PRIVACY-PRESERVING DATA COMPUTING AND ANONYMOUS AUTHENTICATION PROTOCOLS

YANG XIAOYI

PhD

The Hong Kong Polytechnic University

2023

The Hong Kong Polytechnic University

Department of Computing

Privacy-Preserving Data Computing and Anonymous Authentication Protocols

Yang Xiaoyi

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

August 2022

CERTIFICATE OF ORIGINALITY

Abstract

Privacy-preserving technology has been actively studied lately since it is crucial for data security. Privacy-preserving data computing, for instance, enables data to be analyzed and at the same time protected from disclosure. Anonymous authentication mechanism can effectively ensure reliability and integrity. However, existing work suffers from problems such as low efficiency and poor performance in specific application scenarios. This thesis mainly studies privacy-preserving data computing and anonymous authentication protocols. Specifically, this thesis focus on the investigations of three important mechanisms of privacy-preserving technology, namely, private set intersection cardinality (PSI-CA), federated learning with secure aggregation and anonymous reputation system.

The contributions of this thesis are summarised as follows.

We propose a lightweight delegated PSI-CA protocol based on multi-point oblivious pseudorandom function and collision-resistant hash function. In addition, we develop PC-CONTrace, a privacy-preserving contact tracing system by utilizing this protocol. We evaluate the efficiency of the system under different set sizes and compare it with related schemes from the aspects of functionality and performance.

We propose an accountable and verifiable aggregation protocol for federated learning. We employ homomorphic proxy re-authenticators and homomorphic proxy re-encryption to execute secure aggregation, while integrating the blockchain to realize the function of penalty for malicious behavior. To demonstrate the useability of the protocol, we evaluate the specific cryptography schemes and develop a blockchain-based prototype system to test the performance of the protocol.

We propose an anonymous and publicly linkable reputation system with distributed trust (DTrustRS). We define the system model of DTrustRS, formalize its security and give a concrete construction. We adopt the re-randomizable signatures paradigm to prove the security of DTrustRS in the random oracle model under a q-type assumption. We evaluate DTrustRS and compare it with related works to demonstrate its validity.

Publications

The following papers are based on the contents of this thesis.

- **Xiaoyi Yang**,Yanqi Zhao, Qian Chen, Yong Yu, Xiaojiang Du, Mohsen Guizani. Towards Accountable and Verifiable Secure Aggregation for Federated Learning. IEEE Network. 36(5): 173-179 (2022).
- **Xiaoyi Yang**, Yanqi Zhao, Sufang Zhou, Lianhai Wang, Meijuan Huang. A lightweight Delegated Private Set Intersection Cardinality Protocol. Computer Standards & Interfaces 87(2024): 103760.

The following papers are my co-authored works, they are beyond the scope of this thesis.

- Yanqi Zhao, **Xiaoyi Yang**, Qi Feng, Yong Yu. SM2 Digital Signature based Anonymous Credential Protocol. Journal of Software (in chinese). online available: http://www.jos.org.cn/jos/article/abstract/mk001
- Yanqi Zhao, **Xiaoyi Yang**, Yong Yu, Baodong Qin, Xiaojiang Du, Mohsen Guizani. Blockchain-Based Auditable Privacy-Preserving Data Classification for Internet of Things. IEEE Internet of Things Journal. 9(4): 2468-2484 (2022).

- Meijuan Huang, Bo Yang, Yi Zhao, Kaitai Liang, Liang Xue, **Xiaoyi Yang**. CCA-Secure Deterministic Identity-Based Encryption Scheme. The Journal of Universal Computer Science. 25(3): 245-269 (2019).

Acknowledgements

I would like to express my most sincere gratitude to all the teachers, teammates, family members and friends who have given me support, care and help during my PhD journey.

First of all, I would like to appreciate Prof. Xiapu Daniel LUO, Prof. Man Ho Allen Au and Dr. Liu Yan Dennis Wang for the opportunity, supports and tolerance they provided me. I would especially like to thank Prof. Man Ho Allen Au, whose profound knowledge, rigorous and responsible attitude deeply influenced me.

I am very grateful to the joint research team of the Hong Kong Polytechnic University and the Hong Kong University, including Yilei Wang and Xiao Yang for their precious friendship, companionship and help, as well as Jiazhuo Lu, Borui Gong, Haiyang Xue, Dongqing Xu, Kang Li, Xinyu Li, Mengling Liu, Jingjing Fan, Xingye Lu, Rupeng Yang, Zuoxia Yu and others for their help.

In addition, I would like to express my sincere thanks to Prof. Yong Yu at the Shaanxi Normal University for his valuable support and help. I also would like to take this opportunity to particularly thank all the co-authors of all my research papers, especially Prof. Xiaojiang Du and Dr. Sufang Zhou.

Finally, I would like to thank all my family members, thanks my parents for their selfless love and warmth. My parents always support me unconditionally, understand me and care about me. I would like to thank my parents for their silent efforts, giving me inexhaustible motivation and making me face life positively and optimistically. Special thanks to my boyfriend Yanqi Zhao for his priceless support and encouragement. We learn and grow together, you set my mind at rest and make me feel hopeful about life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the development of information technology, people are generating data all the time in daily life. Examples of these data include browsing records and social data generated on the Internet, user data collected by wearable devices, and user behavior data acquired by smart home applications. Human beings have already entered the era of big data. The progress of science and technology not only provides more abundant means for the generation of data, but also provides more convenient tools for the recording and preservation of data. According to the research results of International Data Corporation (IDC) [81], the total amount of data generated globally in 2019 is 45ZB, which is expected to grow to 175ZB by 2025. According to relevant research of IBM, data acquired in the past few years accounts for 90% of all data acquired by the entire human civilization, and more than 500 million pictures are uploaded every day around the world, and more than 20 hours of videos are shared every minute.

Facing the explosive growth of data, the use of data brings convenience to human beings, but data security incidents also occur from time to time, including data loss, leakage, counterfeiting and other security issues. For example, in 2015, about 100G of commercial data in the server of Google data center was permanently removed due to power supply interruption due to weather reasons. In 2018, Facebook leaked the personal information data of more than 50 million users to third-party companies, which make huge profits and even affect the U.S. election [78]. In 2019, the American medical collection organization (AMCA) was attacked by hackers, leading to the disclosure of medical data of about 20 million American citizens, including user names, social security numbers, addresses, birth dates and other information [59]. In February 2020, cosmetics company Estee Lauder exposed a database without protection measures on the Internet, resulting in the leakage of sensitive information of 440 million users, including a large number of audit logs and email addresses [38]. In march 2020, Sina, one of the four major portals in China, was hacked. The personal information of more than 538 million users was put up for public sale on the Deep Web and other online sites [85]. In July 2022, the personal data of 5.4 million Twitter users was leaked. Hackers stole the data from Twitter and put it up for sale, asking for more than 30,000 dollars [84]. It can be seen form above that the actual owner of the data does not have true ownership of the data, and the data is vulnerable to falsification, tampering and leaving no trace, data security faces serious challenges. These data are related to transportation, logistics, medical care, social networking, banking and so on, which are closely related to personal privacy, national security and even human development. Therefore, it is essential to protect the security of data. The research of privacy-preserving technology has attracted numerous attention since it can provide technical support for data security.

Privacy-preserving data computing, also known as "privacy computing", enables data to be analyzed and computed while being protected from disclosure, meaning that it is possible to achieve the purpose of "available and invisible" data and realize the transformation and release of data value on the premise of fully protecting data privacy [58]. The numerous outstanding advantages of privacy computing make it widely used in various fields, such as finance, government affairs, medical care and operators. According to relevant surveys, taking China as an example, the market size of privacy computing in China is 490 million CNY in 2021 and is expected to reach 14.51 billion CNY in 2025 [1]. Privacy computing is a complex technology involving various disciplines, including hardware, cryptography, distributed machine learning and other underlying technologies. There are two main approaches to realise privacy computing: one is hardware-based Trusted Execution Environment (TEE) [79], the other is cryptography and distributed systems. The core idea of the TEE is to build a secure hardware region in which data from all parties can be collected for processing. This method provides a secure computing environment, making it easier to adopt, but it also requires the hardware provider to be trusted. Cryptography refers to using algorithms to protect data in the process of processing, which mainly realizes privacy computing from the level of algorithms. This method can theoretically guarantee the availability and invisibility of data, and does not require a trusted third party and has attracted considerable interest lately. In addition, under the background of big data application of artificial intelligence, federated learning (FL) [19], which is quite popular in recent years, is also the main method to promote and apply in the field of privacy computing.

Protecting data security requires not only protecting privacy of data but also protecting the integrity and reliability of data. Anonymous authentication [56] mechanism can effectively ensure the reliability and integrity of data while protecting the personal identity privacy, making it an important tool in privacy computing. Current researches usually use the pseudonym system, anonymous credentials and privacy-preserving signature technologies to realize anonymous authentication. Pseudonym systems allow users to interact anonymously with multiple organizations. Creating a pseudonym usually requires significant computational and storage overhead during the pseudonym generation phase. Anonymous credentials [83] enable users to prove to service providers that their identity credentials belong to a specific set of users according to the requirements of the application services, during which service providers cannot identify the specific identity of users. Most anonymous credential mechanisms require a centralized organization to issue certificates. Privacy-preserving signature [73] techniques mainly include group signatures, ring signatures and etc, where signers achieve identity anonymity by signing on behalf of group members or ring members. Privacy-preserving signatures provide a stronger guarantee for the integrity of data, which makes them widely used and studied in the anonymous authentication research.

1.1 Research Contents

This thesis mainly studies the privacy-preserving data computing and anonymous authentication protocols and their applications. Specifically, the research content can be summarized as follows.

- In order to improve the communication efficiency of privacy-preserving contact tracing system and reduce the computation overhead of the user, we propose a lightweight delegated private set intersection cardinality protocol based on multi-point OPRF and collision-resistant hash function and then we build a privacy-preserving contact tracing system by utilizing our protocol, that is PC-CONTrace. We test the efficiency of the system under different set sizes and compare it with related scheme from the aspects of functionality and performance.

- To defend against malicious clients and guarantee the confidentiality and verifiability of secure aggregation, we propose an accountable and verifiable secure aggregation for federated learning framework in IoT networks. We employ a secure multiparty computation protocol to protect the confidentiality and verifiability of data, which is based on homomorphic proxy re-authenticators and homomorphic proxy re-encryption.

- To reduce the reliance on one trusted authority in anonymous reputation systems while executing distributed regulation by revoking the anonymity of misbehavior users, we propose an anonymous and publicly linkable reputation system with distributed trust (DTrustRS). To achieve the registration function of DTrustRS, we propose a new variant of the Pointcheval-Sanders (PS) signature named aggregatable PS signature (APS). We define the system model of DTrustRS, formalize its security and give a concrete construction.

1.2 Organization

The rest of this thesis is organized as follows.

Chapter 2, Literature Review. This chapter briefly reviews the related literatures, and briefly expounds the research status from four aspects: private set intersection, delegated private set intersection, federated learning with secure aggregation and reputation system.

Chapter 3, Preliminaries. This chapter briefly introduces some basic mathematical knowledge of cryptography, introduces the cryptographic primitives used in subsequent chapters, and summarizes the blockchain infrastructure and data structure.

Chapter 4-6. These three chapters introduce and describe the above three research **specifically**, including specific protocols, systems and corresponding security and performance analysis.

Chapter 7, Conclusion. This chapter concludes this thesis and discusses the future works.

Chapter 2

Literature Review

In this chapter, we review the literatures related to the specific research content of this thesis. We review the related work of PSI and its variants delegated PSI in Section 2.1. In Section 2.2, the research work of federated learning with secure aggregation is reviewed, and we focus on blockchain-based related works. Finally, we review the works related to reputation system in Section 2.3.

2.1 Private Set Intersection

The protocol of PSI was originally proposed based on Diffie-Hellman difficulty hypothesis [62]. This method has high communication efficiency. There are still some protocols constructed based on this hypothesis [50]. However, this kind of protocol requires both parties to use their own keys to compute the exponent of each input set element, so the computation efficiency is low. Communication efficiency and computation efficiency are two major aspects that affect the efficiency of PSI protocol, so that the deficiency of either party will limit the practicability of the protocol.

In order to reduce the computational complexity of PSI protocol, various approaches have been explored. Freedman et al. [44] convert the set intersection into polynomial evaluation by converting the set into polynomial. They realize PSI by realizing the oblivious polynomial evaluation with the help of the additive homomorphic encryption algorithm. Their scheme is the first homomorphic encryption technology based PSI protocol. In addition, some scholars use other cryptographic methods to construct some PSI protocols and extension protocols. De Cristofaro et al. [33] use RSA blind signature to design the PSI protocol, which can also hide the cardinality of the participant set. Chen et al. [30] use fully homomorphic encryption algorithm and its acceleration technology combined with hashing technology to construct the PSI protocol under non-equilibrium environment. However, these protocols are all based on public-key cryptography and have high complexity. Garbled circuit as a common tool for secure multiparty computation is also used in PSI research. Huang et al. [49] proposed the PSI protocol with the help of Yao's circuit, they express the private set intersection function as a circuit and use sorting to reduce the complexity of the general circuit in computing set intersection, but the efficiency was relatively high at that time. Pinkas et al. [68] improved the efficiency of [49] by using permutation-based hashing, which is about 5 times faster than Huang's. After that, Pinkas et al. [70, 71] continued to improve the computational and communication complexity of circuit-based PSI protocols via cuckoo hash and oblivious programmable pseudorandom function (OPPRF), respectively. However, PSI protocols based on garbled circuit usually contain complex circuits with a large number of gates and large circuit depths, so their communication complexity is relatively high.

Dong et al. [39] proposed a PSI protocol applicable to big data by means of the oblivious transfer extension primitive, which has attracted the attention of academia and industry. With the introduction of oblivious transfer and oblivious transfer extension, the PSI protocol can not only meet the security requirements of MPC, protect the privacy of participants, but also achieve a balance between computing cost and communication cost compared with public keys and obfuscating circuits, which makes it practical. Subsequent works from the academia [68, 55, 69, 67, 28] and companies such as Google [75], Facebook [22] and Apple [5] have also successively proposed solutions for this problem that are applicable to different environments. Among the existing protocols, the one with the best computational efficiency is the one given by Kolesnikov et al. [55], while the work from Pinkas et al. [67] has the best communication efficiency. However, in the process of using the protocol in reality, people pay more attention to the cost of implementing the protocol. Therefore, the concept of monetary cost is proposed [67], and the lowest monetary cost PSI protocol is implemented in the case of low speed network bandwidth. The protocol constructed in [28] achieves monetary cost optimization in medium bandwidth environment.

2.1.1 Delegated Private Set Intersection

In order to improve the computing efficiency, it is typical to delegate the computation tasks of the participants to the cloud server, then reducing the computation complexity of the local participants and thus reducing the equipment requirements of the local participants. Delegating PSI computation to the cloud is an important way to reduce the computational complexity of participants.

Kerschbaum [54] first proposed the delegated PSI protocol, which represents the set as a Bloom filter and designs a new homomorphic encryption scheme to realize the delegated computation. Since then, in order to optimize the communication cost of the protocol and equip the protocol with additional desirable features, such as verifiability [2], many schemes have been proposed. Qiu et al. [76] express the set as the root of a polynomial and encrypt the coefficients of the polynomial with the homomorphic encryption algorithm, then entrust the cloud to do the intersection of the private set. Abadi et al. [3] also express the set as the root of a polynomial, but they used the point-value form of the polynomial and homomorphic encryption algorithm to delegate the computation to the cloud. By means of hash table, polynomial point value and pre-set secure channel, they [3] also realize the delegated PSI without encryption. When it comes to PSI-CA, Duong et al. [40] use OPRF to realize the first delegated PSI-CA. This method is mainly realized by constructing a new primitive, distributed key OPRF (Odk-PRF). Odk-PRF is an extension of OPRF, whose PRF keys, inputs, and outputs are shared by multiple participants. The main idea of their work is that the receiver shares its input set to multiple cloud servers, and these cloud servers take the place of the receiver to interactively compute Odk-PRF with the sender, each cloud server gets the Odk-PRF value of its own shared, the selected combiner computes the final OPRF value, and the sender gets the key of the OPRF. The sender calculates the OPRF value of the elements in its own set, packages the values and sends them to the combiner in the cloud server, and then the combiner sends the calculated results to the receiver, so that the receiver can obtain the final result.

2.2 Federated Learning with Secure Aggregation

Google presented federated learning with secure aggregation in 2017 [19]. To execute the secure aggregation, Bonawitz et al. [19] proposed a double-masking structure, which uses Diffie-Hellman key agreement to generate the seed of a pseudorandom generator (PRG), then using the PRG to protect the update parameters. In addition, they make use of t-out-of-n secret sharing to handle the users dropouts problem. However, this scheme relies on a semi-honest assumption for the client and the aggregation part requires at least 4 communication rounds. To improve the communication and computation efficiency of the aggregation, Xu et al. [89] uses the functional encryption and Truex et al. [87] uses threshold homomorphic encryption to realize secure aggregation that resists the dropouts of participants, but they both rely on a trusted party who holds the master keys. So et al. [82] presented Turbo-Aggregate, they use multi-group circular strategy to improve the efficiency of the model aggregation, the overhead of their secure aggregation is $O(NlogN)$, where N is the number of users. Kadhe et al. [52] proposed Fast-SecAgg based on the fast fourier transform multi-secret sharing scheme further improved the efficiency of communication, which is a scalable secure aggregation scheme for privacy-preserving federated learning. However, in order to improve efficiency, their schemes relaxes the constraint of dropouts from supporting any subset of clients of a bounded size to supporting only a random subset.

As a decentralized infrastructure, blockchain has been applied to FL frameworks for privacy protection. For example, Zhang et al. [91] proposed a blockchain-based participant selection protocol for FL. Their scheme executes numerical evaluation to delete the malicious devices and relies on semi-honest server to run the participant-selection algorithm to select the group of devices. For secure aggregation, Zhang et al. [92] presented a secure aggregation scheme for FL, making use of the consensus algorithm of blockchain, in which the client also need to be semi-honest. However, the semi-honest assumption is vulnerable to malicious clients. To address malicious situations, Awan et al. [8] proposed a reliable and accountable privacy-preserving federated learning framework based on blockchain, which involves two non-colluding parties: aggregator and server. They use homomorphic encryption and proxy re-encryption to protect the privacy of the model updates, which is secure against the random dropouts of the malicious client. The blockchain is used to record the user's transactions, so that the malicious client can be traced in their scheme.

2.3 Anonymous Reputation System

Reputation systems have been studied extensively [31], [34], [61]. Anonymity and unforgerability, as the key attributes of reputation system, guarantee the privacy and security of reputation system and are the most studied in the reputation system [4, 16, 57]. In existing works, the most adopted methodologies are differential privacy, electronic cash and cryptographic signatures. Among them, the works based on differential privacy are usually limited to special reputation functions [31], the electronic cash based schemes cannot support fine-grained ratings [4]. Cryptographic signatures, including group signature, ring signature and blind signature, are considered to be more suitable for constructing anonymous reputation systems [88, 90]. To provide trustworthy and honest ratings, the anonymous reputation system with public linkability has been studied recently, also named anonymous and publicly linkable reputation system, which allows users to anonymously rate services and can detect whether the users are deviating from the once-only policy in the rating service [16]. The anonymous and publicly linkable reputation system needs to satisfy anonymity, traceability, and public linkability. In order to satisfy these security requirements simultaneously, most of the existing works are constructed based on group signature [29], [41], so that the research of group signature affects the study of anonymous and publicly linkable reputation system to some extent. The first group signature with anonymity and traceability was proposed by Bellare et al. in [11], which are generalized by a sign-encrypt-prove paradigm. They also proposed a formalized definition for static group signature in this work. Later, Bellare et al. [12] formalized the security properties of dynamic group signature in the standard model, which supports user to join the group dynamically. Bichsel et al. [13] proposed the re-randomizable signatures paradigm for a group signature without encryption. However, in these schemes, the tracing operation is done by a trusted tracer (or opener), which leads to over-reliance on the tracer and can easily lead to a single point of failure problem. To reduce the trust in one single tracer, Blömer et al. [17] and Ghadafi [47] considered group signatures with threshold traceability. Zheng et al. [93] proposed democratic group signatures with threshold traceability, where the group signature scheme can support distributed issuance and threshold opening, but it is poor in anonymity guarantees. Recently, Camenisch et al. [25] proposed threshold dynamic group signatures by utilizing encrypt-and-prove paradigm, which supports threshold issuance and threshold opening. For linkability, Garms et al. [46] proposed group signatures with selectively linkability, where a converter can blindly convert the same user's signatures. Based on the group signatures, Blömer et al. [16] proposed an anonymous and publicly linkable reputation system, and it satisfies the properties of anonymity, traceability, and public linkability. Based on the same system model, Blömer et al. gave two schemes successively, one is universally composable secure [15], the other is instantiated with lattice [14].

Chapter 3

Preliminaries

This chapter briefly introduces the basic concepts involved in subsequent chapters. Firstly, the basic mathematical knowledge of cryptography is introduced in 3.1-3.2, including bilinear mapping and hardness assumptions. Secondly, message authentication code, hash function, digital signature, encryption system, oblivious transfer, oblivious pseudorandom function, zero-knowledge proof system and other cryptographic primitives and their instances are introduced in 3.3-3.10. Finally, the blockchain infrastructure and blockchain data structure are briefly introduced in 3.11.

3.1 Pairing Groups

A pairing group consists of $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with the same order q, such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear mapping.

1. For $\forall\, g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $e(g_1, g_2) \neq \mathbf{1}$.
2. For $\forall\, x, y \in \mathbb{Z}_q$, $\exists\, e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$.

If $\mathbb{G}_1 = \mathbb{G}_2$, we say that $\Phi = (q, \mathbb{G}_1, \mathbb{G}_T, g, e)$ is a type-1 pairing group. If $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, we say that $\Phi = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a type-2 pairing group. If $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no an isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, we say that $\Phi = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a type-3 pairing group [73].

3.2 Hardness Assumptions

Assumption 1.(eBCDH) Given a type-1 pairing group and a tuple of 4 group elements $(g^{1/u}, g^u, g^v, g^w)$ where u, v, w are randomly selected from $\mathbb{Z}_q$, the extended bilinear computational Diffie-Hellman (eBCDH) assumption holds, if no adversary can compute h, such that $h = e(g, g)^{uvw}$ with non-negligible probability [20].

Assumption 2.(PS-2) The PS assumption 2 (PS-2) introduced in [73]. In type-3 pairing group setting, randomly choose $x, y \in \mathbb{Z}_q$, generate $\widehat{X} = g_2^x, \widehat{Y} = g_2^y$. Define an oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_q$ that choose a random element $h \in \mathbb{G}_1$ and output a pair $P = (h, h^{x+ym})$. Given $(g_2, \widehat{X}, \widehat{Y})$ and unlimited access to the oracle $\mathcal{O}(m)$, no adversary can generate a pair P with non-negligible probability, for $h \neq 1_{\mathbb{G}_1}$ and a new m^* not asked to the oracle.

Assumption 3.(XDH/SXDH) Given a type-3 pairing group $(g_1, g_1^\mu, g_1^\nu, g_1^\omega)$ with $\mu, \nu \in \mathbb{Z}_q$, it is hard to decide whether $\omega = \mu\nu$ mod p or random. We say XDH holds, if DDH is hard in $\mathbb{G}_1$. We say SXDH to hold in type-3 pairing group, if DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$ [13].

Assumption 4.(SDL) Given a type-3 pairing group and a tuple of 2 group elements $(g_1^x, g_2^x) \in \mathbb{G}_1 \times \mathbb{G}_2$ computing x is a hard problem. We call it symmetric discrete logarithm assumption. [13].

Assumption 5.(q-MSDH-1) Given a type-3 pairing group $\Phi = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where g_1 and g_2 are generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Given two tuples $(g_1^{x^l}, g_2^{x^l})_{l=0}^{q}$ and (g_1^a, g_2^a, g_2^{ax}) with $x, a \in \mathbb{Z}_q^*$, and no adversary can generate a tuple $(w, P, h^{1/w+x}, h^{a/P(x)})$ with non-negligible probability, where $h \in \mathbb{G}_1$, P is a polynomial in $\mathbb{Z}_q[X]$ with degree at most q, and $w \in \mathbb{Z}_q$ such that the polynomials $X + w$ and P are coprime [72].

3.3 Message Authentication Code

Message authentication code (MAC) protects the integrity and authenticability of data, which is a symmetric cryptography [20]. A message authentication code scheme includes three algorithms, Keygen, Mac and Verify.

- Keygen(1^λ) algorithm generates the secret key k of the authentication.
- Mac(k, m) algorithm generates a tag T with the inputs k and a message m.
- Verify(k, m, T) algorithm checks the validity of the tag T with inputting the tag T, a message m and the key k.

A homomorphic message authentication code scheme has an additional function f, which can generate a new Tag t on message $m_1, \cdots, m_n$ with the inputs tag $t_1, \cdots, t_n$.

3.4 Hash Function

The hash function maps the input of any length to the output of a fixed length [53]. Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function. We say H is collision-resistant if for any pair $x, x^{'} \in \{0,1\}^*$, no efficient algorithm can find a collision such that $H(x) = H(x^{'})$ and $x \neq x^{'}$.

3.5 Digital Signature

Digital signature, an alternative to handwritten signature, can protect the integrity and authenticability of data. A digital signature scheme includes four algorithms.

- $\mathsf{Setup}(1^\lambda)$ algorithm generates a public parameter pp.
- $\mathsf{Keygen}(pp)$ algorithm generates a pair of keys of a signer, the public key pk and private key sk.
- $\mathsf{Sign}(sk, m)$ algorithm enables a signer to sign on a message m with sk and generate a signature σ.
- $\mathsf{Verify}(\sigma, m)$ algorithm enables a verifier to check the correctness of the signature σ on message m, then, returns true or false.

The digital signature scheme should meet the security of Existential Unforgeabilities-chosen Message Attack (EUF-CMA) [37].

3.5.1 Pointcheval-Sanders Multi-signatures

We review the Pointcheval-Sanders Multi-signature scheme (PSM) [25], which works in type-3 pairing group. The PSM construction is as follows:

- $(pp) \leftarrow \mathsf{Setup}(1^\lambda, n, k, \Phi)$: Take security parameter λ, n, k and Φ as input, where n is the number of signers and k is the number of message blocks.

Return the public parameters $pp = (\Phi, g_1, g_2, H_0, H_1)$, where g_1 and g_2 are the generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. $H_0 : \{0,1\}^* \to \mathbb{Z}_q \times \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q^n$ are collision-resistant hash functions.

