RING SIGNATURE CONSTRUCTIONS AND APPLICATIONS BASED ON DISCRETE LOGARITHM AND LATTICE ASSUMPTIONS

CHEN XUE

MPhil

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University
Department of Computing

Ring Signature Constructions and Applications based on Discrete Logarithm and Lattice Assumptions

Chen Xue

A thesis submitted in partial fulfillment of the requirements for
the degree of Master of Philosophy
April 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Chen Xue

Abstract

Ring signatures, in comparison to most digital signature schemes, have garnered significant attention due to their strong anonymity alongside unforgeability. The compelling attributes of ring signatures have garnered substantial scholarly interest in the field of security and privacy. Consistent with this trend, ring signatures are widely applied across a broad spectrum of privacy-preserving scenarios.

Therefore, this thesis is dedicated to the study of ring signature constructions and applications, encompassing Discrete Logarithm (DL)-based and lattice-based ring signatures.

DL-based Ring Signature Construction and Application

Currently, various Σ-based standard signature schemes serve a wide range of privacy-preserving scenarios. However, these schemes are prone to leakage of the signer's identity information due to a lack of consideration for user anonymity. To address this issue, the ring signature is considered a desirable alternative, as it allows any ring member to generate a signature on a message without revealing the signer's identity. But to our best knowledge, in the aforementioned privacy-preserving scheme, replacing the original standard signature with existing (non-standardized) ring signatures would result in the inability to maintain the original properties. Thus, proposing a general mechanism to convert a standardized Σ-based signature to a ring signature is far-reaching.

The first contribution of this thesis is that we propose a general construction for converting Σ-based signatures into ring signatures. To achieve this, we initially introduce a Σ-based general model, providing a generic transformation to convert existing Σ-based signatures into a Σ-protocol form. Subsequently, we incorporate our redesigned one-out-of-many relation within our general model and proceed to devise a general ring signature leveraging on the Fiat-Shamir heuristic. Furthermore, to enhance the efficiency, we employ the Bulletproofs folding technique, enabling the attainment of the logarithmic size ring signature. To showcase the broad range of uses for our general construction, we provide four prominent signatures as case studies. Ultimately, we conduct a security analysis and experimental evaluation of our proposed ring signature. The signing and verifying times are 0.38 - 0.83 times and 0.23 - 0.65 times compared to existing ring signatures, respectively. Our general ring signature exhibits the lowest storage overhead compared to others.

Lattice-based Ring Signature Construction and Application

Ring signatures have been extensively researched for Cloud-assisted Electronic Medical Records (EMRs) sharing, aiming to address the challenge of "medical information silos" while safeguarding the privacy of patients' personal information and the security of EMRs. However, most existing EMRs sharing systems that utilize ring signatures are vulnerable to quantum attacks, posing a severe challenge. To alleviate this issue, some studies have been conducted on lattice-based ring signatures. Nevertheless, there still exist two challenges. Firstly, current schemes fail to verify if multiple EMRs come from the same signer, undermining e-health reliability. Additionally, adversaries can exploit weaknesses in the network security of signers' secret keys to forge signatures. Therefore, it is essential to design a lattice-based ring signature for the EMRs sharing system to tackle the mentioned challenges.

The second contribution is that we propose an efficient lattice-based linkable ring signature (LLRS) for EMRs sharing to ensure patient privacy through anonymity, EMRs security through unforgeability, and checking the linkability for multiple sig-

natures. We then present an enhancement scheme, called FS-LLRS, to additionally offer forward security, ensuring the security of previous ring signatures even if the current key leaked. To achieve this, we introduce a binary tree to divide time and leverage the ExtBasis algorithm to update the secret keys periodically. Ultimately, we conduct a rigorous security analysis and compare our primitives with prior arts. In computational cost, the best performance of our LLRS and FS-LLRS schemes are just 0.17 and 0.34 times compared to others, respectively. Our LLRS scheme only incurs 0.08 times the communication overhead of others.

