt cryptanalysis attacks. If the attacker could recover the PA keys, all the integrity protections within the scope could be useless.

3.2 Assumptions

In line with the threat model of pointer authentication, we consider a powerful attacker who has the capability to read and write almost the entire address space, constrained only by DEP memory policies, for instance, the attacker cannot write to the read-only memory and execute codes in writable memory. In addition, the attacker cannot read or write the System registers directly, especially the registers that hold PA keys, although he/she may obtain the capability to read and write arbitrary kernel memory via kernel vulnerabilities [11]. The attacker's goal is to subvert the control flow of the process. Thus, he/she would attempt to corrupt any return addresses or function pointers in the data section, stack and heap region. Furthermore, with the understanding of cryptanalysis attacks, the attacker could also try to guess the PAC value of a fake pointer, even to infer the PA keys used in some context.

In addition to the ability of the attacker, we made some assumptions about the system as well. We assume that the firmware or secure monitor is trusted, and the TZASC is configured correctly to separate the physical memory space. The hypervisor (if existed) and the kernel boot-up process is also trusted, since the bootloader in EL3 can verify their cryptographic hash of the binary. And then, we further assume that the random number generation used in this work has the expected random entropy. The hardware circuitry relevant to the PA mechanism should work as defined by the ARM specification, leaking no private and confidential information.

4 Design

4.1 Architecture

In this section, we present a novel scheme named PAKMS to provide PA keys for both kernel and user-space applications. Before illustrating it in details, we need to address security challenges arising from the contradiction between the existing management pattern and the threat model mentioned in the previous section:

Confidentiality. The kernel stores PA keys in *thread_struct* to support each process signing and authenticating pointers correctly, which is not feasible for an attacker with the capability to read and write to arbitrary kernel memory. In order to maintain the confidentiality of PA keys, no keys should reside in the kernel memory.

Variability. This is caused by the immutability of the PA keys, which ensures that all signed pointers remain verifiable in their lifetime, which is vulnerable to the attacker with the knowledge of cryptanalysis.

Diversity. The diversity means the PAC of the same pointer can be distinct in different context. Since the process' PA keys remain constant, the diversity mainly relies on the modifier. However, whether the modifier is static or dynamic, there still exists potential risk of pointer reuse attacks, as we discussed in Sect. 3.1.

We propose the PAKMS to provide PA keys for the kernel and user-space applications, which ensures confidentiality, variability and diversity of PA keys (see Fig. 2). In order to satisfy the demand for confidentiality, PAKMS holds a set of *master-keys* in the privileged EL3, which are designed to derive PA keys for the upper exception levels (i.e., the kernel and the application). On the one hand, the *master-keys* are kept in EL3 memory, and securely isolated from other exception levels through the TZASC, at the same time, the derived PA keys are always kept in System registers and never banked to the kernel memory. On the other hand, the PA keys are also derived on a dynamic parameter, namely the Generation (*Gn*), which ensures that the attacker cannot obtain adequate pairs of the original pointer and the signed one to infer the PA keys by cryptanalysis attacks. Besides, *Gn* further offers variability and diversity of PA keys thanks to that it changes based on system time or access frequency. What's more, to enlarge diversity of PA keys, we have adopted distinct parameters for the granularity of process level, thread level and function level. In the followings, we will detail the PAKMS in accordance with the hierarchical structure.

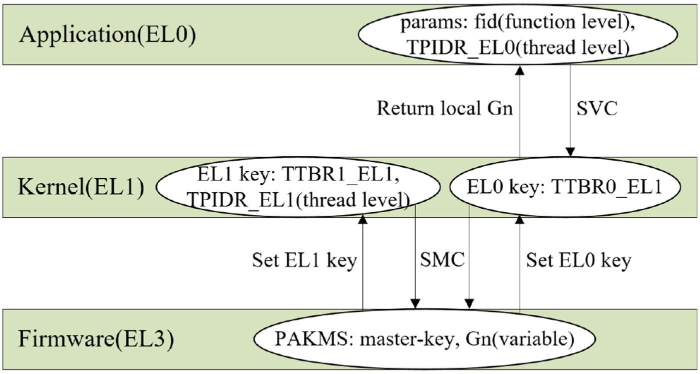

Application(EL0)
params: fid(function level), TPIDR_EL0(thread level)
Return local Gn
SVC
Kernel(EL1)
EL1 key: TTBR1_EL1, TPIDR_EL1(thread level)
EL0 key: TTBR0_EL1
Set EL1 key
SMC
Set EL0 key
Firmware(EL3)
PAKMS: master-key, Gn(variable)

Fig. 2. PAKMS ensures confidentiality, variability and diversity of PA keys.

1) EL3 Isolation: naturally, PAKMS in EL3 can guarantee good confidentiality of PA keys through TZASC configurations. Resided in the system bus and the memory chip, TZASC can support multiple memory regions with different access-control settings (e.g., TZC-400 supports up to 9 memory regions), so that PAKMS can preserve the *mater-keys* and *Gn* in the EL3 memory which is exclusively accessible by the secure world. Besides, as an EL3 runtime service, PAKMS is both versatile and extensible, not only can it provide PA keys for the kernel and the application in Non-secure world, but also can serve the Hypervisor and the Trusted OS in Secure world. What's more, EL3 can provide a more reliable source of randomness for initializing the *master-keys* and *Gn*, and updating the *Gn* with a random strategy.
2) Kernel in the Middle: in our design, the kernel plays two roles on the whole. For one thing, to protect the integrity of kernel-space pointers, the kernel can invoke the PAKMS through a Secure Monitor Call (SMC). According to the SMC Calling Convention [12], an SMC64 call can pass six parameters at most, using registers *X1–X6*, and return results in registers *X0–X3*. Thus, while invoking the PAKMS, we can pass some unique parameters to ensure a rich diversity of PA keys. Optional parameters are the values of register *TTBR1_EL1* and *TPIDR_EL1*. The former holds the base address of the translation table of kernel address space, which can be used to separate keys of the kernel from the user-space applications; and the latter, storing thread identifying information, can be used to produce distinguishable PA keys for kernel threads. And further, function-level parameter can be passed to produce fine-grained PA keys for each function context.

   For another, the kernel plays the role of a bridge between PAKMS and the user-space applications. Since SMC is not available in EL0, the kernel receives a request for PAKMS from applications and passes it to the EL3. Except for the parameters from the application (the function-level parameter), analogously, the register *TTBR0_EL1* and *TPIDR_EL0* can be used to produce PA keys. A local *Gn* will be returned by PAKMS, and can be stored on the stack, in order to correctly authenticate signed pointers later, because *Gn* in EL3 is variable throughout the system.
3) Application Strategies: for SMC instruction is undefined in EL0, the user-space applications have to invoke a Supervisor Call (SVC) to access PAKMS via the

kernel. Then, the kernel would invoke the PAKMS with some context-sensitive parameters, that is the register *TTBR0_EL1*, *TPIDR_EL0* and the function-level parameter. In addition, the PAKMS should return a signed/authenticated pointer directly rather leave the *PAC*/AUT** operations in the application context. Because the user-space tasks can be preempted by other high-priority tasks, which may invoke the PAKMS again, leading to the former task cannot sign/authenticate the pointer rightfully. Hence, the signed/authenticated pointers are returned and *BR** instructions are recommended to make up a function epilogue. Lastly, there should be no way for the application developers to create a signing or authentication gadget manually using the *PAC*/AUT** instructions, for that may form attackable code sequences.

4.2 Algorithm

We illustrate the signing and authenticating routines of PAKMS in Algorithm 1.

Algorithm 1 PAKMS signing and authenticating routines

```
Input: SID: service identifier; IAP: input address pointer; $IGn_l$: input local Generation
of PA keys; FLP: function-level parameter.
Output: OAP: output address pointer; $OGn_l$: output local Generation of PA keys.
initialize master-keys (IA, IB, DA, DA, GA) and global Generation of PA keys
  ($Gn_g$) randomly at boot-up time
$Gn_g \leftarrow$ update with random value by EL3 physical timer
switch SID:
  $Km \leftarrow$ select a master-key from (IA, IB, DA, DA, GA)
  $PLP \leftarrow$ select a process-level parameter, either TTBR0_EL1 or TTBR1_EL1
  $TLP \leftarrow$ select a thread-level parameter, either TPIDR_EL0 or TPIDR_EL1
  case signing:
    set the related PA keys, take APIAKey_EL1 for example:
      $APIAKeyLo\_EL1 = QARMA(Km, PLP \oplus FLP, Gn_g)$
      $APIAKeyHi\_EL1 = QARMA(Km, TLP \oplus FLP, Gn_g)$
    set $OGn_l = Gn_g$
    if preemptive then:
      signing IAP, for example: OAP = PACIA(APIAKey_EL1, IAP, FLP)
      return OAP, $OGn_l$
    else non-preemptive:
      return $OGn_l$
    end if
  case authenticating:
    set the related PA keys, take APIAKey_EL1 for example:
      $APIAKeyLo\_EL1 = QARMA(Km, PLP \oplus FLP, IGn_l)$
      $APIAKeyHi\_EL1 = QARMA(Km, TLP \oplus FLP, IGn_l)$
    if preemptive then:
      authenticating IAP, for example: OAP = AUTIA(APIAKey_EL1, IAP, FLP)
      return OAP
    else non-preemptive:
      return
    end if
end switch
```

After the firmware accomplishes the necessary architectural and platform initialization, including the setup of exception vectors and control registers (specifically, to enable PA instruction in EL3, the register *SCTLR_EL3.{EnIA, EnIB, EnDA, EnDB}* field need to be set up), the PAKMS is initialized with five random *master-keys* and one Gn_g on behave of the Generation of PA keys (Line 1). Then setup the EL3 physical timer to update Gn_g with random value periodically. According to the SMCCC, the framework schedules every SMC call to the corresponding handler through the function identifier, which determines the service and function to be invoked. As shown in Algorithm 1, the service identifier (*SID*) is actually the lower 16 bits of the function identifier, indicating the function number of PAKMS. When PAKMS is invoked, firstly, some context-sensitive parameters are selected according to the value of *SID*, e.g., the *master-key* to use (Line 3 to 6). As there are five *master-keys* in total, it is recommended to use different master-key for code pointers and data pointers. In order to distinguish between the kernel and user-space processes, the process-level parameter can take the register value of *TTBR1_EL1* or *TTBR0_EL1*. And the thread-level parameter, which makes use of the register *TPIDR_EL0* or *TPIDR_EL1*, is related to the thread local storage. After that, the signing and authenticating key generation processes are introduced separately.

In the signing procedure, apart from the parameters mentioned above, the function-level parameter passed from the kernel is also utilized to generate a unique key, using some specific algorithm. In this paper, we choose QARMA for the reason that the *PAC** instructions have provided an efficient implementation (Line 8 to 10). In practice, we take advantage of the *PACGA* instruction, using *Km* as the PA key, $PLP \oplus FLP$ or $TLP \oplus FLP$ as the pointer, and Gn_g as the modifier. Analogously, in the authenticating procedure, the same algorithm are used. However, Gn_g is changeable and may be different from the signing procedure, hence it needs to be backed up through OGn_l (Line 11). To generate the correct key, the authenticating procedure uses IGn_l instead (Line 19 to 21). And for the preemptive situation, we need to return the signed/authenticated pointer calculated by *PAC*/AUT** instructions directly.

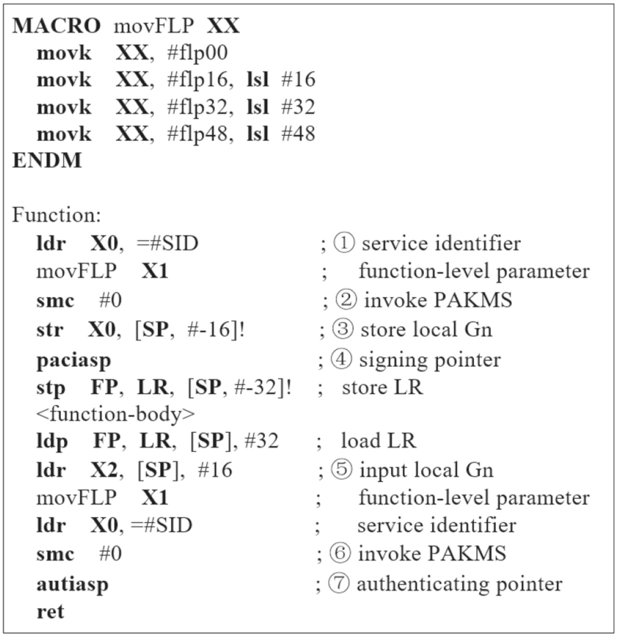
5 Implementation

5.1 EL3 Runtime Service

ARM Trusted Firmware (ATF) provides a reference implementation of ARM interface standards, including SMC Calling Convention (SMCCC), System Control and Management Interface (SCMI), the initialization of Generic Interrupt Controller (GIC) and TrustZone Controller (TZC). By providing the EL3 runtime service framework, ATF makes it convenient to integrate services provided by different developers into the firmware. And it also supports multiple toolchains, such as GCC and LLVM, making it easier to customize.

Our prototype is based on the ATF runtime service framework, making use of the signing and authenticating schemes illustrated in Algorithm 1. Firstly, to enable PA instruction in EL3, the register *SCTLR_EL3.{EnIA, EnIB, EnDA, EnDB}* field need to be set up, then PAKMS need to initialize five *master-keys* and one Gn_g randomly. In order to update Gn_g periodically, we also need to configure the EL3 physical timer and the GIC. Specifically, we need to setup the *CNTPS_CTL_EL1* register fields: *ENABLE* to 1 and *IMASK* to 0. The *IMASK* field controls the interrupt generation. When the timer fires (the value of *CNTPS_CVAL_EL1* <= *CNTPCT_EL0*), an interrupt is asserted to the interrupt controller. The interrupt *ID* used for EL3 physical timer is 29, as recommended by the Server Base System Architecture (SBSA). Then we can update Gn_g and *CNTPS_CVAL_EL1* register in the interrupt handler.

5.2 EL1 Kernel Module



```
MACRO  movFLP  XX
   movk   XX, #flp00
   movk   XX, #flp16,  lsl  #16
   movk   XX, #flp32,  lsl  #32
   movk   XX, #flp48,  lsl  #48
ENDM

Function:
   ldr   X0,  =#SID                  ; ① service identifier
   movFLP   X1                       ;    function-level parameter
   smc   #0                          ; ② invoke PAKMS
   str   X0,  [SP,  #-16]!           ; ③ store local Gn
   paciasp                           ; ④ signing pointer
   stp   FP,  LR,  [SP, #-32]!       ;   store LR
   <function-body>
   ldp   FP,  LR,  [SP], #32         ;   load LR
   ldr   X2,  [SP],  #16             ; ⑤ input local Gn
   movFLP   X1                       ;    function-level parameter
   ldr   X0, =#SID                   ;    service identifier
   smc   #0                          ; ⑥ invoke PAKMS
   autiasp                           ; ⑦ authenticating pointer
   ret
```

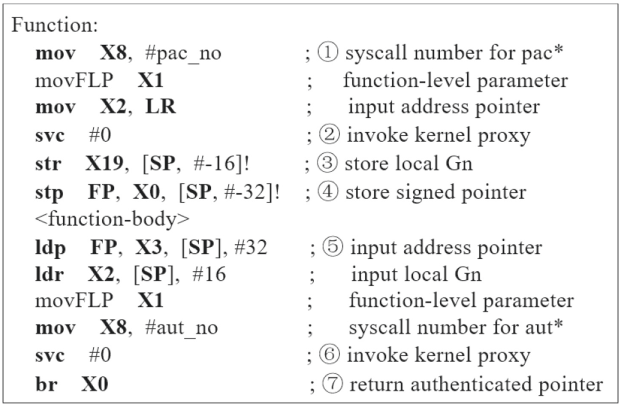
Fig. 3. A non-preemptive kernel can use PAC*/AUT* instructions in its own code sequences (④, ⑦), after switching PA keys by PAKMS (②, ⑥).

Typically, a non-preemptive kernel can use *PAC*/AUT** instructions directly to protect the integrity of the kernel pointers, as illustrated in Fig. 3. Firstly, at the function entry, kernel sets up the necessary parameters and invokes PAKMS to generate and switch the associated PA keys (Fig. 3, ①②). Next, in order to authenticate the signed pointer without error later, a local *Gn* needs to be preserved properly (Fig. 3, ③). Then, the kernel can use any *PAC** instruction to do the signing procedure. For example, *PACIASP* is utilized to sign the return address using stack pointer as a modifier (Fig. 3, ④). Before the function returns, a similar authenticating procedure needs to be executed to get a functional return address. The differences are that the local *Gn* needs to be passed to the PAKMS and the corresponding *AUT** instruction is used to calculate a correct return address (Fig. 3, ⑤⑥⑦). And finally, the function returns to the rightful place using *RET* instruction.

Another important function of the kernel is to redirect the PAKMS requests from the applications in EL0. The kernel only needs to set the proper service identifier according to specific system call, without considering other parameters which are set by the applications in EL0. Furthermore, before returning to the EL0, the kernel can execute security checks to detect and trap on authentication failures, rather return an illegal pointer to the applications. Then the kernel uses *RET* instruction to return to the application, with return values kept in specific registers.

5.3 EL0 Calling Convention

Based on LLVM 10.0, we add some new passes to the optimizer for generating function-level parameters and to the AArch64 backend for emitting suitable low-level instructions. We use optimizer passes to generate function-level parameters based on the pointer's LLVM *ElementType*, and the backend passes to modify the function prologues and epilogues for making up the parameters and invoking system call for PAKMS. Here we only show the sample code listing of the return address signing and authenticating, but assuredly PAKMS can also provide protections for both code pointer and data pointer.



```
Function:
   mov   X8, #pac_no              ; ① syscall number for pac*
   movFLP   X1                    ;    function-level parameter
   mov   X2, LR                   ;    input address pointer
   svc   #0                       ; ② invoke kernel proxy
   str   X19, [SP, #-16]!         ; ③ store local Gn
   stp   FP, X0, [SP, #-32]!      ; ④ store signed pointer
   <function-body>
   ldp   FP, X3, [SP], #32        ; ⑤ input address pointer
   ldr   X2, [SP], #16            ;    input local Gn
   movFLP   X1                    ;    function-level parameter
   mov   X8, #aut_no              ;    syscall number for aut*
   svc   #0                       ; ⑥ invoke kernel proxy
   br   X0                        ; ⑦ return authenticated pointer
```

Fig. 4. The application in EL0 invokes PAKMS indirectly through SVC instruction.

We illustrate part of the function code listing to do the return address signing and authenticating, including the necessary prologues and epilogues (see Fig. 4). First, we set up the special system call number for signing procedure and other essential parameters, then we use the *SVC* instruction to request the kernel proxy as mentioned above (Fig. 4, ①②). After the kernel invokes the PAKMS in EL3, it returns the local *Gn* in register *X19* and the signed pointer in *X0*, which will be preserved on the stack (Fig. 4, ③④). Similar to the signing procedure, the authenticating procedure invokes system call to request the kernel proxy, and the kernel also execute the check routine to detect authenticating failures as early as possible before returns to the application (Fig. 4, ⑤⑥). Finally, the application utilizes the *BR* instruction to return to the location pointed by the register *X0*, which holds the authenticated return address, since the application can be preempted when it returns from the kernel.

6 Evaluation

In this section, we will analyze the security and performance of our proof-of-concept implementation, which is developed and integrated with ARM Trusted Firmware. At the time of writing this paper, an evaluation on an actual platform was not possible, because the PA-capable SoCs (e.g., Apple A12 and Kirin 980) do not dispose of the off-the-shelf development boards. Therefore, we evaluate PAKMS upon the ARM Fixed Virtual Platform (FVP), an emulator supporting the ARMv8.3-A features.

6.1 Security Analysis

The security problems of existing PA-based schemes are stemmed in the immutability of each process' PA keys managed by the kernel. PAKMS, however, can provide different keys to the same process or even the same function context in different time partitions, thanks to the sink of the key management and the variable *Gn* in EL3, which prevents the following security issues to a certain extent.

A signing gadget is a code fragment that can be exploited to sign an arbitrary pointer, which could be utilized to substitute all the pointers signed by the same PA keys. Such gadgets could probably compromise the effectiveness of PA. Another similar attack discussed above, reuse attack, is also a kind of substitute attack reusing the existing authenticatable pointer to construct a loop to perform malicious calculations. Instrumented by LLVM passes, however, all the signing and authenticating code pieces are automatically generated, without certain treacherous patterns of code. Even if the attacker could take advantage of the signing procedure to forge a pointer, he/she still cannot use the pointer in another context, since PAKMS provides rich diversity of PA keys which could be different between function-level contexts. In addition, the keys used in the same context will also change over time, because a dynamic parameter (*Gn*) is used by PAKMS to generate PA keys.

Authentication gadgets, on the other hand, can be exploited to authenticate any forged pointers. And the attacker could verify the correctness of the forged PAC without triggering an exception, which means online guessing attacks are possible. This is because PA mechanism only replaces the PAC with an invalid bit if the authenticating procedure failed, and returns an illegal pointer which causes a translation fault only when it's used. PAKMS solves this problem through a check routine added to the kernel proxy, which can do the early detection on authentication failures, so that all the failures can be recorded to restrict an attacker on the number of guessing.

Table 1. CPU cycles needed by pac* and pakms* instructions to calculate PACs.

Instructions	Execution times & CPU cycles			
	2^8	2^{10}	2^{14}	2^{16}
pac*	3342	13326	229391	917519
pakms*	18702	74766	1212431	4849679

For a sophisticated attacker, cryptanalysis attacks could also be launched to recover the PA keys. Either by the controlled signing procedure, or just by waiting and observing the signing results patiently, the attacker could collect numerous pairs of the original pointer and the signed one. When sufficient data is obtained (only need a set of 65536 chosen plaintexts at least [8]), the attacker could perform offline cryptanalysis on the QARMA to recover the PA keys. We evaluated how many CPU cycles needed by PAC* instruction and PAKMS to calculate PACs, the result is shown in Table 1. PAKMS attempts to mitigate such attacks with a dynamic parameter *Gn* in EL3, and it varies according to the EL3 physical timer, so does the keys based on it. Hence, according to Table 1, we can set the timer less than 4849679 CPU cycles to update *Gn.* So that the attacker cannot collect enough data to recover a specific PA key before other keys are generated for this context. Therefore, PAKMS can prevent such advanced attacks effectively.

6.2 Performance Analysis

We choose nbench-byte 2.2.3 benchmark [13] to evaluate the performance over-head of PAKMS, for the reason that nbench-byte is designed to measure CPU and memory subsystem performance with real-world algorithms, and it is easy to be migrated to new processors or operating systems quickly, allowing us to measure PAKMS instrumentation with scalable and dynamic workload adjustment on FVP. Simultaneously, we build two root file systems based on Yocto Project, us-ing the Linux kernel v5.4.23, and compiled by the customized Clang/LLVM com-piler, one enabled by PAKMS extension and the other not, and the binaries of nbench do the same.

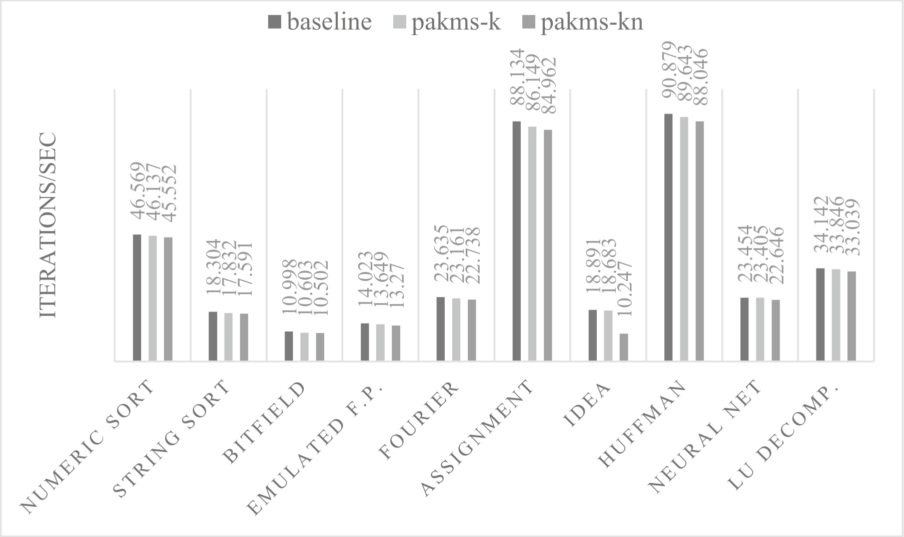


Fig. 5. Performance evaluations of PAKMS using nbench-byte benchmark.

Since PAKMS-enabled nbench cannot run on the original kernel, there are three sets of evaluations in total, as illustrated in Fig. 5, a) the baseline is evaluated with the original kernel and nbench, b) and the pakms-k uses the pakms kernel and the original nbench, c) and the pakms-kn equips both the kernel and nbench with PAKMS extension. In order to analyze and compare experiment results of each algorithm conveniently, we just modify the configuration of each algorithm, but make each evaluation set with the same configurations. Compared with the baseline, both pakms-k and pakms-kn introduce some reasonable overhead, i.e., the iterations per second reduce a little for each algorithm. Except for the IDEA algorithm, which causes 45.7% of performance overhead, most of the other algorithms have an overhead of less than 3%. This is because when the IDEA performs the encryption iterations, all the operations (like ADD, MUL and XOR) need to call its internal functions separately, which leads to frequent calls to PAKMS for signing and authenticating pointers. By optimizing the implementation of IDEA, the PAKMS call frequency can be reduced. In general, for pakms-k, the geometric mean of the performance overhead of all algorithm tests is 1.39%, while pakms-kn is 4.62%.

According to the evaluations, we believe that the performance overhead of PAKMS is reasonable and acceptable. Even in the worst case, IDEA, can reduce the overhead by optimizing the algorithm with less function calls inside, not to mention other more common algorithms. For the ARM Fast Models documentation states that "all instructions execute in one processor master clock cycle" [14], in the future work, we are planning to evaluate PAKMS on the PA-capable hardware and support more signing and authenticating patterns.

7 Related Works

Most of the existing works on ARM Pointer Authentication can be divided into two categories: 1) the usages of PA for pointer protection; 2) the schemes to enhance the security of PA mechanism itself.

The Qualcomm whitepaper [4] came out first with simple backward-edge CFI based on PA, using the SP value as the only modifier, and Apple also proposed forward-edge protection by signing the vtable pointers [5], using zero as the modifier, so that they are both susceptible to reuse attacks, which do no harm to our design due to the rich diversity of PAC. PARTS [15] was the first academic presentation on this subject, which not only implemented a more fine-grained CFI solution for both the return address and code pointer, but also provided the data pointer protection. Our function-level parameter generation required linkage time optimization, as well as its unique function identifiers, and our solution can also offer the comparable security as PARTS. In addition, we further enhanced the security of PA mechanism itself by dynamically changing the PA keys for the kernel and the applications. Other researches also applied the PA to realize a more secure stack canaries [16], and to chain the call stack [17] for fully precise verification of return addresses, however, either of them considered the kernel as a target, which may not readily acceptable for the kernel protection. PTAuth [18] proposed a PA-based heap memory protection system, which can detect and prevent heap-based temporal memory corruptions. However, PTAuth currently does not support multi-threading, and due to shared PA keys, there is still the risk of being bypassed.

PACKER [19] proposed a design for protecting the CFI of the kernel pointers, providing both backward-edge and forward-edge solutions. However, it was also confronted with the problem of sharing the same PA keys in a thread, and it replaced all *BLR* instructions with *BLRAA* instruction, lack of certain compatibility. Based on XOM, Denis-Courmont et al. [20] presented a secure architecture for kernel PA management, which did not depend on traps to higher exception levels. Although the attacker can't access the key stored in the XOM, but in practice, with adequate information of the key, it is possible to recover the key. Ferri et al. [21] took advantage of the hypervisor mode (EL2) and dedicated traps to manage PA keys of applications, certainly the kernel could also be managed this way. However, the traps that the scheme relied on can also be utilized for lazy initialization of PA for hypervisor itself, and were not intended for the key management. As a comparison, PAKMS makes use of the EL3 runtime service framework, which is hardly conflicted with other system services, and besides, it also has the ability to generate PA keys for hypervisor.

8 Conclusion

In this paper, we present an enhanced scheme for PA key management, named PAKMS, which takes advantage of the privilege and exception levels in ARMv8-A to protect the key generation process. PAKMS can dynamically generate PA keys for both the kernel and applications during their lifetime, thanks to the special variable in EL3. According to our evaluation, not only can PAKMS thwart conventional attacks against PA (like substitution or reuse attack), but also elaborate and advanced cryptanalysis attacks. And the prototype of PAKMS based on the ARM Trusted Firmware and Clang/LLVM instrumentation, introducing an acceptable performance overhead.

In the future, we are planning to evaluate the performance of PAKMS comprehensively on the PA-capable hardware and support more signing and authenticating patterns, and also we need to make PAKMS to setup a sufficiently secure and efficient threshold for each system automatically. In addition, the integration with hypervisor and the migration of PAKMS-based process between them are also under consideration.

Ackonwledgements. This research is supported by the Prospective Applied Research Projects of Suzhou City (Grant No. SYG201845), the National Natural Science Foundation of China (Grant No. 61272452) and the National Program on Key Basic Research Project (973 Program) (Grant No. 2014CB340601).

References

1. Shacham, H.: The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 552–561 (2007)
2. Roemer, R., Buchanan, E., Shacham, H.: Return-oriented programming: systems, languages, and applications. ACM Trans. Inf. Syst. Secur. (TISSEC) **15**(1), 1–34 (2012)
3. Abadi, M., Budiu, M., Erlingsson, U.: Control-flow integrity principles, implementations, and applications. ACM Trans. Inf. Syst. Secur. (TISSEC) **13**(1), 1–40 (2009)

4. Qualcomm Technologies: Pointer authentication on ARMv8.3 (2017). https://www.qualcomm.com/media/documents/files/whitepaper-pointerauthentication-on-armv8-3.pdf
5. Apple Inc.: Pointer Authentication (2019). https://github.com/apple/llvm-project/blob/apple/main/clang/docs/PointerAuthentication.rst
6. Li, R., Jin, C.: Meet-in-the-middle attacks on reduced-round QARMA-64/128. Comput. J. **61**(8), 1158–1165 (2018)
7. Yang, D., Qi, W.F.: Impossible differential attack on QARMA family of block ciphers. IACR Cryptology ePrint Archive 2018/334 (2018)
8. Li, M., Hu, K., Wang, M.: Related-tweak statistical saturation cryptanalysis and its application on QARMA. IACR Trans. Symmetric Cryptol. **2019**(1), 236–263 (2019)
9. Hua, Z., Gu, J., Xia, Y.: vTZ: virtualizing ARM trustzone. In: 26th USENIX Security Symposium (USENIX Security 2017), pp. 541–556 (2017)
10. Avanzi, R.: The QARMA block cipher family. Almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017)
11. Zhang, T., Shen, W., Lee, D.: PeX: a permission check analysis framework for Linux kernel. In: 28th USENIX Security Symposium (USENIX Security 2019), pp. 1205–1220 (2019)
12. ARM Limited: SMC calling convention (2019). http://infocenter.arm.com/help/topic/com.arm.doc.den0028b/ARM_DEN0028B_SMC_Calling_Convention.pdf
13. BYTE Magazine (2011). http://www.math.utah.edu/~mayer/linux/byte/bdoc.pdf
14. ARM Limited: Fast Models User Guide (2019). https://static.docs.arm.com/100965/1111/fast_models_ug_100965_1111_00_en.pdf
15. Liljestrand, H., Nyman, T., Wang, K.: PAC it up: towards pointer integrity using ARM pointer authentication. In: 28th USENIX Security Symposium (USENIX Security 2019), pp. 177–194 (2019)
16. Liljestrand, H., Gauhar, Z., Nyman, T.: Protecting the stack with PACed canaries. In: Proceedings of the 4th Workshop on System Software for Trusted Execution, pp. 1–6 (2019)
17. Liljestrand, H., Nyman, T., Ekberg, J.E.: Authenticated call stack. In: Proceedings of the 56th Annual Design Automation Conference 2019, pp. 1–2 (2019)
18. Farkhani, R.M., Ahmadi, M., Lu, L.: PTAuth: temporal memory safety via robust points-to authentication. CoRR (2020)
19. Yang, Y., Zhu, S., Shen, W.: ARM pointer authentication based forward-edge and backward-edge control flow integrity for kernels. arXiv preprint arXiv:1912.10666 (2019)
20. Denis-Courmont, R., Liljestrand, H., Chinea, C.: Camouflage: hardware-assisted CFI for the ARM Linux kernel. In: 2020 57th ACM/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE (2020)
21. Ferri, G., Cicero, G., Biondi, A.: Towards the hypervision of hardware-based control flow integrity for arm platforms. In: ITASEC (2019)

Privacy-Preserving ECC-Based Three-Factor Authentication Protocol for Smart Remote Vehicle Control System

Hongwei Luo[1,2](✉), Qinyao Zhang[1], and Guoai Xu[1,2]

[1] Institute of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
kevin.lhw@antfin.com

[2] National Engineering Laboratory of Mobile Network Security, Beijing 100876, China

Abstract. With the rapid development of auto industry and the Internet, smart remote vehicle control (SRVC) system is now showing in more and more automobile brands which allows people to remotely control vehicles through the Internet. However, SVRC's convenience brings security challenges that the related SRVC's protocol needs to be enhanced with identity authentication mechanism for both users and their vehicles, in case of illegal intrusion without identity authentication. In this paper, we first analyze Chatterjee et al.'s scheme and find that their scheme is not immune from some common attacks. Then, we design a privacy-preserving elliptic curve cryptosystem (ECC)-based three-factor authentication scheme based on the SRVC's features, which can authenticate the identities of users and vehicle, and generate a session secret key to protect the users' privacy. Security analysis shows that our protocol has many security attributes that it not only can protect users' anonymity and untraceability, but also resist many known attacks. And performance analysis shows that our protocol can run efficiently in SRVC system. Lastly, we conclude our work and give the future research direction in SVRC.

Keywords: Three-factor authentication · Privacy-preserving · Smart remote vehicle control (SRVC) · Elliptic curve cryptosystem (ECC)

1 Introduction

In the past few decades, with the upgrading of automobile control systems, the human-vehicle interaction mode has been constantly changing [1]. In the beginning, people used traditional physical car keys to open the doors and start the car. Later, car manufacturers embedded electronic control devices in the car, which enables people to control the car doors remotely by pressing the button on radio car keys.

The remote-control function of radio car keys is mainly realized by the embedded RFID chip. In the communication process between the radio key and the car electronic control device, generally, a symmetric encryption algorithm (such as AES) is used to protect the confidentiality of the message. But the work in [2] shows that these radio

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 56–72, 2021.
https://doi.org/10.1007/978-3-030-80851-8_5

car keys are subject to relay attacks. The attacker can actively or passively control the car from a long distance through the relay device. What's more, once the long-term symmetric secret key stored in the radio key is acquired, the vehicle control right will easily be changed.

Furthermore, with the development of the Internet of Things (IoT) and the Internet of Vehicles (IoV), smart remote vehicle control (SRVC) systems are gradually appearing in much more cars. This charming system allows people to use smart devices to control the vehicle. So, users can not only send commands of opening and closing the door or starting the car through the network, but also read the real-time data of the vehicle and perform multiple operations such as controlling windows and air conditioning.

However, the convenience of remote network communication brings many challenges to the SRVC system. Since the openness of the network, attackers can initiate a variety of attacks, such as interception, modification and replay of network communication messages. Therefore, it is necessary to propose an authentication protocol, as a first line to defend the attacks talked before, for the SRVC system to realize mutual authentication and protect communication data and user privacy.

1.1 Threat Model

By the Delov-Yao adversary model [19], we suppose that the adversary can intercept, modify, delete and replay all messages transmitted via insecure public channels in the protocol. The adversary may have the ability to obtain the data stored in the smart-card or in the car enterprise servers. Besides, the insider attacker can also perform an insider attack to obtain confidential information from the enterprise database.

1.2 Related Works

In this part, we provide a review of authentication and key agreement schemes related to the SRVC system.

Since Lamport [3] proposed the first password authentication scheme, a large number of schemes have been presented to problems of different network communication models based on many cryptography primitives. Chen et al. [5] used ECC earlier in their dynamic ID-based remote mutual authentication scheme for cloud computing. Then, to get a better security property, taking the advantage of Elliptic Curve Diffie-Hellman Problem (ECDHP), many mutual authentication schemes for IoT used ECC secret keys as participants's identity certificates. In 2013 and 2014 Lai et al. [9, 10] proposed two authentication and key agreements of IoT using ECC, which provided group authentication. However, their protocols lacked location and identity privacy considerations. Chu et al. [8] offered their identity authentication scheme for IoT, which could resist several attacks in 2013. But users in their scheme need to authenticate many times in multi-server environment. Interestingly, Poranbage et al. [7] presented a two-phase authentication protocol distributed IoT applications using ECC. Their scheme is able to resist malicious users and Denial of Service (DoS) attacks, but not efficient and cannot resist node capturing attack. As an improved work, Truong et al. [15] overcame Lee et al.'s [16] limitations and proposed a dynamic ID-based user authentication scheme. And then Kumari et al. [4] analyzed Truong's scheme and found that Truong's scheme suffers from password guessing attack and no forward secrecy. For different design scenario, Feng et al. [17] introduced a three-factor mobile multi-server authentication scheme using ECC. Alotaibi et al. [6] proposed a three-factor authentication for WSN using ECC in 2018 which can resist clone and fork attacks. Mo, Jiaqing, et al. [12] proposed a ECC-based two-factor user authentication protocol for the mobile cloud computing environment, which has reasonable computation cost and communication overhead at the mobile client side as well as the server side. Based on a comprehensive threat model, Aghili et al. [18] presented a lightweight scheme for E-Health Systems in IoT with three-factor authentication access control and ownership transfer in 2019. In 2020, Chatterjee et al. [11] proposed a robust lightweight ECC-based three-way authentication scheme for IoT in cloud. Panda et al. [13] also introduced a mutual authentication protocol for IoT environment which has relatively good security performance.

1.3 Motivation and Contribution

The existing SRVC protocol is mainly designed by automobile manufacturers and written into the intelligent vehicle control App and the in-car smart terminal. Given the non-openness of the car manufactures' code on authentication, by the only effective way of comparing the IoT model with the SRVC model, we find that SRVC system has some unique features listed below as our first contribution.

(1) The participants of protocols in SRVC system are smart device, car enterprise server and in-car smart terminal.
(2) A smart device has to communicate with the car directly for the need of privacy protection, so there shall be session key exchange phase after mutual authentication phase in a SRVC protocol.
(3) Different from the GWN and sensors in IOT protocols, the enterprise server and in-car smart terminals have stronger computing power and more stable network connection, which means more computation can be done and longer message can be transmitted to ensure the safety of protocol.
(4) The key agreement in common IoT communication is sensitive to time cost, but the SRVC is different. People open applications on smart devices and connect to the vehicles. The key agreement phase is completed at this stage. Due to a large number of computing requirements, most applications are slow when they are opened. Therefore, time performance i in the SRVC key agreement phase is not as important as in common IoT systems.
(5) Compared to IOT protocols, the security and privacy protection capabilities of SRVC protocols are more important for they are closely related to the user's life safety.

By the use of these features shown above, our second contribution is that the mutual authentication and session key agreement protocol are given special for SRVC. Specifically:

- We prove that Chatterjee et al.'s ECC-based three-way scheme of IoT [11] suffers from several attacks.

- We put forward a novel authentication protocol for SRVC system using ECC. The proposed protocol satisfies various security attributes and can also prevent the disclosure of important privacy of users.
- We conducted a security analysis of the protocol, which demonstrates that the proposed scheme can resist known attacks.
- The performance evaluation shows that our scheme can run efficiently on SRVC system.

1.4 Notations

The notations are introduced in Table 1.

Table 1. Notation of the paper

Symbols	Description
U, S, T	User, car enterprise server and smart terminal of the car
UID, SID, TID, DID	IDs of U, S, T, and the IoT node
k_U, k_{US}, k_{ST}, k_T	Public keys computed by U, S, T
d_U, d_{US}, d_{ST}, d_T	Private keys selected by U, S, T
$k_D,\ k_N, k_G$	Public keys of smart device, IoT node and gateway server in Chatterjee et al.'s scheme
$d_D,\ d_N, d_G$	Private keys of smart device, IoT node and gateway server in Chatterjee et al.'s scheme
h	One-way hash function
h_B	Bio hash function
$PW,\ B$	U's password and biometric
R_{Zp}	Random numbers chosen from the field Z_p
K_{sess}	Session key
E_p	A p-order elliptic curve group defined over $GF(q)$
G	Generate of E_p

1.5 Paper Organization

The rest of the paper is organized as follows. The review of Chatterjee et al.'s scheme is presented in Sect. 2. In Sect. 3, we analyze Chatterjee et al.'s Scheme and prove that it has security limitations. Our protocol is proposed in Sect. 4. The security analysis is discussed in Sect. 5. Section 6 shows the performance evaluation. Finally, concluding remarks and future research are discussed in Sect. 7.

2 Review of Chatterjee et al.'s Scheme

2.1 System Initialization and Registration Phase

This phase includes registration process of the user, the smart device and the IoT node.

1. User registration with smart device: User chooses his identity *UID*, password *PW* and biometric *B*. Then smart device stores them in the form of hashed values.
2. Smart device registration with gateway server: Smart device sends ID and publishes its public key k_U with certificate. Upon receiving the message, gateway server stores *UID* and publish its public key k_G.
3. IoT device registration with gateway server: IoT device sends its identity and public key certificate to the gateway server. Then the gateway server stores its ID and sends its certificate back.

2.2 Authentication and Key Agreement Phase

In this phase, the smart device, the IoT node and the gateway server authenticate each other over a public channel.

Firstly, the user inputs his ID, password and biometric into the smart device and the smart device computes hashed value to authenticate the user.

For the purpose of login, smart device first generates a nonce n_i. Then it computes $d_U \cdot k_G$ and $h(DID \oplus n_i \oplus d_U \cdot k_G)$ and sends $(DID, n_i, h(DID \oplus n_i \oplus d_U \cdot k_G))$ to the IoT node.

Upon receiving login request from the smart device, it generates a random number $p \in R_{Zp}$ and computes $h(NID \oplus d_N \cdot k_G \oplus n_i), p \oplus h(d_N \cdot k_G)$ and sends it with the login request to the IoT gateway.

The gateway server first computes $d_G \cdot k_D$, and checks if $h(DID \oplus n_i \oplus d_U \cdot k_G) = h(DID \oplus n_i \oplus d_G \cdot k_U)$, $h(NID \oplus d_N \cdot k_G \oplus n_i) = h(NID \oplus d_G \cdot k_U \oplus n_i)$. If so, the gateway server can authenticate the smart device and the IoT node. Then the server generates a nonce n_j, computes $p = p \oplus h(d_N \cdot k_G) \oplus h(d_G \cdot k_N)$, $h\big(d_G \cdot K_D \oplus n_j\big)$, and sends $\big(n_j,\ h\big(d_G \cdot K_D \oplus n_j\big),\ p \oplus h(d_G \cdot k_D)\big)$ to the smart device.

The smart device first checks if $h\big(d_D \cdot K_G \oplus n_j\big) = h\big(d_G \cdot K_D \oplus n_j\big)$. If they are equal, the smart device generates a random number $r \in R_{Zp}$, computes p with $p \oplus h(d_G \cdot k_D) \oplus h(d_D \cdot k_G)$ and computes $SK = r \cdot p \cdot G$. Then the smart device sends $\big(n_j,\ h\big(p \cdot r \cdot n_j\big),\ r \oplus p\big)$ to the IoT node.

The IoT node now computes r with the message from the smart device and compares $h\big(r \oplus p \oplus n_j\big)$ and check the equality to the value in the message. Finally, it computes $SK = r \cdot p \cdot G$.

3 Cryptanalysis of Chatterjee et al.'s Scheme

Based on Delov-Yao threat model, since the login authentication and key agreement phase take place on the public channel, messages transmitted can be intercepted and modified by an adversary. All data stored on the gateway server is available for the privileged insider. In the light of the aforementioned scenario, we present the security problems of Chatterjee et al.'s scheme. The listing and the discussion of various security problems of Chatterjee et al.'s scheme are given below:

1. User impersonation attack
   Since the public and private keys are generated in advance, if the adversary is able to obtain any two private keys, he can perform a user impersonation attack. Taking d_N and d_G for instance, the adversary can easily get *DID* for it is transmitted in plaintext, then he generates a nonce n_i^*, and computes $h\big(NID \oplus n_i^* \oplus d_G \cdot k_D\big)$ and sends $\big(DID, n_i^*, h\big(NID \oplus n_i^* \oplus d_G \cdot k_D\big)\big)$ to the gateway server. When the gateway server receives the message, it checks the validation of the message with $h\big(NID \oplus n_i^* \oplus d_G \cdot k_D\big)$, and obviously the message can pass the check. When the IoT Node transmits message to the adversary, he is able to compute p with $d_N \cdot k_D$. Then the adversary selects a random r and computes *SK* and sends r to the IoT node. Finally, a session between the adversary and the IoT node is established.
2. User tracking attack
   In login authentication phase, the smart device sends *DID* to the gateway server in plain text. Although *DID* is not directly related to the user's identity, it is fixed for a specific and not updated with each session. When monitoring the public channel, the adversary can launch a tracking attack with this fixed value. Then he might collect sensitive information related to the user like user's location, user's traveling routes and so on.
3. DoS attack
   Since none of the three participants in the scheme check the freshness of messages and responds messages with calculated values. The adversary can eavesdrop a message and resend it a large number of times, making the gateway server or the IoT node out of service.

4 The Proposed Scheme

In this section, we put forward a three-factor authentication scheme for SRVC system based on ECC. It includes three kinds of participants, i.e., the user U, the car enterprise server S and the vehicle smart terminal T. In registration phase, S is responsible for storing and issuing hashed user information to U and T. In login and authentication phase, S is responsible for implement mutual authentication and negotiate a session key between U and T.

4.1 Pre-deployment Phase

When the car enterprise produces a car, it also generates the unique *TID* of the car and stores the value in the enterprise database.

S selects E_p and G. E_p is a p-order elliptic curve group defined over Galois field $GF(q)$ where q is a prime or in the binary space 2^n and G is a generator of the group E_p. S publishes the parameters $\big(E_p, G\big)$.

4.2 User Registration Phase

This phase is depicted as Fig. 1. Below are the details.

Step1. A user U first gets the TID of his car from the automobile enterprise. Then U randomly picks a UID, a random nonce r_U and choose a password PW. U computes $h(PW||r_U)$ and send parameters $(UID,\ h(PW||r_U), TID)$ via a secure channel.

Step2. Upon receiving $(UID,\ h(PW||r_U), CID)$, S picks two random numbers r_S, r_T and a nonce n_0, computes $A_0 = h(UID||h(PW||r_U)) \oplus n_0$, $C_{UT} = A \oplus h(TID)$, $h(SID||r_S), h(SID||r_T), B_{US} = A \oplus h(SID||r_S), C_{US} = h(A||(SID||r_S))$, $C_{ST} = h(h(CID)||h(SID||r_T))$ and stores $(C_{UT},\ B_{US},\ C_{US},\ C_{ST})$ to its database. Next, S saves $(C_{US},\ h(SID||r_S))$ on a smart-card and hands it to the user U. Then S sends (C_{ST}, TID) to T. Both two data transmissions are carried out via secure channels.

Step3. When U receives the smart-card, he imprints his biometric data B, computes $C_1 = h(UID||PW) \oplus h(SID||r_S)$, $C_2 = h_B(B) \oplus h(UID||h(PW||r_U))$, and replaces $h(SID||r_S)$ in smart-card with C_1 and C_2. Now smart-card contains $(C_1, C_2,\ C_{US})$.

Step4. Upon the receipt of (C_{ST}, CID), T computes $h(TID)$ and stores $(h(TID), C_{ST})$ in a secure storage space.

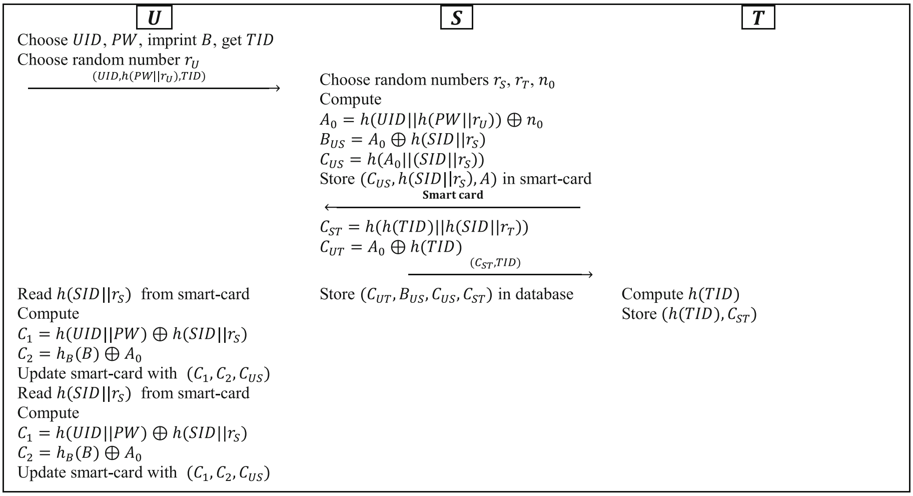

U
S
T
Choose UID, PW, imprint B, get TID
Choose random number r_U
$(UID, h(PW||r_U), TID)$
Choose random numbers r_S, r_T, n_0
Compute
$A_0 = h(UID||h(PW||r_U)) \oplus n_0$
$B_{US} = A_0 \oplus h(SID||r_S)$
$C_{US} = h(A_0||(SID||r_S))$
Store $(C_{US}, h(SID||r_S), A)$ in smart-card
Smart card
$C_{ST} = h(h(TID)||h(SID||r_T))$
$C_{UT} = A_0 \oplus h(TID)$
(C_{ST}, TID)
Read $h(SID||r_S)$ from smart-card
Compute
$C_1 = h(UID||PW) \oplus h(SID||r_S)$
$C_2 = h_B(B) \oplus A_0$
Update smart-card with (C_1, C_2, C_{US})
Read $h(SID||r_S)$ from smart-card
Compute
$C_1 = h(UID||PW) \oplus h(SID||r_S)$
$C_2 = h_B(B) \oplus A_0$
Update smart-card with (C_1, C_2, C_{US})
Store $(C_{UT}, B_{US}, C_{US}, C_{ST})$ in database
Compute $h(TID)$
Store $(h(TID), C_{ST})$

Fig. 1. Registration phase of the proposed scheme

4.3 Login and Authentication Phase

Summary of this phase is in Fig. 2 and below are details.