- $(sk, pk) \leftarrow \mathsf{Keygen}(pp)$: Take the public parameters pp as input, and generate a key pair (sk, pk). Randomly choose $x, y_1, \cdots, y_{k+1} \in \mathbb{Z}_q^*$ and compute $\widehat{X} = g_2^x, \widehat{Y}_1 = g_2^{y_1}, \cdots, \widehat{Y}_{k+1} = g_2^{y_{k+1}}$. Set $sk = (x, y_1, \cdots, y_{k+1})$, $pk = (\widehat{X}, \widehat{Y}_1, \cdots, \widehat{Y}_{k+1})$.

- $(apk) \leftarrow \mathsf{KAggreg}(pp, pk_1, \cdots, pk_n)$: Take pp and $pk_1, \cdots, pk_n$ as input, and output an aggregated public key apk. Compute $H_1(pk_1, \cdots, pk_n) \to (r_1, \cdots, r_n)$ and return $apk = \prod_{i=1}^n pk_i^{r_i}$.

- $(\sigma) \leftarrow \mathsf{Sign}(pp, sk, M = (m_1, \cdots, m_k))$: Take pp, private key sk and message blocks M as input, and output a signature σ. Compute $H_0(M) \to (m', h)$ and return $\sigma = (a, b, c) = (m', h, h^{x + \sum_{i=1}^k y_i m_i + y_{k+1} m'})$.

- $(\sigma) \leftarrow \mathsf{SAggreg}(pp, (pk_i)_{i=1}^n, (\sigma_i)_{i=1}^n, M = (m_1, \cdots, m_k))$: Take pp, the public key $(pk_i)_{i=1}^n$, the signatures $(\sigma_i)_{i=1}^n$ and message $M = (m_1, \cdots, m_k)$ as input, and output an aggregate signature σ. Parse $\sigma_i = (a_i, b_i, c_i)$. If $a_1 = a_2 = \cdots = a_n$ and $b_1 = b_2 = \cdots = b_n$, compute $H_1(pk_1, \cdots, pk_n) \to (r_1, \cdots, r_n)$ and

$$c = \prod_{i=1}^n c_i^{r_i} = b_1^{\sum x_i r_i + \sum_{j=1}^k v_j m_j + v_{k+1} a_1},$$

where $v_j = \sum_{i=1}^n y_{i,j} r_i$ and $v_{k+1} = \sum_{i=1}^n y_{i,k+1} r_i$. Output the $\sigma = (a, b, c) = (a_1, b_1, c)$.

- $(b') \leftarrow \mathsf{Verify}(pp, apk, M = (m_1, \cdots, m_k), \sigma)$: Take pp, the aggregated public key apk, message $M = (m_1, \cdots, m_k)$ and aggregate signature σ as input, and output a bit $b' \in \{0, 1\}$. Parse $\sigma = (a, b, c)$. If $b \neq \mathbf{1}_{\mathbb{G}_1}$ and the equation
$$e(b, \widehat{X} \prod_{i=1}^{k} \widehat{Y}_i^{m_i} \widehat{Y}_{k+1}^{a}) = e(c, g_2),$$
holds, return $b' = 1$; otherwise, return $b' = 0$.

The PSM signature is proved to be EUF-CMA under the q-MSDH-1 assumption in [25].

3.6 Homomorphic Proxy Re-Encryption

Proxy re-encryption (PRE) is a key conversion mechanism applied on ciphertexts [6]. In a PRE scheme, an agent who is semi-trusted converts the ciphertext encrypted with the public key pk_u into ciphertext encrypted with a delegated public key pk_D. In this process, the agent does not have access to the encrypted message, which reduces the risk of data leakage. Homomorphic proxy re-encryption (HPRE) means that there exists an additional evaluation algorithm which can execute homomorphic operation [36]. An homomorphic proxy re-encryption scheme as follows:

- $\mathsf{Setup}(1^\lambda)$: This algorithm inputs security parameter λ, outputs the public parameter pp, where $pp = (q, g, \mathbf{g} = e(g, g), \mathbb{G}, e)$.

- $\mathsf{Keygen}(pp)$: This algorithm inputs public parameter pp, outputs public key and private key (rpk_i, rsk_i). Randomly select $(a_{i,1}, a_{i,2} \in \mathbb{Z}_q)$ and compute $rpk_i = (\mathbf{g}^{a_{i,1}}, g^{a_{i,2}})$, $rsk_i = (a_{i,1}, a_{i,2})$.

- $\mathsf{RKeygen}(rsk_i, rpk_j)$: This algorithm inputs rsk_i and rpk_j, outputs the transformation key rk. Parse $rsk_i = (a_{i,1}, a_{i,2})$ and $rpk_j = (\mathbf{g}^{a_{j,1}}, g^{a_{j,2}})$, compute $rk = (g^{a_{j,2}})^{a_{i,1}}$.

- $\mathsf{Enc}_1(rpk, m)$: This algorithm inputs rpk and m, outputs the ciphertext c. Parse $rpk_i = (\mathbf{g}^{a_{i,1}}, g^{a_{i,2}})$, random select $k \in \mathbb{Z}_q$ and compute $c = (\mathbf{g}^k, \mathbf{g}^m \cdot (\mathbf{g}^{a_{i,1}})^k, 1)$.

- $\mathsf{Enc}_2(rpk, m)$: This algorithm inputs rpk and m, outputs the ciphertext c. Parse $rpk_i = (\mathbf{g}^{a_{i,1}}, g^{a_{i,2}})$, random select $k \in \mathbb{Z}_q$ and compute $c = (g^k, \mathbf{g}^m \cdot (\mathbf{g}^{a_{i,1}})^k, 2)$.

- $\mathsf{REnc}(rk, c)$: This algorithm inputs rk and ciphertext c, outputs the re-encryption ciphertext c. Parse $c = (c_1, c_2, 2)$ and compute $c = (e(c_1, rk), c_2, R)$.

- $\mathsf{Dec}_1(rsk, c)$: This algorithm inputs rsk and ciphertext c, outputs the plaintext. Parse $rsk = (a_1, a_2)$ and $c = (c_1, c_2, c_3)$. If $c_3 = 1$, compute $\mathbf{g}^m = c_2 \cdot c_1^{-a_1}$ and if $c_3 = R$, compute $\mathbf{g}^m = c_2 \cdot c_1^{-1/a_2}$.

- $\mathsf{Dec}_2(rsk, c)$: This algorithm inputs rsk and ciphertext c, outputs the plaintext. Parse $rsk = (a_1, a_2)$ and $c = (c_1, c_2, 2)$, compute $\mathbf{g}^m = c_2 \cdot e(g, c_1^{-a_1})$.

- $\mathsf{EV}(f, \overrightarrow{c})$: This algorithm inputs the function f and the vector $\overrightarrow{c}$, outputs the aggregated ciphertexts c. Parse f as $(w_1, \cdots, w_n)$ and $\overrightarrow{c}$ as $(c_i)_{i \in [n]}$, compute $c = \Pi_{i=1}^n c_i^{w_i}$.

3.7 Homomorphic Proxy Re-Authenticators

Homomorphic proxy re-authenticator (HPRA) enables multiple users to sign on data with their own keys, and an aggregator can convert these users' signatures into one MAC under a server's key without having any knowledge of it [36]. Meanwhile, there is a function that can be evaluated by the aggregator, where the resulting MAC is the evaluation corresponding to the respective function. The homomorphic proxy re-authenticators (HPRAs) have become an important building block for data aggregation and authentication.

There are three entities in a HPRA scheme, namely, User, Aggregator and Server.

- **User**: The user generates its public key and private key. Then, it signs on the data it collects and sends the authenticated data and its signature to the aggregator.
- **Aggregator**: The aggregator converts the users' signatures into a MAC under a server's key.
- **Server**: The server initializes the system and sends the secret key of MAC to each user. It can check the MAC received from the aggregator and verify the data received from the users.

We review a HPRA scheme [36].

- $\mathsf{Setup}(1^\lambda)$: This algorithm inputs security parameter λ, outputs the public parameter pp, where $pp = (q, (g_i)_{i\in[l]} \in \mathbb{G}^l, \mathbf{g} = e(g, g), e, H), H : \mathbb{Z}_q \to \mathbb{G}$ is a collision-resistant hash function.

- $\mathsf{Keygen}(pp)$: This algorithm inputs public parameter pp, outputs public key and private key (pk, sk). Randomly choose $x \in \mathbb{Z}_q$ and compute $pk = (g^x, g^{1/x})$. Set $id = g^x$ and $sk = x$.

- $\mathsf{VGen}(pp)$:This algorithm inputs public parameter pp, outputs a MAC key mk. Randomly choose $\alpha \in \mathbb{Z}_q$, set $mk = \alpha$.

- $\mathsf{Sign}(sk, (m_1, ..., m_l), \tau)$: This algorithm inputs private key sk, message blocks $(m_1, ..., m_l)$, and a public parameter τ, outputs a signature σ' on messages $(m_1, ..., m_l)$. Compute $\sigma' = (H(\tau||g^x) \cdot \Pi_{i=1}^{l} g_i^{m_i})^x$.

- $\mathsf{Verify}(pk, (m_1, ..., m_l), \tau, \sigma)$: This algorithm inputs public key pk, $(m_1, ..., m_l)$, τ and signature σ. Compute

$$e(H(\tau||g^x) \cdot \Pi_{i=1}^{l} g_i^{m_i}, g^x) \overset{?}{=} e(\sigma, g) \wedge (m_1, ..., m_l) \overset{?}{=} (m_1', ..., m_l').$$

- $\mathsf{SRGen}(sk, aux)$: This algorithm inputs sk and auxiliary parameter aux, outputs rk. Return $rk \leftarrow \emptyset$.

- $\mathsf{VRGen}(pk_i, mk, rk_i)$: This algorithm inputs pk, the Mac key mk, and rk_i, outputs the rotation key ak_i. Parse $pk_i = (g^{x_i}, g^{1/x_i})$, $mk = \alpha$, compute $ak_i = (g^{1/x_i})^\alpha$.

- $\mathsf{Agg}((ak_i)_{i\in[n]}, (\sigma_i)_{i\in[n]}, \tau, f)$: This algorithm inputs $(ak_i)_{i\in[n]}, (\sigma_i)_{i\in[n]}$, τ, f, outputs aggregated signature μ. Parse f as $w_1, \cdots, w_n$ and for $i \in [n]$ parse σ_i as $(\sigma_i', (m_1, ..., m_l)_i)$ and compute $\Lambda = (Eval(f, ((m_1, ..., m_l)_i)_{i\in[n]}), \mu, \tau)$, where $\mu = \Pi_{i\in[n]} e(\sigma_i'^{w_i}, ak_i)$.

- AVerify(mk, Λ, ID, f): This algorithm inputs Mac key mk, the aggregated signature Λ, ID, and f, check the validity of signature. Parse $ID = (g^{x_i})_{i\in[n]}$ and f as $w_1, \cdots, w_n$. Compute

$$\mu' \overset{?}{=} (\Pi_{i=1}^{n} e(g^{w_i}, H(\tau||g^{x_i})) \cdot e(\Pi_{i=1}^{l} g_i^{m_i}, g))^{\alpha}.$$

3.8 Oblivious Transfer

A classic t-out-of-n oblivious transfer (OT) [21] protocol consists of two participants, a sender and a receiver. The sender inputs its messages $(m_1, ..., m_n)$ while the receiver inputs its t choices $b_1, ..., b_t$. After the OT protocol, the receiver outputs $m_{b_1}, ..., m_{b_t}$ and the sender has no outputs. The security of OT requires that the sender learns nothing about the choices of the sender and the sender cannot learn anything about m_i if $i \notin \{b_1, ..., b_t\}$.

3.9 Oblivious PRF

An oblivious pseudorandom function (OPRF) [43] is a protocol involving two parties, the sender and the receiver. The receiver selects a single value x as its input, while the sender does not need any input. At the end of the protocol, the sender gets a PRF key k which means the sender can compute $F_k(y)$ for any y, the receiver learns and only learns the PRF evaluation of its own input $F_k(x)$. The security of OPRF requires that if $x \neq y$, the PRF value $F_k(y)$ looks random to the receiver.

3.9.1 Multi-Point Oblivious PRF

A multi-point OPRF [28] is a protocol where the receiver selects multiple values $x_1, x_2, ...x_n$ as its input and learns multiple PRF values $F_k(x_1), F_k(x_2), ..., F_k(x_n)$ under the same PRF key k, while the sender still does not need any input and gets the PRF key k at the end of the protocol. The security of multi-point OPRF requires that the receiver can only learns $F_k(x_1), F_k(x_2), ..., F_k(x_n)$, in addition, for any $y \notin \{x_1, x_2, ..., x_n\}$, $F_k(y)$ looks random to the receiver. The multi-point OPRF seems to be more suitable for addressing the PSI and related problems since all PRF evaluations are based on the same key k, which means that there is no need to worry about misjudgments caused by the difference of the corresponding key.

CM's multi-point oblivious PRF. The work of [28] presents an efficient multi-point OPRF base on OT extension [51], in which the PRF key is an $m \times \omega$ binary matrix Q where m is the number of receiver's input elements $|X|$, and ω is a secure parameter. At the beginning of their OPRF protocol, the receiver generates two $m \times \omega$ binary matrices T and U, where T is a random matrix while U is generated according to T and the input elements. Specifically, denote the column vectors of U and T by U_i and T_i respectively, and let $v = F_k(x)$ in which $F : \{0,1\}^\lambda \times \{0,1\}^{l_1} \rightarrow [m]^\omega$ is a pseudorandom function and k is known to both parties. For all $i \in [\omega]$, the receiver sets $U_i[v[i]] = T_i[v[i]]$, then for all remaining blank spaces sets $U[] = \overline{T[]}$. After the precomputation, the receiver runs ω oblivious transfers with the sender taking these two matrices as input, while the sender's inputs are ω choices $s[1], ..., s[\omega]$. As a result, the sender obtains a new $m \times \omega$ matrix Q where Q_i is either equal to U_i or T_i, which is key of this oblivious PRF. When the sender wants to do OPRF evaluation on some y, it computes $u = F_k(y)$

and its OPRF value is $H(Q_1[u[1]]||...||Q_\omega[u[\omega]])$ where H is a hash function. For each $x \in X$, the receiver can gets its OPRF value through a similar way, which is $H(T_1[v[1]]||...||T_\omega[v[\omega]])$. It can be seen that the receiver can only computes OPRF values on its own input elements since the receiver knows nothing about $s[1], ..., s[\omega]$, while the sender can learn nothing about the receiver's private input. The details are shown in Table 3.1.

3.10 Non-Interactive Zero-Knowledge Proof Systems

The non-interactive zero-knowledge proof system (NIZK) [37] includes the following three algorithms.

- $(CRS) \leftarrow \mathsf{Setup}(1^\lambda)$: Input a security parameter λ, and output a common reference string CRS.
- $(\pi) \leftarrow \mathsf{Proof}(CRS, h, w)$: Input CRS, a statement h and a witness w, and output a proof π.
- $(1/0) \leftarrow \mathsf{Verify}(CRS, \pi, h)$: Inputs CRS, a statement h and a proof π, if the verification holds, output 1; otherwise, output 0.

A non-interactive zero-knowledge proof system should satisfy completeness, soundness and zero-knowledge. In this thesis, we follow the conception of NIZK in [24] which can proof the relation over discrete logarithm values.

3.10.1 Proving Knowledge of PSM Signature

The PSM signatures allow that one can efficiently prove in the knowledge of a PSM signature, as mentioned in [25]. Here, we give a concrete description. The

Parameters:

- The sender S and the receiver R agree on two security parameters that are λ, σ, four protocol parameters that are m, ω, l_1, l_2, $H_1 : 0, 1^* \rightarrow 0, 1^{l_1}$ and $H_2 : 0, 1^{\omega} \rightarrow 0, 1^{l_2}$ are hash functions, $F : 0, 1^{\lambda} \times 0, 1^{l_1} \rightarrow [m]^{\omega}$ is a pseudorandom function.

Input:

- S has no input.
- R has input $X = \{x_1, x_2, ..., x_n\}$.

1. Precomputation

- S randomly samples a string $s \overset{\$}{\leftarrow} \{0, 1\}^{\omega}$
- R dose the following:
  - Generate an $m \times \omega$ random binary matrix $T \overset{\$}{\leftarrow} \{0, 1\}^{m \times \omega}$ and initialize an $m \times \omega$ binary matrix D. Denote the ith column vector of D by D_i, then set $D_1 = ... = D_{\omega} = 1^m$.
  - Sample a uniformly random PRF key $k \overset{\$}{\leftarrow} \{0, 1\}^{\lambda}$.
  - For each $x \in X$, compute $v = F_k(H_1(x))$. Set $D_i[v[i]] = 0$ for all $i \in [\omega]$.
  - Generate a matrix $U = T \oplus D$.

2. Oblivious Transfer

- S and R run ω oblivious transfers where R takes $\{T_i, U_i\}(i \in \omega)$ as inputs and S takes ω choices $s[1], ..., s[\omega]$ as inputs.
- As a result, S obtains the key of the OPRF that is a new $m \times \omega$ matrix Q where $Q_i \in \{T_i, U_i\}(i \in \omega)$.
- For each $x \in X$, R gets its OPRF value $OPRF(x) = H_2(T_1[v[1]]||...||T_{\omega}[v[\omega]])$.

3. OPRF Evaluation

- R sends the key k of the PRF to S.
- For any element y, S computes $v = F_k(H_1(y))$ and gets its OPRF value $OPRF(y) = H_2(Q_1[v[1]]||...||Q_{\omega}[v[\omega]])$.

Table 3.1: CM's multi-point oblivious PRF

PSM signature can prove with zero-knowledge knowledge of a multi-signature since the signature elements can be re-randomized and the verify algorithm does not check the hash relation between a and the message block $M = (m_1, \cdots, m_k)$. With the aggregation key apk, a prover can prove knowledge of $(M, \sigma = (a, b, c))$, such that

$$e(b, \widehat{X} \prod_{i=1}^{k} \widehat{Y}_i^{m_i} \widehat{Y}_{k+1}^{a}) = e(c, g_2).$$

As shown in Table 3.2, the concrete construction of proving knowledge of a PSM signature, where H as a random oracle, since the Schnorr proof [80] is zero-knowledge with Fiat-Shamir heuristic [42].

Prover		Verifier
$u_1, u_2 \in \mathbb{Z}_q$		
$(b', c') = (b^{u_1}, (cb^{u_2})^{u_1})$	$\xrightarrow{b', c'}$	
$\alpha_1, \cdots, \alpha_{k+2} \in \mathbb{Z}_q$		
$S = \Pi_{i=1}^{k} e(b', \widehat{Y}_i)^{\alpha_i} \cdot$		
$e(b', \widehat{Y}_{k+1})^{\alpha_{k+1}} \cdot$		
$e(b', g_2)^{\alpha_{k+2}}$		
$c = H(b', c', S)$		
for $i \in [k]$,		
$s_i = \alpha_i - cm_i$,		
$s_{k+1} = \alpha_{k+1} - ca$		
$s_{k+2} = \alpha_{k+2} + cu_2$		
	$\xrightarrow{c, s_1, \cdots, s_{k+2}}$	Verify
		$c = H(b', c', \Pi_{i=1}^{k} e(b', \widehat{Y}_i)^{s_i} \cdot$
		$e(b', \widehat{Y}_{k+1})^{s_{k+1}} \cdot e(b', g_2)^{s_{k+2}}) \cdot$
		$e(c', g_2)^{c} e(b', \widehat{X})^{-c})$

Table 3.2: The proving knowledge of PSM signature

3.11 Blockchain

Blockchain infrastructure consists of six layers: data layer, network layer, consensus layer, incentive layer, contract layer and application layer. As a distributed ledger, blockchain adopts block chain structure, including Block, Block header, Block body and Chain structure. Figure 3.1 shows the specific structure of blockchain. The existing blockchain is divided into public blockchain, private blockchain and consortium blockchain according to type.

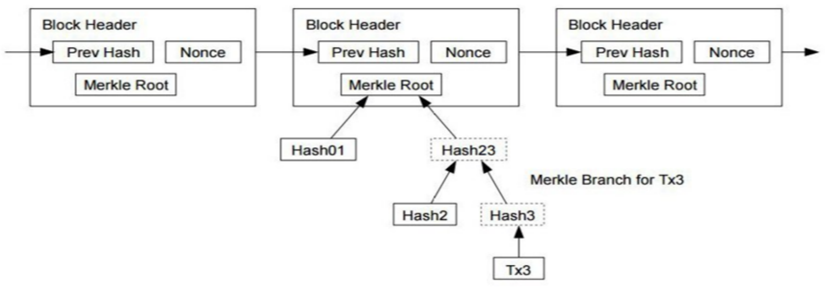


Figure 3.1: The structure of blockchain

In the public blockchain, any organization, group or individual can participate in a transaction, and participate in reaching a consensus, and the resulting transaction can be validated. The public blockchain is the earliest type of blockchain, and cryptocurrencies such as Bitcoins and Ethereum are based on the public blockchain architecture.

In the consortium blockchain, a number of pre-selected nodes are designated as bookkeepers within a specific organization or group, and block generation is jointly decided by the pre-selected nodes. Other nodes can participate in the transaction, but are not responsible for the transaction package and generating the block.

Any participant can query related transactions through the open port of the blockchain. Hyperledger Fabric is the representative one.

Private blockchain only uses blockchain technology to keep accounts, which can be companies, organizations or individuals who have exclusive access to the blockchain and can effectively control data access and compilation. Private blockchains can better protect the privacy of the organization itself, and the transaction data will not be made public to the whole network.

Chapter 4

Delegated PSI-CA Protocol

In the chapter, we propose a lightweight delegated PSI-CA protocol, which does not need to do additional pre-operations, so the computation complexity and the communication complexity on the client side are further reduced compare to the existing works. In addition, we build a privacy-preserving contact tracing system by utilizing our protocol, which can be publicly checked, named PC-CONTrace. The background of delegated PSI-CA and contact tracing system are described in Section 4.1. The specific delegated PSI-CA protocol and its security discussion are described in Section 4.2. Our concrete contact tracing system PC-CONTrace is given in Section 4.3. The implementation and performance evaluation are described in Section 4.4. Finally, we conclude the chapter in Section 4.5.

4.1 Background

Secure multiparty computation (MPC) is a significant means to realize privacy computing, which mainly uses cryptography technology [32, 65, 66]. Private set intersection (PSI) is an important problem in MPC, it refers to that multiple participants can jointly compute the common intersection of their private sets through interaction without revealing any other information about their respective sets, which means the privacy of the set of participants is guaranteed. As an important research field of MPC, PSI not only plays a significant role in scientific computation, but also has a wide range of applications in real life since a lot of data can be abstracted into sets. Typical application scenarios include privacy-preserving similar documents detection [18], private contact discovery [35], secure human gene detection [10], private proximity testing [64], privacy-preserving social network relationship discovery [63], online recommendation services and matchmaking websites. Under the condition that the data is held by different managers, PSI achieves a win-win situation of privacy preserving and information sharing. However, the classic PSI is not fully applicable for some specific scenarios, so a variant of the PSI has been widely studied, known as Private Set Intersection Cardinality (PSI-CA). Different from PSI, PSI-CA only allows the parties learn the size of the intersection but not the specific intersection elements. This security requirement just meets the needs of an emerging technology, privacy-preserving contact tracing, so that PSI-CA has recently been used in contact tracing to protect people's privacy [40].

Due to the rapid and severe spread of COVID-19 around the world, the normal life of human beings has been greatly affected. Among the many ways to fight against the global epidemic, contact tracing is considered to be one of the most effective means. Contact tracing can identify people who have contact with the sick people in the population, then the target population can be quickly isolated and detected, so that the spread of the epidemic can be effectively controlled. At present, many kinds of mobile apps have been designed for contact tracing, most of which use Bluetooth to obtain signals from nearby mobile phones. A huge amount of personal contact information are being collected and analyzed every day, which is an important concern for personal privacy. Therefore, some privacy-preserving contact tracing systems have been introduced recently [86, 27, 77, 5, 75, 60], most of them require the user to spend a large amount of mobile data for transmission or require the user to complete heavy computation in the resource-limited mobile terminal, which leads to the cost of a lot of time and money. In order to overcome such problems, a delegated PSI-CA protocol is introduced by Duong et al. [40] and they further applied it to their contact tracing system. In their delegated PSI-CA protocol, the client can delegate its computation to some untrusted cloud servers, so that the client-side computation is greatly reduced. However, their work is based on a single-point oblivious pseudorandom function [55], as a result the communication and computation consumption on the client side is still not ideal, about 1.5 times and 3times the size of the client's contact set, respectively, which is still unacceptable for users in densely populated cities or areas. Therefore, privacy-preserving contact tracing system with low communication overhead and faster computing for densely populated areas is highly desirable, delegated PSI-CA with low communication complexity is also a challenging research topic. In addition, during the epidemic, inaccurate information often leads to public panic or even damages to people's personal rights and interests. As a result, it is essential that the results of the contact tracing system can be publicly checked when necessary to circumvent the problems caused by misinformation.

4.1.1 Contributions

In order to improve the communication efficiency of privacy-preserving contact tracing system and reduce the computation overhead of the user, we first propose a lightweight delegated PSI-CA protocol based on multi-point OPRF [28] and collision-resistant hash function, which is the first multi-point OPRF based delegated PSI-CA construction to the best of our knowledge, the computation in this protocol is delegated to cloud servers. Our protocol does not need to do extra pre-operations that are multiple of the set size, so the computation complexity and the communication complexity on the client side are further reduced. In addition, our protocol is secure under all cloud servers collusion, as long as the back-end server does not participate in the collusion and the user does not collude with the so-called combiner.

Then, we build a privacy-preserving contact tracing system by utilizing our delegated PSI-CA protocol, PC-CONTrace, which can be publicly checked due to the application of blockchain technology. We test the efficiency of the cryptographic algorithms under different set sizes and compare it with related scheme from the aspects of functionality and performance. The experimental results show that for the same number of confirmed cases, the more people a user contacts each day, the more advantage our system has, which means that our system is more practical in densely populated areas.

4.2 The Construction of Delegated PSI-CA Protocol

In this section, we propose a lightweight delegated PSI-CA construction based on multi-point OPRF [28] and collision-resistant hash function.

4.2.1 Problem Definition

There are three kinds of participants in a delegated PSI-CA protocol [40]: a back-end server S, a client C, a set of cloud servers CS. Let $\prod$ be a delegated PSI-CA protocol, $\prod$ computes the PSI-CA function f as follows: $\prod : (\{0,1\}^*)^N \times (\{0,1\}^*)^n \times \perp \rightarrow f_{|\bigcap|} \times \perp \times \perp$, where N and n are the size of back-end server's and client's input set respectively, $\perp$ denotes the empty input and output.

4.2.2 Security Model

Adversarial model. Informally, a *semi-honest adversary* follows the protocol exactly as the protocol requires but may record the intermediate result and attempt to obtain additional knowledge from transcripts of the execution. If there are n semi-honest participants, each has private input $m_i (i \in [n])$ collaborate to execute protocol $\prod$ of the private computation function F and get the final result once the protocol is complete. Let $M = (m_1, m_2, ..., m_n)$, and the message sequence obtained by each participant during the protocol execution is represented as $view_i^{\prod}(M) = (m_i, r_i, \Theta)$, where Θ represents the messages received by the ith participant, r_i is the random number generated by the ith participant during the protocol.