Acknowledgments

This thesis was finished with the guidance of my chief supervisor and co-supervisor, the discussion with my research partners, the help and encouragement of many persons, and the constant effort by myself. Thanks to these people, I am lucky enough and have gained a lot. Thus, deeply from my heart, it is truly an honor for me to thank everyone who has helped me, supported me, and encouraged me, on the road of pursuing my M.Phil. degree.

First and foremost, I wish to express my sincere and heartful gratitude to my chief supervisor, Dr. Gao Shang, and my co-supervisor, Prof. Xiao Bin. Many thanks for their constant assistance and leadership during my academic journey. In particular, I am grateful to Prof. Xiao and Dr. Gao for seeing my potential and admitting me to pursue the M.Phil. degree under their supervision. Their recognition, understanding, support, and encouragement in my academic journey are greatly appreciated. Their kind help and insightful recommendations served as crucial to me during my M.Phil. research process. Their comprehensive knowledge of cryptography and information security, along with their serious academic attitude to research, has left a profound and enduring influence on me. My heartfelt thanks go out to my supervisors, who have been extremely helpful to me as I have conducted my research.

Furthermore, I wish to thank my parents and family for their unlimited support and love in every aspect. I am grateful for my parents' unwavering financial support. Whenever I encountered difficulties, they always provided me with unlimited comfort

and love, offering me emotional sustenance. I also regret not being able to be with my parents and family during my studies in Hong Kong. The love from my parents and family has given me endless motivation. I will carry their hopes and go further on the academic journey.

Moreover, I want to thank my boyfriend, who is also my research partner, Mr. Shiyuan Xu. I am grateful for his guidance and help in my academic pursuits. I really thank his love, understanding, and encouragement. Throughout our two years of studying away from our hometown, Beijing, we have supported and accompanied each other. For all that he has done for me, I am truly thanks and warmed.

Besides, I wish to thank my research colleagues. I appreciate all of their help and the worthwhile discussions we had. I am fortunate to have such wonderful friends, colleagues, and study partners in my research endeavors.

At this point, I am filled with mixed emotions. In the meantime, I would like to thank myself. I have struggled and felt helpless when confronted with the 'reject' along the road, but I have never given up and have made an effort to overcome the challenges. I am appreciative of my perseverance and will to always pursue excellence. I am appreciative of all the many days and nights I have dedicated to studying and hard work. I am fully conscious of my shortcomings, and I am committed to pushing myself even harder in future studies. Hope me to keep with my original heart. May I always be filled with passion and never stop striving for greatness!

To put it briefly, I have profited from far too many people to name them all here. Thanks to everyone who has helped me. I hope all of you bright future ahead!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent decades, ring signatures have been extensively researched and widely applied in privacy-preserving systems to safeguard data security and user privacy. Ring signature, a cryptographic primitive, was first proposed by Rivest et al. (ASIACRYPT 2001) [60], and ensures the anonymity of the signer within a dynamically formed ring. The signing process does not reveal the signatory's identity, and each involved individual holds a key pair, comprising a secret key (aka. signing key) and a public key (aka. verification key). In 2004, Abe et al. [1] presented a novel approach for constructing a generic ring signature, enabling the use of any hash-and-sign signature or Σ-protocol based signature scheme. The superior anonymity and unforgeability of ring signatures have sparked extensive research in the contents of privacy-preserving fields.

This thesis is dedicated to researching the constructions and applications of ring signatures, focusing on two aspects: ring signatures based on DL hardness and ring signatures based on lattice hardness.

1.1 DL-based Ring Signature Construction and Application

Digital signatures [32] are a vital component of modern cryptography [41, 11]. They have been serving as an essential tool in ensuring computer security [30, 6, 27] and the authenticity, integrity, and unforgeability of transmitted data. Signatures possess diverse properties, prominently utilized in vehicular networks [72, 59], government announcements [54], electronic medical record sharing [21], financial sector [43], the blockchain-based system [62, 70], and other aspects. Nowadays, various Σ-based standard signatures are in the spotlight due to their structural conciseness, e.g., Schnorr signatures [57, 61] and ECDSA signatures [39], etc.