Step1. Firstly, U reads C_1, C_2 and Cus from the smart-card. With the input of UID', PW' and B', U computes $t_1 = C_1 \oplus h(UID'||PW')$, $t_2 = C_2 \oplus h_B(B)$. Next, U computes $C'_{sc} = h(t_2||t_1)$ and checks if $C'_{ST} = C_{ST}$. If they are equal, which means $A_i = t_2$ and $h(SID||r_S) = t_1$, U selects a random number $d_U \in {}_RZ_p$, picks a random nonce n_1 and computes $k_U = d_U \cdot G$, $D = n_1 \oplus h(SID||r_S)$, $E = h(A||h(SID||r_S)) \oplus k_U$, $V_1 = h(A||h(SID||r_S)||k_U||n_1||T_1)$ where T_1 is the current timestamp, and sends $message1 = (A,\ D,\ E, V_1, T_1)$ to S via a public channel. Otherwise U stops login.

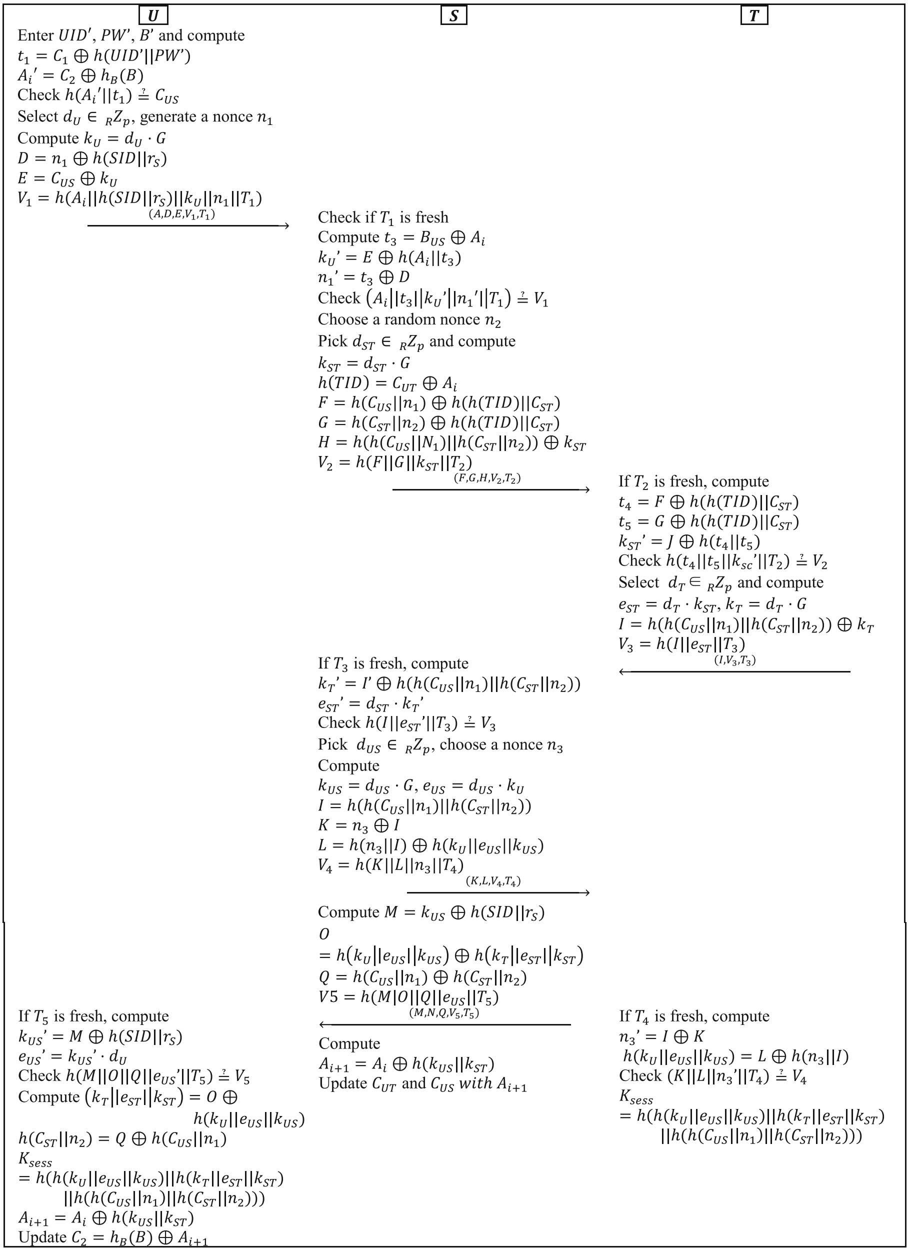
Step2. Upon receiving the login request $message1$ from U, S first check T_1. If T_1 is fresh, U computes $t_3 = B_{US} \oplus A$, $k'_U = E \oplus h(A||t_3)$, $n'_1 = t_3 \oplus D$, $V'_1 = \left(A||t_3||k'_U\middle|\middle|n'_1\middle|\middle|T_1\right)$ and check the if $V'_1 = V_1$. Upon the equality of the two is confirmed, S picks a nonce n_2, and computes $h(TID) = C_{UT} \oplus A$, $F = h(C_{US}||n_1) \oplus h(h(TID)||C_{ST})$, $G = h(C_{ST}||n_2) \oplus h(h(TID)||C_{ST})$. *Next, S selects $d_{ST} \in {}_RZ_p$ and computes $k_{ST} = d_{ST} \cdot G$, $H = h(h(C_{US}||N_1)||h(C_{ST}||n_2)) \oplus k_{ST}$* and $V_2 = h(F||G||k_{ST}||T_2)$ where T_2 is the current timestamp. Then S sends $message2 = (F, G, H, V_2, T_2)$ to T via a public channel.

Step3. When T receives $message2$ from S, T first check the freshness of T_2. If so, T computes $t_4 = F \oplus h(h(TID||C_{ST}))$, $t_5 = G \oplus h(h(TID||C_{ST}))$, $k'_{ST} = J \oplus h(t_4||t_5)$, $V'_2 = h(t_4||t_5||k'_{sc}||T_2)$ and check if $V'_2 = V_2$. If so, T selects $d_T \in {}_RZ_p$ and computes $e_{ST} = d_T \cdot k_{ST}, k_T = d_T \cdot G, I = h(h(C_{US}||n_1)||h(C_{ST}||n_2)) \oplus k_c$, $V_3 = h(I||e_{ST}||T_3)$. Then T_3 sends parameters $message3 = (I, V_3, T_3)$ back to S.

Step4. After receiving $message3$ from T and checking the freshness of T_3, S first computes $k'_T = I' \oplus h(h(C_{US}||n_1)||h(C_{ST}||n_2)), e'_{ST} = d_{ST} \cdot k'_T, V'_3 = h\left(I\middle|\middle|e'_{ST}\middle|\middle|T_3\right)$ and checks if V'_3 is equal to V_3. For the correct match, S selects $d_{US} \in {}_RZ_p$ and computes $k_{US} = d_{US} \cdot G$, $e_{US} = d_{US} \cdot k_U$. Next S picks a nonce n_3 randomly and computes $I = h(h(C_{US}||n_1)||h(C_{ST}||n_2))$, $K = n_3 \oplus I$, $L = h(n_3||I) \oplus h(k_U||e_{US}||k_{US})$, $V_4 = h(K||L||n_3||T_4)$ where T_4 is the current timestamp and sends $message4 = (K, L, V_4, T_4)$ to T via a public channel. S then computes $M = k_{US} \oplus h(SID||r_S)$, $O = h(k_U||e_{US}||k_{US}) \oplus h(k_T||e_{ST}||k_{ST})$, $Q = h(C_{US}||n_1) \oplus h(C_{ST}||n_2)$, $V5 = h(M|O||Q||e_{US}||T_5)$ and sends $message5 = (M, N, Q, V_5, T_5)$ to U via a public channel where T_5 is the current timestamp.

Step5. T receives the parameters in $message4$ from S and checks the freshness of T_4 firstly. Next, T computes $n'_3 = I \oplus K$, $h(k_U||e_{US}||k_{US}) = L \oplus h(n_3||I)$, $V'_4 = (K||L||n'_3||T_4)$ and check if $V'_4 = V_4$. If so, T computes the Session Key $K_{sess} = h(h(k_U||e_{US}||k_{US})||h(k_T||e_{ST}||k_{ST})||h(h(C_{US}|n_1)||h(C_{ST}||n_2)))$.

Step6. U first checks if T_5 is fresh upon receiving $message5$ from S. Then U compute $k'_{US} = M \oplus h(SID||r_S)$, $e'_{US} = k'_{US} \cdot d_U$, $V'_5 = h\left(M||O||Q\middle|\middle|e'_{US}\middle|\middle|T_5\right)$ and check the equality of V'_5 and V_5. If so, U computes $h(k_c||e_{sc}||k_{sc}) = O \oplus h(k_U||e_{US}||k_{US})$, $h(C_{ST}||n_2) = Q \oplus h(C_{US}||n_1)$. Finally, U computes $K_{sess} = h(h(k_U||e_{US}||k_{US})||h(k_T||e_{ST}||k_{ST})||h(h(C_{US}|n_1)||h(C_{ST}||n_2)))$.



Enter UID', PW', B' and compute
$t_1 = C_1 \oplus h(UID'||PW')$
$A_i' = C_2 \oplus h_B(B)$
Check $h(A_i'||t_1) \overset{?}{=} C_{US}$
Select $d_U \in {}_R Z_p$, generate a nonce n_1
Compute $k_U = d_U \cdot G$
$D = n_1 \oplus h(SID||r_S)$
$E = C_{US} \oplus k_U$
$V_1 = h(A_i||h(SID||r_S)||k_U||n_1||T_1)$

$\xrightarrow{(A,D,E,V_1,T_1)}$

Check if T_1 is fresh
Compute $t_3 = B_{US} \oplus A_i$
$k_U' = E \oplus h(A_i||t_3)$
$n_1' = t_3 \oplus D$
Check $\left(A_i||t_3||k_U'||n_1'||T_1\right) \overset{?}{=} V_1$
Choose a random nonce n_2
Pick $d_{ST} \in {}_R Z_p$ and compute
$k_{ST} = d_{ST} \cdot G$
$h(TID) = C_{UT} \oplus A_i$
$F = h(C_{US}||n_1) \oplus h(h(TID)||C_{ST})$
$G = h(C_{ST}||n_2) \oplus h(h(TID)||C_{ST})$
$H = h(h(C_{US}||N_1)||h(C_{ST}||n_2)) \oplus k_{ST}$
$V_2 = h(F||G||k_{ST}||T_2)$

$\xrightarrow{(F,G,H,V_2,T_2)}$

If T_2 is fresh, compute
$t_4 = F \oplus h(h(TID)||C_{ST})$
$t_5 = G \oplus h(h(TID)||C_{ST})$
$k_{ST}' = J \oplus h(t_4||t_5)$
Check $h(t_4||t_5||k_{sc}'||T_2) \overset{?}{=} V_2$
Select $d_T \in {}_R Z_p$ and compute
$e_{ST} = d_T \cdot k_{ST}$, $k_T = d_T \cdot G$
$I = h(h(C_{US}||n_1)||h(C_{ST}||n_2)) \oplus k_T$
$V_3 = h(I||e_{ST}||T_3)$

$\xleftarrow{(I,V_3,T_3)}$

If T_3 is fresh, compute
$k_T' = I' \oplus h(h(C_{US}||n_1)||h(C_{ST}||n_2))$
$e_{ST}' = d_{ST} \cdot k_T'$
Check $h(I||e_{ST}'||T_3) \overset{?}{=} V_3$
Pick $d_{US} \in {}_R Z_p$, choose a nonce n_3
Compute
$k_{US} = d_{US} \cdot G$, $e_{US} = d_{US} \cdot k_U$
$I = h(h(C_{US}||n_1)||h(C_{ST}||n_2))$
$K = n_3 \oplus I$
$L = h(n_3||I) \oplus h(k_U||e_{US}||k_{US})$
$V_4 = h(K||L||n_3||T_4)$

$\xrightarrow{(K,L,V_4,T_4)}$

Compute $M = k_{US} \oplus h(SID||r_S)$
O
$= h\left(k_U||e_{US}||k_{US}\right) \oplus h\left(k_T||e_{ST}||k_{ST}\right)$
$Q = h(C_{US}||n_1) \oplus h(C_{ST}||n_2)$
$V5 = h(M|O||Q||e_{US}||T_5)$

$\xleftarrow{(M,N,Q,V_5,T_5)}$

If T_5 is fresh, compute
$k_{US}' = M \oplus h(SID||r_S)$
$e_{US}' = k_{US}' \cdot d_U$
Check $h(M||O||Q||e_{US}'||T_5) \overset{?}{=} V_5$
Compute $\left(k_T||e_{ST}||k_{ST}\right) = O \oplus h(k_U||e_{US}||k_{US})$
$h(C_{ST}||n_2) = Q \oplus h(C_{US}||n_1)$
K_{sess}
$= h(h(k_U||e_{US}||k_{US})||h(k_T||e_{ST}||k_{ST})||h(h(C_{US}||n_1)||h(C_{ST}||n_2)))$
$A_{i+1} = A_i \oplus h(k_{US}||k_{ST})$
Update $C_2 = h_B(B) \oplus A_{i+1}$

Compute
$A_{i+1} = A_i \oplus h(k_{US}||k_{ST})$
Update C_{UT} and C_{US} $with\ A_{i+1}$

If T_4 is fresh, compute
$n_3' = I \oplus K$
$h(k_U||e_{US}||k_{US}) = L \oplus h(n_3||I)$
Check $(K||L||n_3'||T_4) \overset{?}{=} V_4$
K_{sess}
$= h(h(k_U||e_{US}||k_{US})||h(k_T||e_{ST}||k_{ST})||h(h(C_{US}||n_1)||h(C_{ST}||n_2)))$

Fig. 2. The login and authentication phase of the proposed scheme

5 Security Analysis

In this section, we first show that our scheme provides mutual authentication and key agreement between the user and the car by using BAN logic. Then we demonstrate different security attributes related to our protocol.

5.1 BAN-Logic Based Proof of Authentication

In this subsection, the BAN logic is used to prove that a session key will be agreed between the user and the car node after the execution of the proposed scheme. Table 2 shows the notations used in the BAN logic.

Table 2. Notations in the BAN logic

Symbols	Discription
P, Q	Principals
X	Statements
$\#(X)$	X is fresh
$P \lhd X$	P sees X
$P \mid \sim X$	P once said X
$P \mid \equiv X$	P believes X
$P \overset{K}{\leftrightarrow} Q$	P and Q use the shared key K to communicate
$P \Rightarrow X$	P has jurisdiction over X
$< X >_Y$	X combined with Y
The message-meaning rule	$\frac{P\mid\equiv P \overset{K}{\leftrightarrow} Q, P\lhd<X>_K}{P\mid\equiv Q\mid\sim X}$ or $\frac{P\mid\equiv \overset{K}{\rightarrow} Q, P\lhd\{X\}_{K-1}}{P\mid\equiv Q\mid\sim X}$
The freshness-conjuncatenation rule	$\frac{P\mid\equiv\#(X), P\mid\equiv Q\mid\sim X}{P\mid\equiv Q\mid\equiv X}$
The jurisdiction rule	$\frac{P\mid\equiv Q\Rightarrow X, P\mid\equiv Q\mid\equiv X}{P\mid\equiv X}$

Our scheme should be able to achieve the following goals:

G1: $\mathrm{U}\mid\equiv\left(U \overset{SK}{\leftrightarrow} \mathrm{T}\right)$

G2: $\mathrm{T}\mid\equiv\left(U \overset{SK}{\leftrightarrow} \mathrm{T}\right)$

The idealized forms of our scheme are listed below:

M1:
$\mathrm{S} \rightarrow (U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T,\ e_{US})_{h(SID|r_S)}$

M2: $\mathrm{S} \to T(\overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T,\ V_4)_{h(n_3||I)}$

The basic initial assumptions of our scheme are as follows:

A1: $U| \equiv \left(U \overset{h(SID|r_S)}{\longleftrightarrow} S\right)$
A2: $U| \equiv \#(e_{US})$
A3: $U| \equiv S \Rightarrow \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T\right)$
A4: $T| \equiv \left(S \overset{h(n_3||I)}{\longleftrightarrow} T\right)$
A5: $T| \equiv \#(V_4)$
A6: $T| \equiv S \Rightarrow \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T\right)$

The proof process is as follows

From M1, it is easy to get the following statement:

S1:
$S \lhd (U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T,\ e_{US})_{h(SID|r_S)}$

From S1, A1 and the message-meaning rule, we get

S2:
$U|\equiv S| \sim \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T,\ e_{US}\right)$

From S2, A2 and the freshness-conjuncatenation rule, we get

S3: $U|\equiv S| \equiv \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T\right)$

From S3, A3 and the jurisdiction rule, we get

S4: $U| \equiv \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T\right)$

Since $K_{sess} = h(h(k_U||e_{US}||k_{US})||h(k_T||e_{ST}||k_{ST})||h(h(C_{US}||n_1)||h(C_{ST}||n_2)))$, we get

S5: $U| \equiv \left(U \overset{K_{sess}}{\longleftrightarrow} T\right)$ (G1)

From M2, it is easy to get the following statement:

S6:
$S \lhd (U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T,\ V_4)_{h(n_3||I)}$

From S6, A4 and the message-meaning rule, we get

S7:
$T|\equiv S| \sim \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T,\ V_4\right)$

From S7, A5 and the freshness-conjuncatenation rule, we get

S8: $T|\equiv S| \equiv \left(U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T\right)$

From S8, A6 and the jurisdiction rule, we get

S9: $T| \equiv U \overset{h(k_U||e_{US}||k_{US})}{\longleftrightarrow} T,\ U \overset{h(k_T||e_{ST}||k_{ST})}{\longleftrightarrow} T, U \overset{h(C_{US}||n_1)}{\longleftrightarrow} T, U \overset{h(C_{ST}||n_2)}{\longleftrightarrow} T)$

Since $K_{sess} = h(h(k_U||e_{US}||k_{US})||h(k_T||e_{ST}||k_{ST})||h(h(C_{US}||n_1)||h(C_{ST}||n_2)))$, we get

S10: $T| \equiv \left(U \overset{K_{sess}}{\longleftrightarrow} T\right)$ (G2)

Through G1 and G2, we can get that both U and T believe that a session key K_{sess} is agreed between them.

5.2 Further Security Analysis

This subsection demonstrates that the proposed scheme is immune to known attacks and provides various desirable security properties.

Mutual Authentication

Our scheme has the property of mutual authentication, which means all participants in the scheme are convinced of each other. The three participants are authenticated by each other with the pre-calculated values $\{A_i, B_{US}, C_{ST}, h(TID)\}$, the randomly generated nonces $\{n_1, n_2, n_3\}$ and the temporary computed $\{e_{US}, e_{ST}\}$ associated with ECC keys, which cannot be forged by an adversary.

Session Key Agreement

U and T generate a shared session key K_{sess}. The session key is composed of ECC keys k_x, nonces n_i and values e_{US}, e_{ST} generated by Diffie-Hellman key exchange, which guarantees forward secrecy. All values mentioned above are generated randomly, which provides the resistance of known key attack and session-specific temporary information attack.

Forward Secrecy

Forward Secrecy is a feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if long-term secrets used in the session key exchange are compromised. In the proposed scheme, session key is computed as $h(h(k_U\|e_{US}\|k_{US})\|\ h(k_T\|e_{ST}\|k_{ST})\|h(h(C_{US}\|n_1\|)\|h(C_{ST}||n_2)\|))$, where $\{k_x\}$ are public keys of each participants and n_1, n_2 are random nonces. No value can be retrieved for all keys and nonces are generated temporarily which means they are different from each session. Besides, due to the intractability of ECDHP, private keys of U, S and T in each session cannot be computed by the adversary.

Known Key Security

In the proposed protocol, the session key K_{sess} is computed depending on the nonces $\{n_1, n_2\}$ and ECC secret keys $\{k_U, k_{US}, k_{ST}, k_T\}$ generated by all participants. These nonces are randomly picked and keys are computed by random numbers selected in R_{Z_p} which are different from session to session. Besides, the values are not transmitted in plain text so that no adversary can get the true values of them without the knowledge of all other well protected messages. Thus, obtaining one session key does no help for computing other session keys. So, our protocol provides known-key security.

User Anonymity

Out scheme has the property of user anonymity for any adversary cannot find or compute user information from our messages. Information of U, S and T is only transmitted in plaintext form during user registration via secure channel which can be monitored by no adversary. In login and authentication phase information of the three is protected by one-way hash function. Moreover, data in any message are randomized by nonces $\{n_i\}$ in one authentication session and by elliptic curve keys $\{d_x, k_x\}$ between sessions. Therefore, any adversary has the ability to link any two sessions.

Resist User Tracking Attack

In our proposed scheme, every message transmitted via public channels are randomized by nonces n_1, n_2 and ECC keys selected from R_{Zp}. The only one value without protection is A_i. However, A_i is updated to $A_{i+1} = A_i \oplus h(k_{US}||k_{ST})$ every time the session key generated. Therefore, when the adversary eavesdrops on the delivered messages in different sessions, he cannot confirm that two messages are from a fixed user in our scheme. Hence, our scheme provides resistance to user tracking attack.

Resist Forgery and Impersonation Attack

Our scheme is able to resist forgery and impersonation attack, because when an adversary attempts to generate a legitimate login message (A_i, D, E, V_1, T_1), he does know UID, PW, B so that he cannot generate any data except a fake timestamp.

Resist Privileged Insider Attack

If a privileged insider adversary attempts to get user-related information from a database stored on the server, he might obtain $(DID,\ B_{US},\ C_{US},\ C_{ST})$. DID, B_{US}, C_{US} are all computed with A_i which updates every session. C_{ST} are also computed by one-way hash function. Even if the insider can eavesdrop all messages transmitted in a session, he cannot disclose any vital information of user.

Resist Replay Attack

In our proposed scheme, every message sent over public channel contains a timestamp T_i, which also has been hashed with other random values in V_i. Take *meesage*3 for example, if the adversary replays a *meesage*3, as the timestamp T_3 is not fresh, ultimately the authentication fails. If the adversary replaces T_3 with a fresh timestamp T_3', as the random ECC keys in each session are different, he cannot compute a correct V_3 as a consequence.

Resist Lost-Smart-Card Attack

We propose our scheme under the assumption of adversary might obtain a lost smart card. If the attacker gets (C_1, C_2, C_{US}) in the card, when he attempts to compute A, he cannot get the correct value due to lack of UID and PW. Even if the adversary obtains

UID and *PW* somehow, he cannot compute a legitimate $h(SID||r_S)$ because the biometric B is not stored in the smart-card, let alone compute D, E and V_1 correctly.

Resist Man-in-the-Middle Attack

We suppose that an adversary can intercept, modify and delete messages transmitted over a public channel. However, session keys are computed by U and T, and no plaintext of key information is transmitted via the public channel. Though the adversary is able to intercept any message in login and authentication phase of our scheme, he cannot compute the session key or obtain any identity information without the help of private keys $\{d_U, d_{SU}, d_{ST}, d_T\}$. Therefore, our scheme can resist man-in-the-middle-attack.

Resist Denial-of-Service Attack

This attack occurs when an attacker transmits a huge number of request messages to either U, S or T. In our proposed protocol, each message exchanged contains a timestamp. The legitimacy of each message is also checked by calculating V_n. Thus, any timed out or unauthorized message gets detected and is rejected. As a consequence, DoS attack can be well defended by our protocol.

6 Performance Evaluation

In this section, we present the performance evaluation of our scheme and compare our work with relevant schemes.

6.1 Security Features

Table 3 presents the security analysis results. According to the threat model stated in Sect. 1, the analyzed security attributes include the basic functional features such as mutual authentication, user anonymity and forward secrecy, etc. Resistance of usual like impersonation attack, insider attack and replay attack are also evaluated.

6.2 Computational Overhead

Table 4 shows the computational overhead of our scheme. Let the time to compute one hash operation, one bio hash operation, one ECC point multiplication operation be T_h, T_{h_B} and T_E, respectively. Numbers of operations done by 3 participants are given in Table 4. During calculation, we neglect the computational overhead of some lightweight operations such as xor, concatenation, comparison.

As is shown in Table 3, our proposed protocol provides registration, login, authentication and key agreement function with little computational resource. In SRVC system, the smart device, the car enterprise server and the smart terminal of car have relatively strong computing power. Therefore, our protocol works smoothly on SRVC system.

Table 3. Security features comparison. Y = Yes, N = No

Security properties	Panda et al. [13]	Mo et al. [12]	Chatterjee et al. [11]	Our scheme
Mutual authentication	Y	Y	Y	Y
Session key agreement	Y	Y	Y	Y
Forward secrecy	Y	Y	N	Y
Known key security	N	N	Y	Y
User anonymity	N	Y	N	Y
Resist user tracking Attack	N	Y	Y	Y
Resist user/server impersonation Aattack	N	N	Y	Y
Resist privileged insider attack	N	Y	Y	Y
Resist replay attack	Y	N	Y	Y
Resist lost-smart-card attack	N	N	Y	Y
Resist man-in-the-middle attack	Y	N	Y	Y
Resist denial of service attack	N	N	N	Y

Table 4. The computational overhead of our scheme

Phase	The user	The server	The terminal
Registration	$2T_h + 1T_{h_B}$	$5T_h$	$1T_h$
Login	$3T_h + 1T_{h_B} + 1T_E$	/	/
Auth and key agreement	$7T_h + 1T_E$	$14T_h + 4T_E$	$16T_h + 2T_E$
Total	$12T_h + 2T_{h_B} + 2T_E$	$19T_h + 4T_E$	$17T_h + 2T_E$

7 Conclusion and Future Work

In this paper, we first introduce the smart remote vehicle control system's features, which are similar but also has differences with IoT systems. Then based on the Delov-Yao adversary model, we analyze Chatterjee et al.'s three-way ECC-based IoT authentication protocol, and demonstrate the limitations in this protocol such as lack resistance of DoS attack, impersonation attack and user tracking attack. To give a more security for SRVC system, then we propose a Privacy-Preserving ECC-based three-factor authentication for SRVC system. Security and performance analysis show that our protocol can resist known attacks and has better effectiveness. Finally, the performance analysis shows that our protocol can run efficiently under the system.

In the future, we will study how to extend our protocol to more application scenarios of SRVC system, such as vehicle cancellation, vehicle transaction, temporary vehicle key authorization, etc.

Acknowledgments. This research was funded by the National Key Research and Development Program of China (No. 2018YFB0803600).

References

1. Bono, S., Green, M., Stubblefield, A., Juels, A., Rubin, A.D., Szydlo, M.: Security analysis of a cryptographically-enabled RFID device. In: USENIX Security Symposium, vol. 31, pp. 1–16 (2005)
2. Francillon, A., Danev, B., Capkun, S.: Relay attacks on passive keyless entry and start systems in modern cars. In: Proceedings of the Network and Distributed System Security Symposium, Eidgenössische Technische Hochschule Zürich, Department of Computer Science (2011)
3. Lamport, L.: Password authentication with insecure communication. Commun. ACM **24**(11), 770–772 (1981)
4. Kumari, S., Li, X., Wu, F., Das, A.K., Odelu, V., Khan, M.K.: A User anonymous mutual authentication protocol. KSII Trans. Internet Inf. Syst. **10**(9), 4508–4528 (2016)
5. Chen, T.H., Yeh, H.L., Shih, W.K.: An advanced ECC dynamic id-based remote mutual authentication scheme for cloud computing. In: 2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering, pp. 155–159. IEEE (2011)
6. Alotaibi, M.: An enhanced symmetric cryptosystem and biometric-based anonymous user authentication and session key establishment scheme for WSN. IEEE Access **6**, 70072–70087 (2018)
7. Porambage, P., Schmitt, C., Kumar, P., Gurtov, A., Ylianttila, M.: Two-phase authentication protocol for wireless sensor networks in distributed IoT applications. In: 2014 IEEE Wireless Communications and Networking Conference, pp. 2728–2733. IEEE (2014)
8. Chu, F., Zhang, R., Ni, R., Dai, W.: An improved identity authentication scheme for internet of things in heterogeneous networking environments. In: 2013 16th International Conference on Network-Based Information Systems, pp. 589–593. IEEE (2013)
9. Lai, C., Li, H., Lu, R., Shen, X.S.: SE-AKA: a secure and efficient group authentication and key agreement protocol for LTE networks. Comput. Netw. **57**(17), 3492–3510 (2013)
10. Lai, C., Li, H., Lu, R., Jiang, R., Shen, X.: SEGR: a secure and efficient group roaming scheme for machine to machine communications between 3GPP and WiMAX networks. In: 2014 IEEE International Conference on Communications, pp. 1011–1016. IEEE (2014)
11. Chatterjee, S., Samaddar, S.G.: A robust lightweight ECC-based three-way authentication scheme for IoT in cloud. In: Elçi, A., Sa, P.K., Modi, C.N., Olague, G., Sahoo, M.N., Bakshi, S. (eds.) Smart Computing Paradigms: New Progresses and Challenges. AISC, vol. 767, pp. 101–111. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-9680-9_7
12. Mo, J., Hu, Z., Chen, H., Shen, W.: An efficient and provably secure anonymous user authentication and key agreement for mobile cloud computing. Wirel. Commun. Mob. Comput. (2019)
13. Panda, P.K., Chattopadhyay, S.: A secure mutual authentication protocol for IoT environment. J. Reliable Intell. Environ. **6**(2), 79–94 (2020). https://doi.org/10.1007/s40860-020-00098-y

14. Zhou, L., Li, X., Yeh, K.H., Su, C., Chiu, W.: Lightweight IoT-based authentication scheme in cloud computing circumstance. Future Gener. Comput. Syst. **91**, 244–251 (2019)
15. Truong, T.T., Tran, M.T., Duong, A.D.: Modified dynamic ID-based user authentication scheme resisting smart-card-theft attack. Appl. Math. Inf. Sci. **8**(3), 967 (2014)
16. Lee, Y.C.: A new dynamic id-based user authentication scheme to resist smart card theft attack. Appl. Math. Inf. Sci. **6**, 355–361 (2012)
17. Feng, Q., He, D., Zeadally, S., Wang, H.: Anonymous biometrics-based authentication scheme with key distribution for mobile multi-server environment. Future Gener. Comput. Syst. **84**, 239–251 (2018)
18. Aghili, S.F., Mala, H., Shojafar, M., Peris-Lopez, P.: LACO: lightweight three-factor authentication, access control and ownership transfer scheme for e-health systems in IoT. Future Gener. Comput. Syst. **96**, 410–424 (2019)
19. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)

Secure Computation

Efficient and Private Divisible Double Auction in Trusted Execution Environment

Bingyu Liu, Shangyu Xie, and Yuan Hong(✉)

Department of Computer Science, Illinois Institute of Technology, Chicago, USA
{bliu40,sxie14}@hawk.iit.edu, yuan.hong@iit.edu

Abstract. Auction mechanisms for exchanging divisible resources (e.g., electricity, cloud resources, and network bandwidth) among distributed agents have been extensively studied. In particular, divisible double auction allows both buyers and sellers to dynamically submit their prices until convergence. However, severe privacy concerns may arise in the double auctions since all the agents may have to disclose their sensitive data such as the bid profiles (i.e., bid amounts and prices in different iterations) to other agents for resource allocation. To address such concerns, we propose an efficient and private auction system ETA by co-designing the divisible double auction mechanism with the Intel SGX, which executes the computation for auction while ensuring confidentiality and integrity for the buyers/sellers' sensitive data. Furthermore, ETA seals the bid profiles to achieve a Progressive Second Price (PSP) auction for optimally allocating divisible resources while ensuring truthfulness with a Nash Equilibrium. Finally, we conduct experiments to validate the performance of private auction system ETA.

Keywords: Secure computation · Auction mechanism design · TEE

1 Introduction

In the past decade, divisible resources (e.g., electricity, computation and storage resources in the cloud, stock shares, and network bandwidth) have been frequently exchanged or allocated in a peer-to-peer mode. All the buyers and sellers can trade any amount of the resources in such markets. In such markets, all the buyers/sellers generally compete with each other to maximize their payoffs. Then, divisible double auction mechanisms [48] have been extensively studied to allow both buyers and sellers to dynamically submit their prices until convergence. For instance, all the participants will converge to achieve a Nash Equilibrium (NE) [23,31] in a game.

However, severe privacy concerns may arise in double auctions [4]. For instance, if the bid profiles (i.e., amounts and prices) are disclosed, rival agents may be able to win more payoffs by reporting untruthful bids. Such violation of truthfulness would explicitly deviate the market of exchanging divisible resources

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 75–92, 2021.
https://doi.org/10.1007/978-3-030-80851-8_6

[27]. Even worse, agents (aka. buyers or sellers) may collect such information from their competitors, and misuse such private data, e.g., reselling the data [4]. Thus, it is desirable to explore a divisible double auction mechanism while preserving all the agents' privacy (e.g., sealing all the bid profiles).

To this end, we propose an efficient and truthful auction system `ETA` by co-designing the divisible double auction mechanism with the Intel SGX, which is a Trusted Execution Environment (TEE) supported by an architecture extension of Intel [16]. Then, `ETA` is designed in two folds. On one hand, the divisible double auction mechanism will be designed based on the Progressive Second Price (PSP) [28] auction, which is extended from the Vickrey-Clarke-Groves (VCG) [38] auction and ensures truthfulness for all the buyers/sellers. No buyers or sellers can gain any additional payoff by changing their strategies since the best responses result in an equilibrium. On the other hand, Intel SGX allows applications to execute within a protected environment called an *enclave* [37] that ensures the confidentiality and integrity for all the buyers and sellers' sensitive data. In summary, we make the following major contributions in this paper:

- We propose an efficient and private auction mechanism `ETA` that executes truthful divisible double auction without disclosing private information.
- The auction mechanism (based on the PSP auction) in `ETA` ensures Individual Rationality, Incentive Compatibility and Pareto Efficiency. The privacy of all the buyers and sellers (i.e., bid profiles, and valuation function) can also be guaranteed in `ETA`. We formally analyze such properties for `ETA`.
- We design and implement the system prototype for `ETA` with the Intel SGX, and conduct experiments to validate the performance using real datasets.

2 Divisible Double Auction

In the divisible double auction, we denote the sets of buyers and sellers as $\mathcal{B}$ and $\mathcal{S}$, respectively. b_m and s_n are defined as two-dimensional bid profiles (bid price, and the maximum amount to buy or sell) as follows: (1) buyer $m \in \mathcal{B}$: $b_m = (\alpha_m, d_m)$ with bid price α_m and amount d_m to buy, and (2) seller $n \in \mathcal{S}$: $s_n = (\beta_n, h_n)$ with bid price β_n and amount h_n to sell. Each buyer or seller is required to submit bid b_m or s_n before the auction starts. $b = (b_m, m \in \mathcal{B})$ represents the bid profiles of all the buyers while $s = (s_n, n \in \mathcal{S})$ denotes the bid profiles of all the sellers. $r = (b, s)$ is defined as the bid profiles for all the agents. These are private information, supposed to be sealed among all the participants.

2.1 Models

We represent the strategy of each agent with a non-negative valuation function $\widehat{V}_m(\cdot)$ for buyers and negative cost function $\widehat{C}_n(\cdot)$ for sellers. These functions are also considered as private information, which quantifies the value or cost of the resources by each buyer or seller.

In the mechanism design, we adopt generic assumptions [28,38] for the valuation function $\widehat{V}_m(\cdot)$: (1) $\widehat{V}_m$ is differentiable and $\widehat{V}_m(0) = 0$, and (2) $\widehat{V}'_m(\cdot)$ is non-increasing and continuous. A_m and A_n represent the allocation of buyer m and seller n, respectively. In the k-th iteration of the double auction, $A_m^{(k)}, A_n^{(k)}$ represent the allocation of buyer m and seller n, respectively. For all agents, we assume that $\widehat{V}_m(A_m^k) > \widehat{V}_m(A_m^{k+1})$ where $\forall m \in \mathcal{B}$ and $A_m^k < A_m^{k+1}$ while $\widehat{C}_n(A_n^k) < \widehat{C}_n(A_n^{k+1})$ where $\forall n \in \mathcal{S}$ and $A_n^k < A_n^{k+1}$, since buyers have diminishing marginal utility while sellers have increasing marginal cost.

We also denote the payoff function for buyer m and seller n as $f_m(r)$ and $f_n(r)$, respectively, for representing their payoffs w.r.t. the bid profiles of all the buyers/sellers r. In addition, ρ_m is the payment made by buyer m while ρ_n is the payment received by seller n. Also, $\rho(r_i, r_{-i})$ is defined as the difference between all the buyers' aggregated valuation if any buyer i is not in the auction minus the aggregated valuation if i is in the auction [25,28,48]. Similarly, $\rho(r_j, r_{-j})$ is defined as the difference between all the sellers' aggregated cost if any seller j is not in the auction minus the aggregated cost if j is in the auction. Thus, we have:

$$\rho(r_i, r_{-i}) = \sum_{m \neq i} \alpha_m [A_m(0; r_{-i}) - A_m(r_i; r_{-i})]$$

$$\rho(r_j, r_{-j}) = \sum_{n \neq j} \beta_n [A_n(0; r_{-j}) - A_n(r_j; r_{-j})] \tag{1}$$

Then, given the optimal allocation profile for buyer $m \in \mathcal{B}$ and seller $n \in \mathcal{S}$ as A_m^* and A_n^*, we can define the payoff of the buyer m and seller n as:

$$f_m(r) = \widehat{V}_m(A_m^*) - \rho(r_i, r_{-i}), \forall m \in \mathcal{B}$$

$$f_n(r) = \rho(r_j, r_{-j}) - \widehat{C}_n(A_n^*), \forall n \in \mathcal{S} \tag{2}$$

Definition 1 (Incentive Compatibility). *The divisible double auction is incentive compatible [25] if the following conditions hold:*

$$\sum_{m \in \mathcal{B}} \widehat{V}_m(r) - \rho_m \geq \sum_{m \notin \mathcal{B}} \widehat{V}_m(r) - \rho(r_i, r_{-i}), \forall m \in \mathcal{B}$$

$$\rho_n - \sum_{n \in \mathcal{S}} \widehat{C}_n(r) \geq \rho(r_j, r_{-j}) - \sum_{n \notin \mathcal{S}} \widehat{C}_n(r), \forall n \in \mathcal{S} \tag{3}$$

Incentive compatibility guarantees that every buyer/seller cannot make better payoff with untruthful bid if other buyers/sellers report the true bids.

Definition 2 (Feasible). *The divisible double auction mechanism is feasible, we have:*

$$\sum_{\forall m \in \mathcal{B}} d_m \leq \sum_{\forall n \in \mathcal{S}} h_n \tag{4}$$

This mechanism is *feasible* requires that the amount of resources that sold by sellers is weakly larger than the amount that buyers intend to purchase. In other words, it ensures that the overall supply satisfies the demand.

Definition 3 (Clearing Price). *If there exists a feasible and efficient allocation, such that, $A^*(\cdot)$ at the price θ, the social welfare (defined as $F(\cdot) = \sum_{m \in \mathcal{B}} V_m(A_m) - \sum_{n \in \mathcal{S}} C_n(A_n)$) is maximized to achieve the best response. Then, price θ is defined as the* clearing price*.*

We say that the clearing price θ [5] supports the optimal allocation $A^*(\cdot)$ with the maximum social welfare.

Definition 4 (Nash Equilibrium). *Given the bid profiles r^*, a Nash Equilibrium (NE) holds for double auction such that:*

$$f_m(b_m^*, r_{-m}^*) \geq f_m(b_m, r_{-m}^*), \forall m \in \mathcal{B}$$

$$f_n(s_n^*, r_{-n}^*) \geq f_n(s_n, r_{-n}^*), \forall n \in \mathcal{S} \tag{5}$$

where $r_{-m} = r \setminus b_m$ is a bid profile for all the buyers except buyer m and $r_{-n} = r \setminus s_n$ is a bid profile for all the sellers except seller n.

ETA aims at achieving the best response (approximate allocation efficiency) to maximize social welfare of allocation to all the buyers and sellers. It also guarantees that the truthfulness of bid profiles is the best response for all the agents. More importantly, each buyer or seller's submitted bid profiles (amounts, prices) and its valuation/cost function are protected in ETA.

2.2 Auction Properties

Recall that ETA will be designed based on the Progressive Second Price (PSP) [28] auction. Thus, it has the following properties.

- ***Weakly dominant strategy*** [28]. Each buyer/seller truthfully participates in the auction would gain more payoff than the untruthful response. The auction mechanism in ETA pursues weakly dominant strategies since it is extended from the PSP auction mechanism (second price).
- ***Budget balanced***. We assume "no budget deficit", the total budget of the buyers is no less than the total payment requested by the sellers: $\sum_{\forall m \in \mathcal{B}}(\alpha_m \cdot d_m) \geq \sum_{\forall n \in \mathcal{S}}(\beta_n \cdot h_n)$.
- ***Individual rationality***. All the buyers and sellers will have non-negative payoffs in the auction.
- ***Pareto efficiency*** [9,43]. The divisible resources are supposed to be sold to buyers with the highest valuation.
- ***Privacy***. Buyers/sellers' bid profiles (bid prices and amounts) and valuation/cost functions are kept private; every pair of potential buyer and seller only know their transaction amount and the clearing price.

3 Mechanism Design

In this section, we design the divisible double auction mechanism and its program Prog_x in Intel SGX. The program ensures that all the bid profiles achieve the best response in multiple iterations for the Nash Equilibrium and eventually converge with a termination condition.

Initialization. While executing Prog_x, the bid profiles of all the buyers and sellers will be checked to guarantee that $(\alpha_i)_{\max} \geq (\beta_j)_{\min}$. Otherwise, it requests them to update the bid profiles. Meanwhile, it ensures that the potential amount of the resources in the auction C is smaller than the overall demand and supply. The auction will start once the above conditions are satisfied.

```
**Divisible Double Auction** $\mathrm{Prog}_x(\mathcal{B}, \mathcal{S}, r)$
**Input:** $\forall m \in \mathcal{B}, \forall n \in \mathcal{S}, r = (b, s)$
**Output:** $b^*_m, s^*_n, \forall m \in \mathcal{B}, \forall n \in \mathcal{S}$
set iteration $k := 1$
**initialize** $(\alpha_i)_{\max} \geq (\beta_j)_{\min}$ and $C < \min\{\sum_{i \in B} d_i, \sum_{j \in S} h_j\}$
**while** *true* **do**
    $A^*_m(b, C) := \min\{d_m, \{[C - \sum_{i \in \mathcal{B}_m(b)} d_i], 0\}_{\max}\}$
    $A^*_n(s, C) := \min\{h_n, \{[C - \sum_{i \in \mathcal{S}_n(s)} h_j], 0\}_{\max}\}$
    $Q(r, C) := \min\{\sum_{i \in \mathcal{B}} A^*_i(r, C), \sum_{j \in \mathcal{S}} A^*_j(r, C)\}$
    $\widehat{\mathcal{P}} := \frac{p_b(r, C) - p_s(r, C)}{\omega_{\max} + \sigma_{\max}}$
    $\widetilde{C}(r, C) := Q(r, C) + \widehat{\mathcal{P}}$
    $b^*_m = \arg\max\{f_m(b_m, b_{-m})\}, m \in \mathcal{B}$
    $s^*_n = \arg\max\{f_n(s_n, s_{-n})\}, n \in \mathcal{S}$
    **terminate** until convergence
    iteration $k := k + 1$
**endwhile**
```

Fig. 1. Divisible double auction

Iteration. Given the total amount for the auction (potential amount) C, it will be updated as below:

$$\widetilde{C}(r, C) = Q(r, C) + \frac{p_b(r, C) - p_s(r, C)}{\omega_{\max} + \sigma_{\max}} \tag{6}$$

where $Q(r, c) = \min\{\sum_{m \in \mathcal{B}} A^*_m, \sum_{n \in \mathcal{S}} A^*_n\}$, $p_b(r, C) = \min\{\alpha_i, A_i \geq 0\}$ and $p_s(r, C) = \max\{\beta_j, A_j \geq 0\}$. And $\widehat{\mathcal{P}} = \frac{p_b(r,C) - p_s(r,C)}{\omega_{\max} + \sigma_{\max}}$. Specifically, $Q(r, c)$ is the smaller one of the total demand and total supply; $\widehat{\mathcal{P}}$ is a coefficient for the gradients of marginal valuations (costs); $p_b(r, C)$ and $p_s(r, C)$ are two variables defined to help converge much faster in iterations via updating the potential amount; $\omega_{\max}$ is an upper bound for buyers' valuations $\omega_{\max} \geq \max \sup_{A_m} \{|\widehat{V}_m^{'}(A_m)|\}$ while $\sigma_{\max}$ is a upper bound for sellers' costs $\sigma_{\max} \geq \max \sup_{A_n} \{|\widehat{C}_n^{'}(A_n)|\}$. Using the gradients of marginal valuations/costs, the potential amount can efficiently reach a Nash Equilibrium.

In each iteration, A_m^* and A_n^* are the optimal allocation of buyers and sellers, respectively. Each agent updates its best response. Given (r, C), we derive the optimal allocation for each buyer $A_m^* = \min\{d_m, \{[C - \sum\limits_{i \in B_m(b)} d_i], 0\}_{\max}\}$ and seller $A_n^* = \min\{h_n, \{0, [C - \sum\limits_{j \in \mathcal{S}_n(s)} h_j]\}_{\max}\}$, where $B_m(b) = \{i \in \mathcal{B} | \alpha_i > \alpha_m\} \cup \{\alpha_i = \alpha_n \text{ and } i < m\}$ and $S_n(s) = \{j \in \mathcal{S} | \beta_j > \beta_n\} \cup \{\beta_i = \beta_m \text{ and } j < n\}$. Given the potential amount C obtained from initialization, the updated potential amount $\widetilde{C}(r, C)$ can be iteratively derived.

Best Response. We denote the best response of any buyer m as b_m^* and any seller n as and s_n^*. After updating the potential amount, we have the bid profiles $r = (b, s)$ and a pair of potential amount $(C, \widehat{C})$. Then, we define the best response as follows.

$$b_m^*(r, C, \widehat{C}) = \arg\max\{f_m(b_m, b_{-m})\}$$
$$s_n^*(r, C, \widehat{C}) = \arg\max\{f_n(s_n, s_{-n})\} \tag{7}$$

The auction program $\mathsf{Prog_x}$ will find the best responses in each iteration and finally converges to a Nash Equilibrium. Note that the valuation and cost functions might be different. In this dynamic auction game, all the buyers/sellers recompute their best response to the current strategy (bid profiles) of other agents. We now study the game of the divisible double auction mechanism as follows (all of them are proven in the Appendix).

Theorem 1. *The divisible double auction in* ETA *ensures (1) Weakly dominant strategy, (2) Individual rationality, (3) Pareto efficiency, (4) Incentive compatibility, and (5) Feasibility.*

4 ETA System Design

In this section, we design the ETA system for deploying divisible double auction with the Intel SGX [16], which guarantees the confidentiality and integrity for all the computation.

4.1 SGX Formalization

Intel SGX provides a solution to run programs in a secure container, which is referred as an *enclave*, on an untrusted OS. Sensitive data and codes within the protected memory regions can be isolated by the *enclave*, and can be protected against powerful attackers (e.g., controlling the OS). We use relay R to represent the physical SGX host, which is the interface of the *enclave*. Other components cannot directly access to the *enclave*, unless it relies on R.

Our program Prog_x (shown in Fig. 1) will be executed in the *enclave*, which is trusted by all the buyers and sellers. In this paper, we design the ETA system with the formalization of Intel SGX in [33].

```
SGX Functions $\mathcal{F}_{SGX}[\mathsf{Prog}_\mathsf{x}, R]$
Initialize() :
  upon receiving (Init) from $R$ :
    output := $\mathsf{Prog}_\mathsf{x}$.initialize()
    / generate an output with attestation
    $\psi_{sgx} := \sum_{sgx} \cdot \mathsf{Sig}(\mathsf{sk}_{sgx}, (\mathsf{Prog}_\mathsf{x}, \mathsf{output}))$
    return (output, $\psi_{sgx}$)
Resume() :
  upon receiving Auth(meg) from $R$ :
    output := $\mathsf{Prog}_\mathsf{x}$.resume()
    return output
```

Fig. 2. SGX functions

In order to model the ideal functionality channel with some proprieties such as confidentiality and authenticity, we use a global universal composability (UC) functionality [6] to instantiate the SGX Functions as $\mathcal{F}_{SGX}(\sum_{sgx})[\mathsf{Prog}_\mathsf{x}, R]$ parameterized by a group signature scheme $\sum_{sgx}$. Our program Prog_x is loaded into *enclave* via the "init" call. When it calls "resume", the program is executed based on the given incoming requests (or inputs, denoted as inp), and generates the output with an attestation $\psi_{att} := \sum_{sgx} \cdot \mathsf{Sig}(\mathsf{sk}_{sgx}, (\mathsf{Prog}_\mathsf{x}, \mathsf{output}))$. The signature under TEE hardware key sk_{sgx} and pk_{sgx} could be obtained from the SGX Functions ($\mathcal{F}_{SGX}$). The details are given in Fig. 2.