Definition 4.1. Security of semi-honest MPC protocols [48]. Let $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ be a function, $f_i(m_1, ..., m_n)$ be the ith element of f, $I = \{P_{i1}, .., P_{is}\} \subseteq \{P_1, ..., P_n\}$ be an arbitrary subset of the participants, $f_I(m_1, ..., m_n)$ denote the sequence $f_{i1}(m_1, ..., m_n), ..., f_{is}(m_1, ..., m_n)$, $output^{\prod}(M)$ denote all the honest parties' outputs of the protocol $\prod$. We say a protocol $\prod$ is securely computed in semi-honest model if there is a probabilistic polynomial S such that for any I the following formula holds,

$$\{S(I, (m_{i1}, m_{i2}, ..., m_{is}), f_I(M))\}_{M \in (\{0,1\}^*)^n} \overset{c}{\equiv}$$

$$\{\{view_I^{\Pi}(M)\}_{M \in (\{0,1\}^*)^n}, output^{\Pi}(M)\},$$

where $\overset{c}{\equiv}$ denotes it is computationally indistinguishable, $view_I^{\Pi}(M) = (I, view_{i1}^{\Pi}(M),$ $..., view_{is}^{\Pi}(M))$, which means $view_I^{\Pi}(M)$ contains only the sequence of messages received by the participants in I during the protocol execution, excluding messages passed between honest parties that are invisible to the participants in I.

A malicious adversary usually refers to the party that does not follow the protocol during the process of the protocol, they can make arbitrary attack, such as refusing to participate in the protocol, modifying the private input as needed, and terminating the protocol early [28].

Collusion security. Collusion usually means that two or more dishonest participants share their information with each other in the process of the protocol, the colluded participants usually want to obtain more information than they should or sabotage the implementation of the protocol [40].

In this work, we consider the semi-honest model. In addition, there can be collusion between adversaries as long as the back-end server is not corrupted in the protocol and the client cannot collude with the so-called combiner.

4.2.3 The Construction from Multi-Point OPRF

Our delegated PSI-CA protocol is presented in Figure 4.1, which is under the semi-honest security model. Our work is mainly based on the efficient multi-point

Parameters:

- Set size n and N.
- A back-end server S, a client C, and β cloud servers $CS_1, ..., CS_\beta$.
- A collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{l_1 * \beta}$, a pseudo-random function $F_C : \{0,1\}^\lambda \times \{0,1\}^{l_2 * \beta} \rightarrow \{0,1\}^{l_2 * \beta}$.
- A multi-point OPRF primitive $OPRF$ described in Table 3.1.

Input:

- Back-end server S has input $Y = \{y_1, ..., y_N\}$.
- Client C has input $X = \{x_1, ..., x_n\}$.
- Cloud server $CS_{j \in [\beta]}$ has no input.

Protocol:

I. Initialization phase
   - The client randomly selects a key k' of F_C and sends k' to S.

II. Distribution phase
   - The client uses the hash function H to hash its input X.
   - For each hash value $H(x_i), i \in [n]$, the client divides it into β shares such that the length of the share $|H(x_i)_j| = l_1$, and $H(x_i)_1 || H(x_i)_2 || ... || H(x_i)_\beta = H(x_i)$, then sends $H(x_i)_j$ to CS_j.

III. Server computation phase
   1. Each cloud server CS_j takes its shares $H(x_1)_j, ..., H(x_n)_j$ as input and invokes the multi-point OPRF protocol with S, where S acts as a sender and CS_j acts as a receiver.
   2. After the multi-point OPRF protocol, the cloud server CS_j gets n PRF values $OPRF_j(H(x_i)_j), i \in [n]$, while the back-end server gets β OPRF keys which are $k_1, ..., k_\beta$.
   3. For all $j \in [\beta - 1]$, each CS_j sends $OPRF_j(H(x_1)_j), ..., OPRF_j(H(x_n)_j)$ to the combiner CS_β.
   4. For all $i \in [n]$, the combiner CS_β computes $OPRF(x_i) = OPRF_1(H(x_i)_1) || ... || OPRF_\beta(H(x_i))_\beta)$, permutes these n values and sends back to C.

5. The back-end server S uses the hash function H to hash its input Y, divides these hash values into β shares through the same way and computes $OPRF(y_i)$ for each $i \in [N]$ via using the OPRF keys.

6. For all $i \in [N]$, S computes $F_C(k^{'}, OPRF(x_i))$, and makes the final set $\{F_C(k^{'}, OPRF(y))\}$ public.

7. After collecting all $OPRF(x)$ from the combiner, the client C computes $F_C(k^{'}, OPRF(x))$ and gets the set $\{F_C(k^{'}, OPRF(x))\}$.

IV. Client's output: $\psi = |\{F_C(k^{'}, OPRF(x))\} \cap \{F_C(k^{'}, OPRF(y))\}|$.

Figure 4.1: Delegated PSI-CA protocol

OPRF proposed in [28], the specific OPRF computations are delegated to the cloud servers. In order to protect the private set of clients while delegating computation to the cloud servers, the core idea of our protocol is to share the elements of the set with the cloud servers in a secure and efficient manner. Due to the computing characteristics of CM's multi-point OPRF, we must ensure that the OPRF value of the share of the data in the intersection after sharing is still corresponding to the same, so as to ensure the correctness of the result after recovery. Therefore, we choose quota distribution method to meet this requirement. To ensure that even if all cloud servers collude, including the combiner, the user's private input will still not be exposed, a collision-resistant hash function H is used before sharing, so that each cloud server can only get a part of the hash value of the original input during the protocol process. The security of the hash function ensures that no information about the original input will be disclosed. After interacting with back-end server, each cloud server only gets a portion of the OPRF value, a pseudorandom function F_C is used to ensure that all cloud servers, including the combiner, cannot know the final result, which is the cardinality of the intersection, this is because the key $k^{'}$ of the pseudorandom function F_C is shared only between the client and the back-end server. The detailed interaction process is described in Figure 4.2.

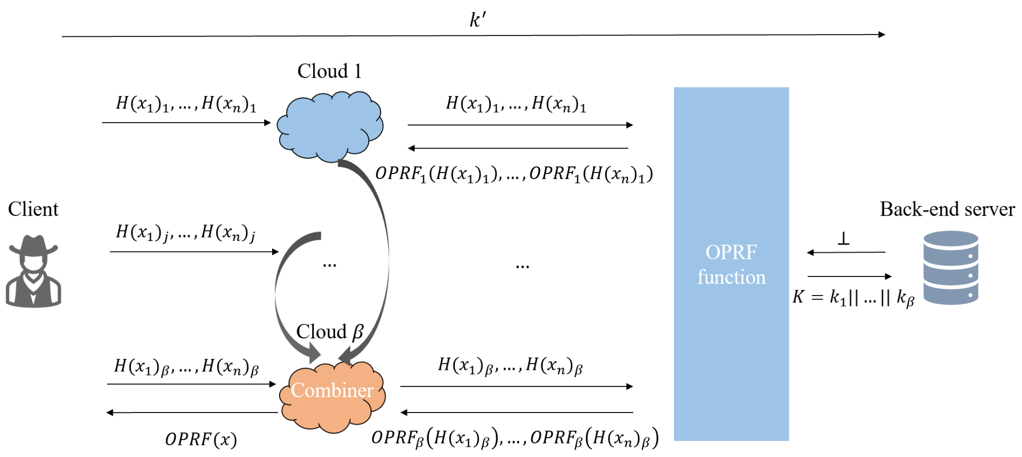

k'
Cloud 1
$H(x_1)_1, \dots, H(x_n)_1$
$H(x_1)_1, \dots, H(x_n)_1$
$OPRF_1(H(x_1)_1), \dots, OPRF_1(H(x_n)_1)$
Client
$H(x_1)_j, \dots, H(x_n)_j$
...
...
OPRF function
Back-end server
$\perp$
$K = k_1 || \dots || k_\beta$
Cloud β
$H(x_1)_\beta, \dots, H(x_n)_\beta$
Combiner
$H(x_1)_\beta, \dots, H(x_n)_\beta$
$OPRF(x)$
$OPRF_\beta(H(x_1)_\beta), \dots, OPRF_\beta(H(x_n)_\beta)$

Figure 4.2: The interaction process of delegated PSI-CA protocol

4.2.4 Security and Discussion

The PSI-CA protocol securely implements the delegated PSI-CA function defined in 4.2.1. in the security model we mentioned in Definition 4.1.

Theorem 4.1. The delegated PSI-CA protocol described in Figure 4.1 is secure in the semi-honest model if the $OPRF$ described in Table 3.1 is secure, H is a collision-resistant hash function and F_C is a pseudorandom function.

Simulating client. For simulating the corrupt client $I = \{C\}$, the simulator S only sees a set of n OPRF values that have been permuted by the combiner who cannot collude with. According to the pseudorandomness of the OPRF protocol, all the values are indistinguishable from random elements, so for each $x_i \notin X \cap Y$ we can replace the value $OPRF(x_i)$ with an independently random element. For $x \in X \cap Y$, we can replace them with any $x^{'} \in X \cap Y$, since the simulator does not know the specific permutation used by the combiner. Therefore, $\{S(I, (m_C), f_I(M))\} \overset{c}{\equiv} \{\{view_I^{\Pi}(M)\}_{M\in(\{0,1\}^*)^n}, output^{\Pi}(M)\}$, that is the simulator can only learn the final result $|X \cap Y|$ and its own input X.

Simulating cloud servers. For simulating the coalition of corrupt cloud servers $I = \{CS_{i1}, ..., CS_{is}\}, s \in [\beta - 1]$, the simulator S can see the received shares sent by the client, randomness in the corresponding OPRF protocol and and transcripts from the OPRF ideal functionality. Two cases need to be considered:

• Security for the client: Each cloud server receives a portion of the user's private input's hash value, so even if all the cloud servers collude, they can only get the hash of the private input but not the original input. In other words, the simulator does not learn anything at this phase, which means the share can be replaced with random. The security of the OPRF ensures that during the process of the OPRF protocol, the simulator cannot learn any information other than the OPRF output. In our security model, the back-end server cannot be involved in collusion, so that we can replace the OPRF outputs with randoms. In the last step of the protocol, the back-end server publishes a set of the PRF values $\{F_C(k', OPRF(y))\}$, where k' is unknown to the simulator, all the values are indistinguishable from random elements to the simulator, which means we can replace them with randoms. Therefore, $\{S(I, (m_{i1}, ..., m_{is}), f_I(M))\} \overset{c}{\equiv} \{\{view_I^{\Pi}(M)\}_{M\in(\{0,1\}^*)^n}, output^{\Pi}(M)\}$, that is the simulator S ultimately knows nothing about the final result $|X \cap Y|$ and the private X.

• Security for the back-end server: In the whole process of our protocol, back-end server only interacts with the cloud servers in the process of OPRF protocol, the security of the OPRF protocol ensures that the simulator cannot learn any information other than the OPRF output during this process, according to the pseudorandomness of the OPRF protocol, all the values can be replaced by random elements. In the last step of the protocol, the back-end server publishes a set of the PRF values $\{F_C(k', OPRF(y))\}$, where k' is unknown to the simulator, all the values are indistinguishable from random elements to the simulator, which means we can replace them with randoms. Therefore, $\{S(I, (m_{i1}, ..., m_{is}), f_I(M))\} \overset{c}{\equiv} \{\{view_I^{\Pi}(M)\}_{M \in (\{0,1\}^*)^n}, output^{\Pi}(M)\}$ that is the simulator ultimately knows nothing about the final result $|X \cap Y|$ and the back-end server's private input Y.

Simulating back-end server. The simulation is elementary since we use the abstraction of the multi-point OPRF functionality.

4.3 Privacy-Preserving Contact Tracing System

In the section, we build a privacy-preserving contact tracing system by utilizing our delegated PSI-CA protocol, which can be publicly checked if necessary, namely PC-CONTrace. The PC-CONTrace system consists of five entities: user, medical centre, clouds, back-end server, Blockchain. We are inspired by the BLE-based approaches of Apple and Google contact tracing project, but we also provide another property with the applying of blockchain, which is that the results can be publicly checked when there is any misinformation. The following is a brief definition of the entities.

- **User**: User exchanges token with other users and executes privacy-preserving contact tracing with back-end server via blockchain.
- **Clouds**: Clouds assist user to execute delegated computation and transfer users' data to back-end server.

- **Back-end server**: Back-end server maintains the contact tracing system and generates system parameters. Back-end server collects users diagnosis token and execute OPRF with Clouds. Then, it publishes the result to blockchain.
- **Medical Centre**: Medical Centre distributes diagnosis certification to user.
- **Blockchain**: Blockchain is a distributed ledger, which records the voucher of the combiner and the final result of back-end server. A User can confirm whether it has been in contact with confirmed cases via the blockchain, or make the results publicly checkable by revealing the final results and combining them with voucher. The parties of blockchain can publish transaction to blockchain and execute smart contract.

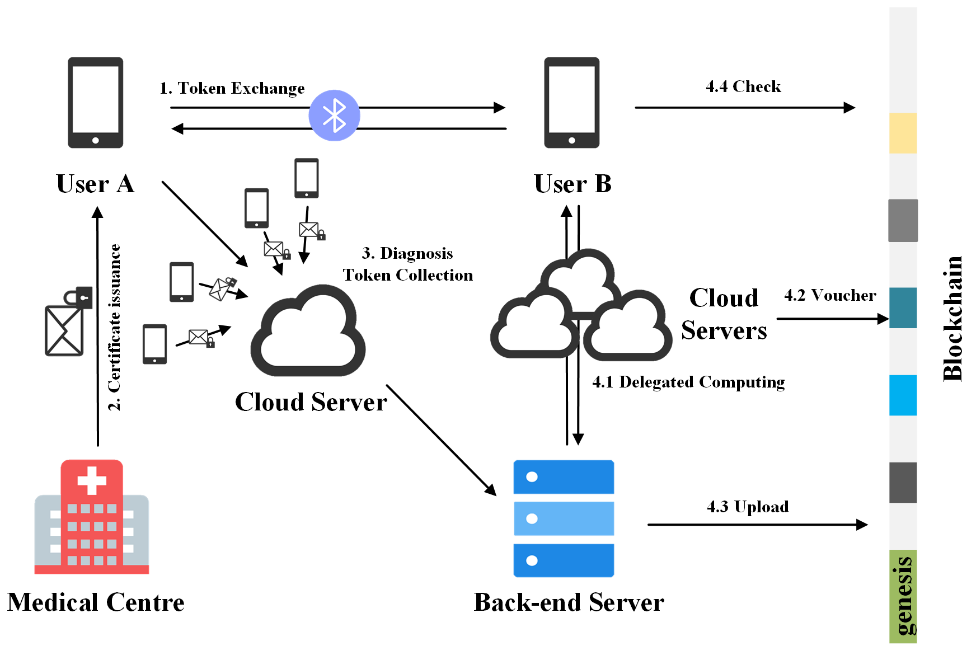


Figure 4.3: The system model of PC-CONTrace

The system model of PC-CONTrace is shown in Figure 4.3. The system model includes four phases.

(1) Based on short-range bluetooth signal of users' smartphone, two users can exchange token.

(2) If anyone is diagnosed with the epidemic by the medical centre, the medical centre will issue a certificate to user by end-to-end encryption.

(3) The diagnosed user encrypts its token and the certificate using the public key of the back-end server and sends it the cloud server. The cloud server permutes all the ciphertexts it received and transmits them to the back-end server. The back-end server decrypts them and verifies the certificates of the diagnosed users. If a certificate is valid, the back-end server store the corresponding diagnosed token.

(4) Executing privacy-preserving contact tracing, which includes four steps.

  (4.1) Cloud-assisted delegated computation, the user executes a delegated PSI-CA algorithm with the back-end server, during which most of the computation is done by several cloud servers.

  (4.2) When the computation is complete, the combiner in the cloud server uploads the hash value of the final result to the blockchain, which can be viewed as a voucher.

  (4.3) The back-end server inputs diagnosis token list and execute the delegated PSI-CA algorithm. Then, it uploads the computing results to blockchain.

(4.4) The user inputs tokens it received and executes check operation on blockchain. Then, it determines if it has been in contact with confirmed cases.

Remark. If the user is judged to have had contact with the confirmed case by the public health institutions due to some misinformation, the user can use its private key k_C to expose the data uploaded by the back-end server in the blockchain so that the results can be publicly checked. It's possible to determine if the user is lying by comparing the hash value of the user's local data with the uploaded voucher. The immutability of blockchain can ensure that users' rights and interests are not infringed.

4.4 Implementation and Performance

In the section, to demonstrate the performance of our PC-CONTrace, we implement the delegated PSI-CA construction in C++ programming by using Clion, a cross-platform IDE for C and C++. We develop our delegated PSI-CA protocol based on two open-source codes on Github that are OPRF-PSI[1] and delegated-psi-ca[2]. We run the experiment on the Linux operation with Ubuntu 16.04 LTS operation system and AMD Ryzen 7 3700X 8-core 3.6 GHz Processor, 16GB RAM. The implement code is available on Github[3].

[1]https://github.com/peihanmiao/OPRF-PSI
[2]https://github.com/nitrieu/delegated-psi-ca
[3]https://github.com/xiaozhao1234/PSI-CA

4.4.1 Implementation Analysis

All implementations of our delegated PSI-CA construction are based on setting the security parameter $\lambda = 128$, the statistical security parameter $\sigma = 40$. At the same time, let N be size of the back-end server's set and n be size of the user's set, and β denotes the number of clouds.

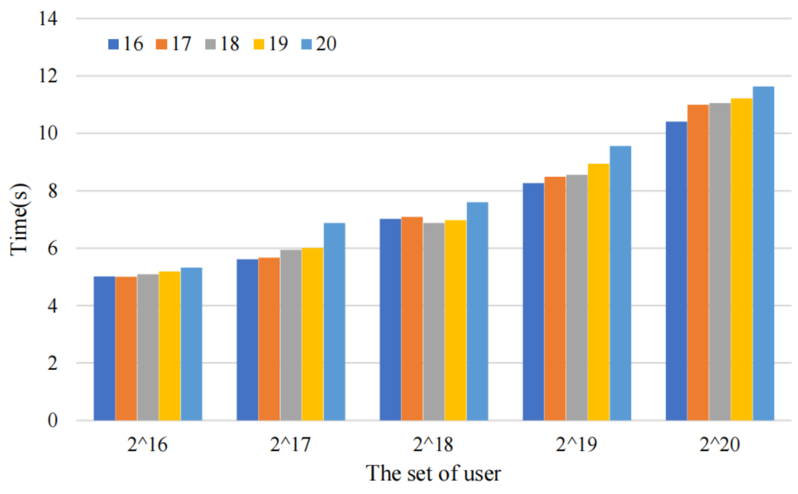


Figure 4.4: The time cost of our Delegated PSI-CA at different n and N

We select the Naor-Pinkas OT as the base OT and implement it with libOTe library[4]. For the instantiation of the function PRF, we apply a pseudorandom generator (PRG) on the top of cipher block chaining MAC. In particular, we chose AES for the instantiation, which is inspired by the idea of [28].

We perform the delegated PSI-CA protocol with the range of set size $N = \{2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}\}$ and $n = \{2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}\}$, and the number of cloud is $\beta = 2$ which is in line with actual deployments. We use the exponent from 16 to 20 to denote the set of the user and the back-end server. We execute the algorithms

[4]https://github.com/osu-crypto/libOte

of our delegated PSI-CA protocol, including the initial algorithm and the operation for the user, the clouds and back-end server in the protocol. The time cost of the algorithms is shown in Figure 4.4.

Specifically, with the increase of the user's and the back-end server's sets' elements, time consumption increases continuously. The increase of user data set has a greater impact on the total system time consumption than the increase of server set, since the size of the user set affects the number of matrix calculations in the OPRF computation process. The server only calculates the OPRF values of its own data set locally, so there is no significant impact on the system computing overhead.

4.4.2 Functionality and Performance Analysis

In the section, we compare the functionality and communication complexity of our PC-CONTrace with Catalic, which is shown in Table 4.1. Due to the applying of the Mix-Net system, our PC-CONTrace can also resist linkage attack such as travel route and infection status. With the help of blockchain, PC-CONTrace supports publicly check property while Catilic does not. In addition, Catalic distributes the user's original private data to the cloud servers in the delegated computing phase, so their scheme can only resist the collusion of $\beta - 1$ cloud servers at most. In contrast, our scheme distributes the hash value of the user's original data to the cloud servers, which makes it impossible to get the user's private data even if all the cloud servers collude. In other words, PC-CONTrace can resist the collusion of any number of cloud servers. Catalic takes use of the single-point OPRF in [55] so its communication size is $1.5n$, while our PC-CONTrace's is exactly n.

Scheme	Travel route	Infection Status	Publicly Check	Collusive Cloud	Comunication Size
Catalic	no	no	no	$bounded$	$1.5n$
PC-CONTrace	no	no	yes	$arbitraty$	n

Table 4.1: Comparison with Catalic in functionality and performance

We implement our PC-CONTrace system and compare it with the system of Catalic. We set the cloud number $\beta = 2$ and the back-end server holds the data set with the size 2^{20}. We control the user set's size from 2^{16} to 2^{20} and compare the time cost of Catalic with our PC-CONTrace. The implement result is shown the Figure 4.5. We can observed that under the condition that the number of clouds and the size of back-end server's set remains, the larger the set of user is, the more obvious the advantage of our system will be. Our PC-CONTrace system costs less time than Catalic.

4.4.2.1 Blockchain Execution Analysis

For our PC-CONTrace system, we use Ethereum geth client to test back-end server upload data to blockchain and the user retrieval process. We use Solidity to develop a privacy-preserving contact tracing smart contract, which has a pre-compile phase with Remix-IDE. We execute the ABI code on Ethereum Virtual Machine (EVM). The gas cost of the transaction of PC-CONTrace is shown in Table 4.2, where the number of the cloud servers is two and $N = 2^{10}$. The storage function enables the back-end server to upload the data to blockchain, which costs more gas since it is related to the size of back-end servers set N. The retrieve function allows the user to check the values, which cost less gas.

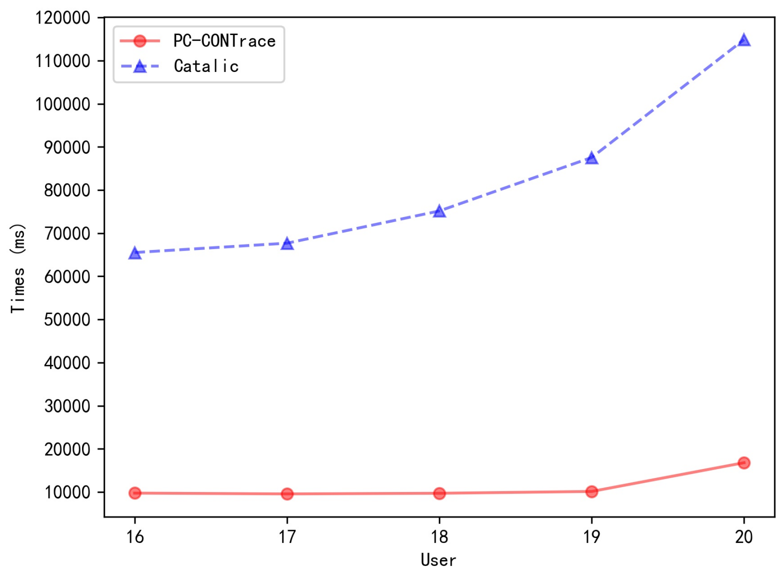


Figure 4.5: The time cost of our PC-CONTrace and Catalic

Function	Transaction cost	Execution cost
PSI contract	277923 gas	208851 gas
Voucher	23908 gas	2620 gas
Storage	2348395 gas	2293860 gas
Retrieve	-	250335 gas

Table 4.2: Gas cost of the transactions of PC-CONTrace

4.5 Summary

In this chapter, we design a lightweight multi-point OPRF based delegated PSI-CA construction, and apply it to build a privacy-preserving contact tracing system named PC-CONTrace. Our system is more adaptable and advantageous in densely populated areas. In addition, we carry on the system implementation of this work, the experimental results show the practicability of our work. The privacy-preserving contact tracing system constructed in this work provides a practical solution to data security and personal privacy issues under the global epidemic prevention and control.

Chapter 5

Privacy-Preserving FL with Secure Aggregation

In the chapter, we propose an accountable and verifiable secure aggregation for federated learning protocol (SA-FL), which guarantees the verifiability of data provenance and is accountable for malicious client. The background of privacy-preserving federated learning is described in Section 5.1. The system model is described in Section 5.2. The concrete construction of SA-FL and security analysis are given in Section 5.3. The performance evaluation is described in Section 5.4. Finally, we conclude the chapter in Section 5.5.

5.1 Background

With the development of IoT and edge computing, large-scale IoT devices are connected to internet to collect users' data. Artificial intelligence (AI) and machine learning (ML) have been widely applied in various fields of human society to analyze massive data and improve server quality (QoS). Google, IBM, Amazon and Alibaba are all actively exploring AI and ML technology and building relevant platforms to improve its services and make market decisions [45].

Traditional machine learning approaches usually rely on centralized management and use the cloud platforms for the model training, so they are also known as cloud-centric machine intelligence. In the cloud-centric machine intelligence model, as shown in Figure 5.1, the client interacts with the cloud and generates logs which can be used as training examples. The client collects the data and sends its data to the cloud platform, the cloud further optimized the training model. At last, the cloud returns the training model to clients as feedbacks which is also used to serve future client requests. The cloud-centric machine intelligence model typically collects large amounts of client data, which increases the risk of privacy violations, such as the Facebook and Cambridge Analytica event [1] and the Wyze data leak event [2]. Facebook provides its platforms data to Cambridge Analytica company for making data analysis, which leaked the data of more than 50 million people. Wyze, a provider of IoT security devices, suffered a server breach that exposed the details of over 2.4 million customers in 2019. It can be seen that data privacy protection is increasingly important when performing data analysis.