Recently, numerous privacy-preserving systems leveraging Σ-based standard signatures have been proposed [55, 47, 26, 28, 37, 58] to guarantee data security. However, they ignored user anonymity, which is crucial in privacy-preserving systems. For example, in the e-health [44], guaranteeing the users' anonymity can protect the privacy of their private medical records. As for e-voting [51], preserving users' anonymity can establish a credible and fair election process. Thus, missing anonymity will leak the users' private information and compromise the equitability of privacy-preserving systems.

To tackle this concern, scholars have conducted a large number of studies and found that ring signature [60, 24, 9] seems to be a feasible choice due to its strong anonymity and unforgeability [20, 75]. In line with this trend, numerous ring signatures have been proposed [33, 12, 29]. Unfortunately, replacing an original standard signature in privacy-preserving systems with a newly selected ring signature to provide user anonymity can be cumbersome, and may result in the loss of advantages that the original signature offered, i.e. lower time and storage overhead. More importantly, it should be noted that a large number of ring signatures have not yet been standardized

(as we will discuss later), which means that multiple parties would need to agree on the same scheme and implementation.

Hence, it becomes imperative to propose a general construction that allows for the conversion from existing (standardized) signatures to ring signatures. Through this technique, scholars can directly enhance the security level of their privacy-preserving systems by converting a signature they currently use into a ring signature. As far as we know, no established method has been proposed to achieve this transformation until now.

1.1.1 Related Works

Recently, there has been an increasing scholarly focus on ring signatures. In 2015, Groth et al. [33] introduced one-out-of-many proofs, utilizing the binary Kronecker's delta vector to present the $O(\log n)$ ring signature scheme with $O(n \log n)$ exponentiation costs. Subsequently, Bootle et al. [12] conducted optimizations on Groth et al.'s scheme [33] and introduced an accountable ring signature that exhibits similar performance to the aforementioned scheme. In 2021, Yuen et al. [74] presented a $O(\log n)$ ring signature with $O(n)$ exponentiation cost, which is regarded as the optimal scheme till now. The scheme of Feng et al. [29] can be seen as a case study of Yuen et al. [74].

Despite the availability of various constructions for ring signatures, there does not exist a standardized framework established by institutions. Since ring signatures are not being standardized, both parties are still required to negotiate and communicate regarding certain realistic details. It leads to less generic implementations and presents bottlenecks in the pursuit of enhancing the security of privacy-preserving schemes.

Moreover, it is worth noting that existing schemes adopt a range of signatures based on Σ-protocols, e.g. ECDSA signature [39], Schnorr signature [57, 61], EdDSA signa-

ture [40] and SM2 signature [5], which unfortunately fail to provide user anonymity. If scholars aim to enhance the security of their systems, e.g. anonymity, they are required to choose an existing ring signature and integrate it into their privacy-preserving system. However, this process can be complex, and the chosen ring signature may not be suitable for their specific requirements.

Building on the above, the challenge addressed in Chapter 3 is to develop a general construction approach that enables the conversion from existing standardized Σ-based signatures to ring signatures. Additionally, we offer four classical case studies to construct standard ring signatures by our method to show its applications.

1.2 Lattice-based Ring Signature Construction and Application

Cloud-assisted Electronic Medical Records (EMRs) sharing systems have become increasingly popular as a convenient and effective method for medical institutions to share patients' medical information [22, 65]. Meanwhile, it also brings the users' privacy and the EMRs security issues, i.e., the disclosure of patients' private information and the malicious forgery of EMRs [67, 69, 42, 19]. With regard to these concerns, a cryptographic technique, ring signature [60, 24, 9, 23], has garnered extensive attention for its strong anonymity and unforgeability. Using the ring signature in the medical data sharing process can safeguard patients' privacy and ensure the unforgeability of medical data.

However, most existing ring signatures utilized in medical data sharing systems [50, 44, 34] are based on classical assumptions. Unfortunately, the emergence of quantum computer [66, 36] has led to the proliferation of quantum algorithms [25, 2], posing a serious risk to classical cryptography. In particular, the Shor algorithm [63, 64] can efficiently compute mathematical and discrete logarithmic decomposi-

tions in polynomial time and attackers can employ Grover's algorithm [35] to extract user keys. In this way, these existing schemes are susceptible to quantum attacks, significantly impacting the security of medical data sharing systems.