4.2 ETA System

As shown in Fig. 3, there are four main components in ETA: (1) **Enclave** is used for protecting program codes and data, which guarantees the confidentiality and integrity of computations. Remote attestation allows remote users to establish encrypted and authenticated channels to *enclave*; (2) **Relay** R is the interface of Intel SGX. It can be used as a physical SGX host, and other components cannot directly access to the *enclave* without connecting with the R; (3) **Key Management Committee** generates key pairs (pk, sk) for buyers/sellers' input

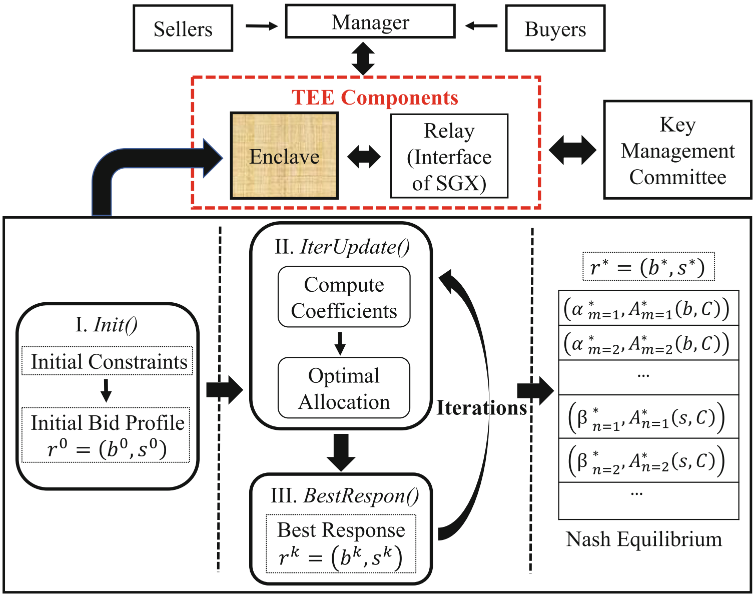


Fig. 3. ETA system (Intel SGX-based)

encryption (encrypted by the public key pk). Private key sk is used inside the TEE components for decryption and preparing for the further computation; (4) **All the Agents** ($\forall m \in \mathcal{B}$, $\forall n \in \mathcal{S}$ and manager $\mathcal{P}$). All the agents participates in the auction while the manager handles all the incoming requests and delivers results as the administrator. Also, TEE components will be triggered by the manager.

Note that the manager is not a trusted third party (TTP). All the all the buyers/sellers' inputs loaded via a secure channel are not visible to the manager. Moreover, it can even deviate arbitrarily from program or collude with other agents. However, manager cannot affect the correct computation/execution of the program and will be penalized for program interrupts or aborts.

4.3 ETA Execution Program

ETA is executed in 3 phases: (1) Setup, (2) Compute, and (3) Delivery, as illustrated in Fig. 4. Assume that secure channels are established between the TEE components and all the buyers and sellers.

(1) Setup. The TEE initializes the system with the "init" call and prepares for loading our auction program $\mathsf{Prog_x}$. Once it receives the "init" request, the key pair $(\mathsf{pk}_{sgx}, \mathsf{sk}_{sgx})$ is created, and then pk_{sgx} which is bound to the TEE code (Program) by the initial attestation will be distributed. Meanwhile, the TEE will obtain the key pairs $(\mathsf{pk}, \mathsf{sk})$ from the Key Management Committee and distribute the public key pk to buyers and sellers for encrypting their inputs.

```
**ETA Execution Program** $(\mathcal{B}, \mathcal{S}, R, E)$
$InitEnclave()$ :
    receive($\mathsf{init}(), request$) to load $\mathsf{Prog}_\mathsf{x}$ inside $E$
    boost *enclave* with $\mathsf{Prog}_x$.**initialize()**
    call $\mathsf{Prog}_x$.**resume()** to handle($\mathsf{Enc}_\mathsf{pk}(\mathsf{inp}), l_{id}$)
    distribute $(\mathsf{pk}, \psi_{sgx})$ for attestation
    / generate key pairs for inputs encryption
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}(1^n)$
$InitAgents()$ :
    upon receiving $\mathsf{init}()$ to obtain $\mathsf{pk}$ for encryption
    buyers/sellers make deposits $\xi_{b_m}$ or $\xi_{s_n}$
    $\mathcal{P}$ make deposits $\xi_\mathcal{P}$
$Compute()$ :
    boost *enclave* with $\mathsf{Prog}_x$.**resume()**
    decrypt $(\mathsf{Enc}_\mathsf{pk}(\mathsf{inp}), l_{id})$ with $\mathsf{sk}$
    execute and compute $\mathsf{Prog}_x$
$Delivery()$ :
    fetch($\mathsf{output}, \psi_{sgx}, l_{id}$)
    forward $\mathsf{output}$ to the honest buyers/sellers by $\mathcal{P}$
```

Fig. 4. ETA execution program

Next, all the buyers and sellers encrypt their input data (denoted as inp overall) with pk, prepare the deposits ξ_{b_m} or ξ_{s_n}) and send meg $:=$ ($\mathsf{Enc}_\mathsf{pk}(\mathsf{inp})$, l_{id}, ξ_{b_m}, ξ_{s_n}) to manager $\mathcal{P}$ where inp denotes the inputs of all the buyers/sellers. Note that l_{id} represents a fresh and unique identifier (ID). In practice, it will use 128-bit MAC of the AES-GCM encryptions as ID.

Note that all the buyers and sellers send the messages through secure authenticated channels. The manager is responsible for batching transactions of all the inputs from all the buyers and sellers. Also, validation of messages will be checked by the manager. If it is valid, manager will forward all the requests to the TEE Components. Otherwise, once malicious behaviors are detected, the deposits will be stored into the TEE for next computation instead of refund.

(2) Compute. TEE components handle the incoming requests (inputs) in parallel. TEE components retrieve the program Prog_x with the "resume" call and decrypt the inputs with the private key sk. The execution of the auction program Prog_x is conducted in a sandboxed environment (*enclave*). Then, a software adversary and/or a physical adversary cannot interrupt the execution or inspect (monitor) data that lives inside the *enclave*. The result of the Prog_x is a secret output which will be securely returned to the manager with a black-box program execution.

(3) Delivery. Once the execution has been completed inside the TEE, the output (of all the buyers/sellers in the divisible double auction) is generated and signature is provided to prove the computation correctness. As a result, the output message (output, $\psi_{sgx}, l_{id}, \xi_{b_m}, \xi_{s_n}$) will be delivered to the corresponding honest buyers/sellers by the manager.

4.4 Threat Model

To ensure confidentiality and integrity for computation, we use the TEE's attestation [44]. In particular, the computation is executed correctly inside the Intel SGX (all the agents trust the *enclave*). However, the remaining software stack outside the *enclave* and the hardware is not trusted. The adversary may corrupt any number of agents, assuming that honest agents will trust their own codes and platform (leakage resulted from its software bugs are out of the scope).

Furthermore, we assume that multiple agents do not trust each other mutually. They can be potentially malicious such as stealing the bid profiles information and modifying the execution flow. During the execution, each buyer or seller may send, drop, modify and record arbitrary messages. Even worse, any buyer or seller may crash and stop responding. Note that the side-channel attacks against *enclave* and DoS attacks are not considered in this paper.

4.5 Security Analysis

Enclave Isolation. Intel SGX enables the program (data) to be executed inside the secure container (*enclave*) for confidentiality and integrity. The adversary cannot interrupt the computation executed in a sandboxed environment (*enclave*). Note that *enclave* is created in its virtual address space by an untrusted hosting application with OS support. Once *enclave* starts initialization, data and codes inside it will be isolated from the rest of the system and secured.

Also, the encrypted data are sent from buyers/sellers to *enclave* through secure channels. However, other malicious servers cannot eavesdrop on the encrypted data, and cannot interrupt the communication.

Data Sealing. Intel SGX reads the encrypted data as inputs and decrypts the data with its private key inside the *enclave*. Once the computation is completed within the *enclave*, the results output will be encrypted again before distribution. Due to the data sealing, the vulnerability of data leakage can be addressed.

Malicious Manager. The manager handles all the incoming requests and delivers the results as the administrator. If the untrusted manager interrupts the delivery rule, delays or tampers with the communication among all the components, he/she will be punished by losing all the initial deposits.

5 Experiments

In this section, we evaluate the performance for ETA in Graphene[1] on the Microsoft Azure.[2] Specifically, considering energy trading as an experimental application, we conduct experiments on real power consumption and generation data (available at the UMASS Trace Repository [2]) which can be used for the application of energy trading in a peer-to-peer mode amongst up to 200 agents, each of which is either a buyer or seller in the divisible double auction.

The auction mechanisms have been designed for many different divisible resources, such as electricity [18,39], cloud resources [13,22], and wireless spectrum [24]. Note that different valuation/cost functions are defined in different applications. We take energy trading application [17,41] as an example. Entities on the power grid may trade their excessive locally generated energy, e.g., the renewable energy resources [1,11]. Since electricity is divisible, ETA can establish a privacy preserving divisible double auction for energy trading. The valuation/-cost functions are defined as $V_m(x_m) = \zeta_m \log(x_m+1)$ and $C_n(y_n) = a_n y_n^2 + b_n y_n$ [3], where ζ_m is a parameter determined by the behavior preference of buyer, a_n and b_n are the parameters for measuring how much the sellers incline to sell. The valuation/cost functions follow the general assumption in Sect. 2. Finally, ETA only outputs the *clearing price* for the auction to all the agents, and each pair of buyer and seller only receive the amount traded between them.

We conduct experiments using the energy trading application [41], and set $\zeta_m = 50$, $a_n = 30$ and $b_n = 0$ in the valuation function $V_m(x_m) = \zeta_m \log(x_m+1)$ and cost function $C_n(y_n)$=$a_n y_n^2 + b_n y_n$ (adopting the same parameters as [48]).

5.1 ETA Performance Evaluation

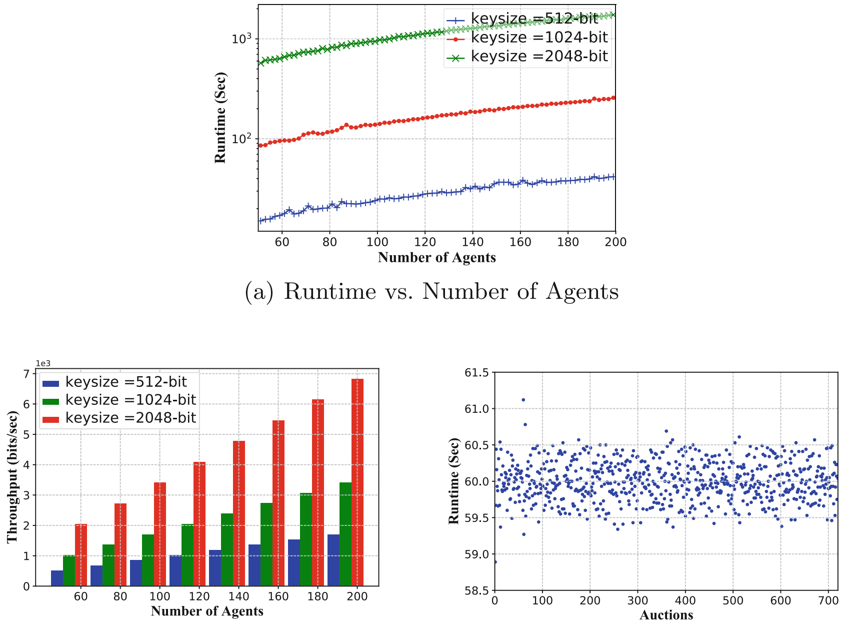
We evaluate the system performance (up to 200 agents) for ETA and demonstrate the results for both auction performance and system efficiency.

Figure 5(a) shows the total runtime on a varying number of agents (from 50 to 200) in an auction. The auction includes both secure computation for the optimal allocation and the peer-to-peer trading in the TEE. It takes up to only 10–15 min for 200 agents even if the 2048-bit key size is adopted for very strong security. Moreover, Fig. 5(b) illustrates the relationship between the number of agents and the throughput (bits/sec). As the number of agents increases, throughput increases linearly as shown in Fig. 5(b).

For evaluating the efficiency in multiple auctions, we use the key size as 1024 bit and 20 agents in ETA for continuously executing 720 auctions. Figure 5(c) shows that the runtime for each auction lies close to 60 s, which would not result in much latency if we execute multiple auctions in real time.

[1] Graphene [36] is a lightweight guest OS, designed for minimal host requirements. Applications can be protected in a hardware-encrypted memory region.

[2] https://azure.microsoft.com/en-us/solutions/confidential-compute/.



(a) Runtime vs. Number of Agents

(b) Throughput vs. Number of Agents

(c) Runtime vs. Auctions (1024-bits)

Fig. 5. ETA performance evaluation

Table 1. Executing ETA for divisible double auction among 12 buyers and 8 sellers; allocation (MWh), potential amount (MWh); social welfare ($)

Criteria	Iterations				
Allocation $(A^*_{m=1}(b,C))$	0.23	0.24	0.24	0.44	0.57
Allocation $(A^*_{m=2}(b,C))$	0.16	0.55	0.57	0.57	0.59
Allocation $(A^*_{m=3}(b,C))$	0.14	0.14	0.24	0.55	0.55
Allocation $(A^*_{n=1}(s,C))$	0.10	0.55	0.74	0.90	0.90
Allocation $(A^*_{n=2}(s,C))$	0.41	0.43	0.64	0.64	0.90
Potential amount (C)	4.38	4.37	2.64	2.64	2.64
Social welfare $(F(\cdot))$	62.9	68.2	70.7	74.0	74.0

5.2 Case Study

Furthermore, we perform a case study by applying ETA for energy trading, and present more detailed results in a divisible double auction. Specifically, 20 agents include 12 buyers and 8 sellers. We will present the fine-grained results in multiple iterations of the auction with respect to the following criteria: (1) allocation, (2) potential amount, and (3) social welfare.

Table 1 shows the detailed results. First, it presents that 5 groups of the allocation for both buyers and sellers ($A^*_m(b,C)$ or $A^*_n(s,C)$), which are generated from multiple iterations (finally converged to the Nash Equilibrium). Due to space limit, we illustrate the results of 5 representative agents (3 buyers and 2 sellers). The allocation of all the participants will increase, and finally they all converge to the final allocated amounts after several iterations. Second, potential amount (C) decreases until convergence. 2.64 MWh will be the total selling/buying amount in the auction. Third, the social welfare ($F(\cdot)$) is derived based on the equation $F(\cdot) = \sum_{m \in \mathcal{B}} V_m(A_m) - \sum_{n \in \mathcal{S}} C_n(A_n)$. It has an increasing trend in multiple iterations and converges to the maximized social welfare \$74.

6 Related Work

Auction mechanisms for divisible resources were widely studied for spectrum allocation [10,40,42]. Spectrum allocation problem was discussed in cognitive radio networks with the combinatorial auctions [10]. Wu and Vaidya [40] modeled the radio spectrum allocation problem as a sealed-bid reserve auction, and investigated strategy-proof mechanism for multi-radio spectrum buyers. Hoefer et al. [15] proposed an approximation algorithm (LP formulation) for combinatorial auctions with a conflict graph. Other similar studies with respect to divisible auctions focused on the revenue maximization [21] or social efficiency maximization [10]. Yu et al. [45] proposed the auction model to handle the uncertain demand conditions. In order to promote high efficiency for the auctions, Lorenzo et al. [30] designed the matching game mechanism in the auction based on the game theory.

The privacy concerns have been raised in auctions for divisible resources [7,19]. To address them, cryptographic techniques [29,32,34] were proposed to achieve both privacy preservation and incentive compatibility for the auctions. Huang et al. [20] proposed a cryptographic scheme for one-side strategy-proof auctions. Some other existing works [8,35] apply the homomorphic encryption to address privacy issues in the auctions. Furthermore, some deployed systems can also be utilized to address the privacy concerns for auctions. In [14], the involved third-party auction platform was designed to preserve privacy for all the participants. The HAWK system [26] was deployed as a decentralized smart contract that privately processes transactions with the private and public portions for the sealed-bid auctions.

However, such techniques are not directly applicable to privacy preserving double auction for divisible resources. Besides, they may require high computational overheads due to heavy cryptographic primitives. Instead, our ETA can efficiently perform secure computation for divisible double auction while ensuring truthfulness.

7 Conclusion

We design an efficient and private system ETA which securely execute double auction for allocating divisible resources among distributed agents within the

Intel SGX. The auction mechanism in ETA ensures individual rationality, incentive compatibility and Pareto efficiency. The input data of both buyers and sellers in the auction can also be protected in ETA. The experimental results have demonstrated a high efficiency for ETA to privately execute the divisible double auction. In the future, blockchain-based auction mechanisms will be investigated by executing the secure computation inside the Intel SGX for trading divisible resources among distributed agents with the cryptocurrencies (e.g., bitcoins).

Acknowledgments. This work is partially supported by the National Science Foundation (NSF) under Grant No. CNS-1745894. The authors would like to thank the anonymous reviewers for their constructive comments.

Appendix

Proof of Theorem 1

Proof. (1) Per the payment rule of ETA (extended from the VCG auction) in Eq. 1, buyer $m \in \mathcal{B}$ will change its own strategies based on other buyers' strategies (so does seller $n \in \mathcal{S}$). As defined in Sect. 2.1, ρ_m is the payment made by buyer m while ρ_n is the payment received by seller n. Also, $\rho(r_i, r_{-i})$ is defined as the difference between all the buyers' aggregated valuation if any other buyer i is not in the auction and the aggregated valuation if buyer i is in the auction. Then, $\rho(r_i, r_{-i})$ can be transformed into the difference between two payoff functions:

$$\begin{aligned}\rho(r_i, r_{-i}) &= \sum_{m \neq i} \alpha_m [A_m(0; r_{-i}) - A_m(r_i; r_{-i})] \\ &= \underbrace{(\max \sum_{m \neq i} f_m(r))}_{\text{without m}} - \underbrace{\sum_{m \neq i} f_m(r^*)}_{\text{with m}} \end{aligned} \tag{8}$$

In addition, as defined in Sect. 2.1, the payoff function for buyer m is $\widehat{V}_m(A^*_m(r)) - \rho(r_i, r_{-i})$. The payoff function is supposed to be maximized if there exists the optimal bid profile r^*, including the optimal allocation profiles for buyers and sellers: $A^*_m(r)$ and $A^*_n(r)$. After integrating Eq. 8, we have the payoff function w.r.t. the buyer m as below:

$$\begin{aligned}&\widehat{V}_m(A^*_m(r)) - \rho(r_i, r_{-i}) \\ &= \left[\widehat{V}_m(A^*_m(r^*)) + \sum_{m \neq i} f_m(r^*)\right] - \left[(\max \sum_{m \neq i} f_m(r))\right] \end{aligned} \tag{9}$$

In Eq. 9, the $\left[(\max \sum_{m \neq i} f_m(r))\right]$ is the same for all the buyers ($\forall m \in \mathcal{B}$). Then, the problem of maximizing buyer m's payoff is reduced to the problem of maximizing $\left[\widehat{V}_m(A^*_m(r^*)) + \sum_{m \neq i} f_m(r^*)\right]$. Intuitively, buyer m would choose the strategy to maximize $\left[\widehat{V}_m(A^*_m(r^*)) + \sum_{m \neq i} f_m(r^*)\right]$. Per Definition 4 and incentive compatibility proven in (4), if each agent responds untruthfully, it would not obtain a higher payoff than truthful response. If buyer m bids truthfully and the objective (to maximize) for double auction mechanism becomes identical to the $\left[\widehat{V}_m(A^*_m(r^*)) + \sum_{m \neq i} f_m(r^*)\right]$. The payoff will be maximized if buyer m bids truthfully. Therefore, the truthful responses in the double auction mechanism are the best strategies for all the buyer $\forall m \in \mathcal{B}$.

Similarly, for any seller $n \in \mathcal{S}$, its payoff function $f_n(r) = \rho(r_j, r_{-j}) - \widehat{C}_n(A^*_n)$ as defined in Sect. 2.1 can also be proven in the same way. Thus, the divisible double auction mechanism in ETA ensures weakly dominant strategy.

(2) If buyer $m \in \mathcal{B}$ provides truthful bid profile, then it has a non-negative payoff function $f_m(r) = \widehat{V}_m(A^*_m) - \rho(r_i, r_{-i}), \forall m \in \mathcal{B}$. Similarly, seller $n \in \mathcal{S}$ can also obtain a non-negative payoff function: $f_n(r) = \rho(r_j, r_{-j}) - \widehat{C}_n(A^*_n), \forall n \in \mathcal{S}$ with truthful bid profile. Thus, the double auction satisfies individual rationality, which indicates that all the agents have non-negative payoffs by participating in the double auction of ETA.

(3) Allocation (A_m, ρ_m) satisfies Pareto efficiency within the budget φ_m ($\rho_m < \varphi_m$) in the divisible double auction ETA if there does not exist a better allocation $(A^{'}_m, \rho^{'}_m)$: $f_m(A_m, \rho_m) > f_m(A^{'}_m, p^{'}_m)$. Suppose that buyer $m \in \mathcal{B}$ is allocated with amount A_m in bid profile r (satisfying individual rationality as above and incentive compatibility as proven in (4)). We now prove the Pareto efficiency (optimality). Given $f^*_m = \max_{A_m} f_m(A_m, \rho(A_m, (r_{-m}))$, buyer m's payoff is upper bounded by f^*_m. If m would like to gain more payoff, then it needs to pay $\rho(A_m, (r_m, r_{-m}))$. Thus, the payoff is supposed to be lowered bounded by f^*_m. Thus, buyer m's payoff is exactly f^*_m for the optimality. Similarly, $f^*_n = \max_{A_n} f_n(A_n, \rho(A_n, (r_{-n}))$ can be proven for sellers. Therefore, the Pareto efficiency is verified in ETA.

(4) Denote the allocation of buyer $m \in \mathcal{B}$ as the A_m, and also denote the allocation in the k-th iteration as A^k_m. To show the incentive compatibility for any buyer $m \in \mathcal{B}$, we verify that for any bid profile $b = (b_m, m \in \mathcal{B})$. Given r_{-m}, there exists a truthful bid profile $b_m = (\alpha_m, d^k_m)$ where $\alpha_m = \widehat{V}^{'}_m(d^k_m)$, such that $f_m(b^k_m, r_{-m}) \geq f_m(b_m, r_{-m}), \forall m \in \mathcal{B}$:

- Case 1: if $\alpha_m < \widehat{V}^{'}_m(d_m)$. Consider a bid b^k_m, such that $d^k_m = A_m \leq d_m$. Based on the diminishing marginal utility of the valuation function for buyers, we have $\alpha^k_m \geq \widehat{V}^{'}_m(d_m) > \alpha_m$. Since we get the maximum social welfare, we have $A^k_m \geq A_m$. Thus, we have $f_m(b^k_m, r_{-m}) \geq f_m(b_m, r_{-m}), \forall m \in \mathcal{B}$.
- Case 2: if $\alpha_m > \widehat{V}^{'}_m(d_m)$. Considering bid b^k_m, such that $d^k_m = d_m$, we have $\alpha_m > \widehat{V}^{'}_m(d_m) = \widehat{V}^{'}_m(d^k_m) = \alpha^k_m$. Also, $A^k_m \leq A_m$ holds for the maximum social welfare. When $A^k_m = A_m$, we have $f_m(b^k_m, r_{-m}) = f_m(b_m, r_{-m}), \forall m \in \mathcal{B}$. When $A^k_m < A_m$, we have:

$$\begin{aligned} & f_m(b_m, r_{-m}) - f_m(b_m^k, r_{-m}) \\ = \; & \widehat{V}_m(A_m) - \widehat{V}_m(A_m^k) + \rho(A_m^k, r_{-m}) - \rho(A_m, r_{-m}) \\ \leq \; & \alpha_m^k(A_m - A_m^k) + F(r) - \alpha_m A_m - F(r^k) + \alpha_m^k A_m^k \\ \leq \; & \alpha_m^k(A_m - A_m^k) - \alpha_m^k(A_m - A_m^k) = 0 \end{aligned} \tag{10}$$

Given Case 1 and 2, we have $f_m(b_m^k, r_{-m}) \geq f_m(b_m, r_{-m}), \forall m \in \mathcal{B}$. Similarly, incentive compatibility can also be proven for all the sellers $\forall n \in \mathcal{S}$.

(5) Assuming that $\sum_{m \in \mathcal{B}} d_m \geq \sum_{n \in \mathcal{S}} h_n$ holds for the initialization, then the potential amount for all divisible resources $C = \min\{\sum_{m \in \mathcal{B}} d_m, \sum_{n \in \mathcal{S}} h_n\}$ holds for the iterative computation in the ETA. Thus, we have $\sum_{m \in \mathcal{B}} d_m = \sum_{n \in \mathcal{S}} h_n$. Furthermore, compared with the other case: $\sum d_m \leq h_n$, the divisible double auction mechanism in ETA satisfies the feasibility. In summary, these complete the proof. □

References

1. Aliabadi, D.E., Kaya, M., Şahin, G.: An agent-based simulation of power generation company behavior in electricity markets under different market-clearing mechanisms. Energy Policy **100**, 191–205 (2017)
2. Barker, S., Mishra, A., Irwin, D., Shenoy, P., Albrecht, J.: SmartCap: flattening peak electricity demand in smart homes. In: IEEE PerCom, pp. 67–75 (2012)
3. Bompard, E., Ma, Y., Napoli, R., Abrate, G.: The demand elasticity impacts on the strategic bidding behavior of the electricity producers. IEEE Trans. Power Syst. **22**(1), 188–197 (2007)
4. Brandt, F., Sandholm, T., Shoham, Y.: Spiteful bidding in sealed-bid auctions. In: Veloso, M.M. (ed.) IJCAI, pp. 1207–1214 (2007)
5. Brero, G., Lahaie, S., Seuken, S.: Fast iterative combinatorial auctions via bayesian learning. In: AAAI, pp. 1820–1828 (2019)
6. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Annual Symposium on Foundations of Computer Science, FOCS 2001, pp. 136–145. IEEE Computer Society (2001)
7. Chen, Z., Huang, L., Li, L., Yang, W., Miao, H., Tian, M., Wang, F.: PS-TRUST: provably secure solution for truthful double spectrum auctions. In: 2014 IEEE Conference on Computer Communications, INFOCOM, pp. 1249–1257 (2014)
8. Chen, Z., Chen, L., Huang, L., Zhong, H.: On privacy-preserving cloud auction. In: 35th IEEE SRDS, pp. 279–288 (2016)
9. Dobzinski, S., Lavi, R., Nisan, N.: Multi-unit auctions with budget limits. Games Econ. Behav. **74**(2), 486–503 (2012)
10. Dong, M., Sun, G., Wang, X., Zhang, Q.: Combinatorial auction with time-frequency flexibility in cognitive radio networks. In: IEEE INFOCOM (2012)
11. Faqiry, M.N., Das, S.: Double-sided energy auction in microgrid: equilibrium under price anticipation. IEEE Access **4**, 3794–3805 (2016)
12. Feng, Z., Qiu, C., Feng, Z., Wei, Z., Li, W., Zhang, P.: An effective approach to 5g: wireless network virtualization. IEEE Commun. Mag. **53**(12), 53–59 (2015)
13. Fujiwara, I., Aida, K., Ono, I.: Applying double-sided combinational auctions to resource allocation in cloud computing. In: SAINT, pp. 7–14 (2010)

14. Gao, W., Yu, W., Liang, F., Hatcher, W.G., Lu, C.: Privacy-preserving auction for big data trading using homomorphic encryption. IEEE Trans. Netw. Sci. Eng. (2020)
15. Hoefer, M., Kesselheim, T., Vöcking, B.: Approximation algorithms for secondary spectrum auctions. ACM Trans. Internet Techn. **14**(2–3), 16:1–16:24 (2014)
16. Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., del Cuvillo, J.: Using innovative instructions to create trustworthy software solutions. In: HASP@ISCA, p. 11 (2013)
17. Hong, Y., Wang, H., Xie, S., Liu, B.: Privacy preserving and collusion resistant energy sharing. In: 2018 IEEE ICASSP, pp. 6941–6945 (2018)
18. Hong, Y., Goel, S., Liu, W.: An efficient and privacy-preserving scheme for P2P energy exchange among smart microgrids. Int. J. Energy Res. **40**(3), 313–331 (2016)
19. Huang, H., Li, X., Sun, Y., Xu, H., Huang, L.: PPS: privacy-preserving strategyproof social-efficient spectrum auction mechanisms. IEEE Trans. Parallel Distrib. Syst. **26**(5), 1393–1404 (2015)
20. Huang, Q., Tao, Y., Wu, F.: SPRING: a strategy-proof and privacy preserving spectrum auction mechanism. In: Proceedings of the IEEE INFOCOM, pp. 827–835 (2013)
21. Jia, J., Zhang, Q., Zhang, Q., Liu, M.: Revenue generation for truthful spectrum auction in dynamic spectrum access. In: ACM MobiHoc, pp. 3–12 (2009)
22. Jin, A., Song, W., Zhuang, W.: Auction-based resource allocation for sharing cloudlets in mobile cloud computing. IEEE Trans. Emerg. Topics Comput. **6**, 45–57 (2018)
23. Johari, R., Tsitsiklis, J.N.: Efficiency loss in a network resource allocation game. Math. Oper. Res. **29**(3), 407–435 (2004)
24. Kebriaei, H., Maham, B., Niyato, D.: Double-sided bandwidth-auction game for cognitive device-to-device communication in cellular networks. IEEE Trans. Vehicular Technol. **65**(9), 7476–7487 (2016)
25. Kojima, F., Yamashita, T.: Double auction with interdependent values: incentives and efficiency. Theor. Econ. **12**(3), 1393–1438 (2017)
26. Kosba, A.E., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: IEEE Symposium on Security and Privacy, pp. 839–858 (2016)
27. Krishna, V.: Auction Theory. Academic Press, Boston (2009)
28. Lazar, A.A., Semret, N.: Design and analysis of the progressive second price auction for network bandwidth sharing. Telecommun. Syst. **13** (2001)
29. Liu, B., Xie, S., Hong, Y.: PANDA: privacy-aware double auction for divisible resources without a mediator. In: AAMAS, pp. 1904–1906 (2020)
30. Lorenzo, B., González-Castaño, F.J.: A matching game for data trading in operator-supervised user-provided networks. In: IEEE ICC, pp. 1–7 (2016)
31. Maheswaran, R.T., Başar, T.: Nash equilibrium and decentralized negotiation in auctioning divisible resources. Group Decis. Negot. **12**(5), 361–395 (2003)
32. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: Robust, privacy protecting and publicly verifiable sealed-bid auction. In: International Conference, ICICS, pp. 147–159 (2002)
33. Shi, E., Zhang, F., Pass, R., Devadas, S., Song, D., Liu, C.: Trusted hardware: life, the composable universe, and everything. Manuscript (2015)
34. Suzuki, K., Yokoo, M.: Secure generalized Vickrey auction using homomorphic encryption. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 239–249. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45126-6_17

35. Suzuki, K., Yokoo, M.: Secure combinatorial auctions by dynamic programming with polynomial secret sharing. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 44–56. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36504-4_4
36. Tsai, C., et al.: Cooperation and security isolation of library OSES for multi-process applications. In: EuroSys, pp. 9:1–9:14. ACM (2014)
37. Tsai, C., Porter, D.E., Vij, M.: Graphene-SGX: a practical library OS for unmodified applications on SGX. In: Silva, D.D., Ford, B. (eds.) USENIX, pp. 645–658 (2017)
38. Tuffin, B.: Revisited progressive second price auction for charging telecommunication networks. Telecommun. Syst. **20**(3–4), 255–263 (2002)
39. Wang, Y., Saad, W., Han, Z., Poor, H.V., Basar, T.: A game-theoretic approach to energy trading in the smart grid. IEEE Trans. Smart Grid **5**(3), 1439–1450 (2014)
40. Wu, F., Vaidya, N.H.: SMALL: a strategy-proof mechanism for radio spectrum allocation. In: IEEE INFOCOM, pp. 81–85 (2011)
41. Xie, S., Wang, H., Hong, Y., Thai, M.: Privacy preserving distributed energy trading. In: IEEE ICDCS (2020)
42. Xu, P., Xu, X., Tang, S., Li, X.: Truthful online spectrum allocation and auction in multi-channel wireless networks. In: IEEE INFOCOM, pp. 26–30 (2011)
43. Yokoo, M., Sakurai, Y., Matsubara, S.: The effect of false-name bids in combinatorial auctions: new fraud in internet auctions. Games Econ. Behav. **46**, 174–188 (2004)
44. Yuan, R., Xia, Y., Chen, H., Zang, B., Xie, J.: Shadoweth: private smart contract on public blockchain. J. Comput. Sci. Technol. **33**(3), 542–556 (2018)
45. Yu, J., Cheung, M.H., Huang, J., Poor, H.V.: Mobile data trading: a behavioral economics perspective. In: IEEE WiOpt, pp. 363–370 (2015)
46. Zhang, D., Chang, Z., Yu, F.R., Chen, X., Hämäläinen, T.: A double auction mechanism for virtual resource allocation in SDN-based cellular network. In: IEEE International Symposium on PIMRC, pp. 1–6 (2016)
47. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: an authenticated data feed for smart contracts. In: ACM Conference on CCS, pp. 270–282 (2016)
48. Zou, S., Ma, Z., Liu, X.: Resource allocation game under double-sided auction mechanism: efficiency and convergence. IEEE Trans. Automat. Contr. **63**, 1273–1287 (2018)

Parallel Implementation and Optimization of SM4 Based on CUDA

Jun Li[1], Wenbo Xie[2], Lingchen Li[2](✉), and Xiaonian Wu[2]

[1] China Industrial Control Systems Cyber Emergency Response Team, Beijing 100000, China
[2] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China

Abstract. SM4 is the 128-bit block cipher used in WAPI standard in China, which has a strong security and flexibility. In this paper, the rapid implementation of SM4 is given. Based on the characteristics of CUDA (Compute Unified Device Architecture), a CPU-GPU (Central Processing Unit-Graphics Processing Unit) scheme of SM4 is proposed by exploiting the structure property. Moreover, this scheme is further improved by introducing the page-locked memory and CUDA streams. The results show that: SM4 optimized parallel implementation under GPU can obtain with a speed-up ratio of 89, and the throughput can reach up to 31.41 Gbps.

Keywords: Block cipher · SM4 · Parallel computing · CUDA · GPU

1 Introduction

With the development of Internet services such as cloud computing, big data, and ecommerce, the information security is attracted more and more attention. And cryptography is the basis method to protect the security of information. The SM4 block cipher has been applied in the WAPI (Wireless LAN Authentication and Privacy Infrastructure) wireless network standardwhich is widely used in China. During to the high computational complexity, SM4 is not suitable for occasions which required the encryption algorithm with a high speed. So, the rapid implementation of SM4 is worth to study further. At present, there are two main platforms for fast implementation of block ciphers: GPU (Graphics Processing Unit) and FPGA (Field Programmable Gate Array). However, FPGA is more difficult to develop and has a longer cycle time than GPU. But

This article is supported in part by the National Natural Science Foundation of China (61872103, 62062026), the Key Research and Development Plan of Guangxi (guike AB18281019), the Innovation Research Team Project of Guangxi (2019GXNSFGA245004), the scientific research project of young innovative talents of Guangxi (guike AD20238082), and the Science Research Foundation of Guilin University of Electronic Technology (UF19050Y).

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 93–104, 2021.
https://doi.org/10.1007/978-3-030-80851-8_7

with the feature of the multi-thread, high-bandwidth, GPU is very suitable to do parallel computation to accelerate the speech of encryption and decryption of block ciphers. CUDA (Compute Unified Device Architecture) is a general-purpose device architecture for GPU computing launched by NVIDIA in 2007, which provides a convenient platform for GPU parallel implement of cryptographic algorithms.

As a professional graphics processing tool, GPU was originally used in the field of computer graphics, then Kedem et al. [1] used GPU to quickly crack the key of the UNIX system, which made GPU begin to apply to cryptographic problems. With the introduction of CUDA, GPU has also developed in the field of general computing. Many symmetric cryptographic algorithms have begun to achieve parallel acceleration on CUDA. Manavski et al. [2] first used CUDA to accelerate the AES in 2007, and proposed a T table to improve the performance of the cipher. Since then, most of the teams' work mostly follows their scheme. In 2008, Harrison et al. [3] completed the CTR mode of AES on CUDA, and discussed how to arrange the serial and parallel execution of the block cipher on the GPU. In 2013, Zhou et al. [4] aimed at the shortcomings of the ECB mode of AES and improved the results based on CUDA. In 2014, Mei et al. [5,6] proposed a new fine-grained benchmarking method and applied it to two popular GPU architectures: Fermi and Kepler, and discussed previously unknown features of their memory hierarchy; they also studied the impact of library conflicts on the latency of the shared memory access. In 2017, Fei et al. [7] accelerated AES under CUDA, and the acceleration obtained was about 3 times faster than that of Manavski, and for the first time he analyzed the performance from the perspective of acceleration efficiency. In the same year, Wang et al. [8] compared the performance of SM4 on two different platforms, and got a speed-up ratio of 64.7. In 2020, Li et al. [9] implemented SM4 under CUDA and discussed different thread block sizes and plaintext block sizes, and obtained a speed-up ratio of 26. In addition, the rapid implementation of other block ciphers are also been proposed based on GPU, such as SHA3 [10], IDEA, Blowfish, and Threefish [11] on GPU.

In this paper, we analyze the property of SM4 and the corresponding hardware to improve the speed of implementation on GPU. We propose a parallel scheme based on the characteristics of CUDA programming technology and further improve it. The speed-up ratio and throughput of parallel scheme of SM4 with different input block sizes are obtained. The results show that the speed-up ratio of our scheme can be improved to 89, and the throughput can reach up to 31.41Gbps.

2 Preliminaries

2.1 CUDA

CUDA is a new computing unified device architecture to applicate the GPU as a data parallel computing device, without the graphics API mapping. It is suitable for multi-threaded programming models. The multi-tasking mechanism of the operating system can be used to call multiple CUDA's cores to run simultaneously. The thread-level parallelism is the basic idea of CUDA, so that these threads are dynamically called and executed. The CUDA uses the CPU as the host and the GPU as the device, allowing the CPU to call the GPU and the GPU to run the computationally intensive part of the program. The CUDA programming architecture is shown in Fig. 1.

There are 6 common memory structures in CUDA: register, shared memory, local memory, constant memory, texture memory and global memory. The first two are on-chip memory, and the access speed is faster than the last four [12]. Figure 2 is the memory structure of CUDA, each type of memory has its own scopes and characteristics. Register is the fastest storage unit, but the number of it is small. Variables that cannot be allocated to register will be overflowed into local memory. Register and local memory are private to each thread. Shared memory is the public memory of each block, available for each thread of the same block, but it may cause memory conflicts. Constant memory and texture memory are read-only memory. Because texture memory is mainly used for graphic processing, it is rarely used in cryptography, while constant memory is a special cache which is suitable for data processing in cryptography. Global memory is the largest memory unit in GPU, all threads can access it, but the speed of it is also the slowest [13,14]. Therefore, for the storage of data variables of different sizes, it is necessary to consider the various characteristics of the memory structure and select the appropriate storage structure.

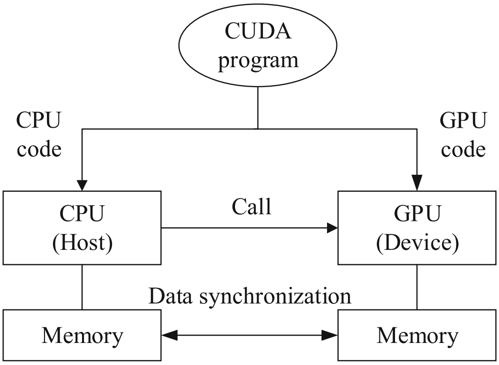


Fig. 1. The programming architecture of CUDA.

2.2 A Brief Description of SM4

SM4 was first proposed in 2006 and became China's national cryptographic industry standard in 2012 [15], which is adopted with a unbalance general feistel structure. The block size and key size are both 128-bit. And the number of iterative rounds is 32. The structure of SM4 is shown in Fig. 3.

Encryption Algorithm. Let (X_0, X_1, X_2, X_3) be the 128-bit input, where $X_i(i{=}\ 0, 1, 2, 3)$ means a 32-bit word, so the function F is

$$F(X_0, X_1, X_2, X_3, rk){=}X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus rk) \tag{1}$$

The round function F includes a linear transformation L and a nonlinear transformation τ, namely $T(.) = L(\tau(.))$. The nonlinear transformation τ is composed of 4 parallel S-boxes. Let $A = (a_0, a_1, a_2, a_3)$ and $B = (b_0, b_1, b_2, b_3)$ be the input and output of τ respectively, which can be defined as,

$$\tau(A) = (b_0, b_1, b_2, b_3) = (Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3)) \tag{2}$$

And the linear transformation L is defined as,

$$L(B) = B \oplus (B{<}{<}{<}2) \oplus (B{<}{<}{<}10) \oplus (B{<}{<}{<}18) \oplus (B{<}{<}{<}24) \tag{3}$$

After 32-round iteration, the output undergoes a reverse order transformation to achieve the ciphertext. The decryption algorithm is similar to the encryption algorithm, except that the decrypted round key is used in the reverse order of encryption.

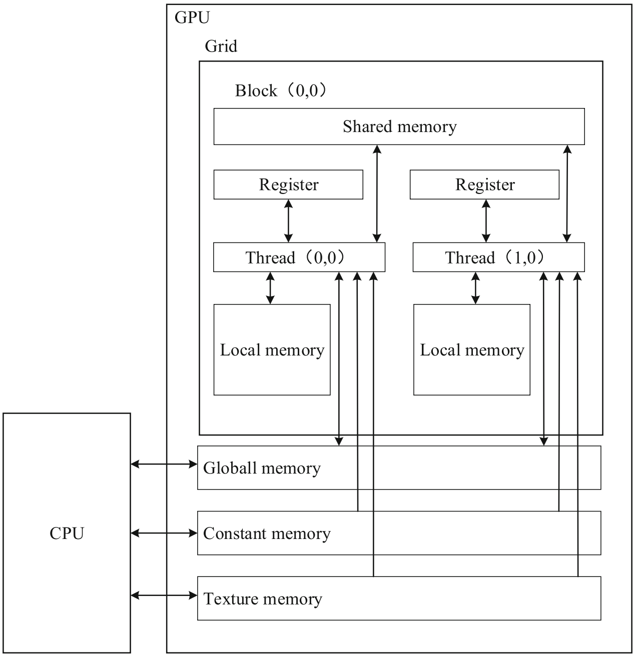


Fig. 2. The memory architecture of CUDA.

Key Schedule. Let $MK = (MK_0, MK_1, MK_2, MK_3)$ be the 128-bit master key of SM4, where $MK_i(i{=}\ 0,1,2,3)$ is a 32-bit word. The round key is expressed as $rk_i(i = 0,1,2,...,31)$, which the size is 32-bit. The key schedule of SM4 can be defined as,

$$(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3) \quad (4)$$

$$rk_i{=}K_{i+4} = K_i \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i) \quad (5)$$

where $FK_i(i = 0,1,2,3)$ and $CK_i(i = 0,1,2,...,31)$ are the constants. T' is almost the same as T with different linear transformation defined as,

$$L'(B) = B \oplus (B{<}{<}{<}13) \oplus (B{<}{<}{<}23) \quad (6)$$

3 Parallel Design of SM4

The basic allocation scheme of SM4 is as follow:

(1) Parallel Granularity, Thread and Grid Allocation
Through the comparison of fine-grained and coarse-grained designs, it is found that the performance of coarse-grained thread scheduling is better, that is, a single thread processes 16 bytes. The allocation of the number of threads is preferably a multiple of 32 [16]. Otherwise, the resources will be wasted. During the upper bound of threads is 1024, we test that 512 threads per block can achieve the best performance. And we set the size of grid is: the size of input/ the number of threads per block/the size of the plaintext block.

(2) Data Allocation
The simplest way is to put the plaintext, S-box, FK and CK into the global memory. However, the access speed of the global memory is very slow, while the storage capacity is large. So the global memory is suitable to store the plaintext. Through comparison tests, it is found that storing the plaintext into the global memory, and the S-box, FK and CK into the constant memory have better performance.

(3) Round Key Allocation
When using multiple-round-key, the round keys used by each GPU thread are different, and the size of keys is large. So GPU is used to generate the round keys and store them in the global memory. When using single-round-key, CPU is chosen to pre-calculate this round key to improve the speed of execution.

So the whole parallel process of SM4 shown in Fig. 4 is: First, randomly generate the input and key according to the length of the input and key that you need, and allocate memory, allocate the size of block and grid. Then, the round key will be generated by GPU or CPU according to the multi-key or single-key mode. Finally, the data is copied to GPU and the kernel function is called for encryption and decryption operations, and then the result is copied back to CPU.

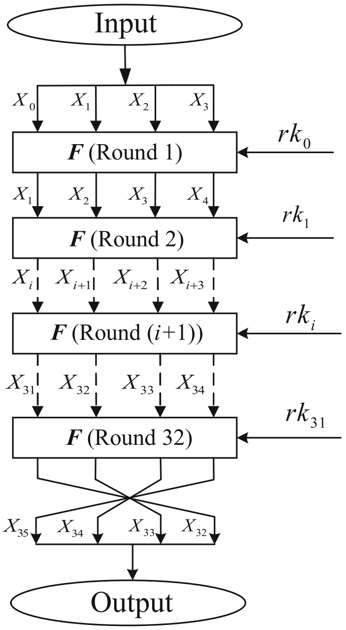


Fig. 3. The structure of SM4.

4 Experiment and Discussion

The environment of experiment in this paper is: GPU: GeForce GTX TITAN X. CPU: Inter(R) Xeon(R) E5-2643 v4 @3.40 GHz. OS: Ubuntu18.04.2 LTS, 64 bits. Version of GCC: 7.5. Version of CUDA: 10.1.

4.1 Basic Experiment

The performance of SM4 based on CPU and GPU is tested under the same size of input. Performing 5 tests, then the average value is computed to reduce the error. All tests consider a single-round-key as the round key. And we use speed-up ratio S to express the improvement of parallel performance of SM4. The calculation equation is defined as,

$$S = T_S/T_p \tag{7}$$

where T_S is the encryption time of CPU, T_p is the encryption time of GPU. The preliminary acceleration of GPU vs. CPU is shown in Fig. 5, and the speed-up ratio can reach up to 53.93.

It can be seen from Fig. 5 that when the size of input is small, the acceleration ratio of the GPU is not very obvious, and the increase is slower. The reason is that the data transmission between the GPU and CPU wastes time when the data is not so big. At the same time, GPU cannot call enough computing units

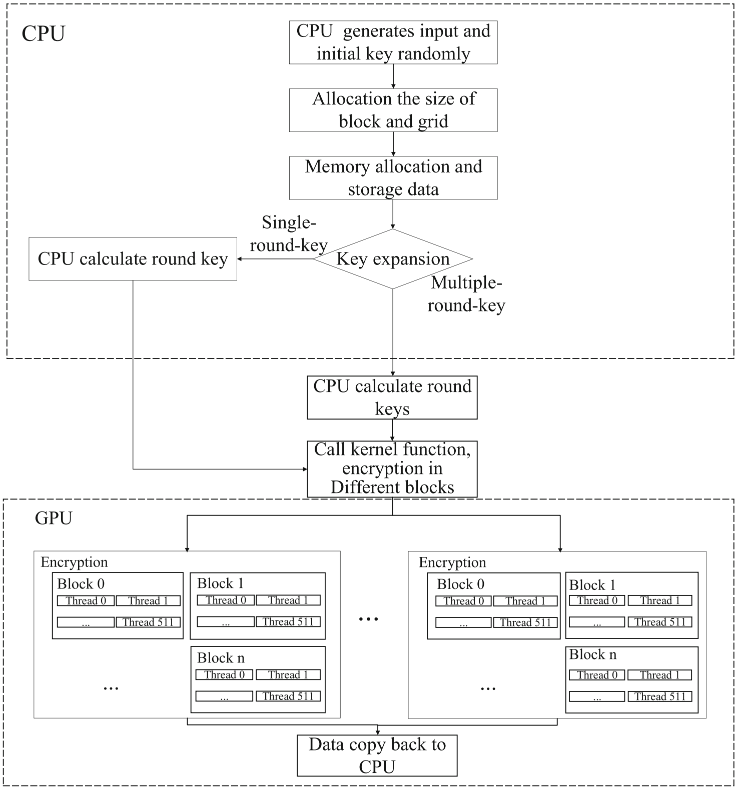


Fig. 4. The parallel process of SM4.

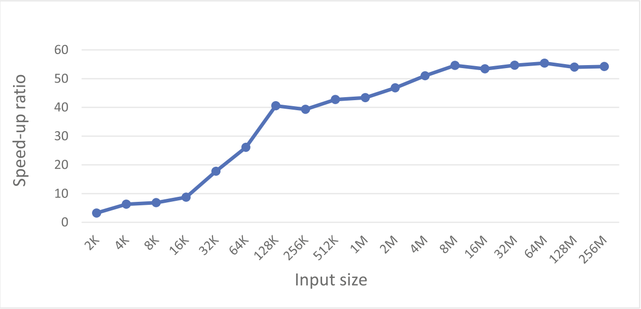


Fig. 5. The structure of SM4.

to perform parallel operations. So the improvement of performance based on GPU is not obvious. However, as the size of input increases, the speed-up ratio increases rapidly. But when the size of input reaches a certain value, that's 256 KB, the growth of the speed-up ratio begins to level off again and finally stabilizes at about 50. This shows that the acceleration of GPU is not unlimited growth, it will also be limited by its own hardware.