To protect the data security, Google introduced the conception of federated learning [19], which imbues mobile devices with advanced machine learning systems and the data dose not need to be centralized. It collaboratively trains the global model by utilizing large-scale distributed devices while protecting the privacy of the participants' local data sets. In the federated learning model, clients,

[1]https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal

[2]https://www.geekwire.com/2019/wyze-data-leak-key-takeaways-server-mistake-exposed-information-2-4m-customers/

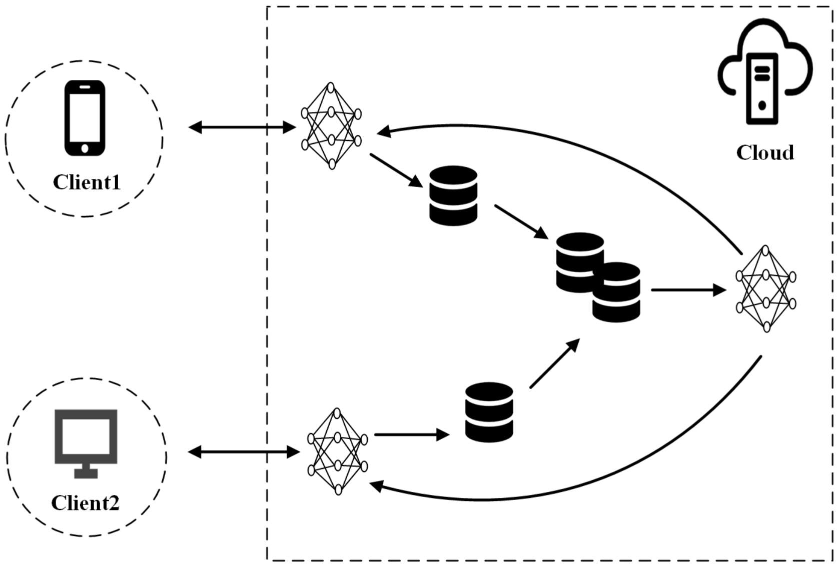


Figure 5.1: The cloud-centric machine intelligence model

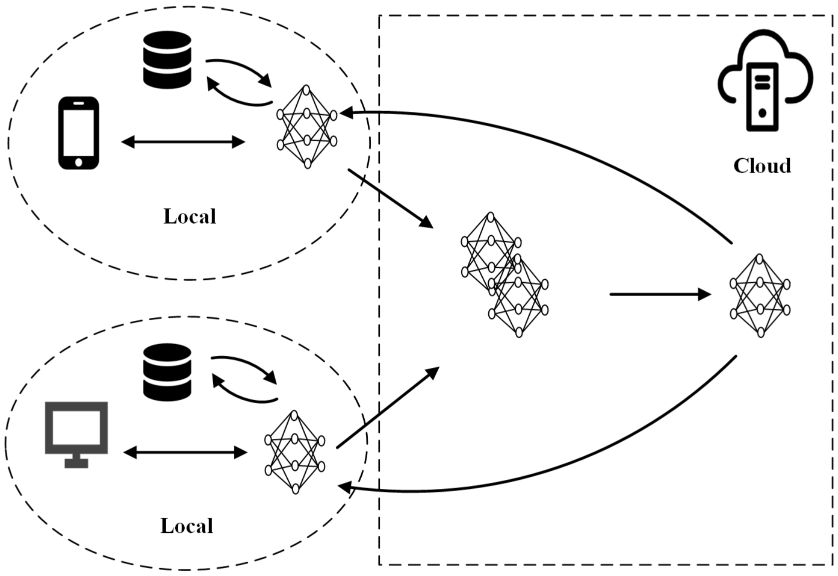


Figure 5.2: Federated learning

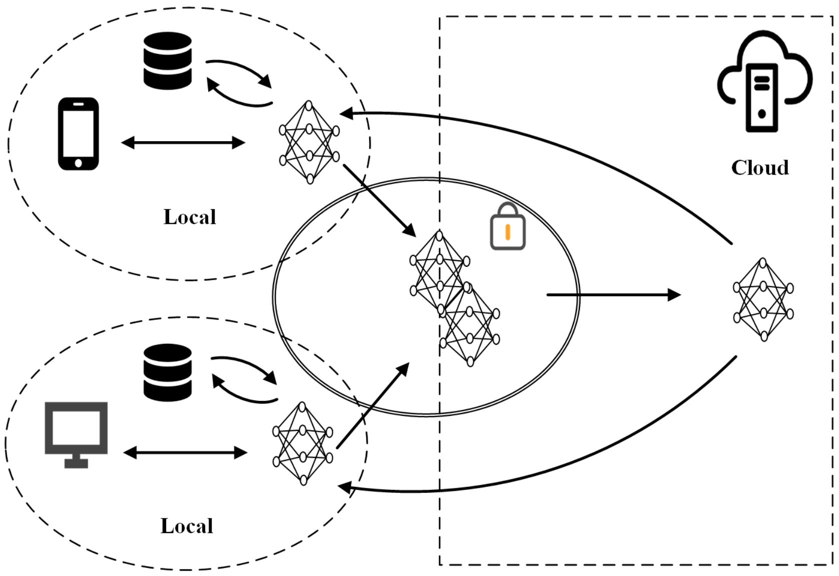


Figure 5.3: Federated learning with secure aggregation

coordinated by a central cloud server, can use their local data sets to cooperatively train a machine learning model without revealing their respective training data sets, as shown in Figure 5.2. A training round in federated learning consists of three phases. First, the cloud server constructs a global machine learning model and sends the model to a group of clients. Second, after receiving the model, the clients train local models with their own data sets and shares the updates of the improved models with the cloud server. Finally, the server aggregates the improved models into a new global model. In the federated learning with secure aggregation model, as shown in Figure 5.3, the aggregation of the model updates is achieved by means of secure multiparty computation, which makes the cloud server learn only the aggregated updates.

Many schemes have been given successively, since the concept was introduced. However, the existing schemes lack validated authentication of the client data, which is of significance in IoT scenarios. In addition, there is no systemic solution for how to confidentially provide the computation result and how to ensure the verifiability of the computation, which is also desirable for federated learning research.

5.1.1 Contributions

To against the malicious clients and guarantee the confidentiality and verifiability of the secure aggregation, we propose an accountable and verifiable secure aggregation for federated learning framework in IoT networks. In the framework, the server is divided into two roles, one of which is the cloud server and the other is the aggregator. The cloud sever verifies the aggregation results and executes the global model updates. The aggregator executes the transformation of the ciphertexts received from each client and secure aggregates the transformed ciphertexts.

We employ a MPC protocol to protect the confidentiality and verifiability of data, which is based on homomorphic proxy re-authenticators and homomorphic proxy re-encryption. The homomorphic proxy re-authenticator guarantees the integrity of the client data and the authenticability of the model updates provenance, while the homomorphic proxy re-encryption scheme ensures the data security and protects the local data confidentiality of each client. Our scheme ensures that the individual local model updates are not leaked to the aggragator and the cloud server only receives the aggrageted ciphertext. In addition, the framework is flexible and dynamically adjustable for participants dropouts. We integrate the blockchain to build the federated learning framework and use the smart contract of blockchain to trace the malicious behavior clients. To demonstrate the useability of our framework, we evaluate the specific cryptography schemes and develop a blockchain-based prototype system to test the performance of the framework.

5.2 System Framework of SA-FL

We now describe the framework of our accountable and verifiable secure aggregation for federated learning.

5.2.1 The Framework Details

The framework is composed of four entities: Client (IoT devices), Server, Aggregator, Blockchain. The framework includes five phases and the Figure 5.4 shows the details:

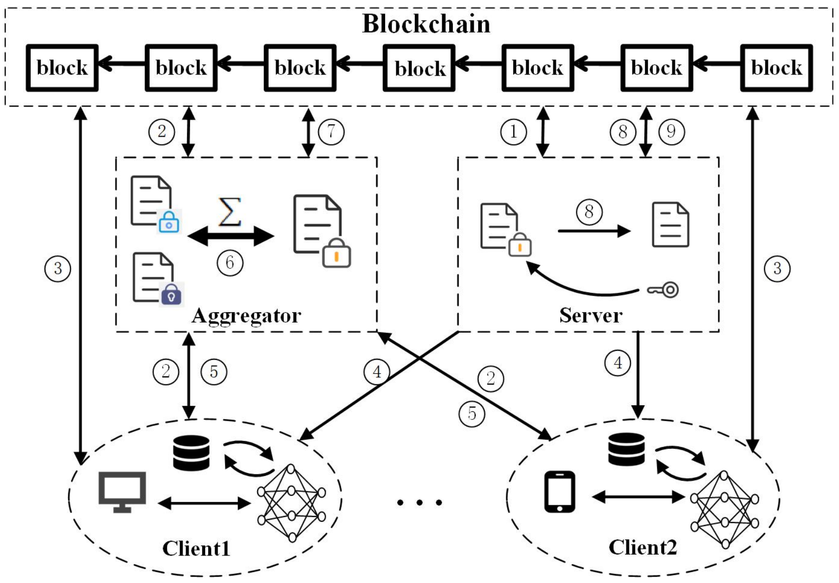


Figure 5.4: The framework of our SA-FL

- **Server**: Server generates the system parameters, deploys a privacy-preserving federated learning smart contract on blockchain, sends each client the global model and executes the model updates. It also traces the malicious client.

- **Client**: Each client registers to the aggregator and calls the smart contract to publish a deposit transaction on blockchain. The client uses its local data sets to collaboratively performs the model updates and sends ciphertexts of the updated parameters to the aggregator.

- **Aggregator**: The aggregator selects the client set and transforms the client's ciphertexts. The aggregator executes the secure aggregation, during which the aggregator cannot learn anything about the data of the client and the server. It constructs a transaction of the aggregated ciphertext and publishes it on blokchain.

- **Blockchain**: The blockchain records the transactions of each party. We assume the blockchain is a secure decentralized infrastructure and has sufficient miners to maintain the update of blockchain.

5.2.2 The General Processing of SA-FL

1. Setup. The server initializes the system and develops the smart contract for the task model. Then, the server deploys smart contracts on blockchain. The client registers to the aggregator and the aggregator selects a set of client and publishes the set on blockchain. The client checks the blockchain and calls the smart contract to submit its deposit on blockchain.

2. Training. The server sends the global training model to the client who pays the deposit on the blockchain. When the client receives the global training model, it executes training on local data sets and generates the model updates. Then, the

client encrypts the updated parameters and sends the ciphertext to the aggregator, At the same time, the client sends a transaction to blockchain.

3. Aggregation. The aggregator receives the ciphertexts from the clients and executes the secure aggregation on these ciphertexts. Then, it generates a transaction for the aggregated ciphertexts and sends a transaction to blockchain.

4. Update. The server checks the transaction on the blockchain and verifies the integrity of the ciphertexts. Then, the server uses its secret key to decrypt the aggregated ciphertexts and obtains the aggregated update.

5. Tracing. The server checks the updated model and interacts with the aggregator to trace the client. If the client has the malicious behavior, the server will punish the client by deducting its deposit. Then, turn to next round.

5.2.3 Design Goals

The accountable and verifiable secure aggregation for federated learning has the following goals:

Privacy: The data of client and the aggregated data cannot be leakaged to other participants, including the other clients, the aggregator. The server only learns the aggregated results.

Authentication: A server can check the reliability of the data sources and verify the integrity of the client data's ciphertext.

Unforgeability: A malicious client should be unable to impersonate other honest clients to forge the signature of honest clients and the aggregator cannot forge the signature of the honest clients, which is not generated by them.

Accountability: A server can trace the malicious client who may random drop out and submit fake local updates. The server can punish the client by deducting its deposit.

5.3 A Concrete Construction

We present a concrete construction of accountable and verifiable secure aggregation for federated learning in this section, which is based on the HPRE, HPRA and blockchain. We assume that each party has an account address in blockchain network. They can deploy the smart contract and publish the transaction on blockchain. Homomorphic proxy re-encryption guarantees the privacy of the client data and the aggregated data and homomorphic proxy re-authenticator guarantees the integrity and verifiability of the client data. With the smart contract, the server can trace the malicious clients.

5.3.1 The Scheme of SA-FL

The process of accountable and verifiable secure aggregation for federated learning is illustrated in Figure 5.5.

- Setup: The server initializes the system and generates the parameters of HPRE and HPRA. (1)The server inputs the security parameters λ to generate a type-1 pairing group $BP = (q, g, \mathbf{g} = e(g, g), \mathbb{G}, \mathbb{G}_T, e)$ where g is a generator of $\mathbb{G}$ and **g** is a generater of $\mathbb{G}_T$. There is a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Set $H_0 : \{0,1\}^* \to \mathbb{G}$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ are collision-resistant hash functions and there is a function $f : \{0,1\}^* \to \{w_i, i \in 1, \cdots, n\}$. The server outputs the parameters $pp = (BP, H_0, H_1, f)$. Then, the server calls

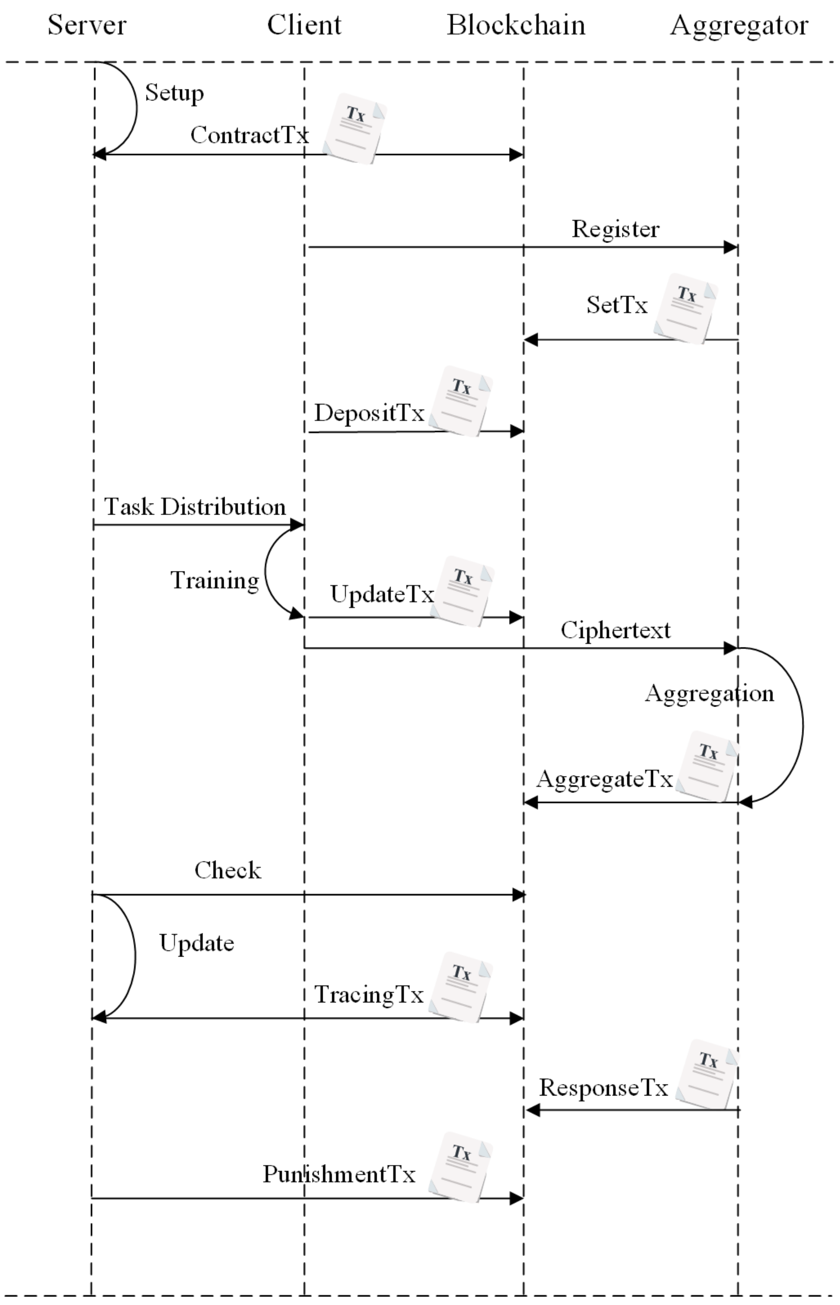


Figure 5.5: The process of our SA-FL

HPRE.Keygen algorithm. It randomly selects $a_1, a_2 \in \mathbb{Z}_q$ and computes $rpk_1 = \mathbf{g}^{a_1}, rpk_2 = g^{a_2}$ and set $sk = (a_1, a_2)$. It randomly selects $\alpha \in \mathbb{Z}_q$ to generate the public authenticated key $pmk = g^{\alpha}$. The server develops a federated learning task contract (FL-Contract), which includes the description of the task model and the system parameters $(pp, rpk = (rpk_1, rpk_2), pmk)$. The server generates a contract transaction $ContractTx$ and publishes it on blockchain.

(2)Using the system parameters, the client gets its public and private keys by calling HPRE.Keygen and HPRA.Keygen algorithms. The client i randomly selects $b_{i,1}, b_{i,2} \in \mathbb{Z}_q$ and computes $rpk_{i,1} = \mathbf{g}^{b_{i,1}}, rpk_{i,2} = g^{b_{i,2}}$. Then, it randomly chooses $x_i \in \mathbb{Z}_q$ and computes $cpk_{i,1} = g^{x_i}$. Set $sk_i = (b_{i,1}, b_{i,2}, x_i)$ and $pk_i = (rpk_{i,1}, rpk_{i,2}, cpk_{i,1})$. The client registers to the aggregator by submitting its public key information pk_i.

(3)The aggregator checks the clients and generates a set list L, which includes n clients. Then, the aggregator constructs the set transaction $SetTx$ with the list L and publishes $SetTx$ on blockchain.

(4)The client calls the federated learning task contract and submits its deposit transaction $DepositTx$ to blockchain.

- Training: (1)The server checks the transaction $SetTx$ and calls HPRA.VGen algorithm. For $pk_i \in L$, parse $pk_i = (rpk_{i,1}, rpk_{i,2}, cpk_{i,1})$, then, the server sends the global training model M to each client in list L by using the security channel.

  (2)The client receives the global training model M and executes training on its local data sets. The client calls HPRE.RKeygen to generate its transformation key. It pares $sk_i = (b_{i,1}, b_{i,2}, x_i)$ and $rpk = (rpk_1, rpk_2)$. It computes transformation key $rk_i = (rpk_2^{b_{i,1}})$ and calls $\mathsf{HPRE.Enc}$ algorithm to encrypt the updated parameters m_i. The client randomly selects $k_i \in \mathbb{Z}_q$ and computes $c_i = (c_{i,1}, c_{i,2}) = (g^{k_i}, \mathbf{g}^{m_i} \cdot rpk_{i,1}^{k_i})$. Then, it calls $\mathsf{HPRA.Sign}$ algorithm to sign on the update parameters. The client parses $pk_i = (rpk_{i,1}, rpk_{i,2}, cpk_{i,1})$. Then, it computes $\sigma_i = (H_0(L||cpk_{i,1}) \cdot g^{m_i})^{x_i}$. The client calls $\mathsf{HPRA.VRGen}$ algorithm to generate the corresponding rotation key $ak_i = (g^{\alpha})^{1/x_i}$. Using the signature σ_i and the ciphertext c_i, the client constructs the transaction of the updates $UpdateTx$, and publishes $UpdateTx$ to blockchain. At the same time, the client sends the ciphertexts and its transform key rk_i and rotation key ak_i to the aggregator.

- $\mathsf{Aggregation}$: The aggregator receives the ciphertext from each client in the list L. It checks the $UpdateTx$ transaction of the clients. Then, the aggregator calls $\mathsf{HPRE.REnc}$ algortihm to generate the re-encryption ciphertext by using the transformation key rk_i. The aggregator parse $c_i = (c_{i,1}, c_{i,2})$ and computes $c_{i,r1} = e(c_{i,1}, rk_i), c_{i,r2} = c_{i,2}$. Set $c_{i,re} = (c_{i,r1}, c_{i,r2})$. Then, it calls $\mathsf{HPRE.EV}$ algorithm to execute the secure aggregation on the re-encryption ciphertext $c_{i,re}$. The aggregator computes $F(c_1, \cdots, c_n) = (w_1, \cdots, w_n)$, $c_{ra} = \Pi_{i=1}^{n} c_{i,r1}^{w_i}, c_{rb} = \Pi_{i=1}^{n} c_{i,r2}^{w_i}$. Then, the aggregator calls $\mathsf{HPRA.Agg}$ algortihm to generate the re-authenticator for the signatures received from each client with the rotation key ak_i, where $\mu = \Pi_{i=1}^{n} e(\sigma_i^{w_i}, ak_i)$. The server only receives the aggregated re-encryption ciphertext (c_{ra}, c_{rb}), which guarantees the privacy of the client data, while the re-authenticator ensures that the ciphertext is calculated from the data of clients in L. The aggregator constructs an aggregation transaction $AggregateTx$ and publishes $AggregateTx$ on blockchain.

- Update: The server checks the $AggregateTx$ and receives the re-encryption ciphertexts (c_{ra}, c_{rb}). The server decrypts the aggregated ciphertexts (c_{ra}, c_{rb}) by calling HPRE.Dec$_1$. It parses $sk = (a_1, a_2)$ to computes $\mathbf{g}^m = c_{rb} \cdot c_{ra}^{-1/a_2}$ and gets the model updates m. Then the server checks the transaction $AggregateTx$ and verifies the reliability of the client data by using the secret authenticated key $smk = \alpha$ of proxy re-authenticator. The server calls HPRA.AVerify to verify authenticator. It computes $\mu' = (\Pi_{i=1}^n e(g^{w_i}, H_0(L||cpk_{i,1}))\cdot e(\Pi_{i=1}^n g^{m_i}, g))^\alpha$. It checks $\mu' \overset{?}{=} \mu$.

- Tracing: The server checks the model updates m and detects the update exceptions. The server interacts with the aggregator to trace a malicious client. (1)The server first generates challenge set for the client, such as the index of user identity (1,2,3,4) in List L. Then, it constructs a tracing transaction $TracingTx$ and publishes it to blockchain.

  (2)The aggregator checks the challenge set and calls HRPE.EV algorithm to execute the secure aggregation on re-encryption ciphertexts of index in challenge set. The aggregator computes $f(c_1, \cdot\cdot\cdot, c_4) = (w_1, \cdot\cdot\cdot, w_4)$, $c_{ra} = \Pi_{i=1}^4 c_{i,r1}^{w_i}, c_{rb} = \Pi_{i=1}^4 c_{i,r2}^{w_i}$. It generates a $ResponseTx$ transaction and publishes the transaction $ResponseTx$ on blockchain.

  (3)The server verifies the response transaction and get the updated model m for (1,2,3,4). Then, it replays the process and regenerate the challenge set (1,2,3,4,5) in List L. The aggregator responses the request. This process can be performed in multiple rounds. The server relies on the random challenge and the updated model to assert the behaviour of client. After that, the server generates a $PunishmentTx$, which includes the client identification and the proof of challenge-response. Finally, the server puts $PunishmentTx$ on blockchain.

5.3.2 Security Analysis

In the section, we analyze the security of accountable and verifiable secure aggregation for federated learning protocol, which achieves the privacy, authentication, unforgeability and accountability. Specifically, we assume the server is semi-honest and it cannot collude with the aggregator.

Privacy: In the protocol, we protect the privacy of data of each client, which is hold by client itself. We also protect the privacy of the aggregated data, which cannot be leakaged to the other clients or the aggregator, but it is transparent to the server. For the client data m_i, we execute the HPRE scheme with $\mathsf{HPRE.Enc}$ algorithm to encrypt the data m_i. The confidentiality of HPRE scheme guarantees the security of client data m_i. The HPRE scheme is an IND-CPA secure homomorphic proxy re-encryption, which is proved achieve the IND-CPA security with the eDBDH assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. The confidentiality of HPRE scheme guarantees the privacy of data of each client. For the aggregated data, the aggregator calls $\mathsf{HPRE.REnc}$ algortihm to generate the re-encryption ciphertexts with the transformation key rk_i. In the process, the client calls $\mathsf{HPRE.RKeygen}$ to generate its transformation key, which is sent to the aggregator. The aggregator executes re-encryption and secure aggregation on the re-encryption ciphertext $c_{i,re}$. The IND-CPA secure homomorphic proxy re-encryption scheme guarantees the confidentiality of the aggregated data. So the protocol guarantees the privacy of data of each client and aggregated data.

Authentication: In the protocol, we protect the reliability of the data sources. In the protocol, the client generates its public key and private key and registers to the aggragator. Then, the aggregator generates a register set list L and constructs the set transaction $SetTx$. It publishes $SetTx$ on blockchain. The publicly verifiable and tamper-resistant of blockchain guarantees the security of L. The client sends the rotation key to the aggregator with security channel. Then, the client calls HPRE.Enc to generate the ciphertexts of data and calls HPRA.Sign algorithm to sign on the data. In the aggregation phase, the aggregator calls HPRA.Agg algorithm to generate the re-authenticator for the signatures received from each client with the rotation key, $\mu = \Pi_{i=1}^{n} e(\sigma_i^{w_i}, ak_i)$. The server calls HPRA.AVerify to check the authenticator. The security of HPRA guarantees the reliability of the data sources. Based on the re-authenticator and the client signature, the protocol provides the authentication of data sources.

Unforgeability: In the protocol, an honest client signature is unforgeable. In the protocol, the client generates its public key and private key and executes federated learning training model at local. For an update parameter m_i, the client calls HPRA scheme to sign on it. The unforgeability of HPRA scheme guarantees the unforgeability of client signature. Specifically, the signer unforgeability of HPRA scheme guarantees the malicious client cannot impersonate other honest clients. The aggregator unforgeability of HPRA scheme guarantees the aggregator cannot forge the signature of the honest clients. The HPRA scheme is unforgeable in eBCDH assumption under the random oracle model. So, the protocol is unforgeable for the malicious client and the aggregator.

Accountability: In the protocol, an honest server can trace the malicious client and punishes it by deducting its deposit. In the tracing phase, the server interacts with the aggregator to execute the challenge-response protocol. The server can sample the client list to generate the challenge set and constructs a tracing transaction $TracingTx$. With the $TracingTx$, the aggregator checks the challenge set and responses the secure aggregation on re-encryption ciphertexts. The aggregator generates a $ResponseTx$ transaction and publishes it on blockchain. The server verifies the response and gets the updated model. Then, the server interacts with the aggregator in multiple rounds. The server relies on the random challenge and the updated model to assert the behaviour of client. After that, the server generates a $PunishmentTx$ and publishes it on blockchain. With the $PunishmentTx$, the smart contract executes deducting client deposit operation. So, the protocol achieves accountability for the malicious client.

5.4 Performance Analysis

We implement the performance evaluation of our SA-FL protocol in this section. We call the specific cryptography algorithms and evaluate its time cost. Then, we deploy a smart contract on blockchain to evaluate the functions of the accountable and verifiable secure aggregation for federated learning and test its efficiency.

5.4.1 Implementation Analysis

Our implementation is based on the AMD Ryzen 9 5900HS with Radeon Graphics 3.30GHz, 16.00GB of RAM with Ubuntu, we use the C++ programming language to develop the cryptography schemes programming. For the pairing library, we select the BN curve for bilinear pairing. We also test the time consumption of the cryptography schemes in a Raspberry Pi to simulate IoT device. The Raspberry Pi 3 Model B is equipped with quad-core 64-bit CPU and 1 GB RAM. We execute the protocol to test its time cost and execute 1000 rounds for each algorithm to get the average running time. The time cost of the client and the server in the Setup, Training and Update phase are shown in Table 5.1, and the time cost of each algorithm less than 1 second.