To tackle this, lattice-based cryptography [53] is considered a more suitable and efficient technique to provide quantum safety compared to other post-quantum cryptographic primitives. Accordingly, scholars have increasingly focused on studying the lattice-based ring signatures [8, 4, 68, 38, 20], offering anonymity, unforgeability, and resistance to quantum attacks. Some of them [8, 4, 68] additionally provide linkability. However, limited research has been conducted on using lattice-based ring signatures to EMRs sharing systems, due to several issues such as efficiency. Among the above schemes, only the lattice-based ring signature (but without linkability) [20] proposed by Chen et al. is efficient and they adopt it to e-health system to guarantee users' privacy and quantum-resistant security of medical data.

Nevertheless, there are still two critical issues requiring solutions, i.e., if medical data is improperly stored or exposed to cyberattacks, it can result in chaos within these data stores or compromise their security. The linkability of ring signatures [48] can be used for the verification of whether multiple EMRs originate from the same user, preventing service unreliability caused by storage confusion. The forward security of ring signatures [49] can regularly update the secret key, ensuring that even if the current key is leaked, previously generated signatures remain secure. In 2018, Boyen et al. [14] firstly introduced a ring signature with both linkability and forward security, however, it is not immune to quantum attacks.

Hence, two questions arise naturally:

1) Can we propose an efficient lattice-based ring signature with linkability and use it in the EMRs sharing system?

2) Can we propose a lattice-based ring signature with both linkability and forward security and use it in the EMRs sharing system?

In Chapter 4, we answer the above two questions affirmatively by constructing an efficient lattice-based linkable ring signature (LLRS) and an enhanced version, lattice-based linkable ring signature with forward security (FS-LLRS). Then, these techniques are used to design a cloud-assisted EMRs sharing system, ensuring secure and efficient e-health services.

1.2.1 Related Works

Ring signature, first introduced by Rivest et al. [60], enables any signer to select a set of public keys, forming a ring to sign the message on behalf of the ring, while concealing their true identities. Abe et al. [1] presented a new approach for constructing ring signature using a three-move type protocol in 2004. Since then, numerous scholars focused on ring signature research and its application in privacy-preserving systems. As e-health developed, scholars have focused on the EMRs security and patient privacy, leading to gradual applications of ring signatures. We compared our scheme with prior arts in Table 1.1.

In 2021, Lu et al. [50] formalized an attribute-based ring signature and used it for Electronic Health Records (EHR). Lai et al. [44] introduced a health data sharing system based on blockchain and ring signature in the following year, providing traceability but reducing anonymity. In 2023, Grover et al. [34] proposed a cloud-based distributed EHR sharing system using ring signature, lacking functionalities, i.e. linkability. Most recently, Bao et al. [7] presented an EHR sharing system that incorporated both linkability and traceability. Among these schemes, this scheme is the only one to offer such functionalities. However, they adopted the group signature technique, which compromised user anonymity. In 2018, Boyen et al. [14] introduced a ring signature with both linkability and forward security. Li et al. [46] provides a forward secure dualring signature in 2023. Nevertheless, it is notable that none of these solutions are resistant to quantum attacks.

Table 1.1: Comparison with Existing Ring Signature Schemes.

Schemes	Linkability	Quantum Resistance	Forward Security	Identity based
Lu et al. [50]	✗	✗	✗	✗
Lai et al. [44]	✗	✗	✗	✗
Grover et al. [34]	✗	✗	✗	✗
Bao et al. [7]	✓	✗	✗	✗
Li et al. [46]	✗	✗	✓	✗
Boyen et al. [14]	✓	✗	✓	✗
Baum et al. [8]	✓	✓	✗	✗
Jia et al. [38]	✗	✓	✗	✓
Alberto et al. [4]	✓	✓	✗	✗
Tang et al. [68]	✓	✓	✗	✓
Chen et al. [20]	✗	✓	✗	✗
Our LLRS Scheme	✓	✓	✗	✗
Our FS-LLRS Scheme	✓	✓	✓	✓

Notes. All schemes satisfy anonymity and unforgeability.