In order to make better use of GPU performance, the next section we will take measures to optimize operations

4.2 Performance Optimization

Data Transmission Optimization. For data transmission optimization, the measures we adopted are: use page-locked memory, also known as pinned memory, instead of pageable memory. By default, the CPU allocates pageable host memory. The GPU cannot directly access data in the pageable memory because the memory data may be shifted or destroyed, it cannot safely use the physical address of the host memory. The GPU needs to perform two-part copy operation to obtain data from the pageable memory, first copy the data to the temporary page-locked memory, and then copy the data from the page-locked memory to the GPU. But by using page-locked memory to allow GPU to access the memory directly, the time of data transmission can be reduced. Figure 6 shows the process of page-locked memory transfer and pageable memory transfer.

Although the allocation and release cost of page-locked memory is higher than that of pageable memory, it can provide higher transmission throughput for large-scale data transmission. Figure 7 shows the performance comparison between pageable memory and page-locked memory. It can be seen that when the size of input is small, the acceleration effect is not obvious. The reason is that the initial cost of page-locked memory is basically the same as the benefit from processing data. However, when transmitting more than 64 KB input, as the size of input increases, the cost of the initial overhead becomes smaller and smaller, and the page-locked memory gains obvious. The final optimized performance increased by 40.73%.

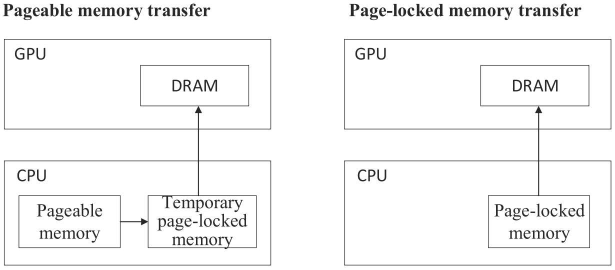


Fig. 6. Two kinds of memory data transmission.

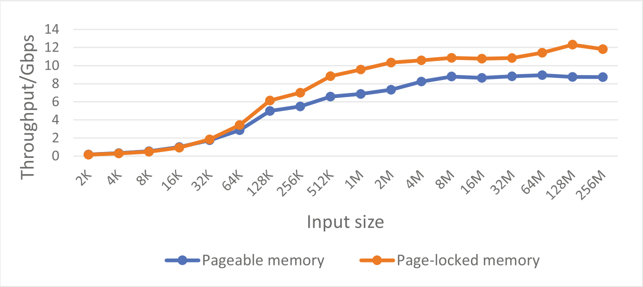


Fig. 7. Performance comparison between pageable memory and page-locked memory.

Parallel Streams Overlapped Execution Optimization. For the optimization of the sequential execution of data transmission and kernel operations, the solution we proposed is: using CUDA streams to overlap data transmission and kernel operations. The basic structure of the SM4 encryption program consists of 3 steps: ① copy the plaintext and key from the host to the device. ② Perform the encryption operation. ③ Return the ciphertext from the device to the host. In order to achieve over-lapping operations, instead of copying the input data to the GPU all at once, the idea of "divide and conquer" is adopted to divide the input data into multiple subsets. Then each sub-problem is independent and can be separately arranged in the CUDA stream for calculation. The output transmission of one stream overlaps the core calculation of another stream. The sequential execution and the overlapping streams execution are shown in Fig. 8.

In order to make better use of the two replication engines of the GPU, we use 8 streams and one queue for each stream for overlapping operations. Figure 9 is a performance comparison between sequential execution and overlapping streams execution. As a result, the optimized performance increased by 33.3% after two rounds of optimization operations. The performance of the original GPU parallel algorithm has been improved by twice. In the end, compared to SM4's CPU serial algorithm, GPU parallel algorithm has nearly 90 times the performance improvement, as shown in Fig. 10.

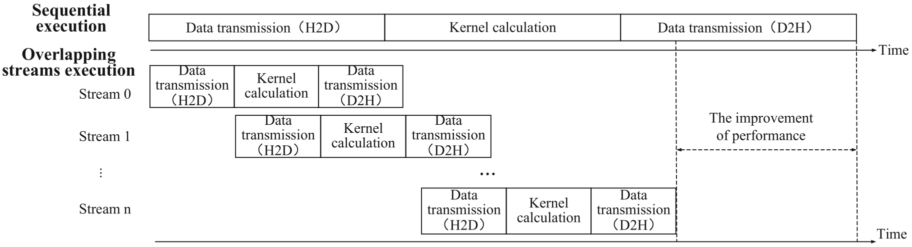


Fig. 8. Sequential execution and overlapping streams execution.

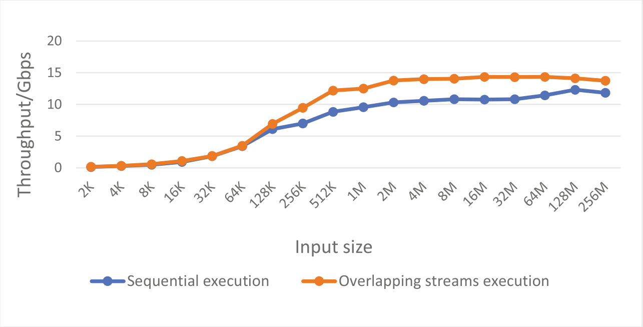


Fig. 9. Performance comparison between sequential execution and overlapping streams execution.

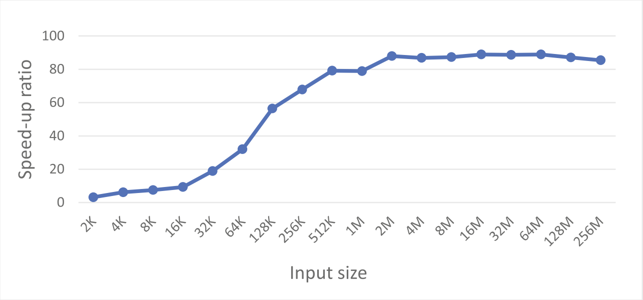


Fig. 10. Optimized speed-up ratio.

4.3 Analysis of Results

The speed-up ration in our scheme is 89 which is higher than reference [8] and [9] respectively as shown in Table 1. However, considering release time, core parameters, operating frequency, bandwidth and other factors of the GPU, we sort the performance of the GPU of the three articles from high to low: our article, [8,9]. In other words, the high speed-up ratio in our article may be caused by the better performance of the GPU itself. Therefore, we compare our result of throughput with [17,18], we can see that in Table 2. The GPU used in [17,18] are Nvidia GeForce RTX 2080 and NVIDIA GTX 1080 respectively, GPUs' performances are better than us. We find that the throughput of [17] can reach up to 27.64 Gbps, and the throughput of our result can reach 31.41 Gbps, which has a performance improvement of 1.13 times. It shows that our work does effectively improve the performance of SM4. Although the throughput is up to 76.8 Gbps in [18], our scheme is still competitive when the price cost is considered.

After analyzing the experiment, it is found that the main factors affecting GPU performance are as follows: ① The data transmission between the host and the device takes a long time. ② The sequential execution of data transmission and kernel operations causes time wasted.

Table 1. Compare with the speed-up ratio.

Input size	GeForce GTX TI TAN X(Us)	GeForce GT 240M([8])	Quadro 600 ([9])
32K	18.96	6.34	4.14
1M	78.94	29.81	23.38
8M	87.31	39.71	25.79
32M	88.69	40.67	25.80

Table 2. Compare with the speed-up ratio of previous works.

	GeForce GTX TI TAN X(Us)	GeForce RTX 2080 ([17])	NVIDIA GTX 1080 ([18])
Throughput (Gbps)	31.41	27.64	76.80
GPU's performance	Good	Better	Best
GPU's price	Medium	High	Higher

5 Conclusion

Based on the CUDA under the Linux OS, we implement the SM4 in parallel and explore the comparison of the performance of the CPU and GPU under the same plaintext in the range of 2 KB to 256 MB under the CPU and GPU. The results show that the GPU performance of parallel implement of SM4 is 88–89 times faster than CPU. Moreover, through introducing the page-locked memory and CUDA streams, the performance of our parallel algorithm can be further improved with the throughput 31.41 Gbps.

References

1. Kedem, G.-Y.: Brute force attack on UNIX passwords with SIMD computer. In: Proceedings of the 8th Conference on USENIX Security Symposium, pp. 93–98. USENIX Association, Washington, D.C. (1999)
2. Manavski, S.A.: CUDA compatible GPU as an efficient hardware accelerator for AES cryptography. In: 2007 IEEE International Conference on Signal Processing and Communications, pp. 65–68. Springer, Dubai (2007)
3. Harrison, O., Waldron, J.: Practical symmetric key cryptography on modern graphics hardware. In: Proceedings of the 17th USENIX Security Symposium, pp. 195–210. USENIX Association, San Jose (2008)

4. Xia, C., Zhou, D., Zhang, K., Liu, C., Chu, X.: CUDA based high-efficiency implementation of AES algorithm. **30**(6), 1907–1909 (2013). (in Chinese)
5. Mei, X., Zhao, K., Liu, C., Chu, X.: Benchmarking the memory hierarchy of modern GPUs. In: Hsu, C.-H., Shi, X., Salapura, V. (eds.) NPC 2014. LNCS, vol. 8707, pp. 144–156. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44917-2_13
6. Mei, X., Chu, X.: Dissecting GPU memory hierarchy through microbenchmarking. IEEE Trans. Parallel Distrib. Syst. **28**(1), 72–86 (2017)
7. Fei, X., Chu, X., Yang, W.: Research and implementation of GPU parallel AES algorithm based on CTR model. J. Chin. Comput. Syst. **36**(3), 529–533 (2015). (in Chinese)
8. Wang, D., Chen, D.: High speed implementation of SM4 encryption algorithm based on CUDA. J. Shijiazhuang Inst. Railway Technol. **16**(1), 59–63 (2017). (in Chinese)
9. Wang, D., Chen, D.: Parallel implementation of SM4 algorithm on GPU. Inf. Netw. Secur. **20**(6), 36–43 (2020). (in Chinese)
10. Yang, J., Wang, W., Xie Z., Han, J., Zeng X.: Parallel implementations of SHA-3 on a 24-core processor with software and hardware co-design. In: IEEE 12th International Conference on ASIC, pp. 953–956. IEEE Computer Society, Guiyang (2017)
11. Cheong, H., Lee, W.: Fast implementation of block ciphers and PRNGs for Kepler GPU architecture. In: International Conference on It Convergence and Security, pp. 1–5. IEEE Computer Society, Los Alamitos (2015)
12. NVIDIA Corporation, https://docs.nvidia.com/cuda/cuda-c-programming-guide. Accessed 23 Sep 2020
13. Jason, S., Edward, K.: CUDA by Example: An Introduction to General-Purpose GPU Programming, 1st edn., Machinery Industry Press, Beijing (2011). (in Chinese)
14. Cheng, J.: Professional CUDA C Programming, 2nd edn., Machinery Industry Press, Beijing (2017). (in Chinese)
15. China's National Cryptography Administration. http://www.sca.gov.cn/sca/c100061/201611/1002423/files/330480f731f64e1ea75138211ea0dc27.pdf. Accessed 18 Nov 2016. (in Chinese)
16. Ma, J., Chen, X., Xu, R., Shi, J.: Implementation and evaluation of different parallel designs of AES using CUDA. In: IEEE Second International Conference on Data Science in Cyberspace, pp. 606–614. IEEE Computer Society, Shenzhen (2017)
17. Zhang, C.: Design and implementation of parallel SM4-GCM based on CUDA. Xi'an University of Electronic Ccience and Technology **2**(1), 104 (2019).(in Chinese)
18. Cheng, W., Zheng, F., Pan, W., Lin, J., Li, H., Li, B.: High-performance symmetric cryptography server with GPU acceleration. In: Qing, S., Mitchell, C., Chen, L., Liu, D. (eds.) ICICS 2017. LNCS, vol. 10631, pp. 529–540. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89500-0_46

Another Algebraic Decomposition Method for Masked Implementation

Shoichi Hirose(✉)

University of Fukui, Fukui, Japan
hrs_shch@u-fukui.ac.jp

Abstract. Side channel attacks are serious concern for implementation of cryptosystems. Masking is an effective countermeasure against them and masked implementation of block ciphers has been attracting active research. It is an obstacle to efficient masked implementation that the complexity of an evaluation of multiplication is quadratic in the order of masking. A direct approach to this problem is to explore methods to reduce the number of multiplications required to represent an S-box. An alternative approach proposed by Carlet et al. in 2015 is to represent an S-box as composition of polynomials with low algebraic degrees. We follow the latter approach and propose to use a special type of polynomials with a low algebraic degree as components, which we call generalized multiplication (GM) polynomials. The masking scheme for multiplication can be applied to a GM polynomial, which is more efficient than the masking scheme for a polynomial with a low algebraic degree. Our experimental results show that, for 4-/6-/8-bit permutations, the proposed decomposition method is more efficient than the method by Carlet et al. in most cases in terms of the number of evaluations of low-algebraic-degree polynomials required by masking.

Keywords: Algebraic decomposition · Boolean function · Masking · S-box

1 Introduction

Background. Side channel attacks introduced by Kocher [10] are serious concern for implementation of cryptosystems. Chari et al. [4] proposed a sound approach based on secret sharing [1,14] against a class of side-channel attacks analyzing power consumption [11]. It is usually called masking [12] in this context. The d-th order masking splits each internal variable into $(d+1)$ shares so that any information of the internal variable cannot be recovered from at most d shares. The complexity of a successful side channel attack against a masked implementation was shown to be exponential in the masking order d [4].

Masked implementation has often been discussed for block ciphers. A block cipher can be manipulated as a function over the finite field $\mathbb{F}_2$ or its extension. For a scalar multiplication or an addition, the number of operations to compute

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 105–114, 2021.
https://doi.org/10.1007/978-3-030-80851-8_8

shares of the result from the shares of an input is $O(d)$. For a square, it is also $O(d)$. For a multiplication, on the other hand, it is $O(d^2)$. Thus, a multiplication is especially called a nonlinear multiplication to refer to the difference from a square.

For efficient masked implementation of block ciphers, it has been actively studied to reduce the number of nonlinear multiplications required to compute an S-box. A similar but different approach is to represent an S-box as composition of polynomials with low algebraic degrees [3].

Our Contribution. We present a method for algebraic decomposition inspired by the method of Carlet et al. [3] and the method to reduce the number of nonlinear multiplications of Goudarzi et al. [7]. The proposed method can be applied to any function from $\{0,1\}^n$ to $\{0,1\}^n$ for even n. It regards a given function $h(x)$ as a pair of bivariate polynomials $(h_0(x_0,x_1), h_1(x_0,x_1))$, where $h_b : \mathbb{F}_{2^{n/2}} \times \mathbb{F}_{2^{n/2}} \rightarrow \mathbb{F}_{2^{n/2}}$ for $b \in \{0,1\}$. Then, it represents h as composition of pairs of linear combinations of $x_0^{2^{i_0}} x_1^{2^{i_1}}$, where $0 \leq i_b \leq n/2-1$ for $b \in \{0,1\}$. We call such a pair of linear combinations a generalized multiplication (GM) polynomial. The difference of our proposed method from the method of Carlet et al. [3] is that the former uses GM polynomials instead of polynomials of low algebraic degrees such as 2 or 3. To a GM polynomial, the masking scheme for a multiplication can be applied, which is more efficient than the masking scheme for a polynomial of low algebraic degree presented by Carlet et al. [3]. Due to this property, for masked implementation, in terms of the number of evaluations of nonlinear functions (GM polynomials, polynomials of low algebraic degree or multiplications), the proposed decomposition method is more efficient than the method by Goudarzi et al. [7] for $n = 4, 6, 8$ and than the method by Carlet et al. [3] for $n = 4, 6$ and for $n = 8$ if the masking order is higher than 1.

Related Work. Ishai, Sahai and Wagner presented a higher-order masking method for multiplication over $\mathbb{F}_2$ in their seminal paper [8]. Rivain and Prouff [13] generalized the method of Ishai et al. [8] to any finite field multiplication and applied it to the AES S-box. Carlet et al. [2] extended the method of Rivain and Prouff [13] and proposed a generic method for masking any S-box based on cyclotomic classes and the Knuth-Eve polynomial evaluation algorithm [6,9]. Coron, Roy and Vivek [5] improved the method of Carlet et al. [2] and presented a heuristic but generic method for masking any S-box. Goudarzi et al. [7] generalized the approach of Coron, Roy and Vivek [5] and proposed a method to treat any S-box from $\{0,1\}^{w_{\mathrm{i}}\nu}$ to $\{0,1\}^{w_{\mathrm{o}}\nu}$ as a tuple of polynomials over $\mathbb{F}_{2^\nu}$.

Inspired by the work of Coron, Roy and Vivek [5], Carlet et al. [3] introduced a new approach to decompose any S-box using polynomials having low algebraic degrees. They also presented masking methods for polynomials having low algebraic degrees.

Organization. Section 2 introduces some notations and definitions necessary for the discussions. Section 3 presents the proposed algebraic decomposition method using GM polynomials and its experimental results. Section 4 discusses application of the proposed decomposition to masking. Section 5 gives a brief concluding remark.

2 Preliminaries

Let ν be a positive integer. Let $\mathbb{F}_{2^\nu}$ be the finite field with 2^ν elements.

2.1 Functions over Finite Fields

A function $h : \mathbb{F}_{2^\nu}^{w_\mathrm{i}} \to \mathbb{F}_{2^\nu}^{w_\mathrm{o}}$ is a tuple of functions $(h_0, h_1, \ldots, h_{w_\mathrm{o}-1})$, where $h_j : \mathbb{F}_{2^\nu}^{w_\mathrm{i}} \to \mathbb{F}_{2^\nu}$ for $0 \le j \le w_\mathrm{o} - 1$. h_j can be represented as

$$h_j(x_0, x_1, \ldots, x_{w_\mathrm{i}-1}) = \sum_{k_0=0}^{2^\nu-1} \cdots \sum_{k_{w_\mathrm{i}-1}=0}^{2^\nu-1} \alpha_{j,k_0,\ldots,k_{w_\mathrm{i}-1}} x_0^{k_0} \cdots x_{w_\mathrm{i}-1}^{k_{w_\mathrm{i}-1}} \ ,$$

where $\alpha_{j,k_0,\ldots,k_{w_\mathrm{i}-1}} \in \mathbb{F}_{2^\nu}$. We only refer to the cases that $(w_\mathrm{i}, w_\mathrm{o}) \in \{(1,1),(2,1),(2,2)\}$ in the remaining parts.

Definition 1 (Algebraic degree). *For a function $h : \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu}$ such that*

$$h(x) = \sum_{k=0}^{2^\nu-1} \alpha_k x^k \ ,$$

its algebraic degree is the maximum of $\mathrm{HW}(k)$ *such that* $\alpha_k \neq 0$ *for* $0 \le k \le 2^\nu - 1$*, where* $\mathrm{HW}(k)$ *is the Hamming weight of the binary representation of* k*.*

Definition 2 (Linearized polynomial). *A function $\ell : \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu}$ is called a linearized polynomial if it can be represented as*

$$\ell(x) = \sum_{k=0}^{\nu-1} \alpha_k x^{2^k} \ ,$$

where $\alpha_k \in \mathbb{F}_{2^\nu}$.

For any linearized polynomial $\ell(x)$, its algebraic degree is 1, and it holds that

$$\ell\Bigl(\sum_{i=0}^{d} x_i\Bigr) = \sum_{i=0}^{d} \ell(x_i) \ . \qquad (1)$$

We introduce generalized multiplication polynomials, which are used in our proposed decomposition method:

Definition 3 (Generalized multiplication polynomial). *We call a function* $m : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$ *a generalized multiplication (GM) polynomial if it can be represented as* $m(x) = (m_0(x), m_1(x))$ *such that, for* $b \in \{0,1\}$, $m_b : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu}$ *and*

$$m_b(x) = \sum_{k=0}^{\nu-1} \sum_{l=0}^{\nu-1} \alpha_{b,k,l} x_0^{2^k} x_1^{2^l} \ ,$$

where $x = (x_0, x_1) \in \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$ *and* $\alpha_{b,k,l} \in \mathbb{F}_{2^\nu}$.

For any GM polynomial $m(x_0, x_1)$, it holds that

$$m\Bigl(\sum_{i=0}^{d} x_{0,i}, \sum_{j=0}^{d} x_{1,j}\Bigr) = \sum_{i=0}^{d}\sum_{j=0}^{d} m(x_{0,i}, x_{1,j}) \tag{2}$$

since

$$\Bigl(\sum_{i=0}^{d} x_{0,i}\Bigr)^{2^k} \Bigl(\sum_{j=0}^{d} x_{1,j}\Bigr)^{2^l} = \Bigl(\sum_{i=0}^{d} x_{0,i}^{2^k}\Bigr)\Bigl(\sum_{j=0}^{d} x_{1,j}^{2^l}\Bigr) = \sum_{i=0}^{d}\sum_{j=0}^{d} x_{0,i}^{2^k} x_{1,j}^{2^l} \ . \tag{3}$$

For Eq. (3), multiplication is the case that $k = l = 0$.

3 Algebraic Decomposition

In the remaining parts of the paper, ν is a positive integer and $n = 2\nu$.

3.1 Algebraic Decomposition Using GM Polynomials

Let $h : \{0,1\}^n \to \{0,1\}^n$. Then, h can be seen as $h(x) = (h_0(x_0, x_1), h_1(x_0, x_1))$, where $x = (x_0, x_1)$ and $h_b : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu}$ for $b \in \{0,1\}$. The decomposition of h proceeds as follows:

1. For $1 \le i \le r$, let $f_i : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$ is a GM polynomial chosen uniformly at random.
2. For $1 \le i \le r$, $g_i : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$ is defined as follows:

$$g_1(x) = f_1(x) \ ,$$

$$g_2(x) = f_2(g_1(x) + (\ell_{0,0}(x_0) + \ell_{0,1}(x_1), \ell_{1,0}(x_0) + \ell_{1,1}(x_1))) \ ,$$

where $\ell_{0,0}$, $\ell_{0,1}$, $\ell_{1,0}$ and $\ell_{1,1}$ are linearized polynomials over $\mathbb{F}_{2^\nu}$ chosen uniformly at random, and, for $3 \le i \le r$,

$$g_i(x) = f_i(g_{i-1}(x)) \ .$$

3. For $1 \leq j \leq t$, $q_j : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \longrightarrow \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$ is defined as follows:
$$q_j(x) = (q_{j,0}(x), q_{j,1}(x)) \ ,$$
where, for $b \in \{0,1\}$,
$$q_{j,b}(x) = \sum_{i=1}^{r} \ell_{j,i,b}(g_{i,b}(x)) + \ell_{j,0,b,0}(x_0) + \ell_{j,0,b,1}(x_1) \ ,$$
and $\ell_{j,i,b}$, $\ell_{j,0,b,0}$ and $\ell_{j,0,b,1}$ are linearized polynomials over $\mathbb{F}_{2^\nu}$ chosen uniformly at random.
4. For $b \in \{0,1\}$, search GM polynomials $\mu_1, \ldots, \mu_t$ over $\mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$, linearized polynomials $\lambda_{0,b,0}, \lambda_{0,b,1}, \ldots, \lambda_{r,b,0}, \lambda_{r,b,1}$ over $\mathbb{F}_{2^\nu}$ and a constant $\delta_b \in \mathbb{F}_{2^\nu}$ satisfying
$$\begin{aligned} h_b(x) = \sum_{j=1}^{t} \mu_{j,b}(q_j(x)) + \sum_{i=1}^{r} \bigl(\lambda_{i,b,0}(g_{i,0}(x)) + \lambda_{i,b,1}(g_{i,1}(x))\bigr) \\ + \lambda_{0,b,0}(x_0) + \lambda_{0,b,1}(x_1) + \delta_b \quad . \end{aligned} \tag{4}$$
If the search fails, then return to the first step.

The amount of computation for an evaluation of h based on the decomposition is summarized in Table 1.

Table 1. The amount of computation based on the proposed decomposition. Both the linearized polynomials and the additions are over $\mathbb{F}_\nu$.

The number of evaluations of GM polynomials	$r + t$
The number of evaluations of linearized polynomials	$2(r+2)(t+2)$
The number of evaluations of additions	$2(r+2)(t+2)$

Similar to the decomposition in [3], the search in the 4th step above can be done by solving a system of linear equations over $\mathbb{F}_{2^\nu}$:
$$A \cdot \boldsymbol{v}_b = \boldsymbol{c}_b \tag{5}$$
for $b \in \{0,1\}$. $\boldsymbol{c}_b$ is a 2^n-dimensional vector over $\mathbb{F}_{2^\nu}$ such that
$$\boldsymbol{c}_b = (h_b(e_1), h_b(e_2), \ldots, h_b(e_{2^n}))^{\mathrm{T}} \ ,$$
where $e_i = (e_{i,0}, e_{i,1}) \in \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$ and $e_{i_1} \neq e_{i_2}$ if $i_1 \neq i_2$. $\boldsymbol{v}_b$ is the vector of unknowns representing the coefficients of GM polynomials $\mu_{1,b}, \ldots, \mu_{t,b}$, linearized polynomials $\lambda_{0,b,0}, \lambda_{0,b,1}, \ldots, \lambda_{r,b,0}, \lambda_{r,b,1}$ and δ_b. The matrix A, which does not depend on the value of b, is defined as follows:
$$A = (A_{q_1} \ A_{q_2} \ \cdots \ A_{q_t} \ A_{g_{1,0}} \ \cdots \ A_{g_{r,0}} \ A_{g_{1,1}} \ \cdots \ A_{g_{r,1}} \ A_{e_{*,0}} \ A_{e_{*,1}} \ \mathbf{1}) \ .$$

$$A_{q_j} = \begin{pmatrix} Q_{j,1}^{(0,0)} \cdots Q_{j,1}^{(0,\nu-1)} \; Q_{j,1}^{(1,0)} \cdots Q_{j,1}^{(1,\nu-1)} \cdots Q_{j,1}^{(\nu-1,0)} \cdots Q_{j,1}^{(\nu-1,\nu-1)} \\ Q_{j,2}^{(0,0)} \cdots Q_{j,2}^{(0,\nu-1)} \; Q_{j,2}^{(1,0)} \cdots Q_{j,2}^{(1,\nu-1)} \cdots Q_{j,2}^{(\nu-1,0)} \cdots Q_{j,2}^{(\nu-1,\nu-1)} \\ \cdots \qquad \cdots \qquad \cdots \qquad \cdots \\ Q_{j,2^n}^{(0,0)} \cdots Q_{j,2^n}^{(0,\nu-1)} \; Q_{j,2^n}^{(1,0)} \cdots Q_{j,2^n}^{(1,\nu-1)} \cdots Q_{j,2^n}^{(\nu-1,0)} \cdots Q_{j,2^n}^{(\nu-1,\nu-1)} \end{pmatrix}$$

is a $2^n \times \nu^2$ matrix, where $Q_{j,i}^{(k,l)} = q_{j,0}(e_i)^{2^k} q_{j,1}(e_i)^{2^l}$ for $1 \le i \le 2^n$, $1 \le k \le \nu - 1$ and $1 \le l \le \nu - 1$.

$$A_{g_{i,b'}} = \begin{pmatrix} g_{i,b'}(e_1)^{2^0} & g_{i,b'}(e_1)^{2^1} & \cdots & g_{i,b'}(e_1)^{2^{\nu-1}} \\ g_{i,b'}(e_2)^{2^0} & g_{i,b'}(e_2)^{2^1} & \cdots & g_{i,b'}(e_2)^{2^{\nu-1}} \\ & & \cdots & \\ g_{i,b'}(e_{2^n})^{2^0} & g_{i,b'}(e_{2^n})^{2^1} & \cdots & g_{i,b'}(e_{2^n})^{2^{\nu-1}} \end{pmatrix}$$

is a $2^n \times \nu$ matrix for $1 \le i \le r$ and $b' \in \{0, 1\}$.

$$A_{e_{*,b'}} = \begin{pmatrix} e_{1,b'}^{2^0} & e_{1,b'}^{2^1} & \cdots & e_{1,b'}^{2^{\nu-1}} \\ e_{2,b'}^{2^0} & e_{2,b'}^{2^1} & \cdots & e_{2,b'}^{2^{\nu-1}} \\ & \cdots & & \\ e_{2^n,b'}^{2^0} & e_{2^n,b'}^{2^1} & \cdots & e_{2^n,b'}^{2^{\nu-1}} \end{pmatrix}$$

is a $2^n \times \nu$ matrix. $\mathbf{1}$ is the column vector whose 2^n coordinates equal 1.

The matrix A has 2^n rows and $t \cdot \nu^2 + 2(r + 1)\nu + 1$ columns. In order for the system of linear equations Eq. (5) to have a solution for any $\boldsymbol{c}_b$, the rank of A must be 2^n and it is required that

$$t \cdot n^2/4 + (r + 1)n + 1 \ge 2^n \ . \tag{6}$$

It is also required that the algebraic degree of the polynomial in the right side of Eq. (4) is $n/2$ with respect to each of x_0 and x_1. Thus,

$$2^r \ge n/2 \ . \tag{7}$$

Once we obtain a matrix A with its rank 2^n from some $g_1, \ldots, g_r$ and $q_1, \ldots, q_t$, we can use it to decompose any function $h : \{0, 1\}^n \to \{0, 1\}^n$.

3.2 Experimental Result

Table 2 shows the values of parameters of successful decomposition minimizing the number of GM polynomials, that is, $r + t$. For $n = 4, 6$, all optimal values satisfying inequalities (6) and (7) and minimizing $r + t$ are achieved. For $n = 8$, optimal values are also achieved. On the other hand, though $(r, t) = (2, 15)$ are also optimal, they cannot be achieved with one hundred trials. For $n = 10$, all optimal values $(r, t) = (3, 40), (4, 39)$ as well as $(r, t) = (3, 41), (4, 40)$ cannot be achieved with one hundred trials.

Table 3 shows the smallest number of nonlinear functions achieved by our decomposition and the decomposition methods by Carlet et al. [3] and by

Table 2. Achievable parameters minimizing $r+t$. #GMP represents the number of GM polynomials. #LinP represents the number of linearized polynomials. #Add represents the number of additions.

n	(r,t)	#GMP	#LinP	#Add
4	$(1,2)$	3	24	24
	$(2,1)$	3	24	24
6	$(2,5)$	7	56	56
8	$(3,14)$	17	160	160
10	$(5,39)$	44	574	574

Goudarzi et al. [7]. For algebraic decomposition by Carlet et al., methods using polynomials of algebraic degrees 2 and/or 3 were presented, and the most efficient method was shown to be the method using only quadratic polynomials (polynomials of algebraic degree 2), which is mentioned in Table 3. The decomposition method to reduce the number of multiplications by Goudarzi et al. [7] is able to process any function over $\{0,1\}^n$ by regarding it as a function over $\mathbb{F}_{\xi}^{n/\xi}$ for any ξ such that $\xi \mid n$. Table 3 mentions only the case that $\xi = n/2$.

In terms of the number of multiplications or GM polynomials, our decomposition is slightly more efficient than the decomposition by Goudarzi et al. [7]. On the other hand, if implementation adopts table lookup for evaluation of multiplications or GM polynomials, then our decomposition needs a lookup table for each GM polynomial, while the decomposition by Goudarzi et al. needs just a single lookup table for multiplication. Thus, the total table size for our decomposition is $2(r+t)$ times as large as that for the decomposition by Goudarzi et al. For example, for $n = 8$, the total table size of GM polynomials for our decomposition is $4352(= 17 \times 256)$ Bytes.

In terms of the number of quadratic polynomials or GM polynomials, our decomposition does not seem so good as decomposition by Carlet et al. [3] apparently. We will see in the next section, however, our decomposition is more effective than the decomposition by Carlet et al. [3] for masked implementation.

Table 3. Comparison of best achievable parameters

	$n=4$	$n=6$	$n=8$
# quadratic polynomials [3]	3	5	11
# multiplications [7]	4	9	18
# GM polynomials (Ours)	3	7	17

4 Application to Masking

Algorithm 1 presents an algorithm of d-th order masking for a GM polynomial. Due to the property of GM polynomials shown by Eq. (3), it is similar to d-th order masking for multiplication. For reference, the algorithm of d-th order masking for a quadratic polynomial [3] is shown in Appendix A.

Algorithm 1: d-th order masking for a GM polynomial $m : \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu} \to \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$, where $\nu = n/2$

```
input  : Shares $(a_0, a_1, \dots, a_d)$ of $a$ and $(b_0, b_1, \dots, b_d)$ of $b$
output: Shares $(c_0, c_1, \dots, c_d)$ of $c = m(a, b)$
for $i = 0$ to $d$ do
    for $j = i + 1$ to $d$ do
        $r_{i,j} \twoheadleftarrow \mathbb{F}_{2^\nu} \times \mathbb{F}_{2^\nu}$;
        $r_{j,i} \leftarrow (r_{i,j} + m(a_i, b_j)) + m(a_j, b_i)$;
for $i = 0$ to $d$ do
    $c_i \leftarrow m(a_i, b_i)$;
    for $j = 0$ to $d$ do
        if $j \neq i$ then
            $c_i \leftarrow c_i + r_{i,j}$;
return $(c_0, c_1, \dots, c_d)$
```

Table 4 shows complexity of d-th order masking for an evaluation of a quadratic polynomial or a GM polynomial. Roughly, d-th order masking for a GM polynomial is twice as efficient as that for a quadratic polynomial. From Tables 3 and 4, in terms of the number of evaluations of nonlinear functions (quadratic polynomials or GM polynomials), the proposed decomposition yields more efficient masking for n-bit S-boxes than the decomposition using quadratic polynomials by Carlet et al. [3] for $n = 4, 6$, and for $n = 8$ if $d \geq 2$.

Table 4. Complexity of d-th order masking. "# eval," "# rand" and "# add," represent the required number of evaluations of a nonlinear function (a quadratic polynomial or a GM polynomial), random sequences and additions, respectively.

	# eval	# rand	# add
Quadratic poly. eval	$(d+1)(2d+1)$	$d(d+1)$	$9d(d+1)/2+1$
GP poly. eval. (Algorithm 1)	$(d+1)^2$	$d(d+1)/2$	$2d(d+1)$

5 Conclusion

We have presented an algebraic decomposition method for masked implementation of any S-box. Essentially, our proposal is to use GM polynomials instead of polynomials with low algebraic degrees for decomposition. Future work is performance evaluation of masked implementaion of S-boxes using the proposed decomposition method.

Acknowledgements. The author was supported in part by JSPS KAKENHI Grant Number JP18H05289.

A Masking for Quadratic Polynomial

Algorithm 2 presents an algorithm of d-th order masking for a quadratic polynomial [3].

Algorithm 2: d-th order masking for a quadratic polynomial $f : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$

```
input  : Shares $(a_0, a_1, \ldots, a_d)$ of $a$
output: Shares $(b_0, b_1, \ldots, b_d)$ of $b = f(a)$
for $i = 0$ to $d$ do
    for $j = i + 1$ to $d$ do
        $r_{i,j} \twoheadleftarrow \mathbb{F}_{2^n}$; $r'_{i,j} \twoheadleftarrow \mathbb{F}_{2^n}$;
        $r_{j,i} \leftarrow r_{i,j} + f(a_i + r'_{i,j}) + f(a_j + r'_{i,j}) + f((a_i + r'_{i,j}) + a_j) + f(r'_{i,j})$;
for $i = 0$ to $d$ do
    $b_i \leftarrow f(a_i)$;
    for $j = 0$ to $d$ do
        if $j \neq i$ then
            $b_i \leftarrow b_i + r_{i,j}$;
if $d$ is odd then
    $b_1 \leftarrow b_1 + f(0)$;
return $(b_0, b_1, \ldots, b_d)$
```

References

1. Blakley, G.R.: Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 National Computer Conference, vol. 48, pp. 313–317 (1979)
2. Carlet, C., Goubin, L., Prouff, E., Quisquater, M., Rivain, M.: Higher-order masking schemes for S-Boxes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 366–384. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34047-5_21

3. Carlet, C., Prouff, E., Rivain, M., Roche, T.: Algebraic decomposition for probing security. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 742–763. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_36
4. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
5. Coron, J.-S., Roy, A., Vivek, S.: Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 170–187. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_10
6. Eve, J.: The evaluation of polynomials. Numerische Mathematik **6**, 17–21 (1964)
7. Goudarzi, D., Rivain, M., Vergnaud, D., Vivek, S.: Generalized polynomial decomposition for S-boxes with application to side-channel countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 154–171. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_8
8. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_27
9. Knuth, D.E.: Evaluation of polynomials by computer. Commun. ACM **5**(12), 595–599 (1962). https://doi.org/10.1145/355580.369074
10. 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). https://doi.org/10.1007/3-540-68697-5_9
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
12. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44706-7_11
13. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_28
14. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)

Practical Crypto Application

Concealed Communication in Online Social Networks

Fabian Schillinger(✉) and Christian Schindelhauer

Computer Networks and Telematics, Department of Computer Science,
University of Freiburg, Freiburg im Breisgau, Germany
{schillfa,schindel}@tf.uni-freiburg.de

Abstract. Online social networks are used frequently: staying in contact with friends and sharing experiences with them is very important. However, users are increasingly concerned that their data will end up in the hands of strangers or that personal data may even be misused. Secure OSNs can help. These often use encryption to keep the communication between the participants incomprehensible to outsiders. However, participants in such social networks cannot be sure that their data is secure. Various approaches show that even harmless-looking metadata, such as the number of contacts of a user, can be evaluated to draw conclusions about a user and the communication. These attack methods are analyzed and existing secure OSNs are examined, whether these attack methods can be utilized to violate the user's privacy. To prevent these privacy attacks, protocols for a secure centralized OSN are developed. Metadata is obscured in the presented OSN and end-to-end encryption is used for secure communication. Additionally, communication channels are concealed like in mix networks such that adversaries cannot determine which user is accessing which data or which user is communicating with whom even with full access to the server.

Keywords: Online social networks · End-to-end encryption · Privacy · Security · Metadata · Attack · Mix network

1 Introduction

Many online social networks (OSNs) vie for the users' favor. They offer different unique selling points to make the user experience as good as possible so that users spend a lot of their time in these networks. Because, in many OSNs, money is earned through advertising, the more users are on the network and the longer they stay there, the greater the advertising income. These can be slightly increased even further if personalized advertising is displayed [14]. This is tailored to the respective user and increases the likelihood that they will react to the advertising and thus increases the profit of the network operators. In doing so, users can lose their anonymity. Personalized advertising is possible by evaluating profiles, contacts, or reactions to messages in social networks, etc. These practices are a thorn in the side of many users. That is why they are increasingly

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 117–137, 2021.
https://doi.org/10.1007/978-3-030-80851-8_9

using privacy-friendly social networks or those they consider privacy-friendly, such as Telegram [25]. Furthermore, it is often not only the operator of a social network that analyzes the personal data. Also third parties, analyze the data they have access to. Often, apps or games in OSNs or code in displayed advertisements can gain such access. Third parties can do harmless things with the data, but they can also display malicious behavior and misuse the collected data for personal theft, for example. This may cause a rethinking of many users which leads to them registering in social networks that supposedly protect their privacy. This often is achieved by using encrypted communication. But privacy can be attacked and the sovereignty of the users over their data undermined no matter, whether the communication is encrypted or the user profiles are protected, but the metadata is accessible and analyzed. Research shows, that the possibility of privacy leakages through metadata should not be taken lightly.

Our Contribution

Our contribution consists of: first, a summary of possible privacy leakages in privacy-preserving online chats and online social networks. Second, an analysis of privacy-preserving OSNs is given, according to the privacy leakages. Third, following these findings, protocols are presented to create concealed channels between participants in an OSN, which relies on a client-server architecture. These concealed channels work comparable to a mix network. The channels provide end-to-end encrypted communication between two or more participants and further do not leak any evaluable metadata to the service provider of the OSN or another attacker. With concealed channels different possibilities are presented to provide the functionalities, an OSN should provide. These are, for example, profiles of the users, private messages between two or more participants, and discussion groups. A prototype of the scheme using a C# server and a HTML and JavaScript frontend is used to evaluate the impact on run-times when using the cryptographic algorithms. The presented approach of a secure and privacy-preserving OSN is analyzed, whether the possible privacy leakages are prevented.

Organization of the Paper

The paper is structured as follows: Sect. 2.1 summarizes different approaches to analyze metadata in online social networks and encrypted communication channels. Based on these works, possible leakages are displayed in Sect. 2.2. Following, in Sect. 2.3, different protocols for privacy-preserving online chats and online social networks are analyzed, whether they are secure against the leakages. In Sect. 3, protocols are presented to achieve secure communication. These protocols are used to construct a privacy-preserving OSN with encrypted communication. For the presented OSN, then, privacy-preserving functionalities are analyzed and compared to the previously found leakages in Sect. 4. Finally, Sect. 6 concludes the work and gives an outlook on future work.

2 Related Work

In the following, different possibilities to undermine user privacy, using metadata are collected. Then, privacy-preserving online social networks are analyzed, whether they are prone to these privacy leakages.

2.1 Privacy Analysis in Online Social Networks

The attributes users post in their profiles in an OSN can be used to predict the attributes of other users, according to [16]. The dataset of the Rice University network and the New Orleans Facebook dataset are used. One of the findings is, that friends are likely to share common attributes and these common attributes form groups. Using these findings, on the one hand, it is possible to predict the social circles of users. Using the social circles it is, on the other hand, possible to predict attributes of users.

In [23] two datasets of the OSNs Flickr and Last.fm, and in [1] three datasets of Flickr, Last.fm, and aNobii are analyzed to predict social links. In these OSNs, users can communicate, form groups of similar topics, create links to other users, and use tags to annotate content, such as shared pictures. Various observations are made: users that have more contacts tend to be more active regarding tagging and membership of groups, assortative mixing of nodes in the OSNs is detected, and different patterns of topic similarity between neighbors are found. For assortative mixing, i.e. the observation, that nodes tend to be linked to nodes with similar properties, various properties are investigated. For example, the degree of nodes, especially the nearest neighbor's degree, average number of tags of nearest neighbors, or average amount of groups. Using these observations similarity between users is calculated to predict social links with high accuracy.

Privacy leakage through metadata of decentralized OSNs is examined in [10]. Adversaries with distinct possibilities to access the data are considered. In a centralized setting, the service provider combines all of them. From the metadata of content, different observations are possible. The size of an object is an indicator for the type, as text messages are smaller than images or videos. From the structure of a group of elements conclusions can be drawn, e.g. amount of images in a shared album. The modification history of an object can reveal information, e.g. about user status updates or intensity of activity. Other information can be obtained from access control mechanisms, like encryption headers. Here, header sizes can allow estimating the number of encryption keys. Adding encryption keys or revoking them, can lead to re-encryption of contents and can allow drawing conclusions about changes in the relations between users. From re-using the same key for different objects one can learn about overlapping access rights. The communication flow can lead to more information. From tracking IP addresses, conclusions about online times or working habits can be drawn, using geo-IP mapping services routes, locations, and traveling information can be tracked. Access logs from shared content can be used to determine user-groups, ownership, and access patterns. Timing information may be obtained from newly created objects and (re-)distribution of keys. Different control-operations, like login, adding friends, or searches can be observed.

Metadata of Twitter is analyzed in [19]. A tweet contains 144 fields of metadata. This allows drawing conclusions about the owner of such a post, without considering the actual content. Using machine learning approaches owners of posts can be identified from a group of 10.000 users with 96.7% accuracy.

Identifying social circles from network structure and user profile information is the task in [15]. Using machine learning approaches, a model is created that accurately identified social circles in Facebook, Google+, and Twitter.

Patterns in user behavior can be recognized even with encrypted traffic. In [5] encrypted traffic of different sequences of actions are collected for popular Android apps. A sample sequence for the Facebook app is to tap on the button to write a post, fill the textbox with some random text, and post this message. Analyzing the network flow as a set of time series it is possible to predict different patterns even with TLS/SSL encrypted traffic.

2.2 Privacy Leakages

From the findings in Sect. 2.1 the following list summarizes possible problems when metadata can be accessed in an Online Social Network where communication is end-to-end encrypted. Some of the problems alone may not necessarily lead to privacy leakages, but the combinations of different problems can lead to severe violations of privacy.

Structure of Network (*NS*)

- Degree of Node (*NS1*). How many contacts does a user have?
- Neighbors of Node (*NS2*). How many contacts do the contacts of a user have?

Structure of Data (*DS*)

- Count of Objects, Groups, Keys, ... (*DS1*). How many contacts, posts, messages, ... does a user, group, ... have?
- Common Objects, Groups, Keys, ... (*DS2*). Which contacts, posts, comments, ... are common between users, groups, ... ?
- Size of Objects or Groups (*DS3*). What is the type of an object (text, media, ...), how many users are in a group?
- History of Objects or Groups (*DS4*). How often does an object or group change?

Timing (*T*)

- Time Series Pattern (*T1*). Are objects or keys accessed in a specific order, especially with a specific timing?
- Timing for Creating Objects and Distributing Keys (*T2*). Which keys are associated with which objects?

- Key Distribution and Re-Encryption (*T3*). Which keys belong to with which objects and which keys are created or deleted when objects are changed?

Control Information (*CI*)

- IP Address Logging (*CI1*). Track user behavior when the same IP accesses content, or different IP addresses access the same content, especially when the time of day is the same across different days.
- Geo-IP Mapping (*CI2*). Where is a user, which locations are associated with a user?
- Access Logs for Objects, Groups, or Ownership (*CI3*). Which user accesses which objects, groups, ...?
- Control Messages and Queries for Login, Friend Request, or Searches (*CI4*). What is a user doing in the OSN?

2.3 Privacy-Preserving Online Social Networks

todo. Various schemes introduce encryption of messages between two or more participants. Some of the schemes are designed for emails, others are designed especially for online message services. Most of the schemes rely on a client-server structure, but there are peer-to-peer approaches, as well:

A scheme for encrypted online chats is *Off-the-record* (OTR) [18]. The scheme uses new session keys k_i for each message i. Each key is negotiated through a Diffie-Hellman key exchange between two participants A, B with keys x_{Ai}, x_{Bi}, where keys x_{zi}, x_{zj} for $z \in \{A, B\}, i \neq j$ are independent. This introduces perfect forward secrecy for the communication. Possible leakage vectors could be: *DS3* and *T1* because an adversary could track the sizes and sending times of messages. When a dedicated server is used for the communication, additionally, *NS*, *CI1*, and *CI2* can be exploited by the server.
A peer-to-peer system for end-to-end encrypted messages between two participants was *Silent Circle Instant Messaging Protocol* (SCIMP) [17]. Elliptic Curve Diffie-Hellman was used to agree on a shared key for encrypted messages. As SCIMP was a peer-to-peer approach all leakages utilizing a server are mitigated, still the leakage vectors *NS*, *DS* and *CI* could be exploited by a service provider or another malicious relay.

Private Facebook Chat (PFC) [21] introduces end-to-end encrypted chats inside Facebook. The key distribution works by dedicated servers, that use the Facebook authentication mechanisms. Nearly all leakage vectors seem exploitable, except for *T2* and *T3*. No matter, whether the Facebook servers or the PFC servers are considered because an adversary on either of the servers can access all metadata.

Multiple peers can communicate securely using the approach described in [11]. A modified Diffie-Hellman protocol is used to find a common secret for the participants with the help of a server. The leakage vectors *NS*, *DS* and *CI* can possibly be exploited, because a server is used to manage the communication and keys.

A comparable approach is discussed in [30]. Here, elliptic curve Diffie-Hellman is used for the key agreement. Therefore, the same leakage vectors *NS*, *DS* and *CI* can possibly be exploited.

In the *Signal* protocol [13] a shared secret between participants is derived from a key chain. The inputs for this chain are found through a modified Diffie-Hellman key exchange. Messages are encrypted and new keys are used for every message. Still, the leakage vectors *NS*, *DS* and *CI* are possible, when the server is attacked.

Threema [26] allows end-to-end encryption of messages. Communication between two peers is encrypted using a shared key. Messages in groups are encrypted for each peer individually, using their public keys. Larger files are encrypted using a symmetric key, which is encrypted with each participant's public key. When the server is attacked, all leakage vectors *NS*, *DS*, *T*, and *CI* could be exploited.

Another end-to-end encrypted online chat is presented in [24]. Messages are encrypted using symmetric AES encryption. The necessary keys are encrypted for all participants of a chat using their public RSA keys. A chat can contain arbitrarily many participants and the number of participants can be changed, then new AES keys are generated and distributed. There are various possible leakage vectors when the server is attacked: all from *NS*, *DS*, *CI*, and *T*, because all necessary data to manage the OSN are stored in plaintext on the server.

Pretty Good Privacy (PGP) [9] and *S/MIME* [22] are methods for the encryption of emails. In both, public-key encryption is used to encrypt a symmetric key for every recipient of an email. The content of an email is encrypted, using the symmetric key. All following answers use the same keys. The main difference is in the verification of public keys: PGP constructs a trust system between participants, whereas S/MIME uses X.509 certificates. Possible leakage vectors are: attNetwork and *DS*, because recipients are known in plaintext. Leakages from *CI* are possible, if the adversary is one of the involved mail servers.