Phase	Raspberry Pi		Laptop	
	Client	Server	Client	Server
Setup	101ms	263ms	9.44ms	36.75ms
Training	367ms	166ms	17.78ms	5.91ms
Update	-	463ms	-	32.76ms

Table 5.1: Time cost of raspberry pi and laptop

We compare our construction with other scheme in terms of the functionality, which is shown in Table 5.2. [19] is weak for the dropouts of client. SA-FL and [8] support the random dropouts for client. SA-FL is authenticated for the data of client identify. The communication of Bonawitz et al. [19] is $O(mn + n^2)$, Awan et al. [8] is $O(mn)$ and SA-FL is $O(mn)$, where m is the size of the data vector and n denotes the client number. We further execute the algorithm of aggregation operation, the aggregator first executes the ciphertexts transformation, then executes secure aggregation for the ciphertexts. We test the time cost of the aggregation in terms of the number of the client, which is shown in the Figure 5.6. Experimental results show that for 100 clients, the [19] costs 8.50ms, [8] costs 5.6ms aggregating the ciphertexts. We also check the aggregation operation under similar condition where no signature algorithm is executed, so our scheme costs 5.5ms.

Scheme	Dropout	Privacy	Authentication	Accountability
Bonawitz et al [19]	bounded	√	√	×
Awan et al [8]	arbitrary	√	×	√
Our SA-FL Scheme	arbitrary	√	√	√

Table 5.2: Functionality comparison

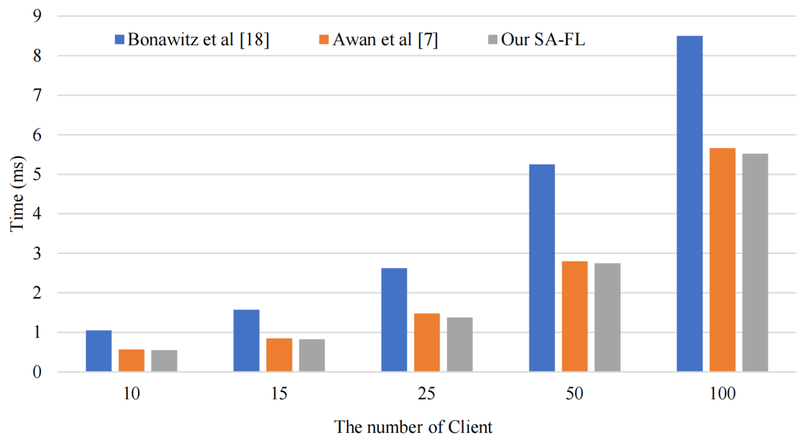


Figure 5.6: The time cost of aggregation operation for different schemes

5.4.2 Experiments and Evaluation

In this section, we conduct simulation experiments to evaluate the prediction performance of the global model. All simulations are implemented on the same computing environment (Windows 10, Intel (R) Core (TM) i5-8400 CPU @ 2.80GHz, NVIDIA GeForce RTX 3070, 16GB of RAM and 4T of memory) with Tensorflow, Keras.

Experimental Settings

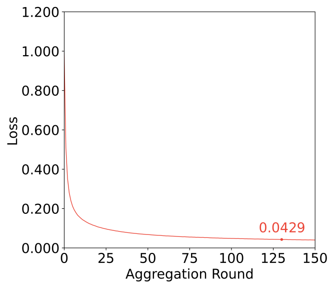
Dataset. In our experiments, we leverage the widespread handwritten digits dataset, MNIST, which contains a training set of 60000 samples and a testing set of 10000 samples. Every sample has been size-normalized and centered in a fixed-size (28 × 28) gray-level image depicting an item from one of ten different labels.

Machine Learning Task. The ML model (deep neural network DNN) is the Convolutional Neural Network (CNN), which contains 2 convolution layers, 2 pooling layers, and 2 fully connected layers. We also set the loss function as the *cross-entropy error* and the active function as the *sigmoid*. Besides, the gradient decent algorithm is set as the stochastic gradient descent (SGD) with the learning rate of 0.001.

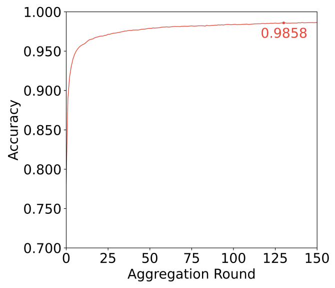
FL Settings. The FL system under our consider consideration involves 50 clients with local datasets extracted from the MNIST dataset. For each local dataset, we randomly select 1200 samples with an equal probability and without replacement. Besides, the central server aggregates local contributions (models) every 5 epochs of local training. In addition, the MNIST testing set is directly used for evaluating the global model performance.

Evaluation

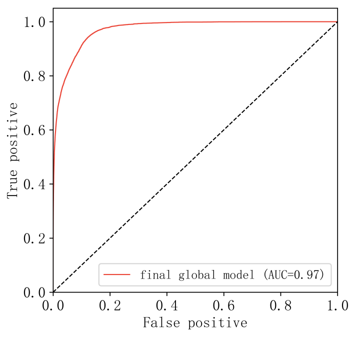
As shown in Figure 5.7(a)(b), after 131 rounds of aggregation, the global model converges. The final loss value and the prediction accuracy on the MNIST testing set are 0.0429 and 98.58%, respectively, which means the global model could classify the testing samples with high accuracy. Also, Figure 5.7(c) illustrates the receiver operating characteristic (ROC) curve and the area under the curve (AUC) of the final converged global model on the testing set. As we can observe, the AUC is 0.97, which reflects an outstanding model generalization the final converged global model achieves. In summary, with the deployment of the proposed FL system, clients could collaboratively train a global model a with rapid convergence rate, high prediction accuracy, and good model generalization.



(a) Loss curve.



(b) Accuracy curve.



(c) ROC curve.

Figure 5.7: Performance analysis

5.4.3 Prototype System

We develop a prototype system of the accountable and verifiable secure aggregation for federated learning. We utilize the Ethereum geth client to build the blockchain network. We use the Solidity to develop the federated learning task contract and pre-compile it with remix IDE, then we test the gas cost of the contract function when generating the transactions. We evaluate the secure aggregation of the aggregator in terms of ten clients, and the estimate of the gas cost is shown in Table 5.3. The $SetTx$ and $TracingTx$ transactions only contain the set of clients identifies and the tracing challenge respectively, so the gas cost of $SetTx$ and $TracingTx$ transactions are low. For a deposit transaction, the client commits to provide real data, and the $DepositTx$ costs 144529Gas. As shown in the Table 5.3, the aggregator uploads the ciphertexts of the secure aggregation, which costs 1233720Gas. In the $Tracing$ phase, the aggregator uploads the selected ciphertexts to response the challenge from the Server, so the $ResponseTx$ costs vast gas. The $PunishmentTx$ uploads the parameters and the tracing data, which costs 357626Gas.

Transaction	Gas units	Gas cost(ether)
ContractTx	727904	0.01455808
SetTx	43724	0.00087448
DepositTx	144529	0.00289058
UpdateTx	456367	0.00912734
AggregateTx	1233720	0.02467440
TracingTx	125677	0.00251354
ResponseTx	5735890	0.1147178
PunishmentTx	357626	0.00715252

Table 5.3: Gas cost of transactions

5.5 Summary

In this chapter, we propose an accountable and verifiable secure aggregation for federated learning, we realize the protection of the confidentially and verifiability of the client data base on HPRA, HPRE and blockchain. In addition, our framework is flexible and dynamically adjustable for the participants dropouts issue. To illustrate the useability of our framework in IoT networks, we also test the performance of it base on blockchain prototype system.

Chapter 6

Anonymous and Publicly Linkable Reputation System

In the chapter, we propose an anonymous and publicly linkable reputation system with distributed trust (DTrustRS), which protects the anonymity of user while executing distributed regulation by revoking the anonymity of misbehavior users. The motivation of DTrustRS is described in Section 6.1. The architectural model is described in Section 6.2. The concrete construction of aggregatable PS signature is given in Section 6.3. The concrete construction of DTrustRS is described in Section 6.4. The performance evaluation is described in Section 6.5. Finally, we conclude the chapter in Section 6.6.

6.1 Background

Reputation system allows the users to rate transactions or services they have previously enjoyed. The rating data is being generated every day. More and more companies are committed to mining behavioral data to extract valuable information for business benefit. However, the users' privacy is thus under the risk of being exposed to third-parties. Anonymous reputation system not only enables a user to rate its services, but also protects user's privacy, which makes it widely concerned. In the research of anonymous reputation system, the anonymous and publicly linkable reputation system has attracted much attention since it can provide more reliable ratings. However, an anonymous and publicly linkable reputation system generally requires strong trust in one single system manager who can revoke the anonymity of misbehavior users if necessary. Once the system manager is corrupted, it can easily trace all the users to compromise their privacy and further impact the reputation of the system. It can be seen that it is necessary to reduce the trust in single authorities and further enhance the robustness of the reputation systems.

Some of the security properties of anonymous and publicly linkable reputation systems are studied on the conception of group signatures [29]. However, group signatures cannot meet all the requirements of anonymous and publicly linkable reputation systems in distributed settings. Group signatures require strong trust in the single issuer and the single opener. We need to find a way to deal with the single trust in anonymous reputation systems while guaranteeing public linkability, that is, be able to assert two ratings for the same service were generated by the same user, to improve the security level of the system. However, the existing construction can not provide them simultaneously.

6.1.1 Contributions

In this chapter, we propose an anonymous and publicly linkable reputation system with distributed trust (DTrustRS). We define the system model of DTrustRS, which initializes with n_I service providers (SP) and n_O system managers (SM), who are involved in adding registered users and opening user signatures, respectively. We formally define the security of DTrustRS which includes anonymity, traceability, and public linkability. The anonymity guarantees that DTrustRS does not reveal the identity of the user. The traceability guarantees that DTrustRS is unforgeable. The public linkability guarantees that anyone can assert two rate signatures for the same statement created by the same user.

We give an efficient construction for DTrustRS. To achieve the registration function of DTrustRS, we propose a new variant of the Pointcheval-Sanders (PS) signature named aggregatable PS signature (APS). We construct the APS scheme and prove its security, which is an independent interest. The construction of DTrustRS follows the re-randomizable signatures paradigm in [13]. We integrate APS and Pointcheval-Sanders Multi-signatures (PSM) to construct the DTrustRS scheme. The signature of DTrustRS is short in length, including $3\mathbb{G}_1$ elements and $3\mathbb{Z}_q$ elements. We prove that DTrustRS is secure in the random oracle model under a q-type assumption. Finally, we demonstrate DTrustRS's validity by comparing it with related works.

6.2 Architectural Model

In this section, we define the system model of DTrustRS and its security requirements.

6.2.1 System Model

The DTrustRS includes three entities: the system managers (SM), users and service providers (SP), as shown in Figure 6.1.

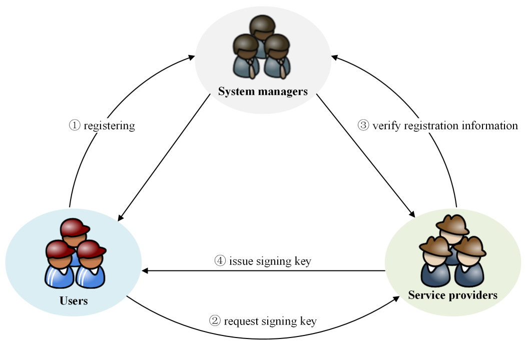


Figure 6.1: The system model of DTrustRS

SM: The system manager authorizes user as a legal user. Each SM issues one credential to legal user and reveals the identity of user.

Users: Each user interacts with SM to register as a legal user. Then, it provides the registration information to service providers and sends a signing key request to a set of service providers. User collects a number of shares and aggregates these shares to form an anonymous reputation for the services of service providers.

SP: The service provider interacts with the SM to verify the registration information of the users. Then, each service provider issues a signing key to user, which allows the user to remark the reputation.

6.2.2 The Syntax of DTrustRS

The anonymous and publicly linkable reputation systems with distributed trust (DTrustRS) is an extension of the anonymous and publicly linkable reputation systems. DTrustRS distributes the role of single authority over several entities.

We initialize it with n_I SP and n_O SM. The DTrustRS scheme Π consists of Setup, KeygenI, KeygenO, KeygenAggr, $\langle$ RegisterO, RegisterU $\rangle$, $\langle$ JoinU, IssueI $\rangle$, Sign, Verify, Open, Judge and Link algorithms.

- $(params) \leftarrow \mathsf{Setup}(1^\lambda, n_I, n_O)$: This algorithm takes a security parameter λ, the number of SP n_I and the number of SM n_O as input and outputs the public parameters $params$.

- $(isk_i, ipk_i, st_i) \leftarrow \mathsf{KeygenI}(params, i \in n_I)$: Service provider runs the algorithm, which takes the public parameters $params$ and the identify i as input and generates a key pair (isk_i, ipk_i) of i with a statement st_i.

- $(osk_i, opk_i, reg_i) \leftarrow \mathsf{KeygenO}(params, i \in n_O)$: System manager runs the algorithm, which takes the system parameters $params$ and the identify i as input, and generates a key pair (osk_i, opk_i) of i and a registration list reg_i.

- $(apk) \leftarrow \mathsf{KeygenAggr}(params, \langle pk_1, \cdots, pk_n \rangle)$: This algorithm takes the public parameters $params$ and a vector of public key $(pk_1, \cdots, pk_n)$ as input to generate an aggregated key pair (apk).

  The system parameters gpk consists of (ipk_i, apk_O), where apk_O is the aggregated key of SM.

- $\langle\, Reg[id], \{reg_i\}_{i=1}^{n_O} \,\rangle \leftarrow \langle\, \mathsf{RegisterU}(params, id, gpk) \rightleftharpoons \{\, \mathsf{RegisterO}(params,$ id, osk_i, opk_i $\}_{i \in n_O}$ $\rangle$: An user with identity id interacts with n_O SM. The user inputs $params$, id and gpk and each SM inputs $params, id$ and its key pair (osk_i, opk_i). Output a register credential $Reg[id]$ and updated the registration list reg_i for each system manager.

- $(gsk[id]) \leftarrow \langle \mathsf{JoinU}(params, Reg[id], gpk) \rightleftharpoons \{\mathsf{IssueI}(params, id, isk_i, ipk_i\}_{i \in n_I} \rangle$: An user with identity id interacts with n_I SP. The user inputs $params$, $Reg[id]$ and gpk and each SP inputs $params, id$ and its key pair (isk, ipk). Output the signing secret key $gsk[id]$.

- $(\sigma) \leftarrow \mathsf{Sign}(params, gpk, gsk[id], M, item)$: A sign user takes the public parameters $params$, gpk, a signing secret key $gsk[id]$, a message M and a specified $item$ as input, and outputs a signature σ over message M.

- $(b) \leftarrow \mathsf{Verify}(params, gpk, \sigma, M, item)$: This algorithm is run by a verifier who wants to verify a signature σ. It takes $params$, gpk, σ, the message M and $item$ as input, and outputs a bit $b \in \{0, 1\}$.

- $\langle \{id, \pi_i\}_{i \in n_O} \rangle \leftarrow \langle \{\mathsf{Open}(params, gpk, item, (\sigma, M),\ reg_i, osk_i)\}_{i \in n_O} \rangle$: This algorithm is run by the SM. Return the identify of user identity who computed a signature σ on M with a proof π.

- $(b') \leftarrow \mathsf{Judge}(params, gpk, item, (\sigma, M), \langle \{id, \pi_i\}_{i \in n_O} \rangle)$: This algorithm verifies the proof in Open algorithm. It takes $params$, id, $item$ and the proof π and gpk as input, and outputs a bit $b' \in \{0, 1\}$.

- $(b'') \leftarrow \mathsf{Link}(params, gpk, item, (M, \sigma), (M', \sigma'))$: This algorithm checks two signatures (M, σ) and (M', σ') whether computed by the same user. It takes $params$, gpk, $item$, (M, σ) and (M', σ') as input, and outputs a bit $b'' \in \{0, 1\}$.

Correctness. A DTrustRS must satisfy the correctness.

For all $(params) \leftarrow \mathsf{Setup}(1^\lambda, n_I, n_O)$, for all $(isk_i, ipk_i, st_i) \in \mathsf{KeygenI}$, $(osk_i, opk_i, reg_i) \in \mathsf{KeygenO}$, $(apk) \leftarrow \mathsf{KeygenAggr}$, $\langle Reg[id], \{reg_i\}_{i=1}^{n_O} \rangle \in$

$[\mathsf{RegisterU}] \times [\mathsf{RegisterO}]$, $(gsk[id]) \in [\mathsf{JoinU}] \times [\mathsf{IssueI}]$, $\langle \{id, \pi_i\}_{i \in n_O} \rangle \in \mathsf{Open}$:

$\mathsf{Verify}(params, gpk, \mathsf{Sign}(params, gpk, gsk[id], M, item), M, item)) = 1 \wedge$ $\mathsf{Judge}(params, gpk, item, (M, \sigma), \langle \{id, \pi_i\}_{i \in n_O} \rangle) = 1 \wedge \mathsf{Link}(params, gpk, item,$ $(M, \sigma), (M', \sigma')) = 1.$

6.2.3 Security Model

The security requirement of DTrustRS is similar to the threshold dynamic group signatures [25], but DTrustRS is distributed opening and publicly linkable.

In the security experiments for DTrustRS, the challenger maintains global variables which include system parameter gpk, honest users signing key gsk, a registration table reg and

$-CU$ denotes corrupted user set, and a corrupted user is controlled by adversary.

$-RU$ denotes the set of user in the registration phase.

$-JIU$ denotes the set of user in the join phase.

$-Q_{Sign}$ is a set of signing queries $(id, M, item)$.

$-Q_{Open}$ is a set of opening queries $(\sigma, M, item)$.

Set is initialized to be empty and gpk, gsk and reg are initialized to be $\perp$.

Oracle. In the security experiments, we use the following oracle.

GSK(id). The oracle returns the secret signing key $gsk[id]$ and adds id to CU.

RegU(id). The oracle adds id to RU. An adversary plays the corrupted SM and oracle executes honestly RegisterU protocol. Return $Reg[id]$.

RegO$_\mathbf{i}$(id). The oracle adds id to RU and CU. An adversary plays the corrupted user and oracle executes honestly RegisterO protocol. Save reg_i in the registration table.

JoinU(id). The oracle adds id to JIU. An adversary plays the corrupted SP and oracle executes honestly JoinU protocol. Return a signing key $gsk[id]$.

IssueI(Reg[id]). The oracle adds id to JIU and CU. The oracle executes honestly IssueI protocol and the adversary plays the corrupted user.

Sign(id, item, M). An adversary queries signing oracle to get a signature on message M for an honest user with identity id in $JIU \backslash CU$. The queried signature is added to Q_{Sign}.

Open$_\mathbf{i}$(item, M, σ). An adversary gets an output of the Open algorithm, and $(item, M, \sigma)$ is added to Q_{Open}.

ReadReg(i, id). The oracle can be used by the adversary to return reg_i[id].

WriteReg(i, id, v). Set $reg_i[id] \leftarrow v$, write v to ith SM for user id.

We define the security experiments of DTrustRS, which include anonymity, traceability, public linkability. Anonymity experiment and traceability experiment are based on [12], [25]. Public linkability experiment is based on [16]. The experiments are defined in Table 6.1.

Definition 6.1. A DTrustRS scheme is anonymous if for every efficient adversary $\mathcal{A}$ wins the $Exp^{ano-b}_{\text{DTrustRS},\lambda,n_I,n_O}$ with negligible advantage. We denote the advantage of adversary $\mathcal{A}$ by

$$\mathbf{Adv}^{ano}_{\text{DTrustRS},n_I,n_O,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \mid \mathbf{Pr}[Exp^{ano-1}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A}) = 1] - \mathbf{Pr}[Exp^{ano-0}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A}) = 1] \mid$$

Definition 6.2. A DTrustRS scheme is traceable if for all efficient adversary $\mathcal{A}$ wins the $Exp^{trace}_{\text{DTrustRS},\lambda,n_I,n_O}$ with negligible advantage. We denote the advantage of adversary $\mathcal{A}$ by

```
$Experiment \quad Exp^{ano-b}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A})$
$(params) \leftarrow \mathsf{Setup}(1^\lambda, n_I, n_O)$
$\langle\{(isk_i, ipk_i, st_i)\}^{n_I}_{i=1}\rangle \leftarrow \langle\{\mathsf{KeygenI}(params, i)\}^{n_I}_{i=1}\rangle$
$\langle\{(osk_i, opk_i, reg_i)\}^{n_O}_{i=1}\rangle \leftarrow \langle\{\mathsf{KeygenO}(params, i)\}^{n_O}_{i=1}\rangle$
$(apk_O) \leftarrow \mathsf{KeygenAggr}(params, \langle opk_1, \cdots, opk_{n_O}\rangle)$
$gpk \leftarrow (ipk_i, apk_O)$
$\mathcal{O} \leftarrow \{(\text{RegU, RegO, JoinU, Sign, Open, GSK, WriteReg})\}$
$(st'_{\mathcal{A}}, (id_0, id_1, M^*, item^*)) \leftarrow \mathcal{A}(gpk, \mathbf{isk}, \mathbf{osk}_{i \neq j, j=1} : \mathcal{O})$
$\sigma^* \leftarrow \mathsf{Sign}(params, gpk, gsk[id^*_b], M^*, item^*)$
$b' \leftarrow \mathcal{A}(st'_{\mathcal{A}}, \sigma^* : \mathcal{O})$
If $id^*_0, id^*_1 \in JIU \backslash CU$ and
$gsk[id^*_0], gsk[id^*_1] \neq \perp$ and $(item^*, M^*, \sigma^*) \notin Q_{Open}$, return $b'$
else return 0

$Experiment \quad Exp^{trace}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A})$
$(params) \leftarrow \mathsf{Setup}(1^\lambda, n_I, n_O)$
$\langle\{(isk_i, ipk_i, st_i)\}^{n_I}_{i=1}\rangle \leftarrow \langle\{\mathsf{KeygenI}(params, i)\}^{n_I}_{i=1}\rangle$
$\langle\{(osk_i, opk_i, reg_i)\}^{n_O}_{i=1}\rangle \leftarrow \langle\{\mathsf{KeygenO}(params, i)\}^{n_O}_{i=1}\rangle$
$(apk_O) \leftarrow \mathsf{KeygenAggr}(params, \langle opk_1, \cdots, opk_{n_O}\rangle)$
$gpk \leftarrow (ipk_i, apk_O)$
$\mathcal{O} \leftarrow \{\text{RegU, RegO, JoinU, IssueI, Sign, Open, GSK, ReadReg}\}$
$(\sigma^*, M^*, item^*) \leftarrow \mathcal{A}(gpk, \mathbf{osk}_{i \neq j, j=1} : \mathcal{O})$
If $\mathsf{Verify}(params, gpk, \sigma^*, M^*, item^*) = 0$ then return 0
$\langle(id^*, \pi_i)\rangle \leftarrow \mathsf{Open}_i(params, gpk, item^*, reg_i, osk_i, (\sigma^*, M^*))$
If $id^* = \perp$ or $\mathsf{Judge}(params, gpk, item^*, (M^*, \sigma^*),$
$\langle\{id^*, \pi_i\}\rangle) = 1$ then return 1
else return 0

$Experiment \quad Exp^{publink}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A})$
$(params) \leftarrow \mathsf{Setup}(1^\lambda, n_I, n_O)$
$\langle\{(isk_i, ipk_i, st_i)\}^{n_I}_{i=1}\rangle \leftarrow \langle\{\mathsf{KeygenI}(params, i)\}^{n_I}_{i=1}\rangle$
$\langle\{(osk_i, opk_i, reg_i)\}^{n_O}_{i=1}\rangle \leftarrow \langle\{\mathsf{KeygenO}(params, i)\}^{n_O}_{i=1}\rangle$
$(apk_O) \leftarrow \mathsf{KeygenAggr}(params, \langle opk_1, \cdots, opk_{n_O}\rangle)$
$gpk \leftarrow (ipk_i, apk_O)$
$\mathcal{O} \leftarrow \{(\text{RegO, IssueI, GSK})\}$
$\{(\sigma_i, M_i, item)\}^{|CU|+1}_{i=1}) \leftarrow \mathcal{A}(gpk, \mathbf{osk}_{i \neq j, j=1} : \mathcal{O})$
For $i = 1; i++; i <= |CU| + 1$
If $\mathsf{Verify}(params, gpk, \sigma_i, M_i, item) = 0$ then return 0
For $i, j \in [1, \cdots, |CU| + 1], i \neq j$
If $\mathsf{Link}(params, gpk, item, (M_i, \sigma_i), (M_j, \sigma_j)) = 0$ then return 1
else return 0
```

Table 6.1: The security experiments of DTrustRS

$$\mathbf{Adv}^{trace}_{\text{DTrustRS},n_I,n_O,A}(\lambda) \overset{\text{def}}{=} \mathbf{Pr}[Exp^{trace}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A}) = 1]$$

Definition 6.3. A DTrustRS scheme is publicly linkable if for any probability polynomial time adversary $\mathcal{A}$ wins $Exp^{publink}_{\text{DTrustRS},\lambda,n_I,n_O}$ with negligible advantage. We denote the advantage of adversary $\mathcal{A}$ by

$$\mathbf{Adv}^{publink}_{\text{DTrustRS},n_I,n_O,A}(\lambda) \overset{\text{def}}{=} \mathbf{Pr}[Exp^{publink}_{\text{DTrustRS},\lambda,n_I,n_O}(\mathcal{A}) = 1]$$

6.3 Aggregatable PS Signature

We propose an aggregatable PS signature (APS), which supports the aggregation of multiple signatures for multiple messages into one signature and allows knowledge of the signature to be efficiently proved like the PSM signatures.

6.3.1 The Syntax of APS

An aggregatable PS signature consists of a tuple of polynomial-time algorithms (Setup, Keygen, AggKgen, VerifyAggKgen, Sign, AggSign, Verify).

- $(params) \leftarrow \mathsf{Setup}(1^\lambda)$: Take a security parameter λ as input, and output the system parameters $params$.
- $(sk, pk) \leftarrow \mathsf{Keygen}(params)$: Take the system parameter $params$ as input to generate a key pair (sk, pk).
- $(apk) \leftarrow \mathsf{AggKgen}(params, pk)$: Take the system parameter $params$ and a public key pk as input to generate an aggregate key apk.

- $(b) \leftarrow \mathsf{VerifyAggKgen}(params, apk, pk)$: Take the system parameter $params$, a public key pk and an aggregate key apk as input to output a bit $b \in \{0, 1\}$.

- $(\sigma) \leftarrow \mathsf{Sign}(params, sk, m, aux)$: Take the system parameter $params$, the secret key sk, a message m and an auxiliary information aux as input, and compute a signature σ.