Other approaches introduce privacy into Online Social Networks:

FlyByNight [12] introduces client-side based encryption of content for Facebook. Each user has a password that is used to encrypt a private key for the key database of the system. Messages between participants are encrypted using their public keys. Proxy cryptography is used when more than two participants communicate. As flyByNight is an extension to Facebook all leakage vectors could be used when considering their servers. When considering only the servers of flyByNight, still, *NS*, *T*, *CI*, *DS1*, and *DS3* could be used, because the server manages all keys and messages. A decentralized OSN is *Safebook* [6]. Social circles from the real-life are used to construct trust relationships. Each node is surrounded by those structures, which are called matryoshkas. This is used to provide data storage and communication privacy. Another layer is a peer-to-peer network that enables application services, such as lookup. The internet is the transport layer in the scheme. Because of the peer-to-peer approach, most of the leakages are prevented, or at least very unlikely. E.g. to exploit *NS*, *DS*, or *T*, trusted peers have to attack the user. Attacks through *CI* are unlikely because of the peer-to-peer structure.

In [7] another decentralized OSN is presented. Again, real-life trust relationships are utilized for trustworthy connections within the network. Multihop routing between trusted peers is used as an anonymization technique. Privacy leakages are unlikely, because, for *NS*, *DS*, or *T* trusted peers have to attack the user.

In the OSN *Persona* [2] users define who can access their information. Attribute-based encryption is used to share secrets within groups of participants that have at least one attribute in common. Further, each user owns a key-pair, such that the public key can be used to encrypt content specifically for this user. Although communication data is encrypted, various usable metadata may accumulate on the server, therefore, exploits of *NS*, *DS*, *T*, and *CI* seem possible.

Snake [3] is an OSN which is written in HTML5 and JavaScript. It uses the WebCrypto API to encrypt messages between peers. One can not conclude which peers communicate, because addresses are masked inside the database. When users establish a friendship they agree on a shared key and addresses to send and receive messages. These addresses change with every new message. Therefore, exploiting *NS* and*DS* seems to be not possible, when considering communication between peers, but *T* and *CI* could be exploited. When a user logs in the server has to prove the necessary encrypted data and exploits of *DS1*, *DS3*, and *DS4* can be possible.

Vuvuzela [28] is a system to ensure private messaging. Most of the metadata of communication is hidden by the system. The participants use locations, called dead drops, to place messages for other users. Managing messages is performed in a round-based approach by multiple servers. A user sends its request messages, like placing messages to or retrieving messages from dead drops to a server. After a round, the collected operations are performed by the server. Then, dead drops are discarded and cannot be accessed in further rounds. All messages that are not delivered, are deleted. Privacy is preserved through different measures: first, a constant amount of messages and sizes of messages for each user per round is set, where messages are delayed to other rounds or empty messages are introduced. Second, the servers work as a mix network to anonymize dead drops. Each message is encrypted for the next server, such that no server can determine, which message comes from which client. The architecture prevents most of the attacks, the only observable information is the number of participants in a communication (*DS3*). Further, the protocol only ensures messages between users, other desired information exchange of OSNs is not supported. This includes all persistent information, like user profiles or groups.

The protocol *Stadium* [27] allows private messaging, comparable to Vuvuzela. Here, messages are, again, exchanged via dead drops and a mix network is set-up between the senders and recipients of messages. The protocol reduces the amount of noise needed in the network, to keep the communication private, compared to Vuvuzela. Still, persistent information cannot be stored, using the protocol.

Another protocol that utilizes mix networks is *Loopix* [20]. Again, no persistent information is stored, apart from the information that is temporarily stored on so called providers, when the recipient is offline. But, as soon, as the information is delivered it is removed from the provider, as well.

2.4 Mix Networks

In [4] mix networks are presented. A mix network is a routing protocol where a message is sent to a proxy, called mix, that forwards the message to another mix or the recipient. When layered encryption is used between the participants the path of a message becomes hard to trace. This allows achieving anonymity of the sender, of the recipient, or both. Consider a network with sender S, recipient R, and mixes $M_1, M_2, \ldots, M_n$ with public encryption keys $s, r, m_1, m_2, \ldots, m_n$ and a message N.

Anonymity of the sender is achieved by encrypting the message to $n = \{\{\{\{N\}_r\}_{m_n} \cdots\}_{m_2}\}_{m_1}$. n is sent to M_1, who decrypts the message and forwards it to M_2, and so on. Finally, R receives $\{N\}_r$ from M_n. Because each recipient of the message only knows the participant before, and after, the flow of the message is concealed and R does not get to know that S is the sender.

The anonymity of the recipient is provided when the recipient generates a return address. The return address is an encrypted sequence of mixes that the sender has to use, where only the first mix is known to the sender. The recipient sends this sequence to the sender, via the mix network.

By combining both schemes, the sender and the recipient of a message can remain anonymous to each other.

Different methods are needed, that outsiders cannot track messages through a mix network, such as removing of duplicate messages. This would allow an attacker to find a connection between a received message and the next mix because the duplicate has to be sent to the same receiver. Further, messages have to be modified by a mix to prevent comparing incoming with outgoing messages. Additionally, messages at a mix have to be collected and either forwarded at random or together with other messages and the forwarding procedure has to produce a different ordering of messages.

3 Proposed Protocols

The following protocols ensure private communication within an Online Social Network (OSN), based on a client-server architecture. No metadata is leaked when users communicate. This is achieved through concealed communication channels.

3.1 A Concealed Secure Channel

In a client-server model, a concealed channel between two clients uses the server. Such a channel can be created when one of the clients has a concealed channel to an address on the server, which is known to the other client and accessed through a concealed channel. This address is called *concealed address*.

Definition 1 (Concealed Address). *A tuple* (c, p_R, p_W, p_O)*, where* c *is an unique address at the server is a concealed address.* p_R*,* p_W*,* p_O *are values to prove that one is allowed to read or write messages to this address, or that one is the owner of the address.*

A *concealed address* can be created by sending a CreateAddress-Message to the server. The message contains three corresponding values, called *address keys*: p_R, p_W, and p_O. The *concealed address* provides an address on the server where clients can read messages from or write messages to, when the according proofs are provided. Another proof determines the ownership of the *concealed address*. The *address keys* are used to provide these proofs.

Definition 2 (Address Key). *An address key is a value* p_R*,* p_W*, or* p_O*. For reading or writing messages, or for proving ownership of an address. Each address key is a public-key of a cryptographic protocol for signatures. An address key can have a wildcard-value* $*$*.*

Clients can prove that they are allowed to read messages from the *concealed address* by using the p_R value, which is called *read address key*. p_W, the *write address key*, is used to verify if a client is allowed to write messages to it. It can be proven that the client is the owner using the *owner address key* p_O. For the verification, the message is signed by the client using the corresponding private-key. To ensure enough entropy in a message, a (signed) random nonce is always part of the message. Then, multiple (similar) requests from the same *concealed address* always have a different signature. This prevents replay-attacks. When using secure cryptographic protocols, the private-key cannot be computed from the public-key. Therefore, providing the correct signature to a message proofs the knowledge of the private-key. This ensures, that a client is allowed to read or write messages, or modify a *concealed address*. The proofs are only a verification against the server. Messages stored at a *concealed address* can be encrypted using a *content key*. Without knowledge of the *content key* no client can decrypt the messages from a *concealed address*.

Definition 3 (Content Key). *A content key is a value* k*, which is used as a key for the encryption and decryption of messages stored at a concealed address.*

The *content key* normally is a symmetric key, known to all participants because of the increased speed compared to public-key cryptography. However, when no symmetric key is established, one of the participants can choose it and encrypt it once with the public-key of each participant. In this case, there are multiple *content keys*. Further, all messages at a *concealed address* can have different *content keys* or are not encrypted, at all. The *content key* and *address keys* are independent from each other. They can be negotiated between the participants or distributed in person and later changed between messages. Using multiple *concealed addresses* concealed communication channels, comparable to a mix network (see Sect. 2.4), can be established. A participant of the OSN, that wants to provide a mix network generates multiple *concealed addresses*, where the *write address keys* are $*$. This allows everybody, knowing one of the addresses, to write messages to it. When the messages are encrypted with, either the public key of the mix or a symmetric key, only known to the mix and the sender, they can be decrypted by the mix, only. These messages, then, contain another encrypted message for another mix or a recipient and an address, where it has to be written to. Messages are collected by the mix, decrypted, reordered, merged if possible, and then written to the specified addresses. Messages can be merged when they have the same target address. Then, they are concatenated and written as a single message. For increased security, duplicate messages are deleted. Using this construction, a communication between a sender and a receiver can be established, where the sender is anonymous, or the receiver is anonymous, or both are anonymous to each other. To reduce the number of messages stored to the server, mixes delete messages, after they are forwarded. However, to prevent attacks with duplicate messages, the mixes have to wait a certain amount of time, depending on how many messages they receive and have to forward. To further decrease the storage needed, the digest values of messages can be stored instead. Together with a sliding limited window of acceptance replay attacks can be prevented. Here, such a window only allows messages that are not older than a specified timespan. When a path is used that achieves anonymity of the sender, this anonymous sender can prove ownership of an address to the server. This allows participants of the OSN to exchange addresses. E.g. a user A can create a *concealed address* c_A with some *address keys*. A can send the *owner address key* via a concealed channel to B. Then, B can prove ownership of the *concealed address*, via another concealed channel. Exchanging *concealed addresses* between users and mixes cannot be tracked by the server or another user (Fig. 1).

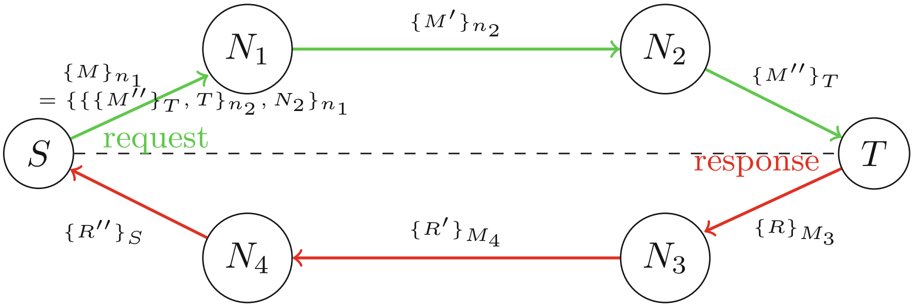


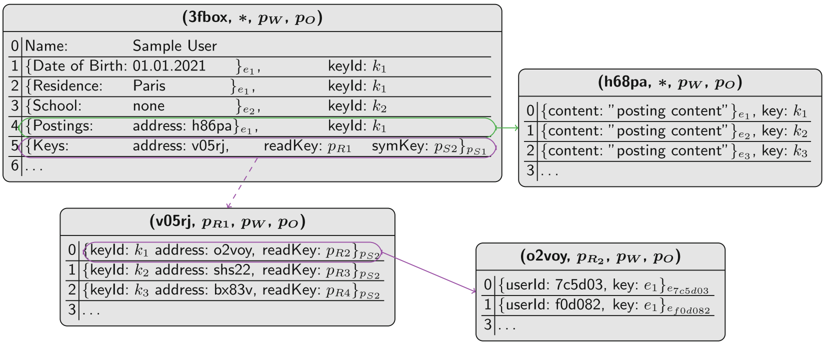
Fig. 1. An exemplary message flow from sender S to receiver T over the nodes N_1 and N_2. $\{M\}_{n_1}$ can be decrypted by N_1 and contains the target address of N_2. Finally, message $\{M''\}_T$ is received by T and decrypted. The return path across N_3 and N_4 is encrypted inside the message, such that T does not need to know the addresses of N_3, N_4, or even S. I.e. M'' is a tuple $(m, N_3, e_{n_3}, \{(N_4, \{S\}_{n_4})\}_{n_3})$, where m is the message for T, e_{n_3} is the public-key of N_3, and the last ciphertext contains the return addresses for nodes N_3 and N_4. These addresses are encrypted and can only be accessed by the respective node.

3.2 User Information and Postings

Sharing personal user information, like the date of birth or residence, is part of social networking. In many OSNs users can create profiles to share such user information. In our approach, it is possible to create a user profile using *concealed addresses*. The user can decide, how the profile is stored at the server: a user profile can consist of a single *concealed address*, containing all the information a user wants to share. This information can be encrypted or stored as plaintext at the *concealed address*. The user then decides who receives the *concealed address*, the *read address key*, and, in case of encrypted information, who receives the *content keys*. Storing the user profile as a whole at the server is possible but not recommended: on the one hand, a user often has to update the whole profile or re-encrypt the contents, when some information or access rights are changed. On the other hand, defining fine-grained access rights is complicated. A user can share some of the information with everybody in public, while other information is shared with specific contacts, only. To achieve this, a user has to distribute multiple different keys. It is recommended that the user creates a *concealed address*, where the *read address key* is $*$. This *concealed address* is used to store all addresses of the profile information, like a *concealed address* containing the public keys of the user, so other users can verify signatures or send encrypted messages, another *concealed address* for the date of birth, another *concealed address* for the residence, and so on. Then each linked *concealed address* can have unique *address keys*. To make the tracking of user information through linked addresses difficult, each linked address itself can contain another linked address, which is encrypted. This prevents an adversary from learning which addresses are connected.

A user may want to share postings. These can contain the personal experience, for example, pictures from the last holidays. This information can be stored at the user profile, like personal information, or the user can link a single, or multiple *concealed addresses*. Each address can contain a list of posts. This allows the user to create different information feeds, where each feed can have different keys, and therefore different access rights.

The construction of a user profile using *concealed addresses* allows not only to post personal profile information and user posts, but any type of information: the user may share a calendar with fine-grained access control over each appointment, different blogs or vlogs, picture albums, or wikis. The user can create different *concealed addresses*, where the *write address keys* are $*$. These addresses work as different pinboards, where other users can post messages visible to the user and everybody knowing the correct *read address keys* and *content keys*. Further, the user can use private *concealed addresses* to store notes, bookmarks, or other private information, like passwords. For every information, the user can create a new *concealed address*, a new *address key*, and *content keys* to be able to define the access rights every time. Using *concealed addresses*, the user can create collections of keys. These collections can be shared with single users or with groups of users. An exemplary user profile is displayed in Fig. 2.



(3fbox, $*$, p_W, p_O)

0 Name: Sample User
1 {Date of Birth: 01.01.2021 $\}_{e_1}$, keyId: k_1
2 {Residence: Paris $\}_{e_1}$, keyId: k_1
3 {School: none $\}_{e_2}$, keyId: k_2
4 {Postings: address: h86pa$\}_{e_1}$, keyId: k_1
5 {Keys: address: v05rj, readKey: p_{R1} symKey: $p_{S2}\}_{p_{S1}}$
6 ...

(h68pa, $*$, p_W, p_O)

0 {content: "posting content"$\}_{e_1}$, key: k_1
1 {content: "posting content"$\}_{e_2}$, key: k_2
2 {content: "posting content"$\}_{e_3}$, key: k_3
3 ...

(v05rj, p_{R1}, p_W, p_O)

0 {keyId: k_1 address: o2voy, readKey: $p_{R2}\}_{p_{S2}}$
1 {keyId: k_2 address: shs22, readKey: $p_{R3}\}_{p_{S2}}$
2 {keyId: k_3 address: bx83v, readKey: $p_{R4}\}_{p_{S2}}$
3 ...

(o2voy, p_{R_2}, p_W, p_O)

0 {userId: 7c5d03, key: $e_1\}_{e_{7c5d03}}$
1 {userId: f0d082, key: $e_1\}_{e_{f0d082}}$
3 ...

Fig. 2. A user profile solely relying on *concealed addresses*. The construction allows a user to hide its personal information, except for the name. The fields are encrypted using keys with Ids $k_1, \ldots, k_3$, which are stored in a *concealed address*, which is not directly, i.e. in plaintext, linked from the profile. When a user receives the key p_{S1}, it can decrypt the entry containing the keys. Then, the *concealed address*, with the keys can be accessed. Each key is encrypted, using a user-specific key. Using these keys, the fields from the profile can be decrypted and their *concealed addresses* can be read.

3.3 Personal Messages and Group Messages

Personal messages or group messages are possible between two or multiple participants. One user creates a *concealed address* and shares the *read address key* and *write address key* with the other participants. The *content key* can be chosen by this user and shared, using the public keys of the other participants. Then, the participants can write messages to this *concealed address* and read the incoming messages. Whenever a new participant is added to a chat the *read address key*, *write address key*, and *content key* is shared with this user, via an encrypted message, using the public key. When a user is removed from the chat the owner of the address, knowing the *owner address key*, can change the *read address key* and *write address key*, and a new *content key* can be generated and shared with the remaining participants.

3.4 Groups

A group in an OSN is a collection of information, messages, posts, pictures, etc., comparable to a user profile. The difference is, that everything in the group may be accessed, modified, or created by multiple users. Additionally, some parts of the group may be changed by some participants. A group can be constructed like a user profile. One *concealed address* is used as a collection of different linked *concealed addresses*, containing the different information shared in the group. Because each *concealed address* can have a unique key, again a fine-grained access system can be created, where some information in the group can be accessed by all members of the group, where other information is hidden to some members. At some addresses, members of the group can add or edit information, at other addresses, only a specified group of administrators of the group can change information. The necessary keys can be shared between the authorized users, via personal messages or shared *concealed addresses*. The construction of groups and user profiles, using *concealed addresses*, allows users to join groups, without exposing their information. Then, they just read the *concealed addresses* of this group. When users want to be visible within the group, they can publish their profile's address or their name, together with a signature, within the group.

3.5 Finding and Verifying Participants, Exchanging Keys

A user has the freedom to construct a profile, such that no other user can find him. This can be achieved when no profile information is published, or all of the user's profile fields are encrypted. Even, when all addresses on the server are tried, the corresponding *read address keys* and *content keys* have to be known. Further, if a user has encrypted profile fields, they cannot be found through the OSN. An attribute-based search for participants can be prevented. Nevertheless, when a user publishes some fields, an attribute-based search is possible. When a user does not want to be found through the OSN, the only way to find him is by receiving the needed *concealed addresses* and keys through a different channel. A channel can be via phone or by meeting in person. The user can generate a fingerprint, like a QR code or a textual representation of the *concealed addresses* and keys via a method described in [8]. This is comparable to the effort of exchanging usernames for any OSN, where the participants do not use their real names.

Public keys of other users can be verified with this method, as well: when participants A and B want to verify their keys, they call a procedure that concatenates all the public keys of A and B in a predefined order, like by ascending id of the public keys. Then a fingerprint is generated by both users A and B and compared. When both fingerprints are equal, they can be sure that the public keys are equal, and no third party has injected a wrong key. When A and B trust each other, they can use this procedure to verify the keys of other participants, as well, and construct a web of trust this way. As soon as two participants have established a trusted channel they can exchange encrypted messages through a *concealed address*. This allows them to further exchange and verify keys. When users have a verified *concealed address* to exchange messages they can work as mixes for each other. By forwarding messages to other mixes, or participants of the OSN.

When a user created a profile comparable to Fig. 2, other users can find him through the OSN, because the server can read the entry containing the name in the profile address. After all, anybody can read the *concealed address*, and the entry is not encrypted. A similar public field can be used in groups, or by mixes, as well.

3.6 Key Revocation

The revocation of keys can be achieved in multiple ways. When the *address keys* for an address are changed, a user is not allowed to access the content. Then, the *address keys* have to be redistributed to the remaining parties. Another possibility is, to move the content from the *concealed address* to another *concealed address* and carry on all communication via the new *concealed address*. Then, the new *concealed address* and the appropriate keys have to be transferred to the remaining parties. Further, the *content key* can be changed for a *concealed address*. Then, the excluded party cannot decrypt new content. Here, the *content key* has to be transferred to the remaining parties. For every method, the needed keys or *concealed addresses* can be transferred via direct messages over known *concealed addresses*.

3.7 User Registration

User registration is not needed in the proposed scheme. Every user can create a *concealed address* or multiple *concealed addresses*. This allows, that users that do not want to create a user profile to still communicate in the proposed OSN. Without registration, there is no need for authentication. Still, the server is not more vulnerable to attacks like DOS than any other server, as flooding can be prevented by the appropriate measures.

4 Privacy Analysis

The privacy of the users in the presented OSN is based on two different approaches that work together. First, all communication between users is encrypted. When using appropriate algorithms, this prevents any third person from reading messages. Here, a public-key cryptosystem can be used, where all communication uses the verified public-keys of the participants. A symmetric cryptosystem can be used, as well. Then, the keys are exchanged via a public-key cryptosystem. This approach works when multiple participants communicate. The second approach is to hide metadata, like who is communicating with whom. Our approach implements a mix network in a client-server architecture, where all participants and mixes communicate via the server. Multiple methods allow untraceable messages through a mix network: Removing of duplicate messages, such that a third person cannot trace a message and the recipient. This method is implemented by our approach, as well. Modification of messages at a mix prevents an outsider from comparing incoming and outgoing messages. This is implemented, first, when a mix decrypts the message and forwards it. Second, the mix can append any random nonce to a message, encrypt the message with the appropriate key of the recipient, and forward the modified message. Another method in a mix network is to rearrange messages, or to collect some messages and forward them at once. In our approach, this is possible, as well. A mix waits until a specific number of messages has arrived and processes them at once. Further, in our approach, a mix can join multiple messages with the same recipient.

E.g. when there are n incoming messages at a mix and m outgoing messages all three, $n = m$, $n < m$, and $n > m$, are possible. Further, in our proposed protocol, a mix can choose to forward the message to another *concealed address* of the same recipient, because participants can have access to multiple *concealed addresses*.

4.1 Attacker Model

An attacker in the OSN mainly is the provider of the OSN, or any person with access to the server and database. This can be a hacker, a disloyal administrator, or any governmental organization agent. The attacker can read all communication between the clients and the server and all entries from the database. Further, untrusted nodes in the OSN can collude with the server. Messages are not deleted by the attacker, but can be delayed. The following assumptions about the cryptographic protocols and the computational power of the attacker are made: First, calculating private-keys from public-keys for signatures and public-key cryptography is not feasible. Second, ciphertexts of the same size, generated by the same algorithm with different keys are indistinguishable. Third, ciphertexts cannot be decrypted, when the private-keys or symmetric keys are not known. Therefore, traffic analysis is prevented, because ciphertexts are indistinguishable and messages are modified at each mix.

4.2 Structure of Network (*NS*)

The leakages of the degree of a node (*NS1*) can be prevented by our approach. An attacker cannot conclude how many nodes a user knows when the user forwards all messages to the same mix or a fixed number of mixes, as long as, one mix is trusted. It is possible to obtain *concealed addresses* of participants via different channels: Users can exchange their *concealed addresses* when meeting in public. When a secured communication channel is set up, it can be used to further exchange new *concealed addresses* of other participants of a network. When the neighbors of a user are unknown, leakages about the neighbors (*NS2*) are prevented, as well.

4.3 Structure of Data (*DS*)

The structure of the stored data is hidden to any third person, as the contents of any *concealed address* are not given. Even, when an attacker can link a *concealed address* to a user, it is not possible to conclude which data is stored, when the contents are encrypted. Even a combination of different contents can be stored: it is possible to store messages, keys, and contacts in the same *concealed address*. Further, finding a connection between a *concealed address* and a user can be prevented. *concealed addresses* can be interchanged between users. When the *address keys* are exchanged, a *concealed address* can be used by multiple users, as well. When a user can read messages from a *concealed address*, it is possible to create new *concealed addresses* with the server, using this *concealed address*. Then, the server cannot link a user with an *concealed address*. The described methods prevent third parties from concluding the amount (*DS1*), size (*DS3*), and history (*DS4*) of objects, groups, or keys, a user is in possession of. Further, no common objects (*DS2*) can be found, because contents inside *concealed addresses* can be encrypted and random nonces can be used to prevent third parties from comparing ciphertexts.

4.4 Timing (*T*)

Timing information is another leakage vector, which can be prevented by applying our scheme. Messages are collected by any mix and processed in a batch to remove any timing information. Therefore, an attacker cannot find time series patterns (*T1*). The contents of messages cannot be distinguished, which prevents to conclude about the timing between creating objects and distributing keys (*T2*) or the re-encryption of contents after a key distribution (*T3*).

4.5 Control Information (*CI*)

Hiding control information in our approach is possible, as well. IP address logging (*CI1*) and Geo-IP mapping (*CI2*) are possible but can be prevented by users when applying VPN connections or proxies. Further, when a user is sending all messages to the same trusted mix, it is still not possible to link the IP address to the owned *concealed addresses*. Together with *concealed addresses*, which can be read by multiple users, this prevents a third party from creating meaningful patterns. Analysis of access logs for objects (*CI3*) can be prevented because using *concealed addresses* allows users to establish sender privacy. Then, it is not possible to draw conclusions between a *concealed address* and a user. Further, it is not possible to make a connection between users that read from a *concealed address*. Control messages and queries for special operations, like for the login procedure (*CI4*) are not created in our approach. In fact, there are four possible actions: read from a *concealed address*, write to a *concealed address*, delete from a *concealed address*, and create a *concealed address*. This prevents third parties from analyzing other actions.

5 Performance Analysis

The proposed system heavily utilizes public-key cryptography. This can introduce long waiting-times when using the OSN. The verification of access to *concealed addresses* requires messages to be signed by the users and all nodes that are utilized in the mix. Additionally the server has to verify all signatures. To assess the feasibility, a prototype of the OSN was created. The prototype is available at: github.com/falti3/concealed. The server is mainly written in C# and provides different web API end-points. The signature-verification procedures are performed by NodeJS running JavaScript. The server-side controller accepts messages to create *concealed addresses*. Messages can be read or deleted from and written to *concealed addresses*. For all incoming messages, the verification of the appropriate key p_R, p_W, or p_O is conducted. To store messages and addresses, a MySQL database is connected. On the client-side, a single-page HTML application with JavaScript is used. All cryptographic protocols are performed by utilizing the WebCrypto API [29]. The specific algorithms are as follows: symmetric encryption of messages (*content keys*) is performed by AES256 with CBC. Signature generation and verification (*address keys*) is performed using RSA with OAEP, modulus length 4096, and SHA256 for the key generation. The HTML application provides the basic functionalities: creating *concealed addresses*, writing messages to *concealed addresses*, and reading and deleting messages from *concealed addresses*. Mixing of messages can be simulated, by listening to addresses, forwarding the incoming messages, and answering to requests. Further, a user profile similar to that in Fig. 2 can be accessed through the client. For three message sizes of 0.5 kB, 500 kB, and 5 MB multiple random messages were created, written, read, and finally deleted again to measure the run-times of the cryptographic operations. The simulations were performed on a laptop with an Intel® Core™ i7-8550U CPU with 1.8 GHz and 16 GB of RAM. The server and MySQL database were stored on a SSD. The measurements are displayed in Table 1. The signature verification on the server was for the read and delete operations on average faster than 3.2 ms. For messages of sizes 50 kB and 5 MB, 6.81 ms and 555.12 ms were measured. This can be optimized: first, when not the whole message is verified by the server, but only a part of the message. Second, when the verification procedure is carried out in C# and not on a different NodeJS process. Still, the durations are acceptable, as most of the messages are smaller than 50 kB. On the client-side, the signature generation was faster than 4 ms for the read and delete operations and the write operations with messages of sizes 0.5 kB and 50 kB. The signature generation for 5 MB messages was 268.94 ms, which is acceptable. Again, this can be reduced, when only parts of the message are signed. For the decryption times, 1.2 ms and 1.83 ms were measured for the smaller messages. 27.66 ms were measured for 5 MB. This is acceptable and faster than the transmission time of the message between the client and the server, in most cases. When transferring messages over a mix network the durations add up, because they are performed sequentially at each node and for every message the server verifies the signature. However, three mix nodes most of the time are sufficient. When transferring files of 5 MB to the OSN still, the combined computation time for the signatures is faster than 1 s. This is acceptable. For smaller messages, that are more common, the delay generated by the cryptographic procedures most likely is not noticeable for an user.

Table 1. To evaluate the performance of the scheme, multiple write, read, and delete message operations were conducted. The following durations were measured, where "sign/encrypt" and "decryption" were measured on the client-side and the "signature verification" was measured on the server. "sign/encrypt" denotes the complete operations to encrypt a message with AES and create the signature with RSA. "signature verification" denotes the operation to verify the signature of an operation on the server. With "decryption" the operation to decrypt an encrypted ciphertext is denoted.

	Operation								
	Write			Read			Delete		
Message size	0.5 kB	50 kB	5 MB	0.5 kB	50 kB	5 MB	0.5 kB	50 kB	5 MB
Sign/encrypt (ms)	1.35	3.84	268.96	1.05	0.88	1.11	1.03	1.22	3.84
Signature verification (ms)	1.67	6.81	555.12	1.17	1.31	2.39	1.21	2.00	3.12
Decryption (ms)				1.20	1.83	27.66			

6 Conclusions

We have presented a summary of different possible privacy leakages within secure online social networks and a discussion on whether different approaches for privacy-preserving OSNs are possibly vulnerable to these leakages. Either, the OSNs are possibly vulnerable or are using a peer-to-peer architecture. Therefore, we presented a way to construct privacy-preserving, encrypted, concealed channels in a client-server architecture. No evaluable metadata is generated when using these channels, according to the previous findings of possible privacy leakages. Using these concealed communication channels different protocols are presented to provide the full functionality of OSNs. These functionalities contain a user profile, where profile fields, contact lists, pinboards, etc. can be encrypted and hidden for the server, an attacker, or any third party. Further, private messages between two or more participants of the OSN are provided. Another presented functionality are groups. A group can contain message boards, calendars, pinboards, etc. The groups can be accessed by multiple participants. Some of the participants may be only able to read content, whereas, other participants can produce content and publish it in a group. Protocols to verify public keys via secure channels are discussed, as well as the interchange of concealed addresses.

One of the main advantages of the OSN, however, can be considered as the main weak points as well: profile information or concealed channels can be hidden from the server, to prevent the server or a third party from evaluating this information. On the other hand, this prevents any user from searching this informations through the server. This means that two participants have to meet via a different channel in order to exchange the information they need in order to ultimately be able to conduct private communication via the OSN. However, this hurdle is comparable to exchanging usernames of any OSN, where the participants do not use their real name.

Acknowledgements. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the framework of SoNaTe (project number 16SV7405).

References

1. Aiello, L.M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., Menczer, F.: Friendship prediction and homophily in social media. ACM Trans. Web **6**(2) (2012). https://doi.org/10.1145/2180861.2180866
2. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. In: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, pp. 135–146 (2009)
3. Barenghi, A., Beretta, M., Di Federico, A., Pelosi, G.: Snake: an end-to-end encrypted online social network. In: 2014 IEEE International Conference on High Performance Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, 2014 IEEE 11th International Conference on Embedded Software and Systems (HPCC, CSS, ICESS), pp. 763–770. IEEE (2014)
4. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
5. Conti, M., Mancini, L.V., Spolaor, R., Verde, N.V.: Analyzing android encrypted network traffic to identify user actions. IEEE Trans. Inf. Forensics Secur. **11**(1), 114–125 (2016)
6. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a privacy-preserving online social network leveraging on real-life trust. IEEE Commun. Mag. **47**(12), 94–101 (2009). https://doi.org/10.1109/MCOM.2009.5350374
7. Cutillo, L.A., Molva, R., Strufe, T.: Privacy preserving social networking through decentralization. In: 2009 Sixth International Conference on Wireless On-Demand Network Systems and Services, pp. 145–152. IEEE (2009)
8. Dechand, S., Schürmann, D., Busse, K., Acar, Y., Fahl, S., Smith, M.: An empirical study of textual key-fingerprint representations. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 193–208 (2016)
9. Finney, H., Donnerhacke, L., Callas, J., Thayer, R.L., Shaw, D.: OpenPGP message format. RFC 4880, November 2007. https://doi.org/10.17487/RFC4880. https://rfc-editor.org/rfc/rfc4880.txt
10. Greschbach, B., Kreitz, G., Buchegger, S.: The devil is in the metadata—new privacy challenges in decentralised online social networks. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 333–339 (2012)
11. Kikuchi, H., Tada, M., Nakanishi, S.: Secure instant messaging protocol preserving confidentiality against administrator. In: 2004 18th International Conference on Advanced Information Networking and Applications, AINA 2004, vol. 2, pp. 27–30. IEEE (2004)
12. Lucas, M.M., Borisov, N.: FlyByNight: mitigating the privacy risks of social networking. In: Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society, WPES 2008, pp. 1–8. ACM, New York (2008). https://doi.org/10.1145/1456403.1456405
13. Marlinspike, M.: The Double Ratchet Algorithm, November 2016. https://signal.org/docs/specifications/doubleratchet/

14. Marotta, V., Abhishek, V., Acquisti, A.: Online tracking and publishers' revenues: an empirical analysis. In: Workshop on the Economics of Information Security (2019)
15. Mcauley, J., Leskovec, J.: Discovering social circles in ego networks. ACM Trans. Knowl. Discov. Data **8**(1) (2014). https://doi.org/10.1145/2556612
16. Mislove, A., Viswanath, B., Gummadi, K.P., Druschel, P.: You are who you know: inferring user profiles in online social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 251–260. Association for Computing Machinery, New York (2010). https://doi.org/10.1145/1718487.1718519
17. Moscaritolo, V., Belvin, G., Zimmermann, P.: Silent Circle Instant Messaging Protocol - Protocol Specification, December 2012. https://web.archive.org/web/20150402122917/silentcircle.com/sites/default/themes/silentcircle/assets/downloads/SCIMP_paper.pdf
18. OTR Development Team: Off-the-Record Messaging Protocol Version 3 (2012). https://otr.cypherpunks.ca/Protocol-v3-4.1.1.html
19. Perez, B., Musolesi, M., Stringhini, G.: You are your metadata: identification and obfuscation of social media users using metadata information. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 12 (2018)
20. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th USENIX Security Symposium (USENIX Security 2017), pp. 1199–1216 (2017)
21. Robison, C., Ruoti, S., van der Horst, T.W., Seamons, K.E.: Private Facebook chat. In: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing, pp. 451–460. IEEE (2012)
22. Schaad, J., Ramsdell, B., Turner, S.: Secure/multipurpose internet mail extensions (S/MIME) version 4.0 message specification. RFC 8551 (2019). https://doi.org/10.17487/RFC8551. https://rfc-editor.org/rfc/rfc8551.txt
23. Schifanella, R., Barrat, A., Cattuto, C., Markines, B., Menczer, F.: Folks in folksonomies: social link prediction from shared metadata. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 271–280. Association for Computing Machinery, New York (2010). https://doi.org/10.1145/1718487.1718521
24. Schillinger, F., Schindelhauer, C.: End-to-end encryption schemes for online social networks. In: Wang, G., Feng, J., Bhuiyan, M.Z.A., Lu, R. (eds.) SpaCCS 2019. LNCS, vol. 11611, pp. 133–146. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24907-6_11
25. Singh, M.: Telegram hits 400m monthly active users, April 2020. https://techcrunch.com/2020/04/24/telegram-hits-400-million-monthly-active-users/
26. Threema: Threema Cryptography Whitepaper, January 2019. https://threema.ch/press-files/cryptography_whitepaper.pdf
27. Tyagi, N., Gilad, Y., Leung, D., Zaharia, M., Zeldovich, N.: Stadium: a distributed metadata-private messaging system. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 423–440 (2017)
28. Van Den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles, pp. 137–152 (2015)

29. Web Cryptography API - W3C Recommendation 26 January 2017. https://www.w3.org/TR/2017/REC-WebCryptoAPI-20170126/
30. Yang, C.H., Kuo, T.Y., Ahn, T., Lee, C.P.: Design and implementation of a secure instant messaging service based on elliptic-curve cryptography. J. Comput. **18**(4), 31–38 (2008)

MobiWear: A Plausibly Deniable Encryption System for Wearable Mobile Devices

Niusen Chen[1], Bo Chen[1](✉), and Weisong Shi[2]

[1] Department of Computer Science, Michigan Technological University,
Houghton, MI, USA
bchen@mtu.edu

[2] Department of Computer Science, Wayne State University, Detroit, MI, USA

Abstract. Mobile computing devices are widely used in our daily life. With their increased use, a large amount of sensitive data are collected, stored, and managed in the mobile devices. To protect sensitive data, encryption is often used but, traditional encryption is vulnerable to coercive attacks in which the device owner is coerced by the adversary to disclose the decryption key. To defend against the coercive attacks, Plausibly Deniable Encryption (PDE) has been designed which can allow the victim user to deny the existence of hidden sensitive data. The PDE systems have been explored broadly for smartphones. However, the PDE systems which are suitable for wearable mobile devices are still missing in the literature.

In this work, we design MobiWear, the first PDE system specifically for wearable mobile devices. To accommodate the hardware nature of wearable devices, MobiWear: 1) uses image steganography to achieve PDE, which suits the resource-limited wearable devices; and 2) relies on various sensors equipped with the wearable devices to input passwords, rather than requiring users to enter them via a keyboard or a touchscreen. Security analysis and experimental evaluation using a real-world prototype (ported to an LG G smartwatch) show that MobiWear can ensure deniability with a small computational overhead as well as a small decrease of image quality.

Keywords: Confidentiality · Plausibly deniable encryption · Wearable mobile devices · Image steganography · Digital watermarking

1 Introduction

Mobile computing devices are ubiquitous today. More and more people choose to use their mobile devices, e.g. smartphones, tablets, smartwatches, to manage their personal private or even mission critical data. To protect confidentiality of sensitive data, full disk encryption (FDE) has been integrated into major mobile operating systems including Android [29] and iOS [15]. FDE can encrypt/decrypt data transparently to users, such that without having access to the secret key, the attacker

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 138–154, 2021.
https://doi.org/10.1007/978-3-030-80851-8_10

will not be able to access plaintext of the data. FDE however, can only defend against a passive attacker which tries to steal sensitive data from external storage [21] of the mobile devices. It cannot defend against an active attacker which can capture the device owner and force him/her to disclose the secret key, i.e., a coercive attack. This type of coercive attackers can be found broadly in the real world. For example, a journalist uses a mobile device to collect criminal evidence of atrocities in a region of oppression, and stores the evidence encrypted; when he/she crosses the border, a border inspector may notice this encrypted ciphertext and require him/her to hand in the decryption key [27]. Plausibly deniable encryption (PDE) has been designed to ensure confidentiality of data against this type of coercive attacks. PDE can allow a victim user to deny the very existence of the hidden sensitive data upon being coerced. Its rationale is, data are encrypted in a special way so that, sensitive private data will be revealed only if a true key is used for decryption but, if a decoy key is used, only non-sensitive public data will be disclosed; when a device owner is coerced, he/she can simply disclose the decoy key and, using the decoy key, the attacker can obtain the non-sensitive data but will be unaware of the existence of the hidden sensitive data.

The concept of PDE has been broadly adapted to mobile devices [5–7,13,14,16,17,21,25,27,28,33] to protect hidden sensitive data against coercive attacks. We have a few observations on those existing mobile PDE systems. First, most of them are specifically built for smartphones [5,7,13,14,25,27,28,33], rather than wearable mobile devices like smartwatches. The PDE systems for the wearable devices have broader applications compared to smartphones. This is because, compared to using a smartphone, using a wearable device (e.g., an Apple watch) to capture criminal evidence (e.g., taking photos or recording videos) is more convenient and less likely to be noticeable[1]. Second, most of them rely on either the hidden volume technique [5,6,9,16,17,21,27,28,33] or the steganographic file system [7,25], and both techniques incline to hide sensitive data among randomness which suffers from several limitations: 1) Filling the randomness will cause expensive extra overhead. 2) An implied assumption needs to be made that, filling the randomness itself is a normal behavior and will not lead to compromise PDE.

This work aims to build a PDE system for wearable mobile devices, using smartwatches as a representative. The resulted system, MobiWear, is the first system which can allow a wearable device user to deny the very existence of hidden sensitive data when facing coercive attacks. Our key insights are two-fold: First, we do not rely on randomness to hide sensitive data; instead, we utilize image steganography. Specifically, having observed that images usually use digital watermarking to protect intellectual property, we choose to embed the sensitive data in the watermarks of images, so that upon being coerced, the hidden sensitive data can be denied as the regular image watermarks. Our image steganography does not incur too much overhead and is suitable for the light-weight wearable devices. In addition, we do not rely on the implied assumption that filling the randomness is a normal system behavior. Second, we carefully adapt the PDE system to the

[1] Note that the examples we show here are only limited to the scope of capturing criminal evidence in a region of oppression or conflict.

wearable devices. A PDE system usually requires some sorts of secrets, i.e., some low-security-level secrets which can be disclosed to the adversary (e.g., the decoy keys), as well some high-security-level secrets (e.g., the true keys) which should be unknown to the adversary. In traditional PDE systems for smartphones [6,7,27], the secrets are entered by users using a keyboard or a touchscreen; in a wearable device however, the screen is usually small, rendering it a "bad" practice to enter the secrets using either a keyboard or a touchscreen. Therefore, we rely on various sensors equipped with the wearable devices to enter the secrets.
Contributions. We summarize our major contributions as follows:

- We have designed the first PDE system specifically for wearable mobile computing devices, by combining the concept of PDE, image steganography, as well as digital watermarking, and adapting the design to the wearable devices. MobiWear is a light-weight PDE system which is well integrated with the hardware features of the wearable devices and well suits the resources-limited wearable devices.
- We have implemented a real-world prototype of MobiWear in an LG G smartwatch and evaluated its performance.
- We have also analyzed the security as well as discuss a few potential security issues.

2 Background

2.1 Wearable Mobile Devices

A wearable mobile computing device is a mobile device which can be worn on the body. The wearable devices can be used for the purpose of general computing, as well as special purposes like fitness tracker. They usually integrate a few special sensors like accelerometers, gyroscopes, magnetometers, heart rate sensors, and pedometers. The most popular wearable device is the smartwatch. Besides the basic functionality of a regular watch, the modern smartwatch may include various extra peripherals to achieve "smartness", e.g., digital cameras, tiny speakers, GPS receivers, pedometers, heart rate sensors, thermometers, accelerometers, altimeters, barometers, compasses, gyroscopes. Compared to a smartphone which is usually much larger in size and equipped with more powerful hardware (e.g., much larger touchscreens, more powerful processors, RAM, and batteries), the smartwatch is small in size and equipped with less powerful hardware, e.g., using small screens which do not well support user input, being equipped with less powerful processors, RAM, and batteries.

2.2 Plausibly Deniable Encryption

Plausibly deniable encryption (PDE) systems are designed to protect sensitive information when a device owner is coerced by an adversary. Upon being coerced, the device owner only reveals the decoy key which can be used to decrypt non-sensitive data. The actual secret key (i.e., true key) which can be used to decrypt the sensitive data will be kept confidential and therefore, sensitive data are protected.

Currently, there are two major techniques which can implement the PDE concept in systems, namely, the steganography [4] and the hidden volume. The steganography-based PDE system hides sensitive data in regular files or randomness arbitrarily filled. However, since the system which manages the public non-sensitive data should not know the existence of the hidden sensitive data and, therefore, the hidden sensitive data may be overwritten by the public data. To mitigate this overwrite issue, several copies of hidden sensitive data are usually maintained across the entire disk. The hidden volume-based PDE system hides the sensitive data in a hidden volume. Its idea is: initially, the entire disk is filled with random data, and two volumes, a public and a hidden volume, are created; the public volume stores the public non-sensitive data, which are encrypted with a decoy key and placed across the entire disk; the hidden volume stores the sensitive data, which are encrypted with a true key and placed at the end of the disk starting from a secret offset; the hidden volume is completely embedded in the empty space of the public volume and, the attacker cannot detect its existence since he/she can not differentiate the encrypted hidden data from the randomness filled initially.

2.3 Image Steganography

Image steganography is often used to hide information in a cover image. Its process is as follows (Fig. 1): Secret data (e.g., texts or images) are stored invisibly in a cover image, generating a stego-image; this stego-image can then be sent to a receiver, where any third party will not be able to find out that the stego-image has hidden the secret data [19]; after having received the stego-image, the receiver can simply extract the secret data with or without a key [22].

In general, image steganography can hide secret data in two domains, the spatial domain and the transform domain. The least significant bit (LSB) steganographic embedding is an extremely simple technique of hiding secret data in the spatial domain. In an RGB image, each pixel consists of 4 channels, alpha (A), red (R), green (G) and blue (B), each of which occupies one byte. Alpha represents the value of transparency, and red, green and blue represent the value of three different colors. The last bit of each byte is called the least significant bit since its value only has a small effect on the pixel value [26] and, therefore, this bit can be used to hide sensitive data. A typical algorithm [3,20] for *embedding* the secret data is as follows: Given a secret key, a cover image and the secret data to be embedded, we first add two flags to the head and the tail of the secret data, respectively, generating the extended secret data. The two flags mark the beginning and the end of the secret data. We then encrypt the extended secret data using the secret key, and the resulted ciphertext will be treated as a collection of bits, which will be embedded sequentially to the pixels of the cover image (i.e., the least significant bit of each byte in a pixel will be used), generating a stego-image. Given the stego-image and the secret key, the *extraction* process of the LSB technique is: we extract the least significant bits from pixels of the

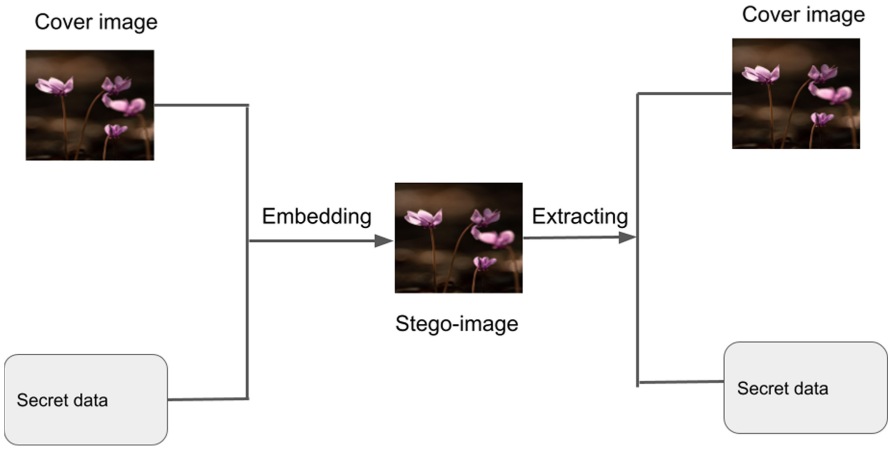


Fig. 1. Image steganography.

stego-image, decrypting them via the secret key, and can identify the beginning and the end of the secret data using the flags added during the embedding process; in general, it is no need to decrypt all the LSB bits, since we can treat the entire LSB bits as a collection of units (e.g., each unit can be 128 bits for AES-128), and decrypt each unit sequentially from the beginning until the “end” flag is found.

There are some variants of the LSB technique, e.g., pixel value differencing (PVD) [31], random pixel embedding method (RPE) [23], and pixel intensity based method [18]. The LSB technique has some advantages, including: 1) being hard to be detected by human eyes; and 2) simplicity of implementation; and 3) high payload compared to transform domain technique. It also has some disadvantages: 1) it is less robust compared to the transform domain technique; 2) the hidden data can easily be destroyed by simple attacks such as scaling and cropping.

2.4 Digital Watermarking

Digital watermarking can reinforce the security of multimedia data, by providing a solution to ensure tamper resistance as well as ownership protection of intellectual property [32]. A simple type of image watermarking is to embed a logo, which is a *visible* watermark. This is usually used for public identification and recognition. Another type of image watermarking is to embed an *invisible* watermark, which has been used broadly in multimedia data (e.g., images, videos) to claim copyrights.

2.5 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio, PSNR, represents a ratio between the maximal possible power of a signal and the power of the noise. A higher PSNR usually indicates a good image quality. PSNR can be computed via Eq. 1, in which MAX_I is the maximal pixel value of the image, and MSE represents the mean square error. The MSE of an m × n image can be computed via Eq. 2, in which I is the original image and K is its noisy approximation.

$$PSNR = 20 \cdot log_{10}(MAX_I) - 10 \cdot log_{10}(MSE) \tag{1}$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{2}$$

3 Model and Assumptions

3.1 System Model

We consider a wearable mobile device, and its architecture is shown in Fig. 2. The architecture mainly contains three layers. The top layer is the application layer, which contains various user apps that directly interact with users and accept I/Os from users, e.g., an image viewer or editor. The middle layer is the operating system for wearable devices which, 1) manages the device's hardware resources, and 2) allows the apps to use the hardware resources via the system APIs. A popular operating system is Wear OS [2]. The bottom layer is the hardware which includes the processor, the RAM, and the flash storage. Note that a flash-based block device like a microSD card is used broadly in wearable devices, in which the flash memory is managed internally by flash translation layer (FTL), exposing a regular block access interface.

3.2 Adversarial Model

We consider a computationally bounded adversary. The adversary is able to capture a victim user together with his/her mobile device. The adversary notices the existence of ciphertext in the device and may coerce the victim user to disclose keys which can be used to decrypt the ciphertext. We need to rely on a few reasonable assumptions:

- The adversary is rationale and will stop coercing the victim user after being convinced that the keys have been disclosed. This is a common assumption for all the PDE designs [6,7,21,27].
- The adversary cannot capture a victim user when he/she is right processing the hidden sensitive data. Otherwise, the sensitive data will be obtained by the adversary trivially.
- We do not consider the attack that the adversary injects malware into the device before capturing it. Otherwise, the malware can monitor the process of embedding hidden data and compromise PDE trivially.

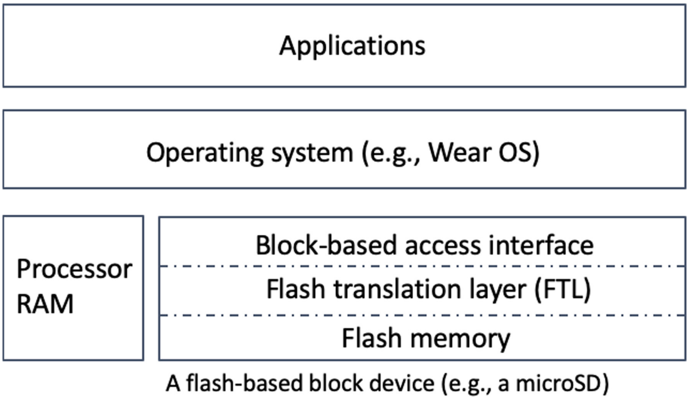

Applications
Operating system (e.g., Wear OS)
Processor
RAM
Block-based access interface
Flash translation layer (FTL)
Flash memory
A flash-based block device (e.g., a microSD)

Fig. 2. The architecture of a wearable mobile device.

- The adversary is assumed to be able to obtain both the original cover image and the stego-image (i.e., the resulted image after the sensitive data are embedded), and to perform forensic analysis over them. The stego-image can be easily obtained once the wearable device is captured. The cover image may be obtained by the adversary considering that the cover image may be obtained from external sources, e.g., the device owner purchased it from others. However, the adversary is assumed to be not able to obtain the watermark image which is created locally by the device owner and has not been disclosed to the public. Note that a cautious user should delete the watermark image locally once used.

4 Design

4.1 Design Rationale

MobiWear is a PDE system specifically designed for wearable mobile devices. To ensure easy deployment of MobiWear, we integrate it into the application layer (Fig. 2) of a wearable mobile device. The design rationale of MobiWear is described in the following.

Upon being coerced, the victim should be able to convince the adversary that the noticeable embedding secrets are nothing but just some normal information. In the hidden volume-based technique [6,27] or steganographic file systems [25], the secret sensitive data are denied as randomness, which are filled by the device. This requires making the assumption that filling randomness is a normal system behavior [6,21,27]. Having observed that images can embed watermarks for intellectual property protection, we choose to hide the sensitive data into the images stored in the device, and deny them as the watermarks embedded into the images.

The process for embedding secret sensitive data into images is as follows (Fig. 3): Given some secret sensitive data, we first pick a cover image as well as a watermark image which will be embedded into the cover image. We then perform two steps: 1) We encrypt the sensitive data using a true key and, the resulted ciphertext will be embedded into the watermark image, obtaining a stego-watermark; 2) We encrypt the stego-watermark using a decoy key and, the resulted ciphertext will be embedded into the cover image, obtaining a stego-image. The stego-image will be stored to the wearable mobile device.

After the victim user is captured together with his/her wearable device, the adversary may notice that something is stored hidden in the stego-image. Note that the adversary is assumed to be able to obtain both the original cover image and the stego-image (Sect. 3.2). The adversary will coerce the victim user for the hidden sensitive data. The victim user will disclose the decoy key and claim that there is a watermark embedded in the stego-image. Utilizing the decoy key, the adversary can successfully extract the watermark (i.e., the stego-watermark) from the stego-image. The adversary will not be able to notice anything special in the stego-watermark, since it does not have access to the actual watermark (Sect. 3.2), and convince that there are no secret sensitive data stored. Only by utilizing the true key, the actual secret sensitive data can be extracted.

4.2 Design Details

How to Input Keys. Due to the small size of a wearable mobile device, using a keyboard or a touchscreen to enter keys is very inconvenient and impractical. We therefore rely on the embedded sensors in the wearable device to enter keys. There are many sensors such as accelerometer, gyroscope, heart rate sensor and compass sensor equipped with the wearable mobile devices. In MobiWear, we choose the gyroscope, which can be used to measure the rotation rate of x-axis, y-axis, and z-axis. Compared to other types of sensors, the gyroscope is a more convenient and robust choice for a wearable device's user to generate different input, each corresponding to a different key. For example, a user who is wearing a smartwatch can simply rotate his/her wrist differently to generate unique keys and, after obtaining the rotation rate from the gyroscope, the device can calculate the rotation degree in a time interval, generating unique keys. In MobiWear, two different keys, a decoy key and a true key, are generated. The decoy key is used to hide/extract the stego-watermark into/from the stego-image, and the true key is used to hide/extract the hidden sensitive data into/ from the stego-watermark.

Information Hiding and Extracting. The process of hiding secret sensitive data includes two steps: 1) hiding the sensitive data into the watermark image, obtaining a stego-watermark; and 2) hiding the stego-watermark into the cover image, obtaining a stego-image. For step 1, the sensitive data are encrypted using the true key and embedded into the watermark image, using the LSB steganographic embedding technique (Sect. 2.3). Similarly for step 2, the stego-watermark is encrypted using the decoy key and embedded into the cover image

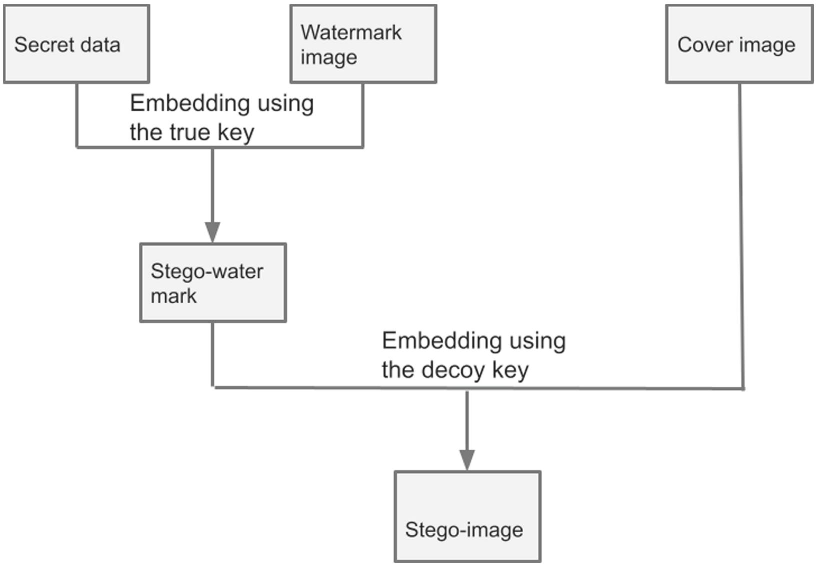


Fig. 3. The process of embedding secret sensitive data.

(via the LSB technique). Extracting sensitive information is the reverse operation of the information hiding. Once both the decoy and the true key are provided, MobiWear can use the decoy key to extract the stego-watermark from the cover image (i.e., via the extraction process of the LSB technique in Sect. 2.3), and then use the true key to extract the sensitive data from the stego-watermark (via the extraction process of the LSB technique). If only the decoy key is provided, only the stego-watermark can be extracted, denying the existence of the hidden sensitive data.

User Authentication. The user is required to enter the decoy key first and, if the decoy key is correct, MobiWear will wait for a certain amount of time (e.g., a few seconds). During this time interval, the user can enter the true key and, if the true key is entered within this time period and it is correct, the secret sensitive data will be extracted and displayed; otherwise, the stego-watermark will be displayed. The entire process for user authentication is shown in Fig. 4. Upon being coerced, the victim will disclose the decoy key and, using the decoy key, the adversary will be able to see the extracted stego-watermark, but will not be aware of the existence of the hidden sensitive data. Although there is a short delay after entering the decoy key, this can be simply denied as the system delay due to the limited computational power of a wearable device.

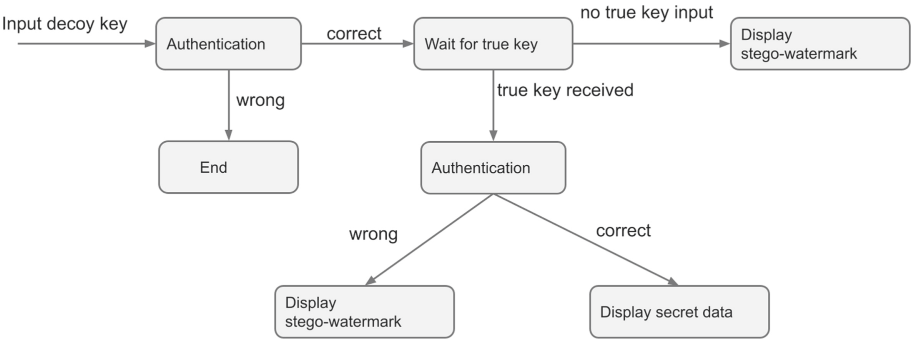


Fig. 4. The process of user authentication.

5 Implementation and Evaluation

5.1 Implementation

We have implemented MobiWear in an LG G watch [1], which has been released by LG and Google. It is equipped with Qualcomm Snapdragon 400 processor, 512 MB memory and 4 GB flash storage. Its embedded sensors include accelerometer, digital compass and gyroscope. The default operating system is Android Wear 1.5.0, which does not well support image viewing, and we have ported Android Wear OS 2.0 instead to resolve this issue. For image steganography, we relied on an open-source image steganography library [3], which has implemented the LSB steganographic embedding technique for Android. The encryption is instantiated using AES-128. Our major implementations are: 1) We support the password input via the embedded sensor gyroscope of the LG G watch; 2) We implement the PDE authentication via both the decoy and the true key; 3) We implement the information hiding and extracting process via the decoy and the true key.

5.2 Evaluation

Computational Overhead. We first evaluate the computational time of MobiWear in hiding/extracting data under different lengths of the secret data, while fixing the size of both the cover image and the watermark image. The results are shown in Fig. 5. We can observe that: 1) For longer secret data, MobiWear needs more time in hiding/extracting secret data. This is because: for longer secret data, MobiWear will need to embed/extract more secret bits into/ from the watermark image, which will increase the time; but the time for embedding/extracting the watermark image into/from the cover image will be identical. 2) Extracting the secret data is slower than hiding them. This is because: when embedding the data upon generating the stego-watermark and the stego-image (Sect. 2.3), two flags which indicate the begin and the end of the data are added and, when extracting them, we need to decrypt the LSB

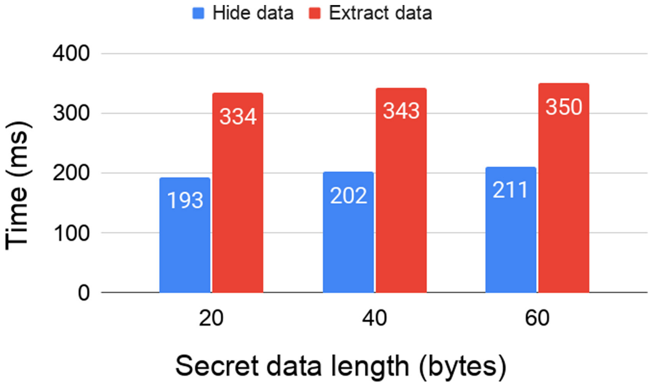


Fig. 5. Computational time for hiding and extracting sensitive data when the size of the cover image and the watermark image is fixed (the cover image is 1.5 MB in size, and the watermark image is 20 KB in size).

bits in terms of units until finding the end flag; this leads to extra overhead in decrypting more data and locating the end flag.

We then evaluate how the size of the watermark image and the cover image will affect the time of hiding/extracting secret data. We fix the length of the secret data as 20 bytes, and evaluate the hiding/extracting time under different sizes of the watermark and the cover image. The results are shown in Fig. 6. We can observe that: 1) The time for hiding/extracting the secret data slightly increases when the size of the cover/watermark image increases. This is because: both the cover and the watermark image need to be loaded into the memory for further processing, which slightly increases the computational time. 2) The time for extracting the secret data is more than that for hiding them. The reason has been mentioned before.

Assessing PSNR. To understand how MobiWear affects the image quality, we compute the PSNR values by varying the lengths of the secret data. We first hide the secret data with length 20, 40 and 60 (in bytes) in a watermark image, generating a corresponding stego-watermark, which is then embedded into the cover image. The size of the watermark image is fixed as 20 KB and the size of the cover image is fixed as 1.5 MB. We calculate the PSNR of each stego-image. The results are shown in Table 1. We can observe that, longer secret data will result in lower PSNR values, which indicates a worse image quality. This is because the longer secret data will occupy more least significant bits and increase the noise ratio. But still, the difference between the cover image and the stego-images is hard to be detected by human, which is justified in Fig. 7.

We also compare the PSNR values between original watermark image and the stego-watermark by varying the lengths of the secret data. The results are shown in Table 2. We can observe that a stego-watermark with longer secret data embedded has a lower PSNR value. This is because longer secret data will occupy more bits in the watermark image, decreasing its image quality. But, the

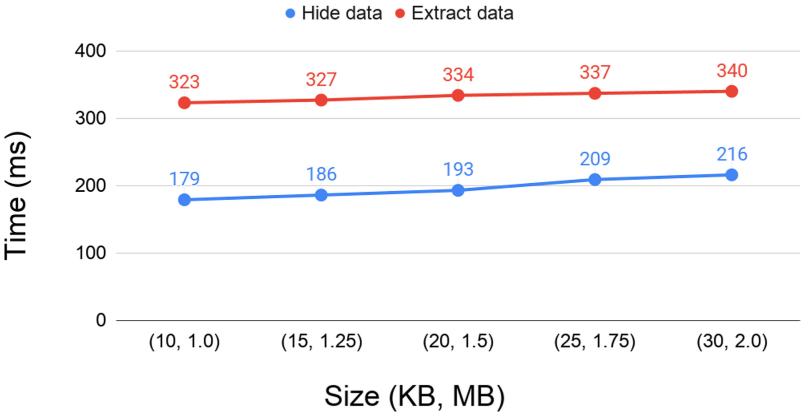


Fig. 6. Computational time for hiding and extracting sensitive data when the size of both the watermark image and the cover image varies. For the x-axis, (a, b) represents (size of watermark image in KBs, size of cover image in MBs).

Table 1. PSNR of stego-images under different lengths of secret data.

Length of secret data (bytes)	20	40	60
PSNR	30.6257	30.6153	30.5500

difference between the original watermark image and the stego-watermarks is also hard to be detected by human, which is justified in Fig. 8.

Table 2. PSNR of stego-watermarks under different lengths of secret data.

Length of secret data (bytes)	20	40	60
PSNR	33.8356	33.7800	30.7610

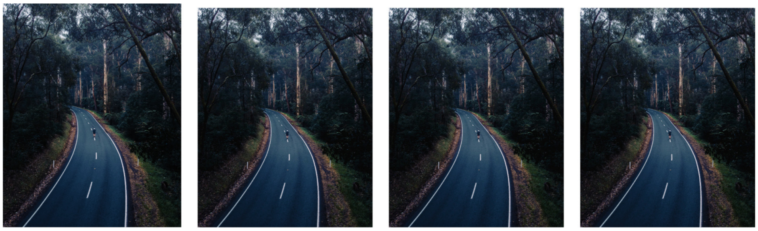

(a) Cover Image (b) Length=20 (c) Length=40 (d) Length=60

Fig. 7. The visualized comparison between the cover image (a) and stego-images (b–d) hiding secret data of different lengths.

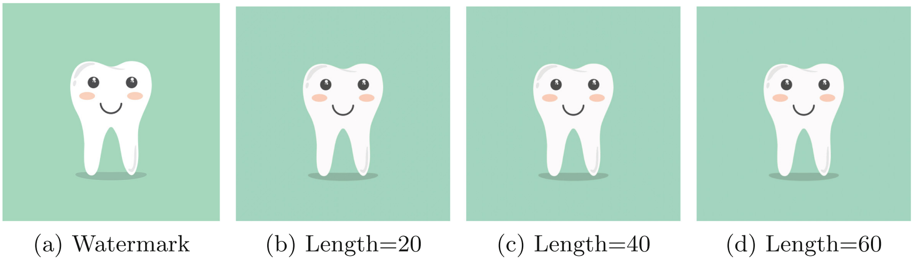

(a) Watermark (b) Length=20 (c) Length=40 (d) Length=60

Fig. 8. The visualized comparison between the original watermark image (a) and the stego-watermarks (b–d) hiding secret data of different lengths.

6 Security Analysis and Discussion

6.1 Security Analysis

After having captured a victim wearable device, the adversary will obtain the stego-image from the device. For simplicity, we assume there is only one image in the device, which can be easily generalized to the actual case that there are multiple images. The adversary will obtain the corresponding cover image somehow (Sect. 3.2). By comparing the cover image and the stego-image, the adversary can identify differences between them and notice something has been embedded stealthily in the cover image. Note that since we use the invisible watermark (Sect. 2.4) in MobiWear and, without having access to the original cover image, the adversary should not be able to notice something has been hidden. By coercing the victim user, the adversary will be able to obtain the decoy key and, using the decoy key, the adversary will then extract the stego-watermark.

The adversary further performs forensic analysis over the extracted stego-watermark. This however, will not give the adversary any advantage of identifying the existence of hidden sensitive data because: The adversary cannot have access to the original watermark image (Sect. 3.2) and, without being able to compare the stego-watermark with the original watermark, there is no way for the adversary to identify whether there are any modifications over the LSBs of the stego-watermark's pixels (solely viewing the stego-watermark will not result in any visualized abnormality).

Therefore, we can conclude that, the adversary will not be able to identify the existence of PDE and the security of MobiWear can be ensured.

6.2 Discussion

About the Length of Sensitive Data which can be Hidden. MobiWear hides the sensitive data into the image and, the length of the hidden data will be limited by the size of the image and the corresponding watermark. Given an ARGB cover image with N pixels and each pixel consisting of 4 bytes (alpha, red, green, blue), we analyze the length of the sensitive data which can be hidden. Using the LSB technique, the maximal size of watermark the cover image can embed is $\frac{N}{2}$ bytes considering 1 bit out of each byte can be used to embed the watermark. Correspondingly, the maximal length of secret sensitive data which can be hidden in the watermark is $\frac{N}{16}$ bytes. For example, a 4 MB cover image will have 1M pixels, and can hide up to 0.0625 MB sensitive data. To hide more sensitive data, we can use more least significant bits, which however will decrease the quality of the image. A practical mitigation is to cut the sensitive data of large size into small chunks, and to hide each chunk using a different cover image.

Deniability Compromises in Memory. Secret sensitive data may leave traces in the memory, leading to compromise of PDE. Considering the volatile nature of RAM, an immediate mitigation strategy is to power-off the device to remove the traces of secret sensitive data in the memory. Other mitigation strategies include utilizing hardware isolation techniques like ARM TrustZone to isolate the memory region which processes the hidden sensitive data [5].

Defending Against Multi-snapshot Adversaries. MobiWear can defend against an adversary which can have access to the victim device once. Note that by using image steganography, MobiWear remains secure even if the adversary can have access to different layers of the system (Fig. 2). If the adversary can capture the device and its owner multiple times, it will have multiple access to the device over time, and a potential PDE compromise could be: If the secret sensitive data are modified and a new stego-watermark needs to be re-embedded into the original cover image and, by comparing the stego-image at different points of time, the adversary may be aware of the existence of hidden sensitive data. A potential mitigation strategy can be, each time when the sensitive data are modified, the device owner should discard the corresponding stego-image which turns obsolete, and hide the new data using a new cover image.

Mitigating Data Corruptions. MobiWear uses the LSB technique to hide secret sensitive data. This is vulnerable to cutting and cropping attack which will destroy the sensitive data. However, the goal of PDE is to ensure confidentiality of the sensitive data, rather than to prevent data from being corrupted. A recommendation for mitigating corruption attacks is to periodically back up the sensitive data, e.g., to a remote cloud server or an offline personal computer.

7 Related Work

7.1 Plausibly Deniable Encryption Systems

Plausibly deniable encryption has been designed to defend against coercive attacks so that even though the key is forced to be disclosed, the critical sensitive data can remain protected. There are mainly two types of PDE systems, the steganography-based PDEs and the hidden volume-based PDEs. There are also PDE systems relying on other techniques like side channel [10] and WOM codes [11].

The Steganography-Based PDE Systems. The first steganography-based PDE system [4] was proposed by Anderson et al. They designed two schemes. The first one is to hide sensitive data in cover files, which however, requires the system to store a large number of cover files. The second one is to fill the entire disk with random data and, to encrypt and hide the secret data in those random data. Based on the second scheme of Anderson et al., McDonald et al. designed StegFS [24], in which they extended a standard Linux file system (EXT2) with PDE support. Peters et al. proposed DEFY [25], a flash file system which supports PDE. Its features include authenticated encryption, fast secure deletion, and support for multiple layers of deniability.

The Hidden-Volume Based PDE Systems. The other technique which can be used to achieve PDE is the hidden volume technique. TrueCrypt [30] and VeraCrypt [12] are open-source projects for disk encryption, with deniability support using the hidden volume technique. Skillen et al. [27,28] proposed Mobiflage which moved the hidden volume technique to mobile computing devices. Mobiflage requires the user to re-boot the device to enter the hidden mode, which is inconvenient. To mitigate this issue, Yu et al. proposed MobiHydra [33] which supports data hiding without the need of rebooting the device. MobiHydra also solved a boot-time attack on the PDE systems. Chang et al. proposed Mobipluto [5,6], a file system friendly PDE system such that any block-based file systems can be deployed on top of the public volume, without worrying about overwriting the hidden sensitive data. Jia et al. [21] further moved the hidden volume to the flash translation layer, eliminating the deniability compromised in the low-level flash memory medium. Having observed that the prior mobile PDE systems cannot defend against a multi-snapshot adversary, Chang et al. designed MobiCeal [7], which combines both the hidden volume technique and the dummy write technique to enable defend against multi-snapshot adversaries.

7.2 Image Steganograyphy

Image steganography has been widely used to claim the ownership or copyright of the products. Wu et al. [31] proposed an efficient steganographic method to embed secret messages into cover images, which is based on a simple visual effect of the human visual perception. Ibrahim et al. [20] proposed an algorithm to hide data in cover images, by using binary codes and pixels inside an image. Based on the proposed algorithm, a system called Steganography Imaging System (SIS) is built which can hide secret message without a noticeable distortion. Chaumontet et al. [8] proposed a DCT-based data hiding method which can hide color information in a compressed gray-level image. Their proposed method consists of three steps, color quantization, color ordering, and data hiding.

8 Conclusion

In this work, we design MobiWear, a plausibly deniable encryption system for wearable mobile devices. MobiWear uses image steganography to hide sensitive data and utilizes the integrated sensors to input secrets. The experiment results indicate that MobiWear can achieve deniability with a small overhead as well as a slight decrease of image quality.

Acknowledgments. This work was supported by US National Science Foundation under grant number 1928349-CNS, 1928331-CNS, and 1938130-CNS.

References

1. LG G watch (2016). https://www.lg.com/us/smart-watches/lg-W100-lg-watch
2. Wear OS (2016). https://wearos.google.com/#stay-connected
3. Agarwal, A.: Image-steganography-library-android (2011). https://github.com/aagarwal1012/Image-Steganography-Library-Android
4. Anderson, R., Needham, R., Shamir, A.: The steganographic file system. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 73–82. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49380-8_6
5. Chang, B., et al.: User-friendly deniable storage for mobile devices. Comput. Secur. **72**, 163 (2017)
6. Chang, B., Wang, Z., Chen, B., Zhang, F.: Mobipluto: file system friendly deniable storage for mobile devices. In: Proceedings of the 31st Annual Computer Security Applications Conference, pp. 381–390 (2015)
7. Chang, B., et al.: Mobiceal: towards secure and practical plausibly deniable encryption on mobile devices. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 454–465. IEEE (2018)
8. Chaumont, M., Puech, W.: A DCT-based data-hiding method to embed the color information in a jpeg grey level image. In: 2006 14th European Signal Processing Conference, pp. 1–5. IEEE (2006)
9. Chen, B., Chen, N.: Poster: a secure plausibly deniable system for mobile devices against multi-snapshot adversaries. In: 2020 IEEE Symposium on Security and Privacy Poster Session (2020)
10. Chen, C., Chakraborti, A., Sion, R.: Infuse: invisible plausibly-deniable file system for nand flash. Proc. Priv. Enhancing Technol. **4**, 239–254 (2020)
11. Chen, C., Chakraborti, A., Sion, R.: Pearl: plausibly deniable flash translation layer using WOM coding. In: The 30th Unsenix Security Symposium (2021)
12. CodePlex. Veracrypt ssd. https://veracrypt.codeplex.com/, 2017
13. Trnka, D.: Steganography software.version 1.3 (2014). https://play.google.com/store/apps/details?id=com.dinaga.photosecret&hl=en_US&gl=US
14. EDS. Free open source on-the-fly disk encryption software.version 2.0.0.243 (2012). http://www.sovworks.com/
15. How to encrypt your devices (2017). https://spreadprivacy.com/how-to-encrypt-devices/
16. Feng, W., et al.: Mobigyges: a mobile hidden volume for preventing data loss, improving storage utilization, and avoiding device reboot. Future Gener. Comput. Syst. **109**, 158 (2020)
17. Hong, S., Liu, C., Ren, B., Huang, Y., Chen, J.: Personal privacy protection framework based on hidden technology for smartphones. IEEE Access **5**, 6515–6526 (2017)
18. Hussain, M., Hussain, M.: Pixel intensity based high capacity data embedding method. In: 2010 International Conference on Information and Emerging Technologies, pp. 1–5. IEEE (2010)

19. Hussain, M., Hussain, M.: A survey of image steganography techniques (2013)
20. Ibrahim, R., Teoh, S.K.: Teganography algorithm to hide secret message inside an image. J. Comput. Technol. Appl. (JCTA) **1**(2), 102–108 (2011)
21. Jia, S., Xia, L., Chen, B., Liu, P.: DEFTL: implementing plausibly deniable encryption in flash translation layer. In: Proceedings of the 24th ACM Conference on Computer and Communications Security. ACM (2017)
22. Johnson, N.F., Jajodia, S.: Exploring steganography seeing the unseen. Computer **31**(2), 26–34 (1998)
23. Liu, L., Chen, T., Cao, C., Wen, X., Xie, R.: A novel data embedding method using random pixels selecting. Inf. Technol. J. **12**(7), 1299 (2013)
24. McDonald, A.D., Kuhn, M.G.: StegFS: a steganographic file system for linux. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 463–477. Springer, Heidelberg (2000). https://doi.org/10.1007/10719724_32
25. Peters, T.M., Gondree, M.A., Peterson, Z.N.J.: DEFY: a deniable, encrypted file system for log-structured storage. In: 22th Annual Network and Distributed System Security Symposium, NDSS (2015)
26. Singh, A.K., Singh, J., Singh, H.V.: Steganography in images using LSB technique. Int. J. Latest Trends Eng. Technol. (IJLTET) **5**(1), 426–430 (2015)
27. Skillen, A., Mannan, M.: On implementing deniable storage encryption for mobile devices. In: 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA (2013)
28. Skillen, A., Mannan, M.: Mobiflage: deniable storage encryption for mobile devices. IEEE Trans. Depend. Secure Comput. **11**(3), 224–237 (2014)
29. Source. Android full disk encryption (2016). https://source.android.com/security/encryption/
30. TrueCrypt. Free open source on-the-fly disk encryption software. version 7.1a (2012). http://www.truecrypt.org/
31. Da-Chun, W., Tsai, W.-H.: A steganographic method for images by pixel-value differencing. Pattern Recogn. Lett. **24**(9–10), 1613–1626 (2003)
32. You, W., Chen, B., Liu, L., Jing, J.: Deduplication-friendly watermarking for multimedia data in public clouds. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) ESORICS 2020. LNCS, vol. 12308, pp. 67–87. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58951-6_4
33. Yu, X., Chen, B., Wang, Z., Chang, B., Zhu, W.T., Jing, J.: Mobihydra: pragmatic and multi-level plausibly deniable encryption storage for mobile devices. In: Information Security - 17th International Conference, ISC 2014, Hong Kong, China. Proceedings, pp. 555–567 (2014)

Recent Advances in the Web PKI and the Technical Challenges in SCMS

Yunkun Wu[1], Xiaokun Zhang[2](✉), Yajun Teng[3], Zhenya Liu[4], Liang Huang[1], Jingqiang Lin[4], and Xuhua Bao[1]

[1] Legendsec Information Technology (Beijing) Inc., Beijing, China
{wuyunkun,huangliang,baoxuhua}@qianxin.com
[2] Academy of Opto-Electronics, Chinese Academy of Sciences, Beijing, China
xkzhang@aoe.ac.cn
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
tengyajun@iie.ac.cn
[4] School of Cyber Security, University of Science and Technology of China, Hefei, China
hhhhlzy@mail.ustc.edu.cn, linjq@ustc.edu.cn

Abstract. The Web PKI plays a more and more important role in network security, as nowadays TLS and HTTPS are being widely adopted. The most significant recent advances in the Web PKI include certificate transparency and push-based revocation, which improve the trustworthiness and performance of TLS and HTTPS, respectively. Meanwhile, SCMS is a specialized PKI system designed for V2V communications. In this paper, we analyze the design principles of certificate transparency and push-based revocation, study the similar requirements in V2V communications, and then summarize the technical challenges to integrate certificate transparency and push-based certificate revocation into SCMS. From the experiences and lessons in the Web PKI, we do believe that the current designs of SCMS are still not completely ready to be deployed in the real world.

Keywords: Public key infrastructure (PKI) · Transport layer security (TLS) · Certificate · Security credential management system (SCMS) · Trust management · Vehicle-to-Vehicle (V2V) communication

1 Introduction

Public key infrastructures (PKIs) [11,28] provide various security services such as confidentiality, authentication, data integrity, and non-repudiation, through certificates signed by a trusted certification authority (CA). The Web PKI (or sometimes called the TLS/HTTPS PKI) is the PKI system that is implemented and deployed for web security, especially for TLS [13] and HTTPS [46]. In the Web PKI, a list of accredited root CAs are trusted by the mainstream operating

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 155–167, 2021.
https://doi.org/10.1007/978-3-030-80851-8_11

systems (OSs) and browsers (e.g., Microsoft Windows, Apple macOS, Google Android, Mozilla Firefox, etc.) [3,40,43]. These root CAs and their subordinate CAs are responsible for signing TLS server certificates binding a domain name (e.g., www.facebook.com or www.gmail.com) to a key pair held by the website. Then, a browser will establish secure TLS sessions with the website, after verifying the TLS server certificate binding the to-be-visited domain.

As TLS and HTTPS are being widely adopted after the structure of PKI systems has been proposed for more than thirty years, several technical advances in the Web PKI are proposed and implemented recently. First of all, *certificate transparency* is proposed and deployed [34,50], to detect fraudulent TLS server certificates and improve the trustworthiness and accountability of CAs. Several security incidents indicate that even accredited famous CA systems may be compromised, deceived or even compelled to issue fraudulent TLS server certificates [10,16,19,25,41,48,56,59,60], which bind a domain name to a key pair held by man-in-the-middle (MitM) or impersonation attackers, instead of the legitimate website. Certificate transparency depends on redundant public log servers to record all CA-signed TLS server certificates, and browsers accept a TLS server certificate only if it is recorded in multiple independent publicly-visible logs [2,21,42]. Thus, a fraudulent certificate will be found by the victim website soon, which acts as a monitor or visits the third-party monitors [9,18,22,45] to search for certificates of interest in the public log servers.

Meanwhile, another significant technical advance of the Web PKI is *push-based certificate revocation*. In the traditional design of the Web PKI, a browser checks the revocation status of a TLS server certificate, through certificate revocation list (CRL) [11] or online certificate status protocol (OCSP) [44]. That is, the browser has to download the CRL file or acquire the OCSP response by itself, through another connection to the PKI system, in addition to the TCP connections to the visited website. The addition connection delays remarkably impact the performance of TLS and HTTPS [31,32,39,49]. On the contrary, push-based revocation requires (a) the certificate holder (i.e., the visited webserver in TLS and HTTPS) to actively send the OCSP message in TLS negotiations, e.g., OCSP stapling [15] and OCSP must-staple [24], or (b) the PKI system to proactively push all or most certificate revocation status data to browsers through the manufacturers, e.g., CRLSet [52], OneCRL [20] and CRLite [29,33]. Push-based revocation eliminates the addition connections for revocation status data.

On the other hand, the security credential management system (SCMS) [58], is a specialized PKI system introduced for vehicle-to-vehicle (V2V) communications. Compared with traditional PKI systems, *short-lived pseudonym certificate*, implicit certificates [7,8], butterfly key expansion, and *linkage-based revocation* are designed in SCMS, to balance security, privacy, and efficiency in the V2V communications. Before SCMS is deployed widely in the real world (except some pilot projects [53]), the experiences and lessons in the large-scale Web PKI are very useful for us. However, we find that *fraudulent certificates* and *certificate revocation* are not carefully considered in the designs of SCMS. Therefore, in this paper, we analyze the possibilities to integrate certificate transparency and push-based certificate revocation into SCMS, and then presents the technical challenges in the integration. These studies help to improve SCMS and also other PKI systems for V2V communications [6,58].

The remainder of this paper is organized as follows. Section 2 describes certificate transparency and push-based revocation, and Sect. 3 in details discusses the technical challenges to integrate certificate transparency and push-based certificate revocation into SCMS. Finally, we conclude this paper in Sect. 4.

2 Recent Advances in the Web PKI

In the traditional Web PKI, a CA is responsible for signing TLS server certificates. In TLS negotiations, a browser does not accept a TLS server certificates until it verifies the CA's signature on the certificate. Meanwhile, the browser also needs to check the revocation status of the TLS server certificate. That is, the browser by itself downloads the CRL file based on the CRL distribution points extension in the certificate, or acquires the OCSP response based on the AIA OCSP extension [11,44]. The CA is usually also responsible for signing CRL files and OCSP responses, or sometimes these functions are implemented by independent CRL issuers or OCSP servers [11,44].

As TLS and HTTPS are widely adopted in the Internet, the original designs of the Web PKI are improved recently. The recent advances include certificate transparency and push-based certificate revocation, which are not included in the original designs of the Web PKI [11,28].

2.1 Certificate Transparency

In the original design of PKIs, a CA is fully trusted to be responsible for signing a certificate only after carefully communicating with the applicant. But security incidents indicate that CA systems may be compromised, deceived or compelled to issue fraudulent certificates [10,16,19,25,41,48,56,59,60]. Therefore, in addition to CAs, browsers and websites, certificate transparency introduces the following PKI components [34].

Log Server. A log server maintains publicly-visible append-only logs that record certificates. It accepts certificates from CAs. All certificate records in a log are organized as a Merkle hash tree, and the root node is periodically signed by the log server, called the signed tree heads (STHs). There are more than one hundred log servers in the Internet in 2020 [23], and a TLS server certificate is recorded redundantly in multiple logs [55].

Monitor. Monitors regularly watch for suspicious certificates in the public logs. A monitor regularly fetches certificates from these logs, decodes the certificates, and searches for the certificates of interest among them. A website may assume the monitor role by itself [34] to search for the certificates binding its domain name, and there are also third-party monitors [9,18,22,45] which process the records in public logs to provide convenient certificate search services for users.

Auditor. Auditors ensure the correct behaviors of log servers. An auditor archives the STHs of a log. Then, by comparing any two STHs and requiring the consistency proof from the log server, an auditor checks whether a log is strictly append-only, i.e., any version of the log is a sub-tree of any later version. It also checks that each recorded certificate corresponds to an entry in the publicly-visible log [34], by verifying the audit path (i.e., the shortest list of additional nodes in the Merkle tree to compute the STH). An auditor may be a stand-alone service, or an internal component of a browser or a monitor.

After signing a TLS server certificate, the CA submits it to some log server. The log server responds with a signed certificate timestamp (SCT), which is a promise to append the certificate to the public log within the maximal merge delay (MMD). Then, SCTs are sent along with the certificate in TLS negotiations. In addition to the CA's signature and the certificate revocation status data, a browser also verifies the log servers' signatures in these SCTs. This implies that a browser preinstalls the public keys of trusted log servers [2,23] as well as the self-signed certificates of trusted root CAs.

It is worth noting that during TLS negotiations the browser does *not* establish addition connections to log servers, to check whether the TLS server certificate has been recorded in the public logs or not. The browser only verifies the log servers' signatures in the SCTs, and in the future some auditors will checks whether each SCT (i.e., a recorded certificate) corresponds to an entry in the log or not.

The above operations of certificate transparency ensure that any certificate accepted by browsers will be visible to the website (or certificate subject), with the help of monitors [35,37]. Thus, the website that is aware of all legitimate certificates issued with its authorization, will find the fraudulent ones among them if any.

2.2 Push-Based Certificate Revocation

In the original designs of the Web PKI, certificate revocation status data are obtained by the certificate verifiers (i.e., browsers) [11,28,44]. On the contrary, push-based certificate revocation eliminates the addition connections to obtain the certificate revocation status data. Typical approaches are listed as below.

OCSP Stapling. OCSP stapling [15] is a TLS extension that enables the website to send an OCSP response in TLS negotiations, and this OCSP message proves the validity (or unrevokedness) of its TLS server certificate. Note that the website acquires the OCSP response from the CA (or OCSP server) in advance. Thus, the browsers do not need the addition connections to check the certificate revocation status. Moreover, in order to defend against the downgrade attacks that exploit revoked TLS server certificates but do not send OCSP stapling extensions in TLS negotiations, the certificate extension of OCSP must-staple is defined [24]. A TLS server certificate with an OCSP must-staple extension, must be sent along with OCSP stapling messages in TLS negotiations; otherwise, it will be immediately rejected by the browsers. OCSP must-staple prevents the downgrade attacks, even when a browser does not download the CRL files or access the OCSP server by itself.

Proactive CRL. Browser manufacturers propose to *proactively* (or periodically) push CRL files to browsers [20,29,52]. Thus, the browsers utilize these CRL files to check the certificate revocation status in TLS negotiations. The dilemma of certificate coverage vs. CRL size exists in the solutions of proactive CRL: in order to cover all (or even the most) of the TLS server certificates, the proactively-pushed CRL file will be greater than 100 MB [33]; on the other hand, small-sized CRL will frequently fail in the checking of certificate revocation status. So CRLite [33] and Let's Revoke [47] recently propose more efficient data structures to encode the revocation status data into small-sized messages, to periodically push the revocation status of *all* TLS server certificates to browsers.

3 The Technical Challenges in SCMS

This section firstly discusses the V2V PKI system. Then, we present the structure of SCMS, and the technical challenges to integrate certificate transparency and push-based certificate revocation into SCMS.

SCMS is designed for V2V communications, as well as other V2V PKI solutions [6,53], especially for the continuous broadcast of basic safety messages (BSMs) by the on-board equipment (OBE) device in each vehicle. It is estimated that BSMs will prevent most of the roadway crashes through active safety applications [54].

3.1 The Expected Properties of V2V PKI Systems

We briefly list the expected properties of V2V PKI systems. These properties shall be considered in certificate signing and certificate verification.

- **Trustworthiness.** As a security infrastructure for cyber-physical systems, the V2V PKI services shall be highly trustworthy. Any defects in this system will cause physical damages and bring direct profits to the attackers, so it needs to be well-protected.
- **User Privacy.** The most primary privacy concern is to protect users against vehicle tracking. That is, it shall be very difficult for the eavesdroppers in physically distant locations to tell whether two BSMs are sent by the same vehicle or not. Frequent change of different pseudonym certificates (e.g., every 10 min) is the common design.
- **High-Volume.** A V2V PKI system shall be able to support hundreds of millions of vehicles, and this number is increasing. Note that the number of pseudonym certificates may be several thousands times (i.e., hundreds of billions), when user privacy is considered and the frequent change of certificates is adopted.

- **Efficiency.** Wireless BSMs are broadcast at least every 100 ms among vehicles, and processed by the embedded OBE devices. Active safety applications usually do not tolerate any latency greater than 0.2 ms. So the security-related operations must be efficient, especially certificate verification in the process of received BSMs.

3.2 SCMS

We list the special features of the certificate services in SCMS as below, while we skip the steps of common certificate services (e.g., CA hierarchy and enrollment certificate) and some special steps but unrelated to fraudulent certificate and certificate revocation (e.g., butterfly key expansion and implicit certificate). More details of SCMS can be found in [58].

In the SCMS PKI, each certified OBE device holds a long-lived enrollment certificate after the device bootstrap. Then, this enrollment certificate is used to apply for multiple short-lived pseudonym certificates, which are then used to sign/verify BSMs. Each OBE device holds 20–40 simultaneously-valid pseudonym certificates for every week, and it applies for pseudonym certificates for signing the BSMs of 1–3 years (i.e., about 1040–6240 pseudonym certificates) in a batch [58]. In order to protect user privacy, any pair of these pseudonym certificates cannot be linked, unless two SCMS internal components collude or these certificates are revoked.

SCMS depends on CRL to revoke certificates, and CRL files are broadcast by road side equipments or satellite radio systems to all OBE devices in the roadway. SCMS does not consider OCSP, because it is impractical for a vehicle to acquire OCSP responses frequently, either the BSM sender or the verifier. That is, CRL files are periodically pushed to all OBE devices, so that an OBE device will locally maintain the identifiers of all revoked certificates. Moreover, linkage-based revocation is designed to reduce the size of CRL files in SCMS, and an entry in linkage-based CRL represents all unexpired but revoked pseudonym certificates for which a certain OBE device applies in a batch [27,58]. With linkage-based revocation, a revocation identifier (or CRL entry) in the CRL file represents a batch of pseudonym certificates: if the relationship of the identifier (i.e., serial number) of a pseudonym certificate and the revocation identifier satisfies the specified rules, the pseudonym certificate is considered as revoked.

3.3 Technical Challenge #1: Certificate Transparency vs. Pseudonym Certificate

Certificate transparency is proposed against compromised CAs of the Web PKI [10,16,19,25,41,56,59,60]. Although SCMS has not been widely deployed in the real world yet, it is reasonable to assume that some CAs in SCMS might be compromised or deceived to sign fraudulent pseudonym certificates for software vulnerabilities and cyber attacks frequently happen. Then, a fraudulent pseudonym certificate enables an attacker (but not the certified OBE device embedded in some vehicle) to arbitrarily broadcast BSMs, to impersonate a vehicle in the road. So certificate transparency and/or other countermeasures [5,17,30,51,57] are also necessary to enhance the trustworthiness of SCMS in the future.

Firstly, as mentioned in Sect. 2.1, certificate transparency depends on the victim website by itself to search for certificates binding its domain name, to detect fraudulent certificates in the public logs. However, in SCMS a *pseudonym* certificate does not bind any meaningful identity or identifier. So no one is able to search for these certificates to detect fraudulent ones, even when all SCMS pseudonym certificates are publicly recorded in log servers. Certificate transparency involves the certificate search based on the identifiers of certificate holders (e.g., domain names) [34], but pseudonym certificates have to mask these identifiers. Therefore, the security principle of certificate transparency does not work well for the pseudonym certificates in SCMS.

Next, we further analyze other solutions which are proposed to tame the absolute authority of CAs [4,38] against possible fraudulent certificates. The existing schemes include:

- **Public key pinning** [17]**.** A browser locally pins the certificates (or public keys) of a TLS server for the visited domain, after a successful TLS negotiation. The pinned public key will be compared with the received certificates in the future TLS negotiations, and then any mismatching is detected immediately by the browser.
- **Restricted scopes of services** [30,48]**.** A CA of the Web PKI is restricted to serve only some scopes of domains, and the restriction rules are enforced by browsers when verifying the TLS server certificates. A certificate violating the rules will be rejected by browsers.
- **Multi-path verification** [1,57]**.** On receiving a server certificate in TLS negotiations, a browser compares it with other copies obtained through different network paths (e.g., an extra Tor circuit). The certificate is accepted, only if they are identical.
- **Subject-controlled policies** [26,51]**.** The certificate subject (or website) specifies its own certificate policies (e.g., a list of authorized CAs), and these policies are published in a publicly-visible means. Any TLS server certificate violating these policies (e.g., a certificate signed by an unauthorized CA) is considered as invalid or fraudulent.
- **Multi-authority certification** [5,51]**.** A TLS server certificate is certified and signed redundantly by multiple independent CAs, and the browser verifies all CAs' signatures. Only when all signatures are valid, the certificate is considered as valid.

Let's analyze the scenarios when these schemes are integrated into SCMS for V2V communications. Public key pinning requires a certificate verifier (i.e., another vehicle or a road side equipment receiving BSMs) to maintain the pubic keys of communication peers, but it is really unsuitable for SCMS because every OBE device holds 20–40 simultaneously-valid pseudonym certificates and each pseudonym certificate expires in days [58]. That is, a pinning mismatching happens frequently but normally in the V2V communications. Restricted scopes of services only mitigate the attack impact of fraudulent certificates but cannot completely prevent or detect the attacks, so it is not enough for cyber-physical systems. Multi-path verification introduces addition connections and greatly degrades the performance, and it brings false alarms when the website (or vehicle) holds multiple valid certificates simultaneously.

Similar to certificate transparency, subject-controlled policies do not work for pseudonym certificates, for the subject certificate policies shall be published by the long-lived enrollment certificate. Thus, in order to validate pseudonym certificates with such a policy will inevitably break the user privacy. Finally, multi-authority certification will remarkably increase the delay of certificate verification due to the expensive public-key cryptographic computation of multiple signature verifications.

Therefore, it is rather difficult but imperative to design different solutions to prevent or detect (possible) fraudulent certificates in SCMS, which work for pseudonym certificates and delay-sensitive communications. Combining (the security principles of) these existing schemes into a specialized solution may work for V2V communications, such as Elaphurus [38] for the Web PKI. However, since the V2V communications do not tolerate high delays, a solution focuses on the steps of certificate signing but not certificate verification may be more practical.

3.4 Technical Challenge #2: Push-Based Certificate Revocation vs. the Great Volume of Pseudonym Certificates

Push-based certificate revocation is proposed to improve the efficiency of certificate verification. If certificate revocation is possible (e.g., the OBE device is compromised), such efficiency improvements of revocation status checking are always required in SCMS. SCMS is designed for V2V communications, which are very delay-sensitive. The push model has actually been adopted in SCMS already, and all CRL files are broadcast to OBE devices in the vehicles [58]. Therefore, because the number of certificates in SCMS is much greater than that in the Web PKI, the dilemma of certificate coverage vs. CRL size also exists in SCMS (or will become even worse), and the efficient data structures to encode certificate revocation status data such as CRLite [33] and Let's Revoke [47] are more imperative.

Due to the great volume of pseudonym certificates, the size of CRL files may bring very heavy burdens when SCMS is deployed in the real world. There are hundreds of millions of vehicles, and hundreds of billions of pseudonym certificates will be signed in SCMS [58]. These numbers are much greater than those of the Web PKI, where there are only 24–32 millions unrevoked TLS server certificates and about 12 millions revoked ones [33]. This implies that the size of CRL files in SCMS will be at least 10 times that of the Web PKI according to the number of users, or the size will be even 10,000 times that of the Web PKI according to the number of pseudonym certificates. For example, Apple signs a

CRL file over 76 MB [39] in the Web PKI, and it is very expensive or even impossible for an embedded OBE device to receive and store such CRL files (or raw certificate revocation status data) through wireless communications. Even if the certificate revocation rates are roughly equal, the CRL file may be about 1 GB in the case of linkage-based revocation (i.e., the CRL size is estimated according to the number of users, but not the number of certificates); while if the certificate revocation rate increases due to some security incidents (e.g., OpenSSL Heartbleed resulted in an irregular 40-fold increase of revocation rate [14,61]), the CRL file will be probably expanded to even several GB. In fact, as typical embedded systems, the OBE devices might be exposed to more physical attack surfaces than TLS webservers with professional administrators, and then the certificate revocation rate of SCMS could become greater.

We next study the recently-proposed efficient designs of data structures for push-based certificate revocation (i.e., CRLite [33] and Let's Revoke [47]). CRLite utilizes the Bloom-filter cascade to encode the identities of both all revoked certificates and all unrevoked ones. This introduces at least two challenges as follows: (a) CRLite requires certificate transparency to fetch all unrevoked certificates from the public logs; otherwise, some valid certificates may be falsely checked as revoked due to the inherent false positive of Bloom filters; and (b) linkage-based revocation of SCMS cannot work compatibly with CRLite, because it requires the *explicit* identifier of a batch of pseudonym certificates in the revocation status data [58] while CRLite cannot extract or recover the identifiers of revoked certificates after they have been encoded into the Bloom-filter cascade. Meanwhile, Let's Revoke depends on a special rule to generate deterministic certificate serial numbers (i.e., a sequence of incremental integers as certificate identifiers, so that only one bit is enough to indicate the revocation status in CRL), but this rule conflicts with linkage-based revocation in SCMS where a certificate serial number is computed by XORing two *random* linkage values, generated by two independent linkage authorities [58]. If the CA assigns another field as the certificate identifier for Let's Revoke, linkage-based revocation will not work and then the size of revocation status data will become 1,000 times.

Besides, the optimization of proactive CRL distributions utilizing the locality of reference [12,36] requires a locally-centralized gateway to cache certificate revocation status data, through either CRL or OCSP. However, such a gateway does not exist in the dynamic wireless V2V communications.

In summary, neither CRLite nor Let's Revoke works compatibly in SCMS. Meanwhile, although linkage-based revocation has been designed in SCMS to reduce the size of CRL, this size reduction is still not enough based on the experiences and lessons in the Web PKI. So it still needs a more efficient data structure to push the revocation status data in SCMS. In the future, we plan to integrate linkage-based revocation and Let's Revoke (or other efficient data structures). For example, Let's Revoke is utilized to revoke a *batch* of certificate signing which results in about 1040–6240 pseudonym certificates for one vehicle, and linkage-based revocation helps to find all identifiers of these pseudonym certificates.

4 Conclusions

Unexpected technical challenges usually appear and then advances are finished, as a security design is deployed in the real world. Although the structure of PKI systems has been discussed, analyzed, implemented and evaluated for more than thirty years, recent advances still happen in the Web PKI as TLS and HTTPS are being widely adopted in the Internet. In particular, certificate transparency and push-based revocation are the most significant recent advances in the Web PKI, to (a) improve the trustworthiness against fraudulent TLS server certificates and (b) eliminate the addition connections for certificate revocation status data, respectively.

SCMS is a specialized PKI system for V2V communications. The number of certificates in SCMS will become much greater than that in the Web PKI, and the V2V communications for BSM broadcast are very delay-sensitive. Therefore, we do believe that the problems solved by certificate transparency and push-based revocation in the Web PKI will appear again when SCMS is deployed in the real world. In this paper, we study these technical challenges in SCMS and find that the existing approaches in the Web PKI do not work compatibly for SCMS. These discussions will be useful references to improve the designs of SCMS in the future.