- $(\sigma) \leftarrow \mathsf{AggSign}(params, \{(\sigma_i, m_i, pk_i)\}_{i=1}^{n}, aux)$: Take the system parameter $params$, the signature-message pair (σ_i, m_i) with corresponding public key pk_i, and an auxiliary information aux as input to compute an aggregated signature σ.

- $(b') \leftarrow \mathsf{Verify}(params, apk, (\sigma, \mathbf{m}), aux)$: Take the system parameters $params$, the aggregate key apk, a signature-message pair $(\sigma, \mathbf{m})$ as input to output a bit $b' \in \{0, 1\}$.

Correctness. An APS must satisfy the correctness.

For all $(params) \leftarrow \mathsf{Setup}(1^{\lambda}) \ \wedge$ for all $(sk, pk) \leftarrow \mathsf{Keygen}(params)$: $\mathsf{VerifyAggKgen}(params, \mathsf{AggKgen}(params, pk), pk) = 1 \wedge \mathsf{Verify}(params, apk, (\mathsf{AggSign}(params, \{(\sigma_i, m_i, pk_i)\}_{i=1}^{n}, aux), \mathbf{m}), aux)) = 1$.

6.3.2 The Security Definition of APS

An aggregatable PS signature should satisfy the unforgeability.

Definition 6.4. An APS satisfies unforgeability if for any probability polynomial-time adversary $\mathcal{A}$ such that

$$\Pr\left[\begin{array}{c} \mathsf{VerifyAggKgen}(params, apk^*, pk) = 1 \wedge \\ \mathsf{Verify}(params, apk^*, (\sigma^*, m^*), aux^*)) = 1 \\ : \\ (params) \leftarrow \mathsf{Setup}(1^\lambda) \\ (sk, pk) \leftarrow \mathsf{Keygen}(params) \\ (\sigma^*, m^*, apk^*, aux^*) \leftarrow \mathcal{A}^{\mathcal{O}sign}(params, pk) \end{array}\right] \leq negl(\lambda),$$

$-\mathcal{O}sign()$ is a sign oracle, which responses the queries from adversary $\mathcal{A}$.

We require that a forged signature-message pair must be not queried by the adversary, and the adversary proves knowledge of the discrete log for his public key.

6.3.3 Construction of APS

- $\mathsf{Setup}(1^\lambda)$: Input a security parameter λ to generate $\Phi = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with g_1 a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$. Set $H_2 : \{0,1\}^* \to \mathbb{Z}_q$, $H_3 : \{0,1\}^* \to \mathbb{G}_1$ be collision-resistant hash functions. Let $params = (\Phi, g_1, g_2, H_2, H_3)$.

- $\mathsf{Keygen}(params)$: Randomly choose $x, y \in \mathbb{Z}_q^*$ and compute $\widehat{X} = g_2^x, \widehat{Y} = g_2^y$. Set $sk = (x, y), pk = (\widehat{X}, \widehat{Y})$.

- $\mathsf{AggKgen}(params, pk)$: Compute $H_2(pk) \to r$ and $apk = (pk)^r = (\widehat{X}^r, \widehat{Y}^r)$. Then, output apk.

- $\mathsf{VerifyAggKgen}(params, apk, pk)$: Compute $H_2\ (pk) \to r$. Parse $pk = (\widehat{X}, \widehat{Y})$ and apk. If $apk = (\widehat{X}^r, \widehat{Y}^r)$, output 1; otherwise, output 0.

- $\mathsf{Sign}(params, sk, m, aux)$: Set auxiliary information $aux \in \{0,1\}^*$. Compute $H_3(aux) \to h$ and $H_3(m) \to h_1$. Output a signature $\sigma = (a, b) = (hh_1, (hh_1)^{x+ym})$.

- $\mathsf{AggSign}(params, \{(\sigma_i, m_i, pk_i = \{\widehat{X}_i, \widehat{Y}_i\})\}_{i=1}^n, aux)$: Parse σ_i as (a_i, b_i). Compute $H_2(pk_i) \to r_i$, for $i = 1, \cdots, n$. $\mathbf{a} = (a_1, \cdots, a_n)$, $b = \prod_{i=1}^n b_i^{r_i}$. Output the aggregated signature $\sigma = (\mathbf{a}, b)$.

- $\mathsf{Verify}(params, apk_i = \{\widehat{X}_i^r, \widehat{Y}_i^r\}_{i=1}^n, (\sigma, \mathbf{m}), aux)$: Parse σ as $(\mathbf{a}, b) = ((a_1, \cdots, a_n), b)$ and $\mathbf{m} = (m_1, \cdots, m_n)$. If $\prod_{i=1}^n e(a_i, \widehat{X}_i^r \widehat{Y}_i^{rm_i}) = e(b, g_2)$, output 1; otherwise, output 0.

Correctness. Given the aggregate key apk and public key pk, by calling VerifyAggKgen algorithm, we can verify the aggregated public key apk. Parse $apk = (ax, ay)$ and compute $H_2(pk) \to r$. If $(pk)^r = (ax, ay) = (\widehat{X}^r, \widehat{Y}^r)$, output 1. Given an aggregated signature σ and message $\mathbf{m}$, by calling Verify algorithm to check the signature. Parse $\sigma = (\mathbf{a}, b) = ((a_1, \cdots, a_n), b)$ and $\mathbf{m} = (m_1, \cdots, m_n)$. If $\prod_{i=1}^n e(a_i, \widehat{X}_i \widehat{Y}_i^{m_i}) = e(b, g_2)$, output 1.

Theorem 6.1. The scheme above is an APS satisfying Definition 6.4 if the q-MSDH-1 assumption holds in Φ.

Proof. Given a type-3 pairing group Φ, if an adversary $\mathcal{A}$ wins the unforgeability game with at least ε probability, then, we can construct an algorithm $\mathcal{B}$, which runs $\mathcal{A}$ as a subroutine to output a tuple $(w, P, h^{1/w+x}, h^{a/P(x)})$ with the non-negligible probability, for a q-MSDH-1 instance $((g_1^{x^i}, g_2^{x^i})_{i=0}^q$ and $(g_1^a, g_2^a, g_2^{ax}))$. At the beginning of the unforgeability game, given the q-MSDH-1 instance $(g_1^{x^i}, g_2^{x^i})_{i=0}^q$ and (g_1^a, g_2^a, g_2^{ax}) to algorithm $\mathcal{B}$. Then, $\mathcal{B}$ interacts with adversary $\mathcal{A}$, which receives a message tuple $(w_1, \cdots, w_q) \in \mathbb{Z}_q$. $\mathcal{B}$ setups the public parameters. It computes $f = g_1^{\prod_{i=1}^{q}(x+w_i)}$, $\widehat{f} = g_2^{\prod_{i=1}^{q}(x+w_i)}$, and sends $(f, \widehat{f}, n)$ to $\mathcal{A}$. $\mathcal{B}$ sets $\widehat{X} = g_2^{ax}, \widehat{Y} = g_2^{a}$ as the public key of target signer $t \in [n]$ and sends it to $\mathcal{A}$. Notice that the $\mathcal{B}$ implicitly sets

$$x_t = \frac{ax}{\prod_{i=1}^{q}(x+w_i)}, y_t = \frac{a}{\prod_{i=1}^{q}(x+w_i)}.$$

For signature queries on message tuple $(w_1, \cdots, w_q) \in \mathbb{Z}_q$, $\mathcal{B}$ computes and stories

$$\sigma_i = (a_i, b_i) = (g^{\prod_{j \neq i}(x+w_j)^{t_i}}, (g^a)^{t_i}),$$

where $t_i \in \mathbb{Z}_q^*$. Then, it sends the signatures $\{\sigma_i\}_{i=1}^{q}$ to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ verifies the signatures

$$e(a_i, \widehat{X}\widehat{Y}^{w_i}) = e(b_i, g_2).$$

$\mathcal{A}$ queries H_3 on aux, $\mathcal{B}$ generates and returns h. $\mathcal{A}$ queries H_3 on a message w, $\mathcal{B}$ computes $\frac{g^{\prod_{j \neq i}(x+w_j)^{t_i}}}{h}$ as an answer to the random oracle query. Whenever adversary $\mathcal{A}$ queries H_2 random oracle on public key pk, $\mathcal{B}$ generates and returns $r \in \mathbb{Z}_q$. Finally, $\mathcal{A}$ returns a forgery σ^* on w^* with apk that includes target public key $\widehat{X}_t, \widehat{Y}_t$. Parse σ as (a, b),

$$\prod_{i=1}^{n} e(a_i, \widehat{X}_i^{r_i}\widehat{Y}_i^{w_i r_i}) = \prod_{i=1(i \neq t)}^{n} e(a_i, \widehat{X}_i^{r_i}\widehat{Y}_i^{w_i r_i}) e(a_t, \widehat{X}_t^{r_t}\widehat{Y}_t^{w_t r_t}) = e(b, \widehat{f}).$$

$\mathcal{B}$ rewinds the hash queries on H_2, and generates $r' \in \mathbb{Z}_q$. Then, $\mathcal{A}$ makes queries. If adversary $\mathcal{A}$ outputs another forgery σ' such that

$$\prod_{i=1(i \neq t)}^{n} e(a'_i, \widehat{X}_i^{r_i}\widehat{Y}_i^{w_i r_i}) e(a'_t, \widehat{X}_t^{r'_t}\widehat{Y}_t^{w_t r'_t}) = e(b', \widehat{f}).$$

Then, $\mathcal{B}$ computes

$$e(a_t'^{r_t'}/a_t^{r_t}, \widehat{X}_t\widehat{Y}_t^{w_t}) = e(b'/b, \widehat{f}).$$

$\mathcal{B}$ outputs a forgery $(w_t, P, a_t'^{r_t'}/a_t^{r_t}, b'/b)$, where $(P = \prod_{i=1}^{q}(x + w_i))$ to challenger. If ε was non-negligible, $\mathcal{B}$ by running $\mathcal{A}$ as a sub-rountine can win the q-MSDH-1 game with non-negligible probability $(\varepsilon^2 - \varepsilon / \mid q \mid)$. Under q-MSDH-1 assumption, ε were negligible, thus, the APS is unforgeability.

6.4 The Construction of DTrustRS

In this section, we construct a concrete scheme for DTrustRS, which is based on the APS and PSM signature [25]. The APS enables SM to open a signature effectively and PSM reduces the signature size. For public linkability, we modify a technique in [7] by integrating the statement information to generate a link identify.

In DTrustRS scheme, we adopt APS to construct the registration protocol and Open algorithm, each user interacts with the SM to register as a legal user. With the registration message, SM can open a signature effectively and generate an opening proof. We adopt PSM signature to issue the member credential. Each registered user interacts with SP and obtains a credential. With the credential, the user can generate a reputation signature. Then, every verifier can check two signatures whether or not from the same user by the link identify.

6.4.1 The Scheme of DTrustRS

Given a security parameter λ to generate a type-3 pairing group Φ. Let H : $\{0,1\}^* \rightarrow \mathbb{Z}_q$, $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q \times \mathbb{G}_1$, $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^n$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$,

$H_3 : \{0,1\}^* \to \mathbb{G}_1$ be collision-resistant hash functions. The scheme of DTrustRS as following:

- $(params) \leftarrow \mathsf{Setup}(1^\lambda, n_I, n_O)$: Generate (g_1, g_2) with g_1 and g_2 are generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Return public parameter $params = (\Phi, g_1, g_2, n_I, n_O)$.

- $(isk_i, ipk_i, st_i) \leftarrow \mathsf{KeygenI}(params, i \in n_I)$: Each SP runs $(vk, sk) \leftarrow \mathsf{PSM.Keygen}(params)$ and let $(ipk_i, isk_i) \leftarrow (vk, sk)$. Then, initialize statement $st_i \in \{0,1\}^*$. The algorithm returns (isk_i, ipk_i, st_i).

- $(osk_i, opk_i, reg_i) \leftarrow \mathsf{KeygenO}(params, i \in n_O)$: Each SM runs $(pk, sk) \leftarrow \mathsf{APS.Keygen}(params)$ and let $(opk_i, osk_i) \leftarrow (pk, sk)$. Then, initialize an empty register list reg_i. The algorithm returns (osk_i, opk_i, reg_i).

- $(apk) \leftarrow \mathsf{KeygenAggr}(params, \langle pk_1, \cdots, pk_n \rangle)$: Take the $params$ and a public key vector of SM $(pk_1, \cdots, pk_{n_O})$ as input to generate an aggregate key $(apk_O) = \langle \{\mathsf{APS.AggKgen}(opk_i)\}_{i=1}^{n_O} \rangle$

  The system parameter gpk is set to (ipk_i, apk_O).

- $\langle\, Reg[id], \{reg_i\}_{i=1}^{n_O} \,\rangle \leftarrow \langle\, \mathsf{RegisterU}(params, id, gpk) \rightleftharpoons \{\mathsf{RegisterO}\ (params,$ $id,\ osk_i,\ opk_i\}_{i \in n_O} \rangle$: As shown in Table 6.2, a user interacts with SM. For $i = 1, \cdots, n_O$, a user inputs $params, id_O$ and $gpk = (ipk_i, apk_O = (\{(\widehat{X}_{0,i}, \widehat{Y}_{0,i})\}_{i=1}^{n_O}))$ to register the id_O. Each system manager computes and stores $reg_i = reg[id]$, and computes and generates $Reg[id] = (\xi, \sigma)$ with id_O.

$\mathsf{RegisterU}$	$\leftrightharpoons$	$\mathsf{RegisterO}$
Randomly choose $sk_{id} \in \mathbb{Z}_q$		
$h \leftarrow H_3(id_O), h_1 = H_3(sk_{id})$		
$h_{sk} = (hh_1)^{sk_{id}}, h_2 = \widehat{Y}_{0,i}^{sk_{id}}$		
$\pi \leftarrow \mathsf{NIZK.Prove}\{(sk_{id}) :$		
$h_{sk} = (hh_1)^{sk_{id}} \wedge h_2 = \widehat{Y}_{0,i}^{sk_{id}}\}$		
	$\xrightarrow{h_{sk},h_2,\pi}$	
		$\mathsf{NIZK.Verify}(h_{sk}, h_2, \pi) \overset{?}{=} 1$
		$\kappa \in \mathbb{Z}_q^*, Z = h_{sk}(hh_1)^{\kappa}$
		$\sigma = (a, b) = (hh_1, (hh_1)^x Z^y)$
	$\xleftarrow{\sigma,\kappa}$	
$Reg[id] = ((\xi = sk_{id} + \kappa), \sigma)$		$reg_i = (h_2, \kappa)$

Table 6.2: The registration protocol

- $(gsk[id]) \leftarrow \langle \mathsf{JoinU}(params, Reg[id], gpk) \rightleftharpoons \{\mathsf{IssueI}(params, id, isk_i, ipk_i\}_{i \in n_I} \rangle$: A user interacts with service providers. We assume that there is a broadcast channel and further this protocol runs over secure channel. The user inputs $params$, $Reg[id]$ and gpk, and computes a proof with $Reg[id]$ that user has registration. It parses $Reg[id] = (\xi, \sigma) = ((\xi_1, \cdots, \xi_{n_O}), ((a_1, \cdots, a_{n_O}), b))$ and randomly chooses $r \in \mathbb{Z}_q$ and computes $\sigma' = (\sigma^r)$.

$$\pi_R \leftarrow \mathsf{NIZK.Prove}\{(\{\xi\}_{i=1}^{n_O}) : e(b', g_2) = \prod_{i=1}^{n_O} e(a_i', \widehat{X}_i \widehat{Y}_i^{\xi_i})\}.$$

Compute $(m', h) \leftarrow H_0(id)$, $h_{sk} = h^{sk_{id}}$,

$$\pi \leftarrow \mathsf{NIZK.Prove}\{(sk_{id}) : h_{sk} = h^{sk_{id}}\}.$$

Set $L[id] \leftarrow (id, \pi_R, \sigma')$ and send $L[id]$ and $(h_{sk}, \pi)$to a SP. As shown in Table 6.3, the user interacts with the SP. Then, output the signing secret key

$gsk[id]$.

JoinU	$\leftrightharpoons$	IssueI
	$\xrightarrow{L[id],h_{sk},\pi}$	
		Parse $L[id] = (id, \pi_R, \sigma')$
		$\mathsf{NIZK.Verify}(h_{sk}, \pi) \overset{?}{=} 1$
		$\mathsf{NIZK.Verify}(\sigma', \pi_R) \overset{?}{=} 1$
		$(m', h) \leftarrow H_0(id)$,
		$c = (h^{x+y_1 m'} h_{sk}^{y_2})$
	$\xleftarrow{c}$	
$\Sigma = (m', h, c)$		
$\mathsf{PSM.Verify}(\Sigma, sk_{id}, ipk_i) \overset{?}{=} 1$		
$gsk[id] = (sk_{id}, \Sigma)$		

Table 6.3: The join protocol

- $(\sigma) \leftarrow \mathsf{Sign}(params, gpk, gsk[id], M, item)$: The user parses $gsk[id] = (sk_{id}, \Sigma = (m', \Sigma_0, \Sigma_1))$. Set $item = (st_i)$. Compute $d \leftarrow H_3(item)$, and randomly choose $\mu \in \mathbb{Z}_q$ to re-randomlize Σ. Compute $\Sigma' = \Sigma^{\mu} = (\Sigma_0', \Sigma_1') = (\Sigma_0^{\mu}, \Sigma_1^{\mu})$ and $S = d^{sk_{id}}$. Then, the user computes

$$\pi \leftarrow \mathsf{NIZK.Prove}\{(sk_{id}, m') :$$

$$S = d^{sk_{id}} \wedge \mathsf{PSM.Verify}(ipk_i, sk_{id}, (m', \Sigma_0', \Sigma_1')) = 1\}(M).$$

The user generates the Schnorr signature of knowledge [80] π on message M such that $S = d^{sk_{id}}$ and

$$e(\Sigma_0'^{sk_{id}}, \widehat{Y}_2) e(\Sigma_0'^{m'}, \widehat{Y}_1) = \frac{e(\Sigma_1', g_2)}{e(\Sigma_0', \widehat{X})},$$

output a signature $\sigma \leftarrow (S, \Sigma', \pi)$.

- $(b) \leftarrow \mathsf{Verify}(params, gpk, \sigma, M, item)$: A verifier parses $\sigma = (S, \Sigma', \pi)$. It computes $\mathsf{NIZK.Verify}(gpk, M,\ (S, \Sigma', \pi))$. If the signature σ is valid, return 1; otherwise, return 0.

- $\langle\{id, \pi_i\}_{i \in n_O}\rangle \leftarrow \langle\{\mathsf{Open}(params, gpk, item, (\sigma, M),\ reg_i, osk_i)\}_{i \in n_O}\rangle$: The SM verifies the signature by calling Verify. If $\mathsf{Verify}(params, gpk, \sigma, M, item) = 0$, then return $\perp$. Otherwise, SM retrieves the list $L[id]$ and reg_i.

  – Parses $reg_i = (h_2, \kappa)$ and computes $d \leftarrow H_3(item)$.

  – For all h_2, it checks whether $e(S, \widehat{Y}_{0,i}) = e(d, h_2)$ holds. Retrieve κ to compute $h_3 = h_2 \widehat{Y}_{0,i}^{\kappa}$ and

  $$\Theta \leftarrow \mathsf{NIZK.Prove}\{(h_2, \kappa) : e(S, \widehat{Y}_{0,i}) = e(d, h_2) \wedge \frac{e(d, h_3)}{e(S, \widehat{Y}_{0,i})} = e(d, \widehat{Y}_{0,i})^{\kappa}\}.$$

  SMs cooperatively retrieve id in the list $L[id]$.

  Parses $L[id] = (id, \pi_R, \sigma')$. Return a proof $(id, \pi = (h_3, \sigma', \Theta))$.

- $(b') \leftarrow \mathsf{Judge}(params, gpk, item, (\sigma, M), \langle\{id, \pi_i\}_{i \in n_O}\rangle)$: This algorithm verifies $\langle\{id, \pi_i\}_{i \in n_O}\rangle$. For all (π_i), it parses $\pi_i = (h_{3,i}, \sigma', \Theta_i)$ and $\sigma' = (a, b) = ((a_1, \cdots, a_{n_O}), b)$ then, computes $\mathsf{NIZK.Verify}(\Theta_i) \stackrel{?}{=} 1$ and checks if

  $$\prod_{i=1}^{n_O} e(a_i, \widehat{X}_{0,i} h_{3,i}) = e(b, g_2).$$

  return 1; otherwise, return 0.

- $(b'') \leftarrow \mathsf{Link}(params, gpk, item, (M_0, \sigma_0), (M_1, \sigma_1))$: This algorithm checks two signatures $(M_0, \sigma_0), (M_1, \sigma_1)$. If $\mathsf{Verify}(params, gpk, \sigma_i, M_i, item) =$

0, for $i \in \{0,1\}$, then return $\perp$. Otherwise, parse $\sigma_i = (S_i, \Sigma_i', \pi_i)$, for $i \in \{0,1\}$. If $S_0 = S_1$ then return 1; otherwise, return 0.

Correctness. The DTrustRS satisfies correctness.

Proof. The correctness of DTrustRS is based on the correctness of the APS and PSM signature scheme, and the completeness of Schnorr proof.

By the public parameters, the SP and SM generate public key. In registration protocol, the user interacts with SM. The user generates a proof of sk_{id} as shown in Table 6.4. Each SM calls NIZK.Verify. Then, SM randomly picks $\kappa \in$

Prover		Verifier
$\rho \in \mathbb{Z}_q^*$		
$v = (hh_1)^\rho, w = \widehat{Y}_0^\rho$		
$c = H(v, w, h_{sk}, h_2)$		
$s_1 = \rho + csk_{id}$	$\xrightarrow{v,w,s_1}$	Verify
		$(hh_1)^{s_1} \overset{?}{=} v \cdot h_{sk}^c, \widehat{Y}_0^{s_1} \overset{?}{=} w \cdot h_2^c$

Table 6.4: The proof of sk_{id}

$\mathbb{Z}_q^*$ to compute $Z = h_{sk}(hh_1)^\kappa = (hh_1)^{sk_{id}+\kappa}$ and sends κ and $\sigma = (a,b) = (hh_1, (hh_1)^x Z^y) = (hh_1, ((hh_1)^{x+y(sk_{id}+\kappa)}))$ to user. The user generates a registration credential $Reg[id]$.

In issue protocol, the user interacts with SP to obtain $gsk[id]$. The user parses $Reg[id] = (\xi, \sigma) = ((\xi_1, \cdots, \xi_{n_O}), ((a_1, \cdots, a_{n_O}), b))$. Randomly choose $r \in \mathbb{Z}_q$ and compute $\sigma' = \sigma^r = (a_1^r, \cdots, a_{n_O}^r, b^r) = (a_1', \cdots, a_{n_O}', b')$ and generate the proof π_R as shown in Table 6.5.

Then, compute $(m', h) \leftarrow H_0(id)$, $h_{sk} = h^{sk_{id}}$ and generate a proof π as shown in Table 6.6.

Set $L[id] \leftarrow (id, \pi_R, \sigma')$ and send $L[id]$ and (h_{sk}, π) to SP. SP computes

Prover		Verifier
For $i = 1, \cdots, n_O$		
Randomly choose $\tau_i \in \mathbb{Z}_q^*$		
$u_i = (\widehat{Y}_{0,i})^{\tau_i}$		
$c = H(\sigma', u_1, \cdots, u_{n_O})$		
$s_i = \tau_i + c\xi_i$	$\xrightarrow{c,(u_i,s_i)_{i=1}^{n_O}}$	Verify
		$\prod_{i=1}^{n_O} e(a'_i, \widehat{X}_{0,i}(\frac{\widehat{Y}_{0,i}^{s_i}}{u_i})^{-c})$
		$\overset{?}{=} e(b', g_2)$

Table 6.5: The NIZK of (ξ, σ)

Prover		Verifier
$\rho \in \mathbb{Z}_q^*, v = h^\rho$		
$c = H(v, h_{sk})$		
$s_1 = \rho + csk_{id}$	$\xrightarrow{v,c,s_1}$	Verify
		$h^{s_1} \overset{?}{=} v \cdot h_{sk}^c$

Table 6.6: The NIZK of sk_{id}

NIZK.Verify and $(m', h) \leftarrow H_0(id)$, $c = (h^{x+y_1m'}h_{sk}^{y_2}) = (h^{x+y_1m'+y_2sk_{id}})$. It sends c to user. The user sets $gsk[id] = (sk_{id}, \Sigma)$. Then, the user generates a signature. The user parses $gsk[id] = (sk_{id}, \Sigma = (m', \Sigma_0, \Sigma_1))$. Set $item = (st_i)$. It computes $d \leftarrow H_3(item)$, and randomly chooses μ to re-randomlize Σ. Compute $\Sigma' = \Sigma^\mu = (\Sigma'_0, \Sigma'_1) = (\Sigma_0^\mu, \Sigma_1^\mu)$ and $S = d^{sk_{id}}$. Then, the user computes a Schnorr signature of knowledge as shown in Table 6.7. Then, a verifier checks the signature by calling NIZK.Verify algorithm.

For Open algorithm, SM verifies the signature by calling Verify. SM retrieves the list $L[id]$ and reg_i. Parses $reg_i = (h_2, \kappa)$ to compute $d \leftarrow H_3(item)$ and check whether $e(S, \widehat{Y}_0) = e(d, h_2)$ holds. Then, retrieves κ to compute $h_3 = h_2\widehat{Y}_0^\kappa$ and π as shown in Table 6.8.

Prover		Verifier
$\nu, \omega \in \mathbb{Z}_q^*$		
$a = e(\Sigma_0'^{\nu}, \widehat{Y}_2)e(\Sigma_0'^{\omega}, \widehat{Y}_1), b = d^{\nu}$		
$\alpha = H(ipk_i, S, \Sigma_0', \Sigma_1', M, a, b)$		
$\beta = \nu - \alpha sk_{id}, \gamma = \omega - \alpha m'$	$\xrightarrow{a,b,\alpha,\beta,\gamma}$	
		$a \cdot e(\Sigma_0'^{\alpha}, \widehat{X}) = e(\Sigma_0'^{\beta}, \widehat{Y}_2) \cdot e(\Sigma_0'^{\gamma}, \widehat{Y}_1)e(\Sigma_1'^{\alpha}, g_2),$
		$b = d^{\beta} S^{\alpha}$

Table 6.7: The proof of a reputation signature

Prover		Verifier
$r_1, r_2, r_3 \in \mathbb{Z}_q^*$		
$A = h_2 g_2^{r_1},$		
$B = d^{r_1},$		
$C = \widehat{Y}_0^{r_2},$		
$D = d^{r_3}$		
$c = H(A, B, C, D, S, h_3)$		
$s = r_3 + cr_1,$		
$r = r_2 + c\kappa$	$\xrightarrow{A,B,C,D,}$	
	$\xrightarrow{S,s,r,c}$	Verify
		$(\frac{e(d,A)}{e(S,\widehat{Y}_0)})^s \overset{?}{=} e(B, g_2)^c e(D, g_2),$
		$(\frac{e(d,h_3)}{e(S,\widehat{Y}_0)})^r \overset{?}{=} (\frac{e(d,h_3)}{e(S,\widehat{Y}_0)})^c e(d, C)$

Table 6.8: The NIZK of (h_2, κ)

By the Judge algorithm, a verifier computes $\mathsf{NIZK.Verify}$ and checks if

$$\prod_{i=1}^{n_O} e(a_i, \widehat{X}_{0,i} h_{3,i}) = e(b, g_2).$$

Link algorithm checks two signatures (M_0, σ_0) and (M_1, σ_1). Compute $\mathsf{Verify}(params, gpk, \sigma_i, M_i, item) = 1$, for $i \in \{0,1\}$ and parse $\sigma_i = (S_i, \Sigma'_i, \pi_i)$, for $i \in \{0,1\}$. The verifiers can verify the linking by computing $S_0 \overset{?}{=} S_1$.