Acknowledgement. This work was partially supported by National Key RD Plan of China (Grant No. 2020YFB1005803) and National Natural Science Foundation of China (Grant No. 61772518).

References

1. Alicherry, M., Keromytis, A.: DoubleCheck: multi-path verification against man-in-the-middle attacks. In: 14th IEEE Symposium on Computers and Communications (ISCC) (2009)
2. Apple Inc.: Apple's certificate transparency policy (2019). https://support.apple.com/en-us/HT205280
3. Apple Inc.: Lists of available trusted root certificates in macOS (2020). https://support.apple.com/en-us/HT202858
4. Bao, X., Zhang, X., Lin, J., Chu, D., Wang, Q., Li, F.: Towards the trust-enhancements of single sign-on services. In: 3rd IEEE Conference on Dependable and Secure Computing (DSC) (2019)
5. Basin, D., Cremers, C., Kim, H., Perrig, A., Sasse, R., Szalachowski, P.: ARPKI: attack resilient public-key infrastructure. In: 21st ACM Conference on Computer and Communications Security (CCS) (2014)
6. Bibmeyer, N., Stubing, H., Schoch, E., Gotz, S., Stolz, J., Lonc, B.: A generic public key infrastructure for securing car-to-X communication. In: 18th World Congress on Intelligent Transport Systems (ITS) (2011)
7. Brown, D.R.L., Gallant, R., Vanstone, S.A.: Provably secure implicit certificate schemes. In: Syverson, P. (ed.) FC 2001. LNCS, vol. 2339, pp. 156–165. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46088-8_15
8. Campagna, M.: SEC 4: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV). Technical report, Certicom Research (2013)

9. Comodo CA Limited: crt.sh: Certificate search (2020). https://crt.sh
10. Comodo Group Inc.: Comodo report of incident (2011). https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html
11. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: IETF RFC 5280 - Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile (2008)
12. Dickinson, L., Smith, T., Seamons, K.: Leveraging locality of reference for certificate revocation. In: 35th Annual Computer Security Applications Conference (ACSAC) (2019)
13. Dierks, T., Rescorla, E.: IETF RFC 5246 - The transport layer security (TLS) protocol (2008)
14. Durumeric, Z., et al.: The matter of Heartbleed. In: 14th Internet Measurement Conference (IMC) (2014)
15. Eastlake, D.: IETF RFC 6066 - Transport layer security (TLS) extensions: extension definitions (2011)
16. Eckersley, P.: A Syrian man-in-the-middle attack against Facebook (2011). https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook
17. Evans, C., Palmer, C., Sleevi, R.: IETF RFC 7469 - Public key pinning extension for HTTP (2015)
18. Facebook Inc.: Facebook: Certificate transparency monitoring (2020). https://developers.facebook.com/tools/ct/search/
19. GlobalSign: Security incident report (2011). https://www.globalsign.com/resources/globalsign-security-incident-report.pdf
20. Goodwin, M.: Revoking intermediate certificates: introducing OneCRL (2015). https://blog.mozilla.org/security/2015/03/03/revoking-intermediate
21. Google Inc.: Certificate transparency in Chrome (2018). https://github.com/chromium/ct-policy/blob/master/ct_policy.md
22. Google Inc.: Google: HTTPS encryption on the web (2018). https://transparencyreport.google.com/https/certificates
23. Google Inc.: Known logs (2018). http://www.certificate-transparency.org/known-logs
24. Hallam-Baker, P.: IETF RFC 7633 - X.509v3 transport layer security (TLS) feature extension (2015)
25. Adkins, H.: An update on attempted man-in-the-middle attacks (2011). https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html
26. Hoffman, P., Schlyter, J.: IETF RFC 6698 - The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA (2012)
27. IEEE Vehicular Technology Society: IEEE 1609.2: Standard for wireless access in vehicular environments (2016)
28. ITU-T: X.509: Information technology - Open systems interconnection - The directory: Authentication framework (1997)
29. Jones, J.C.: CRLite: Speeding up secure browsing (2020). https://blog.mozilla.org/security/2020/01/21/crlite-part-3-speeding-up-secure-browsing/
30. Kasten, J., Wustrow, E., Halderman, J.A.: CAge: taming certificate authorities by inferring restricted scopes. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 329–337. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_28
31. Langley, A.: No, don't enable revocation checking (2014). https://www.imperialviolet.org/2014/04/19/revchecking.html
32. Langley, A.: Revocation still doesn't work (2014). https://www.imperialviolet.org/2014/04/29/revocationagain.html

33. Larisch, J., Choffnes, D., Levin, D., Maggs, B., Mislove, A., Wilson, C.: CRLite: a scalable system for pushing all TLS revocations to all browsers. In: 38th IEEE Symposium on Security and Privacy (S&P) (2017)
34. Laurie, B., Langley, A., Kasper, E.: IETF RFC 6962 - Certificate transparency (2014)
35. Li, B., et al.: Certificate transparency in the wild: exploring the reliability of monitors. In: 26th ACM Conference on Computer and Communications Security (CCS) (2019)
36. Li, B., Lin, J., Wang, Q., Wang, Z., Jing, J.: Locally-centralized certificate validation and its application in desktop virtualization systems. IEEE Trans. Inf. Forensics Secur. (TIFS) **16**, 1380–1395 (2021)
37. Li, B., Chu, D., Lin, J., Cai, Q., Wang, C., Meng, L.: The weakest link of certificate transparency: exploring the TLS/HTTPS configurations of third-party monitors. In: 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (2019)
38. Li, B., Wang, W., Meng, L., Lin, J., Liu, X., Wang, C.: Elaphurus: ensemble defense against fraudulent certificates in TLS. In: Liu, Z., Yung, M. (eds.) Inscrypt 2019. LNCS, vol. 12020, pp. 246–259. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-42921-8_14
39. Liu, Y., et al.: An end-to-end measurement of certificate revocation in the web's PKI. In: 15th Internet Measurement Conference (IMC) (2015)
40. Microsoft Corporation: Microsoft trusted root certificate program: participants (2020). https://gallery.technet.microsoft.com/Trusted-Root-Program-d17011b8
41. Morton, B.: More Google fraudulent certificates (2014). https://www.entrust.com/google-fraudulent-certificates/
42. Mozilla: Mozilla CT policy (2019). https://groups.google.com/a/chromium.org/forum/m/#!topic/ct-policy/Xx1bv8r33ZE
43. Mozilla: Mozilla included CA certificate list (2020). https://wiki.mozilla.org/CA/Included_Certificates
44. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: IETF RFC 2560 - X.509 Internet public key infrastructure online certificate status protocol - OCSP (1999)
45. Opsmate Inc.: SSLMate: Cert spotter (2020). https://sslmate.com/certspotter/
46. Rescorla, E.: IETF RFC 2818 - HTTP over TLS (2000)
47. Smith, T., Dickenson, L., Seamons, K.: Let's revoke: scalable global certificate revocation. In: 27th ISOC Network and Distributed System Security Symposium (NDSS) (2020)
48. Soghoian, C., Stamm, S.: Certified lies: detecting and defeating government interception attacks against SSL (short paper). In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 250–259. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27576-0_20
49. Stark, E., Huang, L.S., Israni, D., Jackson, C., Boneh, D.: The case for prefetching and prevalidating TLS server certificates. In: 19th ISOC Network and Distributed System Security Symposium (NDSS) (2012)
50. Stark, E., et al.: Does certificate transparency break the web? Measuring adoption and error rate. In: 40th IEEE Symposium on Security and Privacy (S&P) (2019)
51. Szalachowski, P., Matsumoto, S., Perrig, A.: PoliCert: secure and flexible TLS certificate management. In: 21st ACM Conference on Computer and Communications Security (CCS) (2014)
52. The Chromium Projects: CRLSets (2015). https://dev.chromium.org/Home/chromium-security/crlsets

53. US Department of Transportation: Safety pilot model deployment (2012). http://safetypilot.umtri.umich.edu/
54. US Department of Transportation: Vehicle-to-vehicle (V2V) communications for safety (2020). https://one.nhtsa.gov/Research/Crash-Avoidance
55. VanderSloot, B., Amann, J., Bernhard, M., Durumeric, Z., Bailey, M., Halderman, A.: Towards a complete view of the certificate ecosystem. In: 16th Internet Measurement Conference (IMC) (2016)
56. VASCO Data Security International Inc.: DigiNotar reports security incident (2011). https://www.vasco.com/about-vasco/press/2011/news_diginotar_reports_security_incident.html
57. Wendlandt, D., Andersen, D., Perrig, A.: Perspectives: improving SSH-style host authentication with multi-path probing. In: USENIX Annual Technical Conference (ATC) (2008)
58. Whyte, W., Weimerskirch, A., Kumar, V., Hehn, T.: A security credential management system for V2V communications. In: 5th IEEE Vehicular Networking Conference (VNC) (2013)
59. Wikipedia: Flame (malware) (2017). https://en.wikipedia.org/wiki/Flame_(malware)
60. Wilson, K.: Distrusting new CNNIC certificates (2015). https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/
61. Zhang, L., et al.: Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. In: 14th Internet Measurement Conference (IMC) (2014)

The First International Workshop on Security for Internet of Things (IOTS 2021)

A Novel Approach for Code Smells Detection Based on Deep Learning

Tao Lin[1](✉), Xue Fu[2](✉), Fu Chen[3](✉), and Luqun Li[2](✉)

[1] Amazon, Seattle, WA 98109, USA
paper@Ltao.org
[2] Shanghai Normal University, Shanghai, China
success@shnu.edu.cn
[3] Central University of Finance and Economics, Beijing, China
chenfu@cufe.edu.cn

Abstract. Compared to software bugs, code smells are more significant in software engineering research. It is not easy to detect code smells through traditional methods. In this work, we propose a novel code smells detection approach based on deep learning. The experiments show that our work achieves high scores in terms of F2 score.

Keywords: Code smells · Deep learning · Convolutional neural network

1 Introduction

Code smells are increasingly generated by modern agile software development. This is because code changes are much more frequent and occur on a daily basis for large software companies and dominant open-source communities.

Although there are many more test approaches to detect code smells, these methods have some defects. Due to the frequent changes, it is increasing probable to generate code smells overheads. Code smells, like software bugs, are a serious problem in modern software.

Nowadays, the code smells are being researched by many practitioners. Software developers are not aware of what is the code smell, although they are aware of software bugs, thanks integrated environment development kits that can provide many instant suggestions and notifications when there are bugs.

The question here is how to detect code smells effectively? And what is the motivation for detecting code smells? Although there are many more test approaches to detect code smells, these methods have some defects. There are mainly two categories of deep learning networks. One is Recurrent Neural Networks, and another is Convolutional networks.

Convolutional networks have already demonstrated its usage by leveraging hierarchy features. In this paper, we use fully convolutional networks for code smells detection based on semantic features. We will use fully convolutional networks for this work.

The original version of this chapter was revised: there is a typo in the chapter title: "leaning" has been corrected to: "learning". The correction to this chapter is available at https://doi.org/10.1007/978-3-030-80851-8_16

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 171–174, 2021.
https://doi.org/10.1007/978-3-030-80851-8_12

We will define the type of neural network we use, and explain how it is used to detect code smells. An advantage of using convolutional network is its ability to identify and use local correspondences.

In recognition and machine learning, convolutional networks are increasingly significant. Convolutional network presents the improvement on image recognition. An example is using convolutional network on local correspondence. In software engineering, we definitely can use these kinds of information for code smells detection.

To our knowledge, this is the first work to train a convolutional network for code smells recognition. The inference is much more improved through convolutional network.

This work is an extension of the authors previous work [8].

Unlike previous works that needs additional information for code smells detection, this work does not use any existing information for code smells detection. One of the major challenges in code smells detection is to find the relationship between code semantics and code location. There is a tradeoff between identifying the correct semantics compared to identifying the correct location of the smells.

Although there are several success stories from image recognition by using deep networks [1]. It is hard to transfer these approaches to software engineering, which is more deterministic. Fully convolutional network has been used for one-layered computation, and has a potential to be deployed to multi-layered environments.

2 Code Smells Detection Based on Convolutional Networks

We can define a multi-dimensional array to represent the convolutional network, h * w * d, where h and w are space dimensions, and d is the channel. The first layer is our source code inputs.

The second layer is the networks for sequence modeling. For example, the inputs are x_0, x_1, x_2, x_3, x_4,... x_n, and the outputs are y_0, y_1, y_2, y_3, y_4,... y_n. The second layer will be y'$_0$, y'$_1$, y'$_2$, y'$_3$, y'$_4$,... y'$_n$.

The outputs will be reshaped to a one-dimensional array, where size will be D *1024. This output array will be dilation blocks. For the encoder task, we should process noise. Each layer in the encoder is processed by normalization and liner analysis.

3 High Level Design

With recent development in software engineering, it is easy to find software bugs using several methods from compiling to running. Our work is based on the state-of-the-art deep learning methods for detection and recognition task.

Firstly, we transform the software source code to XML file, in order to be processed by deep learning models [2]. Then there are two steps: code segment proposal and classification. The code segment proposal leverages the heuristic search to generate following inputs. These segments are processed by CNN classifier. We try to avoid to use R-CNN, otherwise. One of the main reasons is that R-CNN uses selective search algorithm, which is time consuming.

We use the following equation for segments:

$$\text{Seg} = \left(\max rp^{2^{l}}/D + minrq\right) * (I/D)$$

Seg is the segments, r is the rule of limits, and p is process variable, q is the next graph inputs, I is interception, and D is next destination.

3.1 Experiments Results

We use an open-source database published by the authors' previous work [11].

The experiments results are shown as following table:

Table 1. Experiments results

	Precision	Recall	F-score	Kappa
Long method	0.528	0.674	0.754	0.635
Lazy class	0.624	0.678	0.613	0.632
Speculative generality	0.712	0.734	0.689	0.643
Refused bequest	0.698	0.701	0.711	0.677
Duplicated code	0.543	0.568	0.594	0.585
Contrived complexity	0.783	0.792	0.810	0.802
Shotgun surgery	0.597	0.596	0.501	0.601
Uncontrolled side effects	0.801	0.799	0.805	0.810

From Table 1, this work achieves high performance in terms of F2 score, especially for the category of uncontrolled side effects and contrived complexity.

4 Conclusion

In this work, we conducted a research for code smells detection based on deep learning.

Our solution uses convolutional neural network for training a model to detect several common code smells problems in software engineering. The solution achieves satisfied F2 score with the average above 0.75.

Acknowledgements. Part of this work is from the author's PhD study [1], before the author joining Amazon. Professor Fu Chen from Central University of Finance and Economics provided many constructive suggestions and perspectives for this work during author's PhD study. Professor Fu Chen and this work was supported in part by National Science Foundation of China under No. 61672104.

References

1. Lin, T.: A Data Triage Retrieval System for Cyber Security Operations Center. Pennsylvania State University Thesis (2018)
2. Lin, T.: A container - Destructor – Explorer paradigm to code smells detection. J. Chin. Comput. Syst. **37**(3), 17 (2016)
3. Lin, T., Fu, X.: Flame detection based on SIFT algorithm and one class classifier with undetermined environment. Comput. Sci. **42**(6), 6A (2015)
4. Lin, T., Zhong, C., Yen, J., Liu, P.: Retrieval of relevant historical data triage operations in security operation centers. In: Samarati, P., Ray, I., Ray, I. (eds.) From Database to Cyber Security. LNCS, vol. 11170. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04834-1_12
5. Lin, T.: A novel image matching algorithm based on graph theory. Comput. Appl. Softw. **33**(12), 156 (2016)
6. Lin, T.: Graphic user interface testing based on petri net. Appl. Res. Comput. **33**(3), 27 (2016)
7. Lin, T.: A novel direct small world network model. J. Shanghai Norm. Univ. **45**(5), 23 (2016)
8. Lin, T., Gao, J., Fu, X., Lin, Y.: A novel bug report extraction approach. In: Wang, G., Zomaya, A., Perez, G.M., Li, K. (eds.) ICA3PP 2015. LNCS, vol. 9532, pp. 771–780. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27161-3_70
9. Zhong, C., Lin, T., Liu, P., Yen, J., Chen, K.: A cyber security data triage operation retrieval system. Comput. Secur. **76**, 12–31 (2018)
10. Lin, T.: Deep learning for IoT. In: 39th IEEE – International Performance Computing and Communications Conference (2020)
11. Lin, T.: Security Operations Center Retrieval (2021). https://github.com/ltaocs/SecurityOperationsCenterRetrieval

A Thieves Identification Scheme for Prepaid Systems in Smart Grids

Zhiyuan Sui(✉), Hengyue Jia, Fu Chen, and Jianming Zhu

Information School, Central University of Finance and Economics, Beijing, China

Abstract. This paper proposes a privacy preserving thieves identification scheme for prepaid Smart Grid systems. Prepaid systems allow consumers to buy credentials from an operation center before consumption. According to the credentials, consumers can use the corresponding amount of electricity. Based on the dynamic k-times anonymous authentication protocol, the scheme can achieve thieves identification and privacy preservation at the same time without the involvement of a trusted authority in the system. Finally, we point out the path of our future work.

Keywords: Smart grid · Anonymity · Privacy · Security

1 Introduction

With the development of public awareness of environmental conservation, more and more renewable energy come to use. However, the renewable energy sources are so volatile that the power companies have to employ the Information and Communication Technologies (ICT) to balance the power production and consumption. The power company aggregates the power usage reports from consumers to calculate the amount of power consumption. The usage reporting device at each customer site is called as smart meter. The operation center and smart meters form the Smart Grid. The invention of Smart Grids is a great plus to the electricity industry. This mechanism can be exploited to improve energy efficiency and infrastructure reliability. However, trustless data, which does not represent the real consumption, may damage the electricity grid infrastructure [1]. At the same time, the billing information is generated by the usage data. Greedy thieves are not willing to send the trustworthy data and pay for their consumption. Hence, it is a serious challenge to resist fake data and detect power thieves in Smart Grids. On the other hand, privacy is another key requirement in Smart Grids. The usage data from the consumer should not result in finding consumers' usage information, which could lead to disclosure of the consumer's living habits or production outputs, and further causes personalized advertisements or intelligence leakage [2].

To solve the security and privacy preservation challenges, power request model was proposed for prepaid card system. The prepaid smart card system

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 175–180, 2021.
https://doi.org/10.1007/978-3-030-80851-8_13

allows consumers buy credentials from the power company in advance. When the consumer needs power in the existing privacy preserving prepaid card system, he just sends the usage plan with his transformed credentials to the operation center. The operation center will send the power back according to the number of credentials [3]. However, credentials may be collision due to large data base. Moreover, the scheme [3] cannot identify thieves.

Taking the requirements of privacy preservation and thieves identification into account, in this paper, we construct a secure and privacy preserving thieves identification scheme in Smart Grids. There are two important virtues: (1) the scheme resists credential collisions; (2) energy thieves can be identified in the prepaid system.

The rest of this paper is organized as follows. Section 2 discusses related work in the prepaid system domain. Section 3 elaborates and explains necessary system models. In Sect. 4, we describe our proposed scheme. Finally, the paper is concluded in Sect. 5.

2 Related Work

To solve the security and privacy preservation challenges, power request models for smart grid systems have been proposed [3]. In such models, consumers receive tokens signed with blind signatures from the power provider and authenticate themselves using blinded tokens. The tokens represent a corresponding denomination of energy cost. Consequently, the power provider knows the total amount of customer energy costs but do not know the details of the usage information. However, this scheme cannot identify power thieves. Dimitriou et al. [4] added a proof to the blind signature. According to the proof, reused tokens from the same consumer can be identified. However, the tokens are generated by the consumers, and token collisions are inevitable. Zhao et al. [5] employed a fully homomorphic encryption algorithm to aggregate smart meters' usage data. Based on the homomorphism of the ciphers, consumers' billing can be calculated without knowing the plaintexts. However, current fully homomorphic encryption algorithms are less efficient. Xue et al. [6] improved the Paillier encryption algorithm. The exponent on the cipher is also additive homomorphism. Therefore, it can also achieve dynamic billing management with less computational cost. Li et al. [7] divided the usage data into multiple parts according to their denominations. In their study, each smart meter holds a corresponding number of key pairs with respect to the divided sets. Therefore, smart meters belonging to different sets should pay their corresponding billings.

Smart Grids cannot ensure improvement in infrastructure reliability without trustworthy information. At the same time, consumers do not wish their energy usage data to be exposed to the operation center. However, existing thieves identification approaches cannot make use of a trade-off between privacy and security. According to prepaid card systems, this paper proposes a privacy-preserving thieves identification scheme without any trusted authority. In this scheme, a thief can be traced if sending a used credential.

3 The Framework

In this section, the assumed network and security requirements in the scheme are described.

3.1 Network Model

Nowadays, in order to improve energy efficiency and infrastructure reliability, a prepaid system is proposed [3]. In the scheme, the system model mainly consists of two entities: the operation center (OC) and the smart meter (SM). The number of smart meters is large enough for each smart meter to cloak its usage behavior. As depicted in Fig. 1, firstly, SMs join the Smart Grid and obtain their commitments from the OC. Secondly, each SM buys credentials from the OC. Thirdly, a SM sends blinded commitments and credentials in an anonymized form when it needs to purchase electricity.

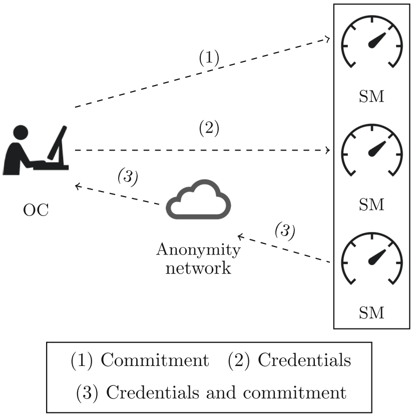

(1)
(2)
(3)
(3)
OC
Anonymity network
SM
SM
SM
(1) Commitment (2) Credentials
(3) Credentials and commitment

Fig. 1. Network model.

3.2 Security Requirements

The privacy-preserving thieves identification scheme is expected to exhibit the following properties:

1. **Privacy Preservation:** Curious eavesdroppers cannot obtain consumers' usage profiles from the usage reports from smart meters.
2. **Unlinkability:** The adversary cannot link different usage reports from the same smart meter.
3. **Authentication:** The operation center ensures that the usage reports are from legitimate smart meters in the Smart Grid.
4. **Traceability:** Energy thieves can be traced if they attempt to send used credentials.

4 Our Proposed Approach

In this section, a privacy-preserving thieves identification scheme is proposed based on the dynamic k-times anonymous authentication scheme.

4.1 Setup

In the privacy-preserving thieves identification scheme, a concrete region is managed by a single operation center. The operation center runs the setup algorithm on input of the security size κ to obtain a group public key and a secret key as follows:

1. Let p be a prime number of size κ, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. Suppose P_1 and P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and e is a bilinear map. On input of κ, the bilinear pairing instance generator returns a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{Z}_p^*, e, P_1, P_2)$.
2. Then, the operation center chooses collision-resistant hash functions $\mathcal{H}_{\mathbb{Z}_p^*} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $\mathcal{H}_{\mathbb{G}^2} : \{0,1\}^* \rightarrow (\mathbb{G}_1, \mathbb{G}_1)$, two elements $V, Q \in \mathbb{G}_1$ and $\gamma \in \mathbb{Z}_p^*$, and computes $P_{\text{pub}} = \gamma P$.
3. Finally, the operation center retains its secret key γ, publishes its public key $(P_1, P_2, V, Q, P_{\text{pub}})$, and builds an empty authentication log **LOG** that records the used credentials.

4.2 Joining

The joining protocol is carried out between the operation center and a smart meter. Each household or company is equipped with a smart meter in the Smart Grid system. A secure public key signature scheme, including a signing algorithm **sig** and a verification algorithm **ver**, is selected for a smart meter. In this protocol, the smart meter must reveal its unique identity ID_j to the operation center as follows. Firstly, the smart meter generates a secure parameter $x_j \in \mathbb{Z}_p^*$, computes a request $C_j = x_j P_1$ and generates a signature $\sigma = \mathbf{sig}(C_j \| \text{ID}_j)$. The smart meter sends the signature σ with the request C_j and identity ID_j to the operation center. Upon the receipt, the operation center computes a hash value $f_j = \mathcal{H}_{\mathbb{Z}_p^*}(\text{ID}_j)$, grants the request $S_j = \frac{1}{(f_j+\gamma)}(C_j + Q)$ and replies S_j. Upon receipt, the smart meter confirms that the equation $e(S_j, f_j P_2 + P_{\text{pub}}) = e(x_j P_1 + Q, P_2)$ holds. Finally, the smart meter retains its secret key x_j and publishes its public key (C_j, S_j).

4.3 Power Purchasing

The power-purchasing protocol is carried out between the operation center and a smart meter. Using this protocol, the smart meters buy credentials from the operation center and are granted the right to obtain electricity.

Periodically, the operation center publishes the bound number $k > 0$ and calculates the set of public credentials $\{(t_i, \hat{t}_i) = \mathcal{H}_{\mathbb{G}^2}(i, n_T) \mid 1 \leq i \leq k\}$, where n_T denotes the timestamp. The public credential $(t_i, \hat{t}_i)$ is called the ith credential base of the operation center. If q smart meters want to buy k units of electricity, represented by k credentials, the smart meters send their identities to the operation center. The operation center calculates $\{f_j = \mathcal{H}_{\mathbb{Z}_p^*}(\mathrm{ID}_j) \mid 1 \leq j \leq q\}$ and $W = \sum_{j=1}^{n}(\gamma + f_j)V$ and $V_j = \frac{1}{\gamma+f_j}W$ for the jth smart meter. The operation center publishes $\{(t_i, \hat{t}_i) \mid 1 \leq i \leq k\}$ and sends V_j to the smart meter ID_j. The smart meter operates a counter μ_j, which is initially set to zero.

4.4 Power Requesting

The power-requesting protocol is run by the operation center and a smart meter to prove knowledge of a smart meter's key x_j and an identity hash value f_j. The smart meter transforms its commitment and credentials, and sends them back to the operation center. Therefore, the operation center cannot locate the source of the public parameters even though the credentials generated by itself.

1. Firstly, the smart meter analyzes the usage data and estimates the amount of energy m. The smart meter increases the counter number by m.
2. If $\mu_j > k$, the smart meter will jump to the power-purchasing algorithm; otherwise, the smart meter sets $\mu_j = \mu_j + m$, chooses m credentials denoted as $(t_1, \hat{t}_1)$, ..., $(t_m, \hat{t}_m)$, and outputs m hash values $\{c_i = \mathcal{H}_{\mathbb{Z}_p^*}(t_i \| \hat{t}_i \| n_T) \mid 1 \leq i \leq m\}$, where n_T denotes the timestamp.
3. After that, the smart meter computes the credentials $\{(\mathit{\Gamma}_i = x_j t_i, \hat{\mathit{\Gamma}}_i = f_j t_i + c_i x_j \hat{t}_i) \mid 1 \leq i \leq m\}$, and generates a proof using the following non-interactive zero-knowledge proof:

$$\left\{ \begin{pmatrix} S_j \\ x_j \\ f_j \\ V_j \end{pmatrix} : \begin{array}{l} e(S_j, f_j P_2 + P_{\mathrm{pub}}) = e(x_j P_1 + Q, P_2) \\ e(V_j, f_j P_2 + P_{\mathrm{pub}}) = e(W, P_2) \\ (\mathit{\Gamma}_i = x_j t_i, \hat{\mathit{\Gamma}}_i = f_j t_i + x_j c_i \hat{t}_i) \end{array} \right.$$

   The smart meter proves its credentials by sending the proof and transformed credential to the operation center. Please refer to [8] for the proof.
4. After receiving the proof and credential, the operation center checks the validity of the timestamp. If it is valid, the operation center compares the credential $(\mathit{\Gamma}_i, \hat{\mathit{\Gamma}}_i)$ with all corresponding credentials in **LOG**, and checks if it is different to the credentials in **LOG**. Finally, the operation center verifies that the proof does prove knowledge of its identity information.

4.5 Identification

If the operation center validates the signature σ and discovers that a received credential has already been used, it will run the identification algorithm. Suppose there are two credentials $(\mathit{\Gamma}_i, n_T)$ and $(\mathit{\Gamma}'_i, n'_T)$, where $\mathit{\Gamma}_i = \mathit{\Gamma}'_i$ and $n_T \neq n'_T$. The operation center can confirm that a credential has been used more than once.

Then, the operation center computes $c = \mathcal{H}_{\mathbb{Z}_p^*}(t_i \| \hat{t}_i \| n_T)$, $c' = \mathcal{H}_{\mathbb{Z}_p^*}(t_i \| \hat{t}_i \| n'_T)$ and $\beta = \frac{c'\hat{\Gamma}_i - c\hat{\Gamma}'_i}{c - c'}$, searches for the ID_j which satisfies $\beta = \mathcal{H}_{\mathbb{Z}_p^*}(\mathrm{ID}_j)t_i$.

The public keys $\{(t_i, \hat{t}_i) \mid 1 \leq i \leq k\}$ are produced by a collision-resistant function, making the transformed credentials $\{(\Gamma_i, \hat{\Gamma}_i) \mid 1 \leq i \leq k\}$ also resistant to collisions. Therefore, malicious smart meters cannot deny their misbehavior.

5 Conclusion

Energy theft occurs frequently in Smart Grids due to its large amount of consumers. In order to defend against energy theft under prepaid smart grid systems, this paper proposes a scheme to achieve identification of thieves without a trusted party. Only an operation center and smart meters are required in the scheme. The operation center checks the smart meters' commitments and corresponding credentials. Therefore, the credentials are resistant to collisions. Great computational capacity is required for operation centers. For the future work, we will study possible ways and improve k-times anonymous authentication scheme to reduce the computational cost during power request procedures.

Acknowledgement. This work is supported by Funds of the National Natural Science Foundation of China (U201509214), (61672104), (62072487) and (61906220).

References

1. Gunduz, M.Z., Das, R.: Cyber-security on smart grid: threats and potential solutions. Comput. Netw. **169**(14), 107094 (2020)
2. Ferraga, M.A., Maglarasc, L.A., Janickec, H., Jiang, J.: A systematic review of data protection and privacy preservation schemes for smart grid communications. Sustain. Cities Soc. **38**, 806–835 (2018)
3. Chim, T.W., Yiu, S.M., Hui, L.C.K., Li, V.O.-K.: Privacy-preserving advance power reservation. IEEE Commun. Mag. **50**(8), 18–23 (2012)
4. Dimitriou, T., Karame, G.: Enabling anonymous authorization and rewarding in the smart grid. IEEE Trans. Dependable Secure Comput. **14**(5), 565–572 (2017)
5. Zhao, S., Li, F., Li, H., Lu, R., Ren, S.: Smart and practical privacy-preserving data aggregation for fog-based smart grids. IEEE Trans. Inf. Forensics Secur. **16**, 521–536 (2021)
6. Xue, K., et al.: PPSO: a privacy-preserving service outsourcing scheme for real-time pricing demand response in smart grid. IEEE Internet Thins J. **6**(2), 2486–2496 (2019)
7. Li, S., Zhang, X., Xue, K., Zhou, L., Yue, H.: Privacy-preserving prepayment based power request and trading in smart grid. Chin. Commun. **15**(4), 24–37 (2018)
8. Nguyen, L., Safavi-Naini, R.: Dynamic k-times anonymous authentication. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 318–333. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_22

Using Smart Contracts to Improve Searchable Symmetric Encryption

Yanbing Wu[1] and Keting Jia[2](✉)

[1] Institute for Advanced Study, Tsinghua University, Beijing 100084, China
wuyb14@mails.tsinghua.edu.cn

[2] Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing 100084, China
ktjia@mail.tsinghua.edu.cn

Abstract. In this paper, we propose a smart contract based searchable symmetric encryption scheme. The existing searchable symmetric encryption protocol can resist malicious servers when using the MAC algorithm; however, it is more effective only under the assumption that the server is running. If the server receives a user's money but does not provide a service to the user (or if the server shuts down after receiving the user's money), the user cannot withdraw the money paid. In addition, if the server wants to reduce computing costs, bandwidth, etc., then it may reduce the number of documents to be searched or omit part of the search results. As a result, there is no guarantee that all files have been searched. We use the Merkle tree to construct search integrity verification. Implementing search integrity verification ensures that it is nearly impossible for searchers to provide integrity verification without searching all documents. Smart contracts use computing resources effectively and help us better search the blockchain. All information is recorded on the blockchain and will not be tampered with. In addition, integrity verification and smart contracts slightly reduce the efficiency but are feasible in practice. Finally, we have theoretically and experimentally verified the safety and feasibility of the proposed scheme.

Keywords: Smart contract · Blockchain · Searchable symmetric encryption · Result integrity · Verifiable · Bloom filter

1 Introduction

A keyword index helps us search for documents containing specified keywords in a certain period of time. Security indexes are extensions of building data structures, e.g., indexes provided by unrelated data structures [13] and historically independent [2,12] data structures. Unfortunately, the standard index structure using hash tables is not suitable for indexing encrypted documents because they leak information about the contents of the document. However, data structures with privacy protection can be used to build secure indexes.

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 181–202, 2021.
https://doi.org/10.1007/978-3-030-80851-8_14

To improve the efficiency of searchable symmetric encryption (SSE), Goh et al. proposed using the Bloom filter method to search symmetric encryption [4]. The Bloom filter proposed by Bloom is a space-saving random data structure that uses a bit array to represent a collection very concisely. It can determine whether an element belongs to this collection. The Bloom filter was used in the early UNIX spell checker [9,11]. The Bloom filter is also used as an index for each document to search the keywords [4] in each document. In the index, a keyword is represented by a codeword derived by applying a pseudorandom function twice, i.e., once using the keyword as input and once using a unique document identifier as input.

In 2017, Li et al. [8] combined blockchain technology with SSE, where a blockchain is used as a peer-to-peer network to store user data. This scheme adds data as blocks to the blockchain, thereby storing the data in the common chain. However, this scheme only supports single keyword search. Tang et al. [1] proposed two frameworks that support blockchain technology, and these frameworks can be applied to most existing SSE schemes to achieve fairness while maintaining the original privacy protection. However, these frameworks are inefficient. Zhang et al. [17] proposed a blockchain based searchable encryption scheme in multicloud environment. Here, multiple cloud service providers are combined to share data through an alliance chain. Then, the encrypted document and document index are stored in IPFs, and the hash value of the document is stored in the blockchain. This scheme can provide retrieval of outsourced encrypted data based on multiple keywords; however, it is based on trusted cloud service providers, which cannot resist malicious cloud servers.

We employ the Bloom filter to construct a searchable symmetric encryption scheme and use smart contracts to maintain the fairness of the searchable symmetric encryption scheme.

The smart contract was originally proposed by Nick Szabo in 1996 [14]. A smart contract is a computer protocol designed to facilitate, verify, or execute a contract. Note that a smart contract in the blockchain is traceable and cannot be tampered with [10]. Many contract clauses can be executed partially or completely in an independent manner. The purpose of smart contracts is to provide better security than traditional contracts and reduce other transaction costs associated with the contract. Various cryptocurrencies are implemented as types of smart contracts. A smart contract is a set of commitments specified in digital form, including an agreement for parties to fulfill these commitments [14].

Our Contribution: We exploit computing resources in the blockchain to propose a searchable symmetric encryption scheme based on smart contracts. To address incomplete retrieval of all files on the server, we first propose a non-interactive integrity verification method for search results. The proposed method uses Merkle trees to construct search integrity verification, which can ensure that it is nearly impossible for searchers to provide integrity verification without searching all documents. In addition, using smart contracts, we can use the computing resources in the blockchain effectively to perform ciphertext searches. As a result, neither the owner nor searcher can cheat. All information (including search results, proof of integrity, and commission payment information) is recorded in the blockchain and cannot be tampered with. By implementing integrity verification and smart contracts, the proposed solution does not reduce efficiency significantly and ensures that searchers cannot obtain information from encrypted documents and search keywords. The proposed scheme guarantees the authenticity and completeness of the results, and, through the Merkle root signature, the proposed scheme ensures that searchers of search results will not be tampered with by miners after submitting the results.

The remainder of this paper is organized as follows. We define notations used in this paper in Sect. 2. In Sect. 3, we provide preliminary information about searchable symmetric encryption and smart contracts. In Sect. 4, we propose the scheme of the non-interactive integrity verification of search results for the first time. In Sect. 5, we present our scheme of smart contract based SSE and discuss the schedule in detail. In Sect. 6, we provide a security definition, security analysis, and performance analysis. Finally, the paper is concluded in Sect. 7.

2 Notations

We define the notations used in this paper in Table 1.

Table 1. Notations

Notation	Description
$\wedge$	Bitwise and
$H_1(), f()$	Pseudorandom function
n	Number of documents
m	Assume that each file has m keywords for brevity
r	Number of function $f()$ used in Bloom filter
s	Key length used in the f function
t	Key length of the encrypted file, the key length of the MAC
l	Security parameter, number of random numbers generated in the integrity verification of search results
n_{tran}	Transaction size
$H()$	Collision resistant hash function
$\|\|$	Concatenation
$bf(y)$	Obtain output of Bloom filter according to y

3 Preliminaries

We propose a searchable symmetric encryption scheme called smart contract-based SSE. The proposed scheme employs a non-interactive result integrity proof method, the Bloom filter, searchable symmetric encryption, and smart contract technology. Here, we introduce the Bloom filter, searchable symmetric encryption and smart contract technology. Eu-jin Guo introduced a Bloom filter method to solve the problem of searchable symmetric encryption [4], which was used to solve the searchable symmetric encryption efficiency problem. We propose an improved scheme in their framework. Szabo proposed the smart contract in 1994 [15], and the proposed method employs smart contracts as a platform to increase the availability of computing resources.

3.1 Scheme of Searchable Symmetric Encryption

To improve the efficiency of searchable symmetric encryption, Eu-jin Guo introduced a Bloom filter method for searchable symmetric encryption [4] that included two participants, i.e., the file owner and the untrusted server. Here, assume the file owner has n files denoted $D_1, D_2, \ldots, D_n$. The file owner encrypts these documents into ciphertexts $C = (C_1, C_2, \ldots, C_n)$ and constructs the corresponding index set of I, and then sends C, I to the server. When the file owner wants to look up documents including keyword w, he calculates trapdoor T_w for keyword w and sends T_w to the server. The server searches for result C_i based on t_w, C and I, and then sends C_i to the file owner. Finally, the file owner decrypts C_i to obtain D_i.

Note that search symmetric encryption is considered secure if the server does not obtain any information about the plaintexts when it only knows ciphertexts or when it does not know any information about the plaintexts and keywords except the search results when it executes search algorithms.

Assume file owner U has n encryption files $C_1, C_2, \ldots, C_n$ on server $\mathcal{S}$. The SSE scheme comprises functions $(BuildIndex(\cdot), SearchIndex(\cdot))$ and three phases (**setup, search, update**). In the **setup** phase, indexes for $C_1, C_2, \ldots, C_n$ are built by the search system, the retrieval task is performed in the **search** phase, and the **update** phase updates the index when documents are changed, added, or deleted. The process of the **setup** phase is shown in Fig. 1.

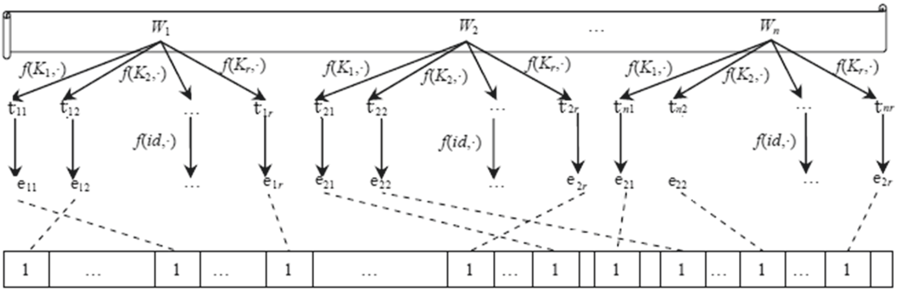


Fig. 1. The secure index setup phase

We introduce the $BuildIndex(\cdot)$ and $SearchIndex(\cdot)$ functions in the following.

- **BuildIndex($C, K_{trapdoor}$):** The function of the input is an encrypted document C with unique file ID $C_{id} \in \{0,1\}^n$ and some keywords $w = (w_0, \ldots, w_t)$, as well as the corresponding keywords trapdoor $K_{trapdoor} = (k_1, \ldots, k_r) \in \{0,1\}^{sr}$, $k_i \in \{0,1\}^s$.
  1. For each document D_i for $i \in [0, n]$, we do the following:
     (a) We compute the trapdoor of each keyword w_i, i.e. $Buildtrapdoor(K_{trapdoor}, w_i) = (t_1 = f(w_i, k_1), \ldots, t_r = f(w_i, k_r)) \in \{0,1\}^{sr}$.
     (b) The corresponding ciphertext ID C_{id} information is added to get the codewords $e : e_1, \ldots, e_r$, where $(e_1 = f(C_{id}, t_1), \ldots, e_r = f(C_{id}, t_r)) \in \{0,1\}^{sr}$.
     (c) The codewords $e : e_1, \ldots, e_r$ is added to the file D_i Bloom filter BF_i.
  2. Output the index of C: $I_C = (C_{id}, BF)$.
- **SearchIndex**(T_w, I_C)**:** The function of the input is the key word trapdoor $T_w = (t_1, \ldots, t_r) \in \{0,1\}^{sr}$ corresponding to keywords W, and the index $I_C = (C_{id}, BF)$ corresponding to encryption file C.
  1. Key word w is encoded according to C_{id} to obtain e, where $e : (e_1 = f(C_{id}, t_1), \ldots, e_r = f(C_{id}, t_r)) \in \{0,1\}^{sr}$.
  2. To determine if each position in y that contains a 1 corresponds to a 1 in the Bloom filter, we must determine if $(bf(e) \wedge BF) == bf(e)$ $(e = e_1, e_2, ..., e_r)$ is correct.
  3. If the above is correct, 1 is output; otherwise, 0 is output.

The **setup**, **search**, and **update** phases are described as follows.

- **Setup phase:** The n files are uploaded to the server after the index is constructed.
  1. First, the proper Bloom filter parameters are derived for each index. In addition, we define $K_{trapdoor} = (k_1, \ldots, k_r) \in \{0,1\}^{sr}$, $k_i \in \{0,1\}^s$.
  2. Each file is assigned a unique ID, which is an integer represented by $i \in [1, n]$.
  3. Each file builds an index as $I_{C_i} \leftarrow BuildIndex(C_i, K_{trapdoor})$.
  4. After each document is compressed and encrypted using standard algorithms, the document and its index can be placed on server $\mathcal{S}$.
- **Search phase:** When file owner $\mathcal{U}$ has a lookup requirement, server $\mathcal{S}$ must be queried for the given keyword w so server $\mathcal{S}$ can return the file containing the given keyword. Then, the following is performed.
  1. File owner $\mathcal{U}$ calculates the trapdoor of w to obtain $T_w \leftarrow Buildtrapdoor(K_{trapdoor}, w)$, and then sends T_w to the server $\mathcal{S}$.
  2. For all the indexes of I_{C_i}, server $\mathcal{S}$ calls SearchIndex (T_w, I_{C_i}) to test matches. All matched files are returned to file owner $\mathcal{U}$.
- **Update phase:** There are three possible scenarios in which an update operation may be used.

1. Add file: When a file is added to the system, a unique ID is first assigned to ciphertext file C, and then index I_{C_i} is created for C.
2. Delete file: When a file is deleted from the system, we must delete both the file and its index file.
3. Changing the contents of an existing document requires assigning a new unique ID to the document and rebuilding the index by calling the $BuildIndex$ function with the new unique ID.

3.2 Smart Contract

Smart contracts are computer programs running on the blockchain. A smart contract comprises program code, stored files, and an account balance. Any user can create smart contracts by posting events to the blockchain. When creating a smart contract, the contract's program code is fixed and cannot be changed. As shown in Fig. 2, the contract storage file is stored in the blockchain. The contract's program logic is executed by miners who reach consensus on the execution results and update the blockchain accordingly. The contract code is executed when the user or another contract receives the message. When executing its code, the contract can read or write from its storage file. A contract can have an account to accept transfers from other accounts or contracts, or it can send transactions to other accounts or contracts. Conceptually, the contract can be considered a special "trusted third party." However, the party is only trusted for correctness (not privacy). Note that the entire state of the contract is visible to all nodes. Figure 2 shows a schematic diagram of a smart contract.

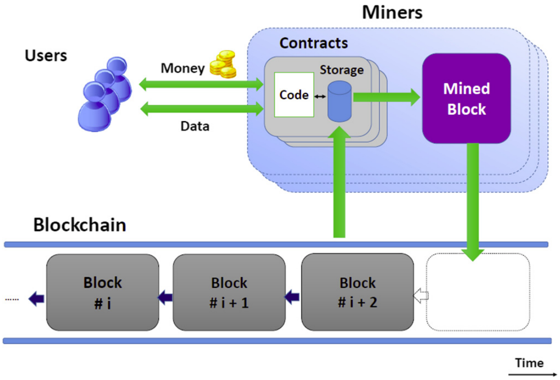


Fig. 2. Schematic of a smart contract

4 Scheme of the Integrity Verification of Search Results

Here, we introduce the non-interactive integrity verification scheme for search results. In this scheme, the Merkle tree is employed to verify the integrity of the search results. The non-interactive integrity verification scheme can be verified on the blockchain. The searcher generates the result integrity certificate after the search is completed. The smart contract can verify whether the searcher has searched all the files according to the proof submitted by the searcher. The final submitted result is complete without any omission.

4.1 Merkle Tree

In cryptography and computer science, the Merkle tree is a tree in which each leaf node is marked with the hash value of a data block, and each non-leaf node is marked with the hash value of the label of its child nodes. Merkle trees, which are generalizations of hash tables and hash chains, allow efficient and safe verification of the contents of large data structures. To prove that a leaf node is part of a given binary hash tree, a logarithmic hash calculation of the number of leaf nodes of the tree is required. Note that this differs from hash lists because the number of hash lists is proportional to the number of leaf nodes. Figure 3 shows an example of a Merkle tree with four leaf nodes.

Merkle trees can be used to verify any type of data stored, processed, and transmitted on or between computers. They ensure that the data blocks received from other nodes in a point-to-point network are undamaged and unchanged, and can even check whether other peer nodes have tampered with the data or sent fake data. A Merkle tree is primarily used for data integrity verification based on a hash. Many systems use Merkle trees for data integrity verification, e.g., the IPFS, Btrfs, and ZFS file systems [18], as well as the Apache wave protocol [16].

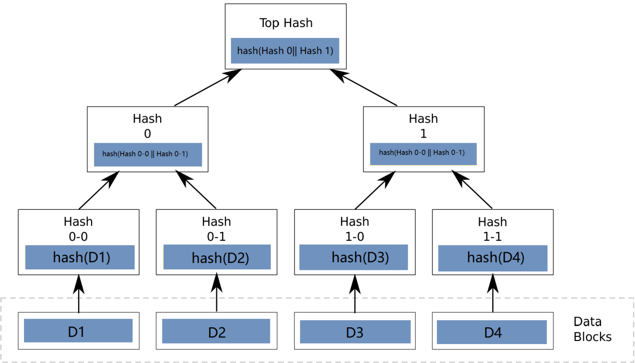


Fig. 3. Example Merkle tree with four leaf nodes

4.2 Proof of Integrity Verification of Search Results

Assuming there are n documents C, the keywords to be searched are represented by W. The file owner encrypts the keywords to trapdoor T_w and publishes trapdoor T_w and index collections I ($I = (C, BF)$). The searcher then searches n encrypted documents according to the trapdoor T_w. The searcher calculates $bf(y)$ ($y = e_1, e_2, ...e_r$) according to T_w and document identifier C_{id}, and the search results are obtained based on $bf(y)$ and BF. To ensure the integrity of search results (that is, the searchers have searched all documents rather than only some documents), and can be use on blockchains, we propose the scheme of the non-interactive integrity verification of search results.

For the search results for the keywords in each document, we obtain the search results for each document where $search_i = (bf(y) \wedge BF_i)$ (the specific calculation process is described in Sect. 5.2.). We then use each result $search_i$ ($i = 1, 2..., n$) as a leaf node to build a Merkle tree. We obtain the corresponding root node Root when we finish constructing the Merkle tree. Here, the searcher uses their public key to sign the Merkle tree root node and obtain sign $= sign_{sk_{search}}(Root)$. Then, we use the hash function to generate a random number to construct the proof of the result's integrity. We compute $H(sign||i)$ ($i = 1, 2, ..., l$) to generate l random numbers, where l denotes the security parameter. Assuming we have n documents that need to be searched, we create an array to save a proof of the integrity of the search results, which we refer to as the array result integrity proof (RIP). We then put the $H(sign||i)$ mod n ($i = 1, 2, ..., l$) search result that is $H(sign||i)$ mod n and its path in the Merkle tree into the array RIP. We also create a Result array to save the search results. When the searcher completes the search, he outputs Result, RIP, and sign(Root). The algorithm used to search the document and construct the RIP is given in Algorithm 1.

Algorithm 1. Search document and prove integrity of search results.

```
Search($T_w, I$): When a searcher sees a demand that he wants to help search the
document, then the searcher will use the trapdoor $T_w = (t_1, \ldots, t_r) \in \{0,1\}^{sr}$ for word
$w$ to test every index $I_{C_i}$ in indexed collections $I$.
for every $C_i$ in C do
    The key word $w$ is encoded according to $C_{id}$ to get $e$, and $e : (e_1 = f(C_{id}, t_1), \ldots, e_r = f(C_{id}, t_r)) \in \{0,1\}^{sr}$.
    To see if each position in $y$ that contains a 1 corresponds to a 1 in the bloom filter, we need to detect $(bf(e) \wedge BF) == bf(e)$ $(e = e_1, e_2, ..., e_r)$ is correct, and add $(bf(y) \wedge BF_i)$ to the array search that $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$.
    If so, add $C_i$ to the array Result.
end for
Use array search to build a Merkle tree and get a Merkle tree root Root.
Searcher use his public key to sign the Merkle tree root Root and get $sign = sign_{PK_{searcher}}(Root)$
Compute $H(sign||i)$ $(i = 1, 2, ..., l)$, the $l$ denote the security parameter, then add the $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ and the Merkle tree path into the array result integrity proof RIP.
Output Result, RIP and sign(Root).
```

When verifying the integrity of the search results, first we verify that the signature of the Merkle tree root. We then verify that the results in the array Result are correct. Here, according to the root of the Merkle tree, we compute $H(sign||i)$ $(i = 1, 2, ..., l)$ to generate l random numbers, based on the generated random numbers and the document to compute $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$. Then, we verify that the path which would be provided in the array RIP of the Merkle tree of $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ is correct. The algorithm used to verify results and the integrity of the results is given in Algorithm 2.

Algorithm 2. Verification of results and result integrity proof

Verify(searcher, Result, RIP, sign(Root)): This function is used to verify the result and result integrity.

```
1: Verify that the signature of the Merkle tree root $Root$ is signed by the searcher.
2: Verify that the results in the array $Result$ are correct, if correct then continues, else incorrect return error 0.
3: According to the root of the Merkle tree to compute $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$.
4: Verify that the path which be given in the array RIP of the Merkle tree of $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ is correct, if correct then continues, else incorrect return error 0.
5: According to the compute process given in the SearchIndex function, the result of the array RIP is correct, if correct return 1, else the error returns 0.
```

In the following, we discuss an example (Fig. 4) to illustrate the proposed scheme. Here, assume there are four documents. Thus, when the search is complete, we obtain four search results: $search_0$, $search_1$, $search_2$, $search_3$. For each search result, $search_0 = (bf(y) \wedge BF_0)$, $search_1 = (bf(y) \wedge BF_1)$, $search_2 = (bf(y) \wedge BF_2)$, $search_3 = (bf(y) \wedge BF_3)$. By using these four search results as a leaf node to build the Merkle tree, we obtain the corresponding root node. Assume Alice is searching for the result; thus, Alice signs the root with her private key. The signature and i are joined together. We set i to 1 and 2. Thus, we obtain two values: $(sign||1)$ and $(sign||2)$. After hash and modulus, we obtain two random numbers between 0 and 3. Here, assume that $H(sign||1)$ is equal to 0, and $H(sign||2))$ is equal to 2. We save $search_0$, $search_2$ and the corresponding Merkle tree path into to the RIP array. The $search_0$, $search_2$ and the corresponding Merkle tree path are used for subsequent result integrity verification. In addition, we assume that $search_1$ is the result of the search criteria. We save the $search_1$ in the array Result. If Bob is the verifier, Bob first verifies that root is Alice's signature. Then, Bob verifies that the results in the array Result are correct; thus, Bob verifies that $search_1$ is correct. Then, Bob according to compute $H(sign||1)\ mod\ 4$ and $H(sign||2)\ mod\ 4$ to generate two random numbers. Here, Bob obtains the two random numbers 0 and 2. Then, Bob verifies that $search_0$, $search_2$ and the corresponding Merkle tree path are correct, which completes the validation.

Assume the search is complete with p $(0 \leq p \leq 1)$, there are n documents that need to be searched, and $l(l \leq n)$ random numbers need to be generated. If the searcher does not fully search the entire document, only $p * n$ documents are searched. When validation of the Merkle tree is set up, the probability that the searcher build a Merkle tree will be able to verify at once is p^l. The larger the l option, the lower the probability that the searcher will be able to pass validation (when the searcher does not search the entire document), and the greater the cost of not fully searching the entire document and provide correct Result, RIP, and sign(Root). Here larger l results in more complete search results.

In the following, we examine an intuitive example; we see that a searcher searches 99% of the files, l is set to 100, and the probability that the searchers are successful in building the Merkle tree is 0.366. From the above mentioned findings, it is obvious that the proposed method is extremely effective in ensuring the integrity of the search results.

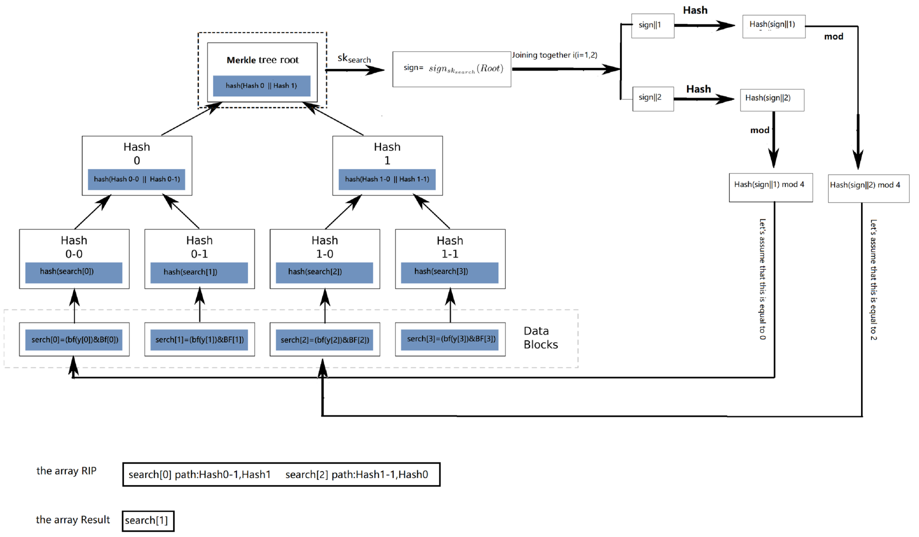

Hash
sign||1
Hash(sign||1)
mod
Merkle tree root
hash(Hash 0 || Hash 1)
sk_search
sign= $sign_{sk_{search}}(Root)$
Joining together i(i=1,2)
sign||2
Hash
Hash(sign||2)
mod
Hash 0
hash(Hash 0-0 || Hash 0-1)
Hash 1
hash(Hash 1-0 || Hash 1-1)
Hash(sign||1) mod 4
Hash(sign||2) mod 4
Hash 0-0
hash(search[0])
Hash 0-1
hash(search[1])
Hash 1-0
hash(search[2])
Hash 1-1
hash(search[3])
Let's assume that this is equal to 0
Let's assume that this is equal to 2
serch[0]=(bf(y[0])&BF[0])
serch[1]=(bf(y[1])&BF[1])
serch[2]=(bf(y[2])&BF[2])
serch[3]=(bf(y[3])&BF[3])
Data Blocks
the array RIP
search[0] path:Hash0-1,Hash1 search[2] path:Hash1-1,Hash0
the array Result
search[1]

Fig. 4. Example of the proposed search result integrity verification of search results (here, assume four files to be searched and l to 2)

5 Smart Contract Based Searchable Symmetric Encryption

In this section, we propose the smart contract-based SSE scheme. The proposed scheme includes search result integrity verification algorithm, an index generation algorithms, a ciphertext search algorithm, and a demand smart contract. First, we describe the overall architecture of the proposed scheme. We then explain the proposed scheme in greater detail.

5.1 Scheme of Smart Contract Based SSE

Here, we introduce the smart contract-based SSE scheme and provide a security proof. There are three roles in the proposed scheme: file owner, searcher, and miner. The file owner has n documents denoted $D = (D_1, D_2, ..., D_n)$. The searcher's task is to search the corresponding document based on the keywords and indexes provided by the file owner. The miner's task is to verify that the search results are correct and pack the transactions into the blockchain. The owner of the file first inputs the keywords to search and provides commission d. When searcher finds searching demand then he searches the file according to the keywords and index directory, and searcher will submit search results and results integrity prove to the miners; if the verification through, miner will packaged the results and integrity proof to block chain.

Our scheme has two phases. In **series phase 1**, the file owner creates a smart contract for search demand. In **phase 2**, for a smart contract, the searcher can submit the results, and the miner can verify the results. The specific process is shown in Fig. 5, and we introduce our scheme below.

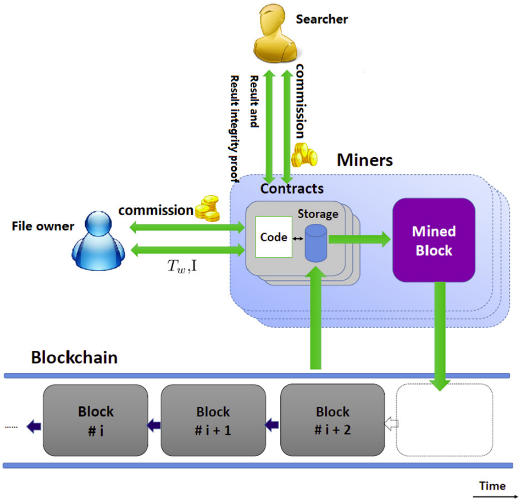


Fig. 5. Smart contract based SSE scheme

The proposed scheme involves two phases. In **phase 1**, the file owner creates a smart contract for the search demand. In **phase 2**, for a smart contract, the searcher can submit the results, and the miner can verify the results. The process is shown in Fig. 5.

In **phase 1**, the file owner wants to search keyword w, and he encrypts the keyword using the key $K_{trapdoor}$ to obtain the search trapdoor $T_w = (t_1 = f(w_i, k_1), \ldots, t_r = f(w_i, k_r))$ $(K_{trapdoor} = (k_1, k_2, ..., k_r))$, Then, the file owner

Algorithm 3. Demand smart contract

```
variable $T_w$, $I$, $d$, owner        \\trapdoor $T_w$ index $I = (C, BF)$, commissions $d$

function DEMAND($T_w$_input, $I$_input, $d$_input)
    $T_w$=$T_w$_input
    $I$ = $I$_input
    $d$ = $d$_input
    owner=msg.sender
end function

function VERIFY(searcher, Result, RIP, sign(Root))
    if sign(R) is signed by searcher then
        if Result are correct then
            if the path of elements in the RIP is correct then
                count=0
                for $i = 1; i <= l; i++$ do
                    if search[$H(sign||i)$] == $RIP[i]$ then
                        count++
                    end if
                end for
                if count==l then
                    return 1
                end if
            end if
        end if
    end if
    return
end function

function PAYCOMMISSIONS(searcher, Result, RIP, sign(Root))
    if Verify(searcher, Result, RIP, sign(R))==1 then
        Sent(owner, searcher, amount, Result)
        Selfdestruct()
    else
        return
    end if
end function

function DESTROY
    if msg.sender == owner then
        Selfdestruct()
    else
        return
    end if
end function
```

creates a smart contract that contains trapdoor T_w, index $I = (C, BF)$ and commissions d.

Algorithm 3 presents a pseudocode framework for a file owner to publish a trapdoor that needs to be looked up. Here, when creating a contract, the file owner inputs trapdoor T_w, index $I = (C, BF)$, the corresponding commission d, and the verification function.

In **phase 2**, when the searcher sees the search demand, he begins to search trapdoor T_w. When the searcher finishes searching the documents, he calls function $Verify(\cdot)$ of the demand smart contracts and uses the results, results integrity proof as parameters. If the verification is conducted through smart contract, the commissions will be sent from the file owner to the searcher, and the demand smart contract calls the function $Destroy(\cdot)$ to destroy itself. Otherwise, authentication failure is returned.

If the file owner does not want to search the document, he calls the function $Destroy(\cdot)$ to destroy the smart contract.

Search keywords can be single or multiple words. For simplicity, we only consider the single keyword case. The proposed smart contract-based SSE scheme is defined as follows:

Definition 1 *(Smart Contract based Searchable Symmetric Encryption). A smart contract based SSE is a set of six polynomial time algorithms SSE = (Gen, Enc, Build, Buildtrapdoor, Search, Demand smart contract):*

- $K \leftarrow Gen(t, s, r)$: This probabilistic algorithm takes security parameters $t,\ s,\ r$ and outputs array key K.
- $C \leftarrow Enc(K_{file}, D)$: This algorithms takes file key K_{file}, data file collection D, and keyword collection W as input, and the file owner outputs encrypted files C.
- $I \leftarrow Build(C, W)$: This algorithms takes encrypt file C and keywords W as input, and the file owner outputs index I.
- $T_w \leftarrow Buildtrapdoor(K_{trapdoor}, w)$: This is the keyword trapdoor generation algorithm. The file owner takes search keyword w, and trapdoor key $K_{trapdoor}$ as input, and obtains trapdoor T_w as the output.
- $(result, RIP) \leftarrow Search(T_w, I)$: The searcher searches the encrypted files according to the file owner's requirements in the smart contract (keyword trapdoor T_w and index I). The search results are returned to the array Result and result integrity proof array RIP.
- Demand smart contract: This is created by the file owner and placed in the blockchain. The file owner initializes the contract and calls the constructor function Demand($\cdot$) to initialize the parameter trapdoor T_w, index $I = (C, BF)$, and commissions d parameters. The searcher calls the function PayCommissions($\cdot$) to submit the result and the RIP. The miner uses the function Verify($\cdot$) to verify the result and the RIP. The file owner calls the function Destroy($\cdot$) to destroy the smart contract.

5.2 Details of Smart Contract Based SSE Scheme

In the proposed smart contract-based SSE scheme, there are three participants, i.e., the file owner, the searcher, and the miner. The file owner has n files denoted $D = (D_1, D_2, ..., D_n)$ that need to be uploaded to the network. Prior to uploading, the file owner selects the secure function. Here, $f : \{0,1\}^* \times \{0,1\}^s \rightarrow \{0,1\}^s$ and $H_1 : \{0,1\}^* \times \{0,1\}^t \rightarrow \{0,1\}^t$ are secure pseudorandom functions. Note that $H : \{0,1\}^* \rightarrow \{0,1\}^t$ is a collision-resistant hash function, and $c = t + t + sr$ denotes the security parameter. The main steps of the algorithm are described as follows.

- **Gen(t, s, r)**. The data owner takes security parameters t, s, r as input, and it outputs secret key array $K = (K_{file}, K_{trapdoor}, K_{MAC})$, where $(K_{file}, K_{trapdoor}, K_{MAC}) \leftarrow \{0,1\}^{t+t+sr}$, $K_{file} \leftarrow \{0,1\}^t, K_{trapdoor} \leftarrow \{0,1\}^t, K_{MAC} \leftarrow \{0,1\}^{sr}$. K_{file} is used to encrypt the data documents, and $K_{trapdoor}$ is used to generate the search trapdoor T_w for the keyword w. K_{MAC} is used to generate a MAC for C_i.
- **Enc(K$_{file}$, D)**: There are two steps in this function. First, the data owner uses key K_{file} to encrypt the documents collection $D = (D_1, D_2, ..., D_n)$: $C_i = Enc_{K_{file}}(D_i)(1 \leq i \leq n)$, $MAC_{C_i} = H_1(K_{MAC}, C_i)$ then set $C \leftarrow ((C_1, MAC_{C_1}), (C_2, MAC_{C_2}), \ldots, (C_n, MAC_{C_n}))$. Then, the data owner generates an index to optimize the search time complexity for future searches. First, the data owner extracts keywords collection $W = (w_1, w_2, \ldots, w_r)$ from each D_i and obtains a total of nr keywords for n documents.
- **Build(C,W)**. Before the n documents are encrypted, the indexes are built as follows. Here, the input is document C comprising a unique identifier $C_{id} \in \{0,1\}^n$ (each encrypted file C_i has a unique identifier $C_{i_{id}}$), and a list of keywords $(w_1, w_2, \ldots, w_m) \in \{0,1\}^*$ (assume each file has m keywords). Here $K_{trapdoor} = (k_1, \ldots, k_r) \in \{0,1\}^{sr}$.
  - I. The following is performed for each document C_i in C.
    1. For each different keyword w_j for $j \in [0, m]$, perform the following.
       - (a) Compute the trapdoor of each keyword: $(t_1 = f(w_j, k_1), \ldots, t_r = f(w_j, k_r)) \in \{0,1\}^{sr}$.
       - (b) The corresponding ciphertext ID C_{id} information is added to get the codeword $e : e_1, \ldots, e_r$, where $(e_1 = f(C_{id}, t_1), \ldots, e_r = f(C_{id}, t_r)) \in \{0,1\}^{sr}$.
       - (c) codeword $e : e_1, \ldots, e_r$ is added to the file D_i Bloom filter BF_i.
    2. The output secure index of C is $I_C = (C_{id}, BF)$.
  - II. Output $I = \{I_{C_1}, \ldots, I_{C_n}\}$
- **Buildtrapdoor($K_{trapdoor}$, w)**: Given trapdoor key $K_{trapdoor} = (k_1, ..., k_r) \in \{0,1\}^{sr}$ and keyword w, $T_w = (f(w, k_1), ..., f(w, k_r)) \in \{0,1\}^{sr}$ is output as the trapdoor.
- **Search(T_w, I)**. When a searcher observes a demand, he wants to help search the file. Then, the searcher uses trapdoor $T_w = (t_1, \ldots, t_r) \in \{0,1\}^{sr}$ for keyword w to test each index I_{C_i} in indexed collections I.

1. The following is performed for C_i in C.
   a. Keyword w is encoded according to C_{id} to obtain e, where $e : (e_1 = f(C_{id}, t_1), \ldots, e_r = f(C_{id}, t_r)) \in \{0, 1\}^{sr}$.
   b. To determine if each position in y that contains a 1 corresponds to a 1 in the Bloom filter, we must determine whether $(bf(e) \wedge BF) == bf(e)$ $(e = e_1, e_2, ..., e_r)$ is correct, and then add $(bf(y) \wedge BF_i)$ to the array search that $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$.
   c. If this is correct, C_i is added the array Result.
2. An array search is employed to build a Merkle tree and obtain the Merkle tree root Root.
3. The searcher uses their private key to sign the Merkle tree root Root and obtains $sign = sign_{PK_{searcher}}(Root)$.
4. Compute $H(sign||i)$ $(i = 1, 2, ..., l)$ and add $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ and the Merkle tree path to the array RIP, where l denotes the security parameter.
5. Output Result, RIP, and sign(Root).

- **Demand smart contract**. This contract is described in Algorithm 3. This contract contains four functions: Demand(·), Verify(·), PayCommissions(·), and Destroy(·), which are described as follows.
  - **Demand(T_w, $I = (C, BF)$, d)**. This function is used by the file owner to initialize the smart contract. The file owner initializes trapdoor T_w, index $I = (C, BF)$, commissions d and the initial contract owner.
  - **Verify(searcher, Result, RIP, sign(Root))**. This function is used to verify the result and RIP.
    1. Verify that the signature of the Merkle tree root *Root* is signed by the searcher.
    2. Check that the results in the array *Result* are correct. If the results are correct, continues; otherwise, return error 0.
    3. Compute $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ according to the root of the Merkle tree.
    4. Verify that the path given in array RIP of the Merkle tree of $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ is correct. If this is correct, then continues; otherwise, return error 0.
    5. According to the compute process given in the SearchIndex function, the result of the array RIP is correct. If this is correct, return 1; otherwise, return error 0.
  - **PayCommissions(searcher, Result, RIP, sign(Root))**. This function is used by the searcher to submit the lookup result and result integrity proof. If the result of the submission is verified, the smart contract transfers the commission from the file owner's account to the searcher's account. Then, the smart contract is destroyed. Here, if validation fails, a validation error is returned.
  - **Destroy()**. This function is used by the file owner to cancel the demand and destroy the smart contract. The function first determines whether the calling function is from the file owner. If so, the destroy operation is performed; otherwise, a call error is returned.

6 Security and Performance Analyses

In this section, we apply a simulation paradigm [4] to define the security of the proposed scheme. We also present a security analysis and security proof. Finally, we analyze the performance of the proposed scheme.

6.1 Security Analysis

The proposed scheme can guarantee the correctness of search data and the integrity of search results. We apply SSE based on the smart contract, and file owners can publish demands through the smart contracts. When a searcher observes the file owner's the file search demand through smart contracts, he starts to search. When the searcher finishes searching, he sends the results to the smart contracts for accuracy and integrity verification. If verification is successful, the searcher obtains the commissions. When the searcher completes the search, a Merkle tree is constructed for each file's search results, and the root of the Merkle tree is signed with the searcher's private key. Then, according to the root of the signature and i, to compute a hash, modulo the number of the files n, $H(sign||i)\ mod\ n$, and then give the search result of the corresponding $search_{H(sign||i)\ mod\ n}$ $(i = 1, 2, ..., l)$ and the path of the result in the Merkle tree. The purpose of the root signature is to ensure that when the result is submitted, the miner will not tamper with the searcher while verifying the result, and that the commission will be sent to the searcher. When the results are submitted, the searcher needs to give the results integrity proof, that are a partial search result random given by hash value and the corresponding Merkle tree path, so our scheme can avoid searcher deliberately omitting the search results to save computational cost or other reasons. The fairness of the proposed scheme is based on the fact that the blockchain cannot be tampered with. Search files on smart contracts are encrypted; thus, when only a ciphertext is retrieved, the searcher cannot learn anything about the plaintext. The search keywords in the smart contract are also encrypted; thus, the server cannot view any information about the plaintext and keywords other than the search results when executing the search algorithm. The information uploaded to the blockchain and smart contract only includes the encrypted keywords (trapdoor T_w), encrypted ciphertext, and Bloom filter; thus, no one (searchers, miners, etc.) can obtain any information related to the keywords and search plaintext through block data and the smart contract. The smart contract-based SSE follows the above two properties; thus, it provides secure SSE.

We present a theorem to prove that the proposed scheme is secure. Here, we used a real/ideal simulation model to prove security, which is fully described in the paper [4]. A number of SSE-related papers [5–7] have used this security model to prove security.

Definition 2 *(IND-CKA2 Security). If a smart contract based SSE protocol is secure, the adversary should not distinguish the real game* $Real^{\Pi}_{\mathcal{A}}(c)$ *and the simulation game* $Ideal^{\Pi}_{\mathcal{A},\mathcal{B},\mathcal{S}}(c)$*.*

Here, let Π be a smart contract based SSE scheme, where $\Pi = (Gen, Enc, Build, Buildtrapdoor,\ Search, Demand\ smart\ contract)$, and $c = t + t + sr$ is the security parameter. $\mathcal{A}$ represents the attacker, $\mathcal{B}$ represents the environment that can simulate the Ethereum system, $\mathcal{C}$ represents the challenger, and $\mathcal{S}$ represents the simulator.

$Real_A^{\Pi}(c)$: Challenger $\mathcal{S}$ can obtain key array K by performing operation $K \leftarrow Gen(j, s, r)$. After $\mathcal{S}$ obtains K, he will give the signature to be verified by the public key to Adversary $\mathcal{A}$. $\mathcal{A}$ gives challenger $\mathcal{C}$ a set of randomly selected files $D = (D_1, D_2, ..., D_n)$. Then, challenger $\mathcal{C}$ can obtain (I, C) by performing operation $Build(Enc(K, D), W)$. After challenger $\mathcal{C}$ obtains (I, C), they send (I, C) to $\mathcal{A}$. Challenger $\mathcal{C}$ returns (commissions, T_w) to attacker $\mathcal{A}$ to perform a polynomial query based on the different keyword w_i selected by the attacker $\mathcal{A}$. Then, attacker $\mathcal{A}$ queries Result and RIP by asking the $\mathcal{C}$. Finally, $\mathcal{A}$ produces a bit b that is returned by the experiment.

$Ideal_A^{\Pi}(c)$: Attacker $\mathcal{A}$ randomly selects D to satisfy condition $|D_{Ideal}| = |D_{Real}|$, where $D \leftarrow \{0, 1\}^*$. Here, simulator $\mathcal{S}$ obtains (I, C) by performing operation $\leftarrow S(L(D))$ ($L(D)$ is defined in [3]). When the simulator calculates (I, C), it sends (I, C) to attacker $\mathcal{A}$. The challenger $\mathcal{C}$ returns Result and RIP to attacker $\mathcal{A}$ to perform a polynomial query in the $\mathcal{B}$ environment based on the different keyword w_i selected by attacker $\mathcal{A}$. Finally, $\mathcal{A}$ produces a bit b that is returned by the experiment.

If we can find a $PPT\ \mathcal{S}$ for all $PPT\ \mathcal{A}$, that makes $\mathcal{D}$,
$|Pr[\mathcal{D}(view_{real}) = 1] - Pr[\mathcal{D}(view_{ideal}) = 1]| \leq negl(c)$ distinguishable for all polynomial sizes. Then, we can prove that Π is IND-CKA2 secure.

Theorem 1. *If the proposed SSE based smart contract scheme is an adaptive IND-CKA2 scheme, then the conditions to be met are H_1, f should be a pseudorandom function, H should be an anti-collision hash function, and $\varepsilon = (Enc, Dec)$ should be an IND-CPA [4] symmetric encryption scheme.*

Proof. We can build a PPT simulator $\mathcal{S} = \mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q$ for attackers, which allows $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_q$ to output two results $View_{real}$ and $View_{ideal}$. However these two results are indistinguishable during computation. Here, through the trace of a given history L, simulator $\mathcal{S}$ can be generated (I^*, C^*, t_w, Result, RIP) using the following methods.

- First, we discuss simulation I^*. Here, if $q = 0$, all files sizes are available to the simulator from L. $\mathcal{S}$ could set I^* to a random string of size $|I|$ by taking advantage of that information. $\mathcal{S}$ sets $C_i^* \leftarrow \{0, 1\}^{|D_i|}$ first, and then sets the state of I^* to $st_{\mathcal{S}}$. Note that state $st_{\mathcal{A}_0}$ has no key value for $K_{trapdoor}$; thus, $\mathcal{S}$ will do a uniform random selection of w for each keyword.
  If $q \geq 1$, attacker $st_{\mathcal{A}}$ first select $(st_{\mathcal{A}_0}, t^*_{w_0^*}) \leftarrow \{0, 1\}^*$ and $Mac_{w_0^*}{}^* \leftarrow \{0, 1\}^*$ randomly and evenly for each keyword w_0^*. Then, $\mathcal{S}$ select $w_i^* (q \geq i \geq 1)$ based on $(w^*_{i-1}, st_{\mathcal{A}_{i-1}})$. $\mathcal{S}$ then randomly selects $(st_{\mathcal{A}_i}, t^*_{w_i^*}) \leftarrow \{0, 1\}^*$ and $Mac_{w_i^*}{}^*$ for each keyword w_i^*.
  We know that I^* and I are indistinguishable because H_1 and $\{0, 1\}^* \times \{0, 1\}^t$ are indistinguishable. In the same manner, $\varepsilon = (Enc, Dec)$ is IND-CPA; thus,

we can also know that C^* is indistinguishable from the real ciphertext c. In the absence of key K_{MAC} in state st_{A_0}, there is no way for an attacker to distinguish between the ${MAC_{C_i}}^*$ and MAC_{C_i} generated in the $Enc(\cdot)$ step. This ${MAC_{C_i}}^*$ is selected randomly and uniformly from $\{0,1\}^t$.
- Now, we simulate T_w^*. If t_w^*, K_{tonke}^* and $t_w, K_{trapdoor}$ (generated by function $Build(\cdot)$) are indistinguishable, the conclusion is good. If $q = 0$, $\mathcal{S}$ will randomly and evenly select t_w^*, K_{tonke}^* from $\{0,1\}^k$. If $q \geq 1$, calculate $(t_{w_i^*}^*, K_{trapdoor}^*) \leftarrow \{0,1\}^*$ and select $w_i^*, q \geq i \geq 1$ based on $st_{\mathcal{A}_{i-1}}, w_{i-1}^*, i \geq 1$. Here, the pseudorandom functions H_1 and $\{0,1\}^* \times \{0,1\}^t$ are indistinguishable; thus, we can say that T_w^* and true T_w are indistinguishable.
- Simulate RIP^*. The pseudorandom function f is indistinguishable; thus, e and e^* generated in the search step are indistinguishable. The anti-collision hash function h is indistinguishable, the resulting RIP and the simulated RIP^* are indistinguishable.

Definition 3. *If the scheme is fair to both searchers and file owners, we say that the scheme satisfies fairness.*

Theorem 2. *If the blockchain is irreversible and the smart contract runs in the blockchain, the proposed scheme will satisfy fairness.*

- When the search task is generated, the commission is stored in the smart contract account. If the searcher fails to find the correct result, they will not obtain the commission. When the searcher finds the correct result, the file owner cannot refuse to pay the commission, and the smart contract will automatically transfer the commission to the searcher.
- The smart contract runs in the blockchain; thus, the file owner cannot change the task after publishing the search task, and the searcher cannot change the correct search result after receiving the commission.
- If both parties execute the agreement honestly, the user can obtain the correct results from the smart contract, and the searcher can obtain a commission.

Definition 4. *If we can verify whether the search results are complete and without omission, we say that the SSE scheme satisfies result integrity verification.*

Theorem 3. *If the smart contract can correctly execute our result integrity proof, then the scheme satisfies result integrity verifiable.*

Proof. After the searcher submits the results, the smart contract will implement the result integrity verification algorithm (Sect. 4.2). If the result is correct and there is no omission, the smart contract will pass verification. If the submitted result is correct but missing, it cannot pass the result integrity verification of the smart contract. □

6.2 Performance Analysis

Here, we analyze the efficiency and security of the proposed scheme and compare it to related schemes.

For users, search time determines the practicability and feasibility of the SSE scheme; thus, we primarily consider calculation and communication costs during the search stage. Table 2 shows how the proposed scheme performs in the search phase compared to related schemes. As shown in Table 2, the search time of the scheme proposed Kamara et al. [5] is the fastest because this scheme is designed to run parallel. However, this scheme does not satisfy fairness does not provide an integrity proof of the result. Thus, if the searcher returns incorrect results or if the searcher disappears after receiving a commission from the user, the user will lose both money and the search results. Therein lies the unfairness. To achieve fairness, another scheme [7] requires at least six transactions and three communications, which may prolong the time required for users to obtain results. The proposed scheme only requires two transactions and two rounds communication.

Table 2. Comparison of results of different schemes

Different scheme	Search time complexity	Communication complexity	Verifiable	Attacker	Fairness	Result integrity verifiable
Parallel [5]	O(mlog(n))	1	Yes	Server is honest-but-curious	No	No
Uc-secure [6]	O(n)	m	Yes	Server is malicious	No	No
BC [7]	O(n)	6	Yes	Server is malicious and user is malicious	Yes	No
Our scheme	O(n)	2	Yes	Server is malicious and user is malicious	Yes	Yes

6.3 Experimental Analysis

We develop an encrypted file search system. Here, we conduct relevant experiments on the number of different files and number of keywords, and we compare the time of completeness of the generated results and search time. The experimental data are shown in Table 3.

Table 3. Time required to search for encrypted files and obtain the result integrity verification

Number of documents	Number of keywords	Search time(s)	Result integrity verification(s)
2000	4	3.745	3.747
4000	4	3.912	3.902
6000	4	4.049	4.053
8000	4	4.282	4.286
10000	4	4.459	4.467
12000	4	4.576	4.571
2000	6	3.754	3.736
4000	6	3.909	3.913
6000	6	4.045	4.049
8000	6	4.283	4.287
10000	6	4.461	4.468
12000	6	4.567	4.573
2000	8	3.761	3.765
4000	8	3.908	3.917
6000	8	4.054	4.053
8000	8	4.291	4.292
10000	8	4.473	4.474
12000	8	4.577	4.581
2000	10	3.777	3.781
4000	10	3.931	3.928
6000	10	4.074	4.069
8000	10	4.296	4.293
10000	10	4.485	4.481
12000	10	4.596	4.594

From the experimental data, we conclude that, after adding the result integrity verification, the overall time is doubled, which is still within the acceptable range; thus, we consider that the proposed system provides sufficient availability.

We run our smart contract on the test network of Ethereum. Running our smart contract will burn 72000 gas per time on average, while the normal transfer on Ethereum test network will burn 20000 gas; therefore, our scheme is practical.

7 Conclusion

In this paper, we have proposed smart contract based SSE. Existing SSE protocols can resist malicious servers using MAC algorithms, but only if the server is actively running. If the server receives a user's money but does not provide service to the user or the server is closed after receiving the user's money, the user cannot withdraw the money. In addition, if the server wants to reduce computational costs and bandwidth, it will reduce the number of documents to be searched and omit a part of the search results; thus, there is no guarantee that all files will be searched. The proposed scheme solves this problem effectively using an integrity proof of search results and a smart contract. The proposed scheme guarantees the authenticity and integrity of the search results, and through the signature of the Merkle tree root that the searcher who searches the results will not be tampering by the miners after the submission of the results.

Acknowledgments. We would like to thank the anonymous reviewers. This work is supported by The National Key Research and Development Program of China (Grant No. 2018YFA0704701), National Natural Science Foundation of China (Grant No. 62072270), National Cryptography Development Fund (Grant No. MMJJ20170121), and Shandong Province Key Research and Development Project (Grant Nos. 2020ZLYS09 and 2019JZZY010133).

References

1. Androulaki, E., Karame, G., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating User Privacy in Bitcoin. IACR Cryptology ePrint Archive 2012:596 (2012)
2. Buchbinder, N., Petrank, E.: Lower and upper bounds on obtaining history independence. Inf. Comput. **204**(2), 291–337 (2006)
3. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
4. Goh, E.-J.: Secure Indexes. IACR Cryptology ePrint Archive 2003:216 (2003)
5. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 258–274. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_22
6. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_21
7. Li, H., Tian, H., Zhang, F., He, J.: Blockchain-based searchable symmetric encryption scheme. Comput. Electr. Eng. **73**, 32–45 (2019)
8. Li, H., Zhang, F., He, J., Tian, H.: A searchable symmetric encryption scheme using blockchain. arXiv preprint arXiv:1711.01030 (2017)
9. McIlroy, M.: Development of a spelling list. IEEE Trans. Commun. **30**(1), 91–99 (1982)
10. Meitinger, T.H.: Smart contracts. Informatik-Spektrum **40**(4), 371–375 (2017). https://doi.org/10.1007/s00287-017-1045-2
11. Mullin, J.K., Margoliash, D.J.: A tale of three spelling checkers. Softw. Pract. Exper. **20**(6), 625–630 (1990)

12. Naor, M., Teague, V.: Anti-persistence: history independent data structures. In: ACM Symposium on Theory of Computing, pp. 492–501 (2001)
13. Roy, S.S., Karmakar, A., Verbauwhede, I.: Ring-LWE: applications to cryptography and their efficient realization. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) SPACE 2016. LNCS, vol. 10076, pp. 323–331. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_18
14. Szabo, N.: Smart contracts: building blocks for digital markets. EXTROPY J. Transhumanist Thought **18**(16), 2 (1996)
15. Szabo, N.: Formalizing and securing relationships on public networks. First Monday **2**(9) (1997)
16. Wan, Z., Cai, M., Lin, X., Yang, J.: Blockchain federation for complex distributed applications. In: Joshi, J., Nepal, S., Zhang, Q., Zhang, L.-J. (eds.) ICBC 2019. LNCS, vol. 11521, pp. 112–125. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23404-1_8
17. Zhang, C., Fu, S., Ao, W.: A blockchain based searchable encryption scheme for multiple cloud storage. In: Vaidya, J., Zhang, X., Li, J. (eds.) CSS 2019. LNCS, vol. 11982, pp. 585–600. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-37337-5_48
18. Zhang, Y., Rajimwale, A., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: End-to-end data integrity for file systems: a ZFS case study. In: 8th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, 23–26 February 2010, pp. 29–42. USENIX (2010)

Towards the Adaptability of Traffic-Based IoT Security Management Systems to the Device Behavior Evolutions

Chenxin Duan[1,2], Jia Li[3], Dongqi Han[1,2], Linna Fan[1,2], Shize Zhang[1,2], Jiahai Yang[1,2](✉), and Zhiliang Wang[1,2]

[1] Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
{dcx19,handq19,fln19,zsz16}@mails.tsinghua.edu.cn

[2] Beijing National Research Center for Information Science and Technology, Beijing, China
{yang,wzl}@cernet.edu.cn

[3] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China
lijia@cert.org.cn

Abstract. Different kinds of Internet-of-Things (IoT) devices have been widely deployed in recent years, bringing great convenience as well as security threats. Given the grim situation of IoT security, various traffic-based security management systems specially designed for IoT systems have also been developed, such as device identification systems and anomaly detection systems. A lot of such systems are trained and evaluated on datasets collected only in short time periods and lack long-term evaluation. Intuitively, the communication behaviors and traffic profile of IoT devices may keep evolving due to factors like software or firmware update and the changes of user habits. It remains to be evaluated whether these IoT security management systems can adapt well to the device behavior evolutions, which matters a lot to the real-world performance. In this paper, we give a systematic discussion about the adaptability of IoT security management systems. We summarize the factors that may cause changes on the traffic profiles of IoT devices and how they can influence the long-term performance of IoT security management systems. We hope our work can serve as a base for further study on the building of adaptive systems for the security of IoT devices.

Keywords: Internet-of-Things · Security management · Adaptability · Device behavior evolutions

1 Introduction

The proliferation of different kinds of Internet-of-Things (IoT) devices has facilitated many aspects of people's daily life, such as smart home, smart city and

This work is supported by the National Key Research and Development Program of China (No. 2017YFB0803004).

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 203–209, 2021.
https://doi.org/10.1007/978-3-030-80851-8_15

industrial control. However, the deployment of these mini devices also poses new challenges to cyber security. For example, Mirai, a large-scale botnet which is mainly composed of compromised IoT devices, caused great damages by launching powerful distributed denial-of-service (DDoS) attacks [2]. It is also demonstrated that such IoT-based botnets are becoming more and more resilient and can even bring down the power grid [7,13]. Given the severe security threats brought by IoT devices, many systems to perform identification [6,9,10,12], monitor [5,8,14,16] and anomaly detection [11,15] to IoT devices by traffic modeling and analysis have been developed to help network managers ensure that the deployed IoT devices are functioning as expected. We refer to these systems as IoT security management systems. Considering IoT devices usually have only very simple functionalities and constrained communication and computing capacities, many IoT security management systems depend on the characterization of traffic generated by IoT devices and the construction of models that depict the normal traffic profiles of IoT devices. Such IoT security management systems are shown to achieve excellent performance and can realize the expected security management purposes.

However, for the evaluations of these IoT security management systems, both of the training and test processes are usually based on the traffic datasets collected in short time periods. Because it is impractical to collect traffic traces spanning long time periods for the training and test of these IoT security management systems in both laboratory and real-world environments. Nevertheless, many IoT devices interact with changing environments. The changes of weather or seasons will also influence the user activities and then change the way they use IoT devices. Moreover, technical issues like software or firmware updates and configurable properties may also make the traffic profiles of IoT devices change dramaticly. Thus, a problem of adaptability, whether existing IoT security management systems can keep their high performance in the long-term running, arises because the characteristics of traffic generated by the same IoT devices may keep evolving in the process of uses.

Adaptability is of great importance for practical IoT security management systems to be deployed in real-world scenarios, without which, the systems will be very fragile and the behavior evolutions of IoT devices can easily jeopardize their performance and then compromise the network management. However, it seems that previous works on IoT security have ignored the evaluation of the system adaptability to the device behavior evolutions and this property needs more attention in the future development of IoT security management systems. In this paper, we try to give a prelimilary but systematic investigation about the problem of adaptability by discussion on the following research questions (RQ):

- RQ1: How device behavior evolutions caused by different factors will affect the traffic profiles of IoT devices?
- RQ2: Can current IoT security management systems based on the traffic profiles adapt well to the device behavior evolutions?
- RQ3: What are the possible solutions to improve the adaptability of IoT security management systems or build self-adaptive systems?

We hope our work can serve as a base for further studies on the building of adaptive systems for the security of IoT devices. The remainder of this paper is organized as follows: Sect. 2 gives a review about the existing IoT security management systems based on the traffic features or profiles; Sect. 3 summarizes and categorizes different factors that may cause evolutions on the generated traffic of IoT devices; Sect. 4 discusses the possible solutions to build adaptive security management systems for IoT devices; Sect. 5 concludes the work.

2 IoT Security Management Systems

Existing IoT security management systems based on the traffic characteristics can be divided into 3 classes: device identification, event fingerprinting and intrusion detection. We will review these 3 classes of works and the traffic features they tend to use respectively.

Device Identification: A single IoT device has only very simple functionality and there are usually various different types of IoT devices in the environments equipped with IoT systems. Thus, knowing what IoT devices are connecting to the network is the first step to enable further management policies towards the IoT devices. Besides, device identification systems can also help with the discovery of unauthorized or vulnerable devices. Some works utilize identifiable fields in the traffic to recognize IoT devices, such as destination IP addresses, domain name queries and certificates [6]. Others leverage features in different resolutions, including packet level, flow level and session level, with machine learning models to classify traffic generated by different IoT devices [9,10,12]. Many device identification methods extract feature vectors in terms of traffic traces collected in short durations, ranging from minutes to hours. Both of the selected features and the durations of instances to be classified may influence the adaptability of the device identification methods. What's more, in the evaluations of these works, the training and test datasets are usually derived from traffic generated by the same devices in different periods, completely ignoring the impact of the deployment environment and user habits, which may matter much to the real-world scenarios, especially in long terms.

Event Fingerprinting: It is presented that the events happening on IoT devices that trigger the changes of their working status (trigger events) often correspond to some fixed traffic patterns, based on which, event fingerprinting systems can be developed to help network managers monitor the working status of deployed IoT devices [8,14]. With the persistent awareness of the working status of all the IoT devices, network managers can even monitor the semantic contexts of the IoT devices and detect context-relevant anomalies like event spoofing and device failure [5,16]. Unlike device identification, event fingerprinting systems tend to use very simple features, short sequences of packet lengths and directions, to detect the events happening on IoT devices by pattern matching [5,8,14,16]. Intuitively, although such simple signatures can be very precise, they are also very fragile at the same time, because the most minor update of the firmware or changes on distinct device configuration parameters (*e.g.*, credentials and device IDs) could destroy the signatures and it will take much effort to maintain the signatures if the changes are frequent.

Intrusion Detection: Intrusion detection systems (IDS) aim to detect all kinds of attack traffic sent by malicious attackers to compromise IoT devices. Considering the simple functionalities of IoT devices, anomaly-based IDSes, which build the profile of the normal traffic generated by IoT devices and regard any traffic deviated from the profile as intrusions, become popular in the networks serving IoT devices [11,15]. However, the traffic generated by some IoT devices may change dramaticly according to the user activities. For example, surveillance cameras are silent when users are at home and generate much large-volume traffic when the users leave their houses; the using frequency of air-conditioners can vary a lot in different seasons, which means that the communication behaviors of IoT devices can drift to deviate from the known normal profiles by themselves, not just intrusions. Without taking these factors into consideration, the long-term performance of IDSes for IoT devices may be quite questionable.

3 IoT Device Behavior Evolutions

In this section, we summarize the factors that may incur evolutions in the traffic generated by IoT devices and their possible implications on the IoT security management systems. Technical issues are the most common reasons that cause IoT devices to behave differently. Regular software or firmware updates are the most common issues that change the communication patterns of IoT devices. The IP addresses of cloud servers and domain names queried by the devices, coming from content delivery networks (CDN), can change frequently with the updates and then compromise device identification systems based on these fields. In addition, many IoT devices support multiple users, configurable credentials or parameters and user-defined trigger-action rules, which can all be reflected in the traffic characteristics. And the impact of these changes on systems based on only simple traffic features, like exact packet lengths, will be catastrophic.

Besides technical issues, the ways in which users interact with the IoT devices will also often bring evolutions to device behaviors. The functionalities of most IoT devices are closely tied with people's living and producing activities, which naturally change with the seasons. The evolutions caused by human activities, such as daily routines, lifestyles and producing plans, are mainly reflected in the using frequencies and durations of different IoT devices, which can reshape many spatial-temporal characteristics of traffic generated by the devices. For intrusion or anomaly detection systems, the changes of user habits must be taken into consideration, otherwise, by regarding traffic traces collected only in some periods as the whole normal profiles, a storm of false alerts will be generated once immense changes of user habits take place.

Having said the behavior evolutions of IoT devices, there may also exist some stable traffic features, like protocols and ports used by the devices, because the hardware and core functionalities of IoT devices can hardly change after coming into uses. However, intuitively, these relatively stable features are not distinguishable enough to achieve the goals of IoT security management. Thus, the adaptability of IoT security management systems to the device behavior evolutions should get more attention. Both of the empirical evaluations and the enhancement of the adaptability of IoT security management systems are in highly demand in the future works.

4 Possible Solutions

In this section, we briefly discuss the possible solutions to cope with the IoT device behavior evolutions. A prelimilary method is to enrich the training datasets so that the datasets can cover all the possible traffic patterns that IoT devices may exhibit as much as possible. However, on one hand, this increases the overhead to prepare the data and makes the performance of systems sensitive to the training datasets. On the other hand, this method may only apply to the evolutions caused by user activities and the changes incurred by device software or firmware updates are usually unpredictable and remain unsolved. What's more, for anomaly detection systems, current user habits are also important factors to determine whether the ongoing device behavior is abnormal. It may hinder the system performance to simply add the training data without proper contexts.

Another possible solution is to enable lifelong learning and detection for the systems, which is a hot topic in current anomaly detection community [3]. Lifelong learning means that when the system makes some mistakes, it can be trained increamentally by directly learning from the administrators' feedback to avoid making the same mistakes again. This method is a kind of remedy afterwards with lots of human intervention and cannot eliminate the performance loss caused by evolutions proactively. Additionally, it is also challenging to make the system get aware of the different contexts between the newly coming feedback and the previously learned model that should perhaps be forgotten sometimes.

Machine learning methods are widely used in existing IoT security management systems. Nevertheless, similar problems, called *concept drift*, have already been investigated in machine learning community [1,4]. Concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. While in IoT scenarios, many security management systems are trying to fit some mapping relationships with the traffic features, which also keep changing over time due to the device behavior evolutions. Therefore, adaptive learning methods that update predictive models online during their operation to react to concept drifts, may provide inspirations to deal with the IoT device behavior evolutions. However, a lot of future work is still needed to dive deep into the regularities of IoT device behavior evolutions and combine general adaptive learning algorithms with the IoT security management objectives.

5 Conclusion

In this paper, we focus on the adaptability of IoT security management systems to device behavior evolutions. We give systematic summary and analyses on the existing IoT security management systems, different kinds of evolutions happening on IoT devices and the possible solutions to build adaptive systems for the security of IoT devices. We find that the adaptability of IoT security management systems did not get enough attention despite its great importance and we hope our work can serve as the base of further study on adaptive IoT security management systems.

References

1. CADE: Detecting and explaining concept drift samples for security applications. In: 30th USENIX Security Symposium (USENIX Security 2021). USENIX Association, Vancouver, B.C. (2021). https://www.usenix.org/conference/usenixsecurity21/presentation/yang
2. Antonakakis, M., et al.: Understanding the mirai botnet. In: 26th USENIX security symposium (USENIX Security 2017), pp. 1093–1110 (2017)
3. Du, M., Chen, Z., Liu, C., Oak, R., Song, D.: Lifelong anomaly detection through unlearning. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1283–1297 (2019)
4. Gama, J.A., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (2014). https://doi.org/10.1145/2523813
5. Gu, T., Fang, Z., Abhishek, A., Fu, H., Hu, P., Mohapatra, P.: IoTGaze: IoT security enforcement via wireless context analysis. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 884–893. IEEE (2020)
6. Guo, H., Heidemann, J.: Detecting IoT devices in the internet. IEEE/ACM Trans. Netw. **28**(5), 2323–2336 (2020)
7. Herwig, S., Harvey, K., Hughey, G., Roberts, R., Levin, D.: Measurement and analysis of Hajime, a peer-to-peer IoT botnet. In: NDSS (2019)
8. Junges, P., François, J., Festor, O.: Passive inference of user actions through IoT gateway encrypted traffic analysis. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 7–12 (2019)
9. Ma, X., Qu, J., Li, J., Lui, J.C., Li, Z., Guan, X.: Pinpointing hidden IoT devices via spatial-temporal traffic fingerprinting. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 894–903. IEEE (2020)
10. Marchal, S., Miettinen, M., Nguyen, T.D., Sadeghi, A.R., Asokan, N.: AuDI: toward autonomous IoT device-type identification using periodic communication. IEEE J. Sel. Areas Commun. **37**(6), 1402–1412 (2019)
11. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. In: Network and Distributed System Security Symposium, NDSS (2018)
12. Sivanathan, A., et al.: Classifying IoT devices in smart environments using network traffic characteristics. IEEE Trans. Mob. Comput. **18**(8), 1745–1759 (2018)
13. Soltan, S., Mittal, P., Poor, H.V.: BlackIoT: IoT botnet of high wattage devices can disrupt the power grid. In: 27th USENIX Security Symposium (USENIX Security 2018), pp. 15–32 (2018)

14. Trimananda, R., Varmarken, J., Markopoulou, A., Demsky, B.: Packet-level signatures for smart home devices. In: Network and Distributed System Security Symposium, NDSS (2020)
15. Wan, Y., Xu, K., Xue, G., Wang, F.: IoTArgos: a multi-layer security monitoring system for internet-of-things in smart homes. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 874–883. IEEE (2020)
16. Zhang, W., Meng, Y., Liu, Y., Zhang, X., Zhang, Y., Zhu, H.: HoMonit: monitoring smart home apps from encrypted traffic. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1074–1088 (2018)

Correction to: A Novel Approach for Code Smells Detection Based on Deep Learning

Tao Lin, Xue Fu, Fu Chen, and Luqun Li

Correction to:
Chapter "A Novel Approach for Code Smells Detection Based on Deep Learning" in: B. Chen and X. Huang (Eds.): *Applied Cryptography in Computer and Communications*, LNICST 386, https://doi.org/10.1007/978-3-030-80851-8_12

In the original version of this book, a chapter title appeared with a typo. This has now been corrected.

The updated version of this chapter can be found at
https://doi.org/10.1007/978-3-030-80851-8_12

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, p. C1, 2022.
https://doi.org/10.1007/978-3-030-80851-8_16

Author Index