Remark. The DTrustRS supports batch verification, which can further compress the signature check time with multiple signatures. When the provider receive the signature $(\sigma_i \leftarrow (S_i, \Sigma'_i, \pi_i))_{i=1}^n$from user $[1, n]$. The verifier can check the verify algorithm and executes batch verification. The verifier computes $\Pi_{i=1}^n a_i \cdot e(\Pi_{i=1}^n \Sigma_{i,0}'^{\alpha_i}, \widehat{X}) = e(\Pi_{i=1}^n \Sigma_{i,0}'^{\beta_i}, \widehat{Y}_2) \cdot e(\Pi_{i=1}^n \Sigma_{i,0}'^{\gamma_i}, \widehat{Y}_1) e(\Pi_{i=1}^n \Sigma_{i,1}'^{\alpha_i}, g_2), \Pi_{i=1}^n b_i = \Pi_{i=1}^n d_i^{\beta_i} S_i^{\alpha_i}$. With the optimized operation, we can compress the pairing group operation from $4n$ to constant 4.

6.4.2 Security Analysis

The security property of DTrustRS is based on the APS scheme, PSM signature scheme, DDH assumption and SDL assumption. Before the formal proving, we follow the lemma below to provide an argument for the proof of anonymity, traceability and public linkability.

Lemma 6.1.(Forgery) If the SDL holds in type-3 pairing groups in random oracle model, no efficient adversary $\mathcal{A}$ can forge a DTrustRS signature with non-negligible probability.

Proof. An adversary $\mathcal{A}$ generates a DTrustRS signature σ on message m with a valid proof π for id*, then, an algorithm $\mathcal{B}$ can solve the SDL problem. With the general forking lemma [74], [9], $\mathcal{B}$ generates an algorithm $\mathcal{A}'$. $\mathcal{A}'$ will output two signatures (σ_0, σ_1) such that $\mathcal{B}$ can extract μsk_{id^*} where sk_{id^*} is randomly selected by $\mathcal{B}$ for id^*. Then, $\mathcal{B}$ can compute μ to win the SDL game.

Given $\mathcal{A}'$ system parameters gpk, and SDL instance $g_1, g_2, \widetilde{g}_1 = g_1^{\mu}, \widetilde{g}_2 = g_2^{\mu}$. $\mathcal{A}'$ runs $\mathcal{A}$ as a sub-routine. $\mathcal{A}'$ maintains a list $List_H = (str, c, count)$. $\mathcal{A}'$ answers the oracle queries.

$-H(str \in \{0,1\}^*)$: $\mathcal{A}'$ retrieves str in list $List_H$, and outputs c. If $(str, *) \notin List_H$, $\mathcal{A}'$ selects $c \in \mathbb{Z}_q$ and returns c. Then, it sets $count \leftarrow count + 1$ and stores (str, c, ctr) to $List_H$.

$-H_0()$: $\mathcal{A}'$ queries the simulator $\mathcal{B}$.

$-$RegisterU(id). Add id to RU and execute registration protocol. Each identity id computes and generates an APS signature, then, returns $Reg[id] = (\xi, \sigma)$.

$-$JoinU$(Reg[id], gpk)$. $\mathcal{A}'$ plays the honest user. Add id to JIU, if $id \neq id^*$, follow the Join protocol. If $id = id^*$, A' makes internal query the $(m'^*, r^*) \leftarrow H_0(id^*)$. Next, generate sk_{id^*}. Compute $h_{sk} = \widetilde{g}_1^{r^* sk_{id^*}}$ and call H to simulate a proof π. With the $Reg[id]$ of APS signature, A' generates the proof π_R. Then, set $L[id] = (id, \pi_R, \sigma')$, . For the view of $\mathcal{A}$, the distribution of $L[id]$ is indistinguishable with the real protocol. Then, $\mathcal{A}'$ computes the PSM signature on μsk_{id^*}. The PSM signature on μsk_{id^*} is

$$(m'^*, \Sigma_1^* = g_1^{r^*}, \Sigma_2^* = (g_1^{r^*})^{x + y_1 \mu sk_{id^*} + y_2 m'^*})$$

Set $gsk[id^*] \leftarrow (\perp, (m'^*, \Sigma_1^*, \Sigma_2^*))$.

$-$Sign$(id, M, item)$: For the $id \in JIU \backslash CU$, if $id \neq id^*$, $\mathcal{A}'$ returns σ as an answer to the sign query, add $(id, item, M, \sigma)$ to Q_{Sign}. If $id = id^*$, fetch $gsk[id^*] \leftarrow$

$(\perp, (m'^{*}, \Sigma_1^*, \Sigma_2^*))$. Pick $t \in \mathbb{Z}_q$ and compute $\Sigma^t = (\Sigma_1', \Sigma_2') = (\Sigma_1^t, \Sigma_2^t)$. Then, simulate a proof π on μsk_{id^*} and m'^{*}, return σ. Add $(id, item, M, \sigma)$ to sign query set Q_{Sign}.

$-$GSK(id): Add id to CU and return $gsk[id]$.

$\mathcal{A}$ forges a signature σ on message m with a proof π for id. If Verify$(params, gpk, (M, \sigma), item) = 0$ or Judge$(params, gpk, item, M, \sigma, id, \pi) = 0$, abort the interaction with $\mathcal{A}$ and sends a random $\mu \in \mathbb{Z}_q^*$ to the SDL challenger. If the forgery is valid for $id \neq id^* \in JIU \setminus CU$, abort the interaction with $\mathcal{A}$ and sends a random $\mu \in \mathbb{Z}_q^*$ to the SDL challenger, where the $\mathcal{A}$ forges a signature for id^* with probability $1/|ID|$. If $id = id^* \in JIU \setminus CU$ and $(id^*, item, M, \sigma) \notin Q_{Sign}$, $\mathcal{A}$ has forged a signature. $\mathcal{A}'$ looks up the tuple (a, j, c), such a tuple exists, $\mathcal{A}'$ outputs (j, σ).

$\mathcal{B}$ runs algorithm $\mathcal{A}'$ and answers its random oracle queries on H_0. By the generalized forking lemma [74], [9], set $q > 8tq_{H_1}/\varepsilon$ by assumption, with probability at least $\varepsilon/8$ and with run time at most $8q_{H_1}/\varepsilon \cdot In(8/\varepsilon) \cdot \tau_{\mathcal{A}}$, $\mathcal{A}'$ returns (σ, σ').

Parse $\sigma = (S, \Sigma, \pi = (a, b, \alpha, \beta, \gamma))$ and $\sigma' = (S', \Sigma', \pi' = (a', b', \alpha', \beta', \gamma'))$, where $S = S', \Sigma = \Sigma'$ and $\alpha \neq \alpha'$ and $\alpha \neq \alpha' \bmod q$. $\mathcal{B}$ computes

$$e(\Sigma_0, \widehat{Y}_2^{(\beta-\beta')/(\alpha'-\alpha)} \widehat{Y}_1^{(\gamma-\gamma')/(\alpha'-\alpha)}) = \frac{e(\Sigma_1, g_2)}{e(\Sigma_0, \widehat{X})}.$$

Therefore,

$$g_2^{y_2(\beta-\beta')/(\alpha'-\alpha)+y_1(\gamma-\gamma')/(\alpha'-\alpha)} = g_2^{y_1 m'^{*}} \widetilde{g}_2^{y_2 sk_{id^*}}.$$

compute $\mu = (y_2(\beta-\beta') + y_1((\gamma-\gamma') - m'^{*}(\alpha'-\alpha)))/(\alpha'-\alpha)y_0 sk_{id^*}$, then send μ to challenger. $\mathcal{A}$ wins the non-frameability game with probability ε. Algorithm

$\mathcal{B}$ then wins the SDL game with probability at least

$$1/|ID| \cdot \varepsilon/8(1-1/q) - n_O|ID|Adv_{\lambda}^{\text{q-MSDH-1}}(\mathcal{B}(\mathcal{A})).$$

Anonymity: The DTrustRS satisfies anonymity if the DDH holds in type-3 pairing groups in the random oracle model.

Proof. If an adversary $\mathcal{A}$ has a non-negligible advantage ε in the anonymity experiment, then, a simulator $\mathcal{B}$ can solve the DDH problem in $\mathbb{G}_1$. Given $(f_1, f_2, f_3, f_4) = (g_1, g_1^{\mu}, g_1^{\nu}, g_1^{\tau})$ to $\mathcal{B}$. $\mathcal{B}$ will output a guess b', where $b = 1$, a Diffie-Hellman tuple $\tau \leftarrow \mu\nu$, or $b = 0$, random tuple $\tau \in \mathbb{Z}_q$. Proof is as follows.

Given a type-3 pairing groups Φ and a tuple (f_1, f_2, f_3, f_4) from a DDH challenger, $\mathcal{B}$ sets $g_1 = f_1$ and generates the other parameters of DTrustRS. By the public parameters $params$, $\mathcal{B}$ executes the key generation algorithms with $\mathcal{A}$. $\mathcal{B}$ randomly selects two identities id_0^* and id_1^*. $\mathcal{B}$ answers the oracle queries as in Lemma 6.1 for the identity $id_b^*, b \in \{0, 1\}$.

$-$RegisterU(id). Add id to RU and execute registration protocol. For each identity id computes and generates an APS signature, return $Reg[id] = (\xi, \sigma)$.

$-$RegisterO$_i(id)$. Add id to RU and CU and execute registration protocol. Computes and stores $reg_i = reg[id]$.

$-$JoinU$(Reg[id], gpk)$. $\mathcal{B}$ plays the honest user. Add id to JIU, if $id \neq id_b^*$, follow the protocol and stores sk_{id}. If $id = id_b^*$, randomly chooses $r_b^* \in \mathbb{Z}_q$ and generates the $H(id_b^*) \leftarrow (m_b'^*, h = f_2^{r_b^*})$. Next, generate $sk_{id_b^*}$. Compute $h_{sk} = h^{sk_{id_b^*}}$, then, call H to simulate the proof π. With the $Reg[id]$ of APS signature, A' generates the proof π_R. Then, set $L[id] = (\pi_R, \sigma', h_{sk}, \pi)$. For the view of $\mathcal{A}$, the distribution of $L[id]$ is indistinguishable with the real protocol. Then, compute a

PS signature on $\mu sk_{id_b^*}$ with a signature from $\mathcal{A}$. The signature is

$$(m'^*, \Sigma_1^* = g_1^{r^*}, \Sigma_2^* = (g_1^{r^*})^{x+y_1\mu sk_{id_b^*}+y_2 m'^*})$$

Set $gsk[id_b^*] \leftarrow (\perp, (m'^*, \Sigma_1^*, \Sigma_2^*))$.

$-$Sign$(id_b^*, item, M)$: For $id = id_b^*$, fetch $gsk[id_b^*] = (\perp, (m'^*, \Sigma_1^*, \Sigma_2^*))$, and pick t to compute $\Sigma = (\Sigma_1', \Sigma_2') = (\Sigma_1^{*^t}, \Sigma_2^{*^t})$. Then, simulate a proof π on μsk_{id^*} and m'^*, return σ. Add $(id, item, M, \sigma)$ to sign query set Q_{Sign}.

$-$Open$_i(item, M, \sigma)$: Add $(item, M, \sigma)$ to Q_{Open}, and Open algorithm is triggered for honest opener. First, by calling Verify to verify the validity of σ. If the signature is valid, outputs a proof $(id, \pi = (h_3, \sigma', \Theta))$. If the signature is forged by $\mathcal{A}$, it is excluded by the Lemma 6.1. If σ is created by an honest user, which has been queried the Sign oracle. $\mathcal{B}$ looks up the list Q_{Sign} to search the corresponding $gsk[id]$. Then, it calls H to simulate a proof

$$\Theta \leftarrow \mathsf{NIZK.Prove}\{(h_2, \kappa):$$

$$e(S, \widehat{Y}_0) = e(d, h_2) \wedge \frac{e(d, h_3)}{e(S, \widehat{Y}_0)} = e(d, \widehat{Y}_0)^{\kappa}\}.$$

Then, return $(id, \pi = (h_3, \sigma', \Theta))$.

$-$GSK(id): Add id to CU, and return $gsk[id]$.

$-$WriteReg(i, id, v): Set $reg_i[id] \leftarrow v$.

$-$ For the challenge $(id_0^*, id_1^*, M, item)$, $\mathcal{B}$ randomly picks $b \in \{0, 1\}$, and retrieves $gsk[id_b^*]$ to compute $\Sigma_1 = f_3, \Sigma_2 = f_3^{x+y_1 m_0'} f_4^{sk_{id_0^*}}$. If the given challenge is a DDH tuple, i.e., $b = 1$, it implies Σ, which has the same distribution with a real signature. If b = 0, the Σ_2 is uniformly distributed in $\mathbb{G}_1$. $\mathcal{B}$ calls oracle H to simulate a signature of knowledge π.

Then, $\mathcal{A}$ will output a guess. $\mathcal{B}$ outputs a guess b'. $\mathcal{B}$ solves the DDH game at least ε advantage.

$$\varepsilon \leq |ID|^2 Adv^{\text{DDH}}_{\mathcal{B}(\mathcal{A})}(\lambda) + \frac{8q}{q-1}|ID| Adv^{\text{SDL}}_{\mathcal{B}(\mathcal{A})}(\lambda) + n_O|ID| Adv^{\text{q-MSDH-1}}_{\lambda}(\mathcal{B}(\mathcal{A})).$$

Therefore,

$$Adv^{ano}_{\text{DTrustRS},\mathcal{A},n_I,n_O}(\lambda) \leq 2(|ID|^2 Adv^{\text{DDH}}_{\mathcal{B}(\mathcal{A})}(\lambda)+$$

$$\frac{8q}{q-1}|ID| Adv^{\text{SDL}}_{\mathcal{B}(\mathcal{A})}(\lambda) + n_O|ID| Adv^{\text{q-MSDH-1}}_{\lambda}(\mathcal{B}(\mathcal{A}))).$$

Traceability. The DTrustRS satisfies traceability if the q-MSDH-1 assumption holds in type-3 pairing groups in the random oracle model.

Proof. If an adversary $\mathcal{A}$ can win the traceability game, then, existing an adversary $\mathcal{B}$ can break the existential unforgeability of PS signature. The security of PS signature is based on the q-MSDH-1 assumption, so we prove the traceability. $\mathcal{B}$ generates an algorithm $\mathcal{A}'$, and utilities the general forking lemma [74], [9]. $\mathcal{A}'$ will output two signatures (σ_0, σ_1), then, $\mathcal{B}$ can forge a PS signature.

Given $params$, public key $\widehat{X}, \widehat{Y}_1, \widehat{Y}_2$ of the PS signature to $\mathcal{A}'$. $\mathcal{A}'$ runs $\mathcal{A}$ as a subroutine, which inputs $(params, \widehat{X}, \widehat{Y}_1, \widehat{Y}_2)$. $\mathcal{A}'$ maintains a list $List_H = (str, c, count)$. $\mathcal{A}'$ answers the oracle queries.

$-H(str \in \{0,1\}^*)$: $\mathcal{A}'$ retrieves str in list $List_H$, and outputs c. If $(str, *) \notin List_H$, $\mathcal{A}'$ selects $c \in \mathbb{Z}_q$ and returns c. Then, it sets $count \leftarrow count + 1$ and stores (str, c, ctr) to $List_H$.

$-H_0()$: $\mathcal{A}'$ queries $\mathcal{B}$.

$-$RegisterU(id). Add id to RU and execute registration protocol. For each identity id computes and generates an APS signature, return $Reg[id] = (\xi, \sigma)$.

$-$RegisterO(id). Add id to RU and CU and execute registration protocol. Compute and store $reg_i = reg[id]$.

$-$JoinU$(Reg[id])$. $\mathcal{A}'$ plays the honest user. Add id to JIU. Follow the protocol to test the signature Σ_2 with

$$e(\Sigma_1, \widehat{X}\widehat{Y}_1^{sk_{id}}\widehat{Y}_2^{m'}) \overset{?}{=} e(\Sigma_2, g_2).$$

If it holds, follow the protocol to get (sk, Σ) and set $gsk[id] = (sk, \Sigma)$; Otherwise, abort the protocol.

$-$IssueI(id). Add id to JIU and CU. The adversary $\mathcal{A}$ generate $L[id]$and sends to $\mathcal{A}'$. $\mathcal{A}'$ internal query $(m', h) \leftarrow H_0(id)$. If verification $\mathsf{NIZK.Verify}(\pi, h, h_{sk}) \overset{?}{=} 1$ succeeds, $\mathcal{A}'$ rewinds $\mathcal{A}$ to extracts sk. If the extraction fail, abort the interaction with $\mathcal{A}$. Otherwise, sends sk to challenger making signing query and receives a signature to answer $\mathcal{A}$.

$-$Sign$(id, item, M)$: For the $id \in JIU \setminus CU$, generate signature σ to answer the signing query. Add $(id, item, M, \sigma)$ to Q_{Sign} and return σ.

$-$Open$(item, M, \sigma)$: Add $(item, M, \sigma)$ to Q_{Open} and run Open algorithm. Return a proof $(id, \pi = (h_3, \sigma', \Theta))$.

$-$GSK(id): Add id to CU and return $gsk[id]$.

$-$ReadReg(i, id): Return $reg_i[id]$.

$\mathcal{A}$ forges a signature σ^* on message M^* with $item^*$. If $\mathsf{Verify}(params, gpk, M^*, \sigma^*, item^*) = 0$ or $\mathsf{Judge}(params, gpk, item^*, M^*, \sigma^*, id^*, \pi) = 0$, abort the interaction with $\mathcal{A}$. If the signature is valid, parse σ^* as (S^*, Σ^*, π^*). Let index j be

jth H queries for $\mathcal{A}$, $\mathcal{A}'$ halts and outputs (j, σ^*) where $j \geq 1$. The $\mathcal{A}$ wins the traceability game of DTrustRS with

$$\widetilde{\varepsilon} \leftarrow \varepsilon - |ID|Adv_{\lambda}^{\text{DLOG}}(\mathcal{B}(\mathcal{A})) - \frac{8q}{q-1}|ID|Adv_{\mathcal{B}(\mathcal{A})}^{\text{SDL}}(\lambda) -$$

$$n_O|ID|Adv_{\lambda}^{\text{q-MSDH-1}}(\mathcal{B}(\mathcal{A}))$$

$\mathcal{B}$ runs $\mathcal{A}'$ and answers its random oracle queries in the assumption that $q > 8tq_H/\varepsilon$, running time at most $8q_H/\widetilde{\varepsilon} \cdot ln(8/\widetilde{\varepsilon}) \cdot \tau_{\mathcal{A}}$, with probability at lest $\widetilde{\varepsilon}/8$. Using generalized forking lemma, $\mathcal{A}'$ returns (σ, σ'), where (σ, σ') were forged signatures which open to $\perp$. Parse $\sigma = (S, \Sigma, \pi = (a, b, \alpha, \beta, \gamma))$ and $\sigma' = (S', \Sigma', \pi' = (a', b', \alpha', \beta', \gamma'))$, where $S = S', \Sigma = \Sigma'$ and $\alpha \neq \alpha'$ and $\alpha \neq \alpha'$ mod q. $\mathcal{B}$ computes $(\gamma - \gamma')/(\alpha' - \alpha) = m'$, $(\beta - \beta')/(\alpha' - \alpha) = sk$, then, $\mathcal{B}$ sends (sk, Σ) to challenger as a forgery of PS signatures. Therefore, $\mathcal{B}$ then wins the existential forgery game with probability at least $\widetilde{\varepsilon}/8$. Therefore,

$$Adv_{\text{DTrustRS},\mathcal{A}}^{trace}(\lambda) \leq 8Adv_{\text{PS},\mathcal{A}}^{\text{EUF-CMA}}(\lambda) + |ID|Adv_{\lambda}^{\text{DLOG}}(\mathcal{B}(\mathcal{A})) +$$

$$\frac{8q}{q-1}|ID|Adv_{\mathcal{B}(\mathcal{A})}^{\text{SDL}}(\lambda) + n_O|ID|Adv_{\lambda}^{\text{q-MSDH-1}}(\mathcal{B}(\mathcal{A}))$$

Public linkability. The DTrustRS satisfies public linkability, if the q-MSDH-1 assumption holds in type 3 pairing groups in the random oracle model..

Proof. If an adversary $\mathcal{A}$ can win the public linkability game, then, existing an adversary $\mathcal{B}$ can break the unforgeability of PS signature. The security of PS signature is based on q-MSDH-1 assumption, so we prove the traceability. $\mathcal{B}$ generates an algorithm $\mathcal{A}'$, and uses the general forking lemma [74], [9]. $\mathcal{A}'$ will output two different signatures (σ_0, σ_1), then, $\mathcal{B}$ can forge a PS signature.

Given $params$, public key $\widehat{X}, \widehat{Y}_1, \widehat{Y}_2$ of the PS signature to $\mathcal{A}'$. $\mathcal{A}'$ runs $\mathcal{A}$ as a subroutine, which inputs $(params, \widehat{X}, \widehat{Y}_1, \widehat{Y}_2)$. $\mathcal{A}'$ maintains a list $List_H = (str, c, count)$. $\mathcal{A}'$ answers the oracle queries.

$-H(str \in \{0,1\}^*)$: $\mathcal{A}'$ retrieves str in list $List_H$, and outputs c. If $(str, *) \notin List_H$, $\mathcal{A}'$ selects $c \in \mathbb{Z}_q$ and returns c. Then, it sets $count \leftarrow count + 1$ and stores (str, c, ctr) to $List_H$.

$-H_0()$: $\mathcal{A}'$ queries $\mathcal{B}$.

$-$RegisterO$_i(id)$. Add id to RU and CU and execute registration protocol. Compute and store $reg_i = reg[id]$.

$-$IssueI(id). Add id to JIU and CU. Follow the protocol, adversary $\mathcal{A}$ gets $gsk[id]$.

$-$GSK(id): Add id to CU, and return $gsk[id]$.

$\mathcal{A}$ outputs $(|CU|+1) = l$ message-signature pairs $((M_1, \sigma_1, item), \cdots, (M_l, \sigma_l, item))$. At least one pair $(M_i, \sigma_i, item)$ is invalid, e.g. $\mathsf{Verify}(params, gpk, M_i, \sigma_i, item) = 0$ or existing at least two publicly linkable signatures for $\mathsf{Link}(params, gpk, item, (M_i, \sigma_i), (M_j, \sigma_j)) = 1$, abort the interaction with $\mathcal{A}$. Otherwise, run the Verify algorithm. Since the signatures are not publicly linkable, there be at least one message-signature pair is valid signature. Parsing σ as (S, Σ, π). Let index j be jth H queries for $\mathcal{A}$, $\mathcal{A}'$ abort and returns (j, σ) with $j \geq 1$. The probability of $\mathcal{A}$ wins the public linkability game of DTrustRS with

$$\widetilde{\varepsilon} \leftarrow \varepsilon - |ID|Adv_{\lambda}^{\text{DLOG}}(\mathcal{B}(\mathcal{A})) - \frac{8q}{q-1}|ID|Adv_{\mathcal{B}(\mathcal{A})}^{\text{SDL}}(\lambda) -$$

$$n_O|ID|Adv_{\lambda}^{\text{q-MSDH-1}}(\mathcal{B}(\mathcal{A}))$$

$\mathcal{B}$ runs $\mathcal{A}'$ and answers its random oracle queries in the assumption that $q >$

$8tq_H/\varepsilon$, running time at most $8q_H/\widetilde{\varepsilon} \cdot ln(8/\widetilde{\varepsilon}) \cdot \tau_A$, with probability at lest $\widetilde{\varepsilon}/8$. By forking lemma, $\mathcal{A}'$ returns (σ, σ') where (σ, σ') were forged signatures. Parse $\sigma = (S, \Sigma, \pi = (a, b, \alpha, \beta, \gamma))$ and $\sigma' = (S', \Sigma', \pi' = (a', b', \alpha', \beta', \gamma'))$, where $S = S', \Sigma = \Sigma'$ and $\alpha \neq \alpha'$ and $\alpha \neq \alpha'$ mod q. $\mathcal{B}$ computes $(\gamma - \gamma')/(\alpha' - \alpha) = m'$, $(\beta - \beta')/(\alpha' - \alpha) = sk$, then, $\mathcal{B}$ sends $(sk, (m', \Sigma))$ to challenger as a forgery of PS signature to win the forgery game.

6.5 Performance Evaluation

In this section, we execute performance evaluation of APS and DTrustRS, then compare with other schemes to analyse the computation complexity.

6.5.1 Efficiency Analysis

We evaluate the cryptographic algorithms of APS and DTrustRS based on Intel(R) Core(TM) i5-2450M CPU 2.50 GHz 4.00GB RAM by windows 7 OS with the Miracl library [1]. We execute the cryptographic algorithms for 1000 rounds to get an execution average time. The time cost of APS is showed in Figure 6.2. The AggSign algorithm effectively compresses the time of sign.

We test the time cost of DTrustRS scheme, as shown in Figure 6.3. The time cost of Sign and Verify algorithms is constant and the time cost is millisecond.

6.5.2 Computation Complexity

We compare DTrustRS with the group signatures and anonymous reputation system in the signature's size and the computation complexity of signature algorithm

[1]https://certivox.org/display/EXT/MIRACL

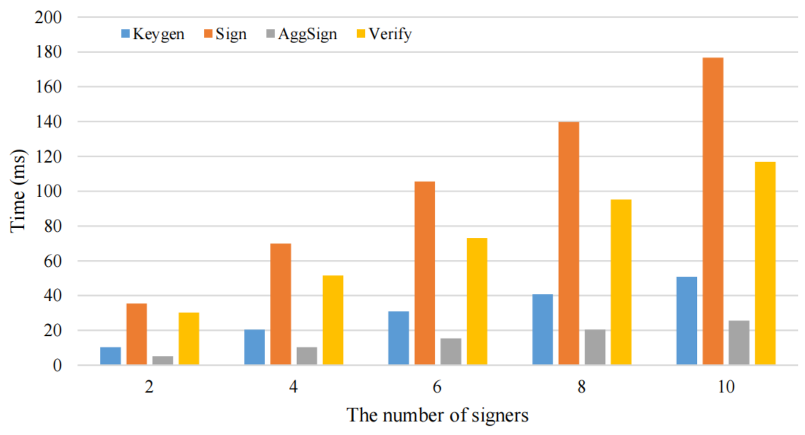


Figure 6.2: The time cost of APS

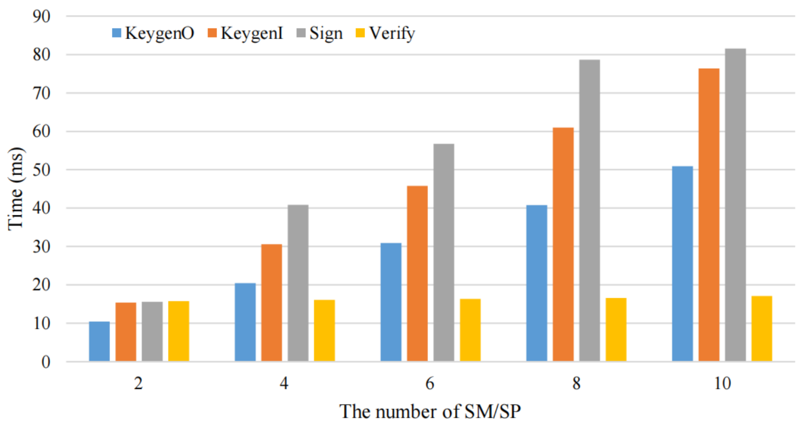


Figure 6.3: The time cost of DTrustRS

and verify algorithm. As shown in Table 6.9.

We compare APS with other signature schemes, including signature size, computation cost and signature verification. The integer k is the number of message blocks and the integer n is the number of signers. We also compare DTrustRS with other group signatures schemes and anonymous reputation system in term of signature size, and cost of generating and verify the signature. $\mathbb{G}_1^l$ denotes l-exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2^l$ denotes l-exponentiation in $\mathbb{G}_2$ and $\mathbb{G}_T^l$ denotes l-exponentiation in $\mathbb{G}_T$. P denotes the pairing operations and P^l denotes to product l pairing values.

Due to the re-randomize property, CL signatures [23] have been applied in many applications, such as electronic cash [26]. As a substitute of CL signatures, PS [72] signatures have the re-randomize property and further reduce the signature size. PSM [25] provided a multi-signature version for PS signatures. Our signature schemes APS provide similar functions, but our scheme can aggregate multiple signatures of multiple messages, which sign size is linear with the number of signers not the message blocks for CL signatures. Bichsel et al. [13] and Pointcheval et al. group signatures in [73] Appendix A.1 follow re-randomizable signatures paradigm to reduce the signature size and DTrustRS sign size additional to $1\mathbb{Z}_q$ and $1\mathbb{G}_1$, but DTrustRS supports distributed issuance and opening. Sonnino et al. [83] anonymous credential systems support threshold issuance but their did not devote to distributed opening, at the same time, its signature exist additional $1\mathbb{G}_2$ element. Camenisch et al. [25] proposed threshold dynamic group signatures which support threshold/distributed issuance and threshold opening. This scheme is shorter than DTrustRS, but DTrustRS makes sure distributed issuance and distributed opening with the function of publicly linkable. The sign size of DTrustRS

Scheme	Sign.size	Sign	Verify
CL [23]	$(2k+3)\mathbb{G}_1$	$(2k+1)\mathbb{G}_1 + 1\mathbb{G}_1^{k+1}$	$(k+1)\mathbb{G}_1 + (4k+3)P + 1P^{k+2}$
PS [72]	$3\mathbb{G}_1$	$1\mathbb{G}_1$	$1\mathbb{G}_2^{k+1} + 2P$
PSM [25]	$3\mathbb{G}_1$	$1\mathbb{G}_1^n$	$1\mathbb{G}_2^{k+1} + 2P$
APS	$(n+1)\mathbb{G}_1$	$1\mathbb{G}_1^n$	$1\mathbb{G}_2^n + 1P + 1P^n$
Bichsel et al. [13]	$3\mathbb{G}_1 + 2\mathbb{Z}_q$	$3\mathbb{G}_1 + 1\mathbb{G}_T$	$1\mathbb{G}_1 + 1\mathbb{G}_1^2 + 2P^2$
Pointcheval et al. [73]	$2\mathbb{G}_1 + 2\mathbb{Z}_q$	$2\mathbb{G}_1 + 1\mathbb{G}_T$	$3\mathbb{G}_1 + 1P^3$
Sonnino et al. [83]	$3\mathbb{G}_1 + 1\mathbb{G}_2 + 3\mathbb{Z}_q$	$4\mathbb{G}_1 + 1\mathbb{G}_2^3 + 1\mathbb{G}_2^2$	$1\mathbb{G}_1 + 1\mathbb{G}_1^2 + 1\mathbb{G}_2^4 + P^2$
Camenisch et al. [25]	$2\mathbb{G}_1 + 3\mathbb{Z}_q$	$4\mathbb{G}_1 + 1P^2$	$4\mathbb{G}_1 + 1P^4$
Garms et al. [46]	$5\mathbb{G}_1 + 7\mathbb{Z}_q$	$3\mathbb{G}_1 + 7\mathbb{G}_1^2$	$1\mathbb{G}_1^2 + 3\mathbb{G}_1^3$
Blömer et al. [16]	$5\mathbb{G}_1 + 8\mathbb{Z}_q$	$13\mathbb{G}_1 + 2\mathbb{G}_1^2 + 1P^4$	$6\mathbb{G}_1 + 6\mathbb{G}_1^2 + 1P^6$
DTrustRS	$3\mathbb{G}_1 + 3\mathbb{Z}_q$	$6\mathbb{G}_1 + 1P^2$	$4\mathbb{G}_1 + 1\mathbb{G}_1^2 + 1P^4$

Table 6.9: Comparison of APS and DTrustRS with other schemes

is shorter than Garms et al. [46] group signatures and Blömer et al. scheme [16]. Blömer et al. [16] anonymous reputation system supports publicly linkable but needs a single trusted system manager to maintain the system.

We compare and test the time cost of DTrustRS with the Blömer et al. [16] reputation system with the different number of server providers. As shown in Figures 6.4 and 6.5. The time cost of Sign and Verify algorithms of DTrustRS is smooth. But the time cost of Sign and Verify algorithms of [16] increases with the number of service providers.

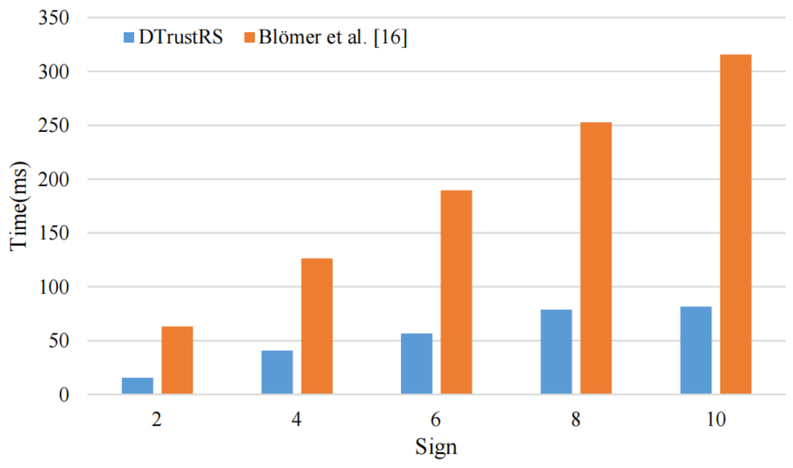


Figure 6.4: The time cost of sign algorithm DTrustRS and Blömer et al. [16]

6.6 Summary

In the chapter, we propose DTrustRS, an anonymous and publicly linkable reputation system with distributed trust, which frees anonymous and publicly linkable reputation system from relying on a single authority. DTrustRS distributes the role of system manager over several entities and enhances the robustness of reputation

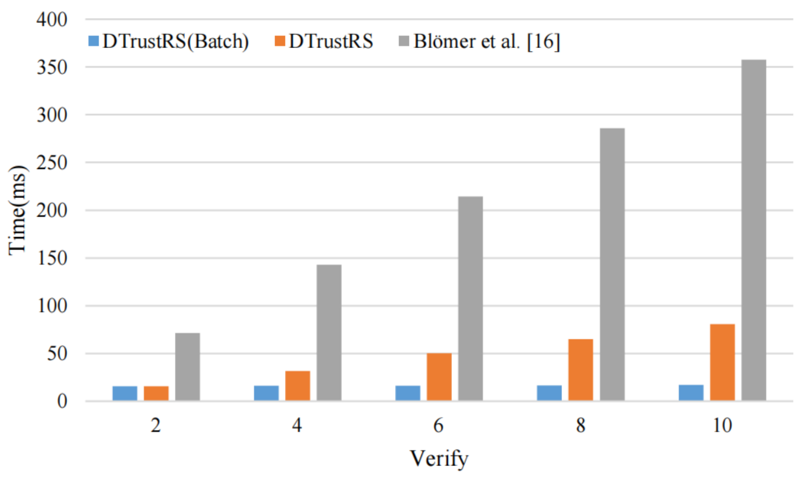


Figure 6.5: The time cost of verify algorithm DTrustRS and Blömer et al. [16]

system. We define the system model of DTrustRS and formalize its security. Then, we provide an efficient construction and prove its security in the random-oracle model under a q-type assumption. We evaluate DTrustRS scheme and compare with other schemes to demonstrate its validity.

Chapter 7

Conclusion

As a valuable resource, data has become an important factor of production. The use of data brings convenience to human beings, but there are also some problems to be solved, such as data privacy leakage and poor data reliability. However, we cannot completely influence the use of data for privacy protection, thus, it is essential to make tradeoffs between data privacy and data utility. Against this backdrop, this thesis investigate protocols and applications in privacy-preserving data computing and anonymous authentication. The research of privacy-preserving technologies have been widely concerned, since it can provide technical support for data security. Privacy-preserving data computing enables data to be analyzed and computed while being protected from disclosure, anonymous authentication can effectively ensure the reliability and integrity of data while protecting the personal identity privacy. The work of this thesis is summarized as follows.

- MPC is a significant means to realize privacy computing, PSI is a widely concerned problem in MPC. We propose a lightweight multi-point OPRF based delegated PSI-CA protocol, and apply it to build a privacy-preserving contact tracing system named PC-CONTrace. Our system supports publicly check property, can resist the collusion of any number of cloud servers and is more adaptable and advantageous in densely populated areas.

- Federated learning is an important machine learning technique. Unlike traditional machine learning approaches, federated learning allows data to be decentralized, and a group of organizations or groups within the same organization can train and improve the shared global machine learning models in a collaborative and iterative manner. We present an accountable and verifiable secure aggregation for federated learning framework base on HPRA, HPRE and blockchain, which realizes the protection of the confidentially and verifiability of the client data. The use of HPRE and HPRA allows our scheme to require only semi-honest aggregator and can be resistant to arbitrary dropouts.

- Reputation systems help users evaluate information quality and incentivize civilized behavior, often by tallying feedback from other users. Anonymous reputation systems protects the user's identity privacy in the process of evaluation. We introduce an anonymous and publicly linkable reputation system with distributed trust, that is DTrustRS, the registration function of which is achieved by our scheme APS. DTrustRS can reduce the trust in a single authority in anonymous and publicly linkable reputation system and enable providers to verify ratings in batches.

7.1 Future Work

Due to the different adversaries and application environments in reality, some problems warrant further investigations.

Facing the current complex epidemic situation, updatable PSI-CA protocol is clearly more advantageous and desirable. It is valuable to design efficient, practical and updatable PSI-CA protocols based on appropriate data structures and cryptographic tools and apply them to updatable contact tracing systems.

Since the research on secure aggregation for FL is still in the exploratory stage, our construction is not yet capable of handling complex data models. In addition, there is no protection in our scheme to against attacks such as poisoning and collusion of aggregator and server. It is worth studying to combine cryptography with TEE or other privacy-preserving technologies to construct more practical secure aggregation schemes for different attacks and data models.

In the open environment, different application scenarios have different requirements for reputation system, our reputation system construction is not flexible enough. Fine-grained, designated constructions are more realistic and practical. How to increase the server's effective control over reputation data, achieve fine-grained control over tracing, and how to selectively design functionalities deserves further research.

Reference

[1] *2022 China Privacy Preserving Computing Industry Report*. `https://www.iresearchchina.com/content/details8_69794.html`.

[2] Aydin Abadi, Sotirios Terzis, and Changyu Dong. “VD-PSI: verifiable delegated private set intersection on outsourced private datasets”. In: *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*. Springer. 2017, pp. 149–168.

[3] Aydin Abadi et al. “Efficient Delegated Private Set Intersection on Outsourced Private Datasets”. In: *IEEE Transactions on Dependable and Secure Computing* 16.04 (2019), pp. 608–624.

[4] Elli Androulaki et al. “Reputation systems for anonymous networks”. In: *Privacy Enhancing Technologies: 8th International Symposium, PETS 2008 Leuven, Belgium, July 23-25, 2008 Proceedings 8*. Springer. 2008, pp. 202–218.

[5] *Apple and google privacy-preserving contact tracing*. `https://www.apple.com/covid19/contacttracing.`.

[6] Giuseppe Ateniese et al. “Improved proxy re-encryption schemes with applications to secure distributed storage”. In: *ACM Transactions on Information and System Security (TISSEC)* 9.1 (2006), pp. 1–30.

[7] Man Ho Au et al. “Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction”. In: *Theoretical Computer Science* 469 (2013), pp. 1–14.

[8] Sana Awan et al. “Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2561–2563.

[9] Ali Bagherzandi, Jung-Hee Cheon, and Stanislaw Jarecki. “Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma”. In: *Proceedings of the 15th ACM conference on Computer and communications security*. 2008, pp. 449–458.

[10] Pierre Baldi, Roberta Baronio, et al. “Countering gattaca: efficient and secure testing of fully-sequenced human genomes”. In: *Proc on Computer and communications security*. 2011, pp. 691–702.

[11] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. “Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions”. In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2003, pp. 614–629.

[12] Mihir Bellare, Haixia Shi, and Chong Zhang. “Foundations of group signatures: The case of dynamic groups”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2005, pp. 136–153.

[13] Patrik Bichsel et al. “Get shorty via group signatures without encryption”. In: *International Conference on Security and Cryptography for Networks*. Springer. 2010, pp. 381–398.

[14] Johannes Blömer, Jan Bobolz, and Laurens Porzenheim. “A Generic Construction of an Anonymous Reputation System and Instantiations from Lattices”. In: *Cryptology ePrint Archive* (2023).

[15] Johannes Blömer, Fabian Eidens, and Jakob Juhnke. “Practical, anonymous, and publicly linkable universally-composable reputation systems”. In: *Topics in Cryptology–CT-RSA 2018: The Cryptographers’ Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*. Springer. 2018, pp. 470–490.

[16] Johannes Blömer, Jakob Juhnke, and Christina Kolb. “Anonymous and publicly linkable reputation systems”. In: *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*. Springer. 2015, pp. 478–488.

[17] Johannes Blömer, Jakob Juhnke, and Nils Löken. “Short group signatures with distributed traceability”. In: *International Conference on Mathematical Aspects of Computer and Information Sciences*. Springer. 2015, pp. 166–180.

[18] Carlo Blundo, Emiliano De Cristofaro, and Paolo Gasti. “Espresso: efficient privacy-preserving evaluation of sample set similarity”. In: vol. 22. 3. IOS Press, 2014, pp. 355–381.

[19] Keith Bonawitz et al. “Practical secure aggregation for privacy-preserving machine learning”. In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191.

[20] Dan Boneh et al. “Signing a linear subspace: Signature schemes for network coding”. In: *International workshop on public key cryptography*. Springer. 2009, pp. 68–87.

[21] Pedro Branco et al. “Roted: Random oblivious transfer for embedded devices”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), pp. 215–238.

[22] Prasad Buddhavarapu, Andrew Knox, et al. “Private Matching for Compute.” In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 599.

[23] Jan Camenisch and Anna Lysyanskaya. “Signature schemes and anonymous credentials from bilinear maps”. In: *Annual international cryptology conference*. Springer. 2004, pp. 56–72.

[24] Jan Camenisch and Markus Stadler. “Efficient group signature schemes for large groups”. In: *Annual International Cryptology Conference*. Springer. 1997, pp. 410–424.

[25] Jan Camenisch et al. “Short threshold dynamic group signatures”. In: *International Conference on Security and Cryptography for Networks*. Springer. 2020, pp. 401–423.

[26] Sébastien Canard et al. "Divisible e-cash made practical". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2015, pp. 77–100.

[27] Justin Chan, Dean Foster, et al. "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing". In: *arXiv preprint arXiv:2004.03544* (2020).

[28] Melissa Chase and Peihan Miao. "Private set intersection in the internet setting from lightweight oblivious PRF". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 34–63.

[29] David Chaum and Eugène van Heyst. "Group signatures". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1991, pp. 257–265.

[30] Hao Chen, Zhicong Huang, et al. "Labeled PSI from fully homomorphic encryption with malicious security". In: *Proc on Computer and Communications Security*. 2018, pp. 1223–1237.

[31] Sebastian Clauß, Stefan Schiffner, and Florian Kerschbaum. "K-anonymous reputation". In: *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. 2013, pp. 359–368.

[32] Ronald Cramer, Ivan Bjerre Damgård, et al. *Secure multiparty computation*. Cambridge University Press, 2015.

[33] Emiliano De Cristofaro and Gene Tsudik. "Practical private set intersection protocols with linear complexity". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2010, pp. 143–159.

[34] Chrysanthos Dellarocas. “Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior”. In: *Proceedings of the 2nd ACM Conference on Electronic Commerce*. 2000, pp. 150–157.

[35] Daniel Demmler, Peter Rindal, et al. “PIR-PSI: Scaling Private Contact Discovery.” In: *Proc. Priv. Enhancing Technol.* 2018.4 (2018), pp. 159–178.

[36] David Derler, Sebastian Ramacher, and Daniel Slamanig. “Homomorphic proxy re-authenticators and applications to verifiable multi-user data aggregation”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2017, pp. 124–142.

[37] David Derler, Sebastian Ramacher, and Daniel Slamanig. “Short double- and n-times-authentication-preventing signatures from ECDSA and more”. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, pp. 273–287.

[38] Marios D Dikaiakos et al. “Location-aware services over vehicular ad-hoc networks using car-to-car communication”. In: *IEEE Journal on Selected Areas in Communications* 25.8 (2007), pp. 1590–1602.

[39] Changyu Dong, Liqun Chen, et al. “When private set intersection meets big data: an efficient and scalable protocol”. In: *Proc on Computer & communications security*. 2013, pp. 789–800.

[40] Thai Duong, Duong Hieu Phan, et al. “Catalic: Delegated psi cardinality with applications to contact tracing”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 870–899.

[41] Yunyi Fang et al. “Blockchain-based privacy-preserving valet parking for self-driving vehicles”. In: *Transactions on Emerging Telecommunications Technologies* 32.4 (2021), e4239.

[42] Amos Fiat and Adi Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194.

[43] Michael J Freedman, Yuval Ishai, et al. “Keyword search and oblivious pseudorandom functions”. In: *Theory of Cryptography Conference*. Springer. 2005, pp. 303–324.

[44] Michael J Freedman, Kobbi Nissim, et al. “Efficient private matching and set intersection”. In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2004, pp. 1–19.

[45] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. “{HAWatcher}:{Semantics-Aware} Anomaly Detection for Appified Smart Homes”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 4223–4240.

[46] Lydia Garms and Anja Lehmann. “Group signatures with selective linkability”. In: *IACR International Workshop on Public Key Cryptography*. Springer. 2019, pp. 190–220.

[47] Essam Ghadafi. “Efficient distributed tag-based encryption and its application to group signatures with efficient distributed traceability”. In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2014, pp. 327–347.

[48] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[49] Yan Huang, David Evans, et al. "Private set intersection: Are garbled circuits better than custom protocols?" In: *NDSS*. 2012.

[50] Mihaela Ion, Ben Kreuter, et al. "On deploying secure computing: Private intersection-sum-with-cardinality". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2020, pp. 370–389.

[51] Yuval Ishai, Joe Kilian, et al. "Extending oblivious transfers efficiently". In: *Annual International Cryptology Conference*. Springer. 2003, pp. 145–161.

[52] Swanand Kadhe et al. "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning". In: *arXiv preprint arXiv:2009.11248* (2020).

[53] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[54] Florian Kerschbaum. "Outsourced private set intersection using homomorphic encryption". In: *Proc on Information, Computer and Communications Security*. 2012, pp. 85–86.

[55] Vladimir Kolesnikov, Ranjit Kumaresan, et al. "Efficient batched oblivious PRF with applications to private set intersection". In: *Proc on Computer and Communications Security*. 2016, pp. 818–829.

[56] Yehuda Lindell. "Anonymous authentication". In: *Journal of Privacy and Confidentiality* 2.2 (2011).

[57] Dongxiao Liu et al. "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain". In: *IEEE Transactions on Industrial Informatics* 15.6 (2019), pp. 3527–3537.

[58] Rongxing Lu, Hui Zhu, et al. "Toward efficient and privacy-preserving computing in big data era". In: *IEEE Network* 28.4 (2014), pp. 46–50.

[59] Dominik Lucke, Carmen Constantinescu, and Engelbert Westkämper. "Smart factory-a step towards the next generation of manufacturing". In: *Manufacturing systems and technologies for the new frontier*. Springer, 2008, pp. 115–118.

[60] P Madhusudan, Ling Ren, and VN Venkatakrishnan. *Privacy-Preserving Secure Contact Tracing*. 2020.

[61] Sergio Marti and Hector Garcia-Molina. "Identity crisis: anonymity vs reputation in P2P systems". In: *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*. IEEE. 2003, pp. 134–141.

[62] Catherine Meadows. "A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party". In: *1986 IEEE Symposium on Security and Privacy*. IEEE. 1986, pp. 134–134.

[63] Ghita Mezzour, Adrian Perrig, et al. "Privacy-preserving relationship path discovery in social networks". In: *International Conference on Cryptology and Network Security*. Springer. 2009, pp. 189–208.

[64] Arvind Narayanan, Narendran Thiagarajan, et al. "Location privacy via private proximity testing." In: *NDSS*. Vol. 11. 2011.

[65] Jiguo Li Ningyu Chen et al. "Efficient CP-ABE Scheme With Shared Decryption in Cloud Storage". In: *IEEE Transactions on Computers* 71.1 (2022), pp. 175–184.

[66] Chintan Patel and Nishant Doshi. “Secure Lightweight Key Exchange Using ECC for User-Gateway Paradigm”. In: *IEEE Transactions on Computers* 70.11 (2021), pp. 1789–1803.

[67] Benny Pinkas, Mike Rosulek, et al. “Spot-light: Lightweight private set intersection from sparse ot extension”. In: *Annual International Cryptology Conference*. Springer. 2019, pp. 401–431.

[68] Benny Pinkas, Thomas Schneider, et al. “Scalable private set intersection based on OT extension”. In: *ACM Transactions on Privacy and Security (TOPS)* 21.2 (2018), pp. 1–35.

[69] Benny Pinkas, Thomas Schneider, and Michael Zohner. “Faster private set intersection based on {OT} extension”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 797–812.

[70] Benny Pinkas et al. “Efficient circuit-based PSI via cuckoo hashing”. In: *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part III 37*. Springer. 2018, pp. 125–157.

[71] Benny Pinkas et al. “Efficient circuit-based PSI with linear communication”. In: *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*. Springer. 2019, pp. 122–153.

[72] David Pointcheval and Olivier Sanders. “Reassessing security of randomizable signatures”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2018, pp. 319–338.

[73] David Pointcheval and Olivier Sanders. “Short randomizable signatures”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2016, pp. 111–126.

[74] David Pointcheval and Jacques Stern. “Security arguments for digital signatures and blind signatures”. In: *Journal of cryptology* 13.3 (2000), pp. 361–396.

[75] *Privacy-safe contact tracing using bluetooth low energy*. `https://blog.google/documents/57/Overview$of$COVID-19$Contact$Tracing$Using$BLE.pdf/`.

[76] Shuo Qiu, Jiqiang Liu, et al. “Identity-based private matching over outsourced encrypted datasets”. In: *IEEE Transactions on cloud Computing* 6.3 (2015), pp. 747–759.

[77] Ramesh Raskar, Deepti Pahwa, et al. “Contact Tracing: Holistic Solution Beyond Bluetooth.” In: *IEEE Data Eng. Bull.* 43.2 (2020), pp. 67–70.

[78] *Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach*. `https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election`.

[79] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. “Trusted execution environment: what it is, and what it is not”. In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE. 2015, pp. 57–64.

[80] Claus-Peter Schnorr. “Efficient identification and signatures for smart cards”. In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 239–252.

[81] *Seagete Data analysis report*. `https://www.seagate.com/our-story/data-age-2025/`.

[82] Jinhyun So, Başak Güler, and A Salman Avestimehr. “Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning”. In: *IEEE Journal on Selected Areas in Information Theory* 2.1 (2021), pp. 479–489.

[83] Alberto Sonnino et al. “Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers”. In: *arXiv preprint arXiv:1802.07344* (2018).

[84] *The Most Recent Data Breaches – August 2022*. `https://firewalltimes.com/recent-data-breaches/`.

[85] *The Top 10 Data Breaches of 2020*. `https://www.securitymagazine.com/articles/94076-the-top-10-data-breaches-of-2020`.

[86] Carmela Troncoso, Mathias Payer, et al. “Decentralized privacy-preserving proximity tracing”. In: *arXiv preprint arXiv:2005.12273* (2020).

[87] Stacey Truex et al. “A hybrid approach to privacy-preserving federated learning”. In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 2019, pp. 1–11.

[88] Xinlei Oscar Wang et al. “Artsense: Anonymous reputation and trust in participatory sensing”. In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 2517–2525.

[89] Runhua Xu et al. “Hybridalpha: An efficient approach for privacy-preserving federated learning”. In: *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 2019, pp. 13–23.

[90] Ennan Zhai et al. “Anonrep: Towards tracking-resistant anonymous reputation”. In: *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*. 2016, pp. 583–596.

[91] Keshan Zhang et al. “Blockchain-based participant selection for federated learning”. In: *International Conference on Blockchain and Trustworthy Systems*. Springer. 2020, pp. 112–125.

[92] Qiong Zhang et al. “Blockchain-based secure aggregation for federated learning with a traffic prediction use case”. In: *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE. 2021, pp. 372–374.

[93] Dong Zheng et al. “Democratic group signatures with threshold traceability”. In: *Cryptology ePrint Archive* (2008).