Limited research [20] has been conducted on introducing lattice-based ring signatures to e-health systems. Accordingly, we compare several lattice-based ring signatures that can be utilized to protect medical privacy. As shown in Table 1.1, the schemes of Baum et al. [8], Alberto et al. [4], and Tang et al. [68] offer linkability, while the schemes of Jia et al. [38] and Tang et al. [68] are identity-based.

In Chapter 4, our efficient LLRS scheme provides both linkability and resistance to quantum attacks. Our enhanced FS-LLRS scheme provides additional forward security while retaining the properties of the LLRS scheme. Further, in our FS-LLRS scheme, we use the identity tag τ_i for the public key generation, to achieve identity-based property.

1.3 Thesis Outline

The rest Chapters of this thesis are structured as follows.

- Chapter 2 introduces the notations, definitions, and cryptographic fundamental knowledge for the following content.
- Chapter 3 presents the general construction for converting Σ-protocols based signatures to ring signatures.
- Chapter 4 presents the lattice-based linkable ring signature with forward security construction and uses it for cloud-assisted electronic medical records sharing system.
- Chapter 5 concludes our contributions and provides some ideas for future work stemming from this thesis.

Chapter 2

Preliminary

To begin with, we provide the notations and some fundamental knowledge of this thesis.

2.1 DL-based Ring Signature

2.1.1 Notations

Let $\mathbb{G}$ be the cyclic group order q, $\mathbb{Z}_q$ denotes the integers group modulo q, and $\mathbb{Z}_q^*$ represents non-zero elements in $\mathbb{Z}_q$. Set uppercase letter $A \in \mathbb{G}$ and lowercase letter $a \in \mathbb{Z}_q$. The vectors can be presented as bold letters $\mathbf{a}$ or $\mathbf{A}$. The $\mathbf{P^a}$ means $\prod P_i^{a_i}$. The $\mathbf{a} \circ \mathbf{b}$ denotes as $(a_1 \cdot b_1, \cdots, a_n \cdot b_n)$. The $x \xleftarrow{\$} \mathcal{S}$ means randomly sample the notation x from finite set $\mathcal{S}$.

2.1.2 Discrete Logarithm Assumption

Definition 1 (**Discrete Logarithm**). *For all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$ such that*

$$\Pr\left[a = g^b \middle| \begin{array}{c} g, a \leftarrow \mathbb{G}; \\ b \leftarrow \mathcal{A}(\mathbb{G}, g, a); \end{array}\right] \leq \mathsf{negl}, \tag{2.1}$$

where the negl *denotes as negligible.*

2.1.3 Σ-Protocol

Σ-protocol is a three-phase (commitment, challenge, and response) interactive protocol, utilized by a prover $\mathcal{P}$ to persuade the verifier $\mathcal{V}$ of the truth of a statement without disclosing its witness.

Specifically, the Σ-protocol is between $\mathcal{P}$ and $\mathcal{V}$, and there also exists a setup algorithm $\mathscr{S}$. The tuple $(\mathscr{S}, \mathcal{P}, \mathcal{V})$ is defined for a Σ-protocol with relation $\mathcal{R}$, and $(s, P) \subseteq \mathcal{R}$, where s is a witness, P is a statement. The Σ protocol satisfies three properties, i.e. the completeness, soundness, and Honest Verifier Zero-Knowledge (HVZK).

Definition 2 (**Perfect Completeness**). *[33, 18] The tuple $(\mathscr{S}, \mathcal{P}, \mathcal{V})$ is called the Σ-protocol, with the perfect completeness, for all adversaries $\mathcal{A}$*

$$\Pr\left[\mathcal{V}(pp, P, \mu, c, f) = 1 \middle| \begin{array}{c} pp \leftarrow \mathscr{S}(\lambda); \\ (s, P) \leftarrow \mathcal{A}(pp); \\ \mu \leftarrow \mathcal{P}(pp, s, P); \\ c \leftarrow \{0,1\}^{\lambda}; \\ f \leftarrow \mathcal{P}(c); \end{array}\right] = 1, \tag{2.2}$$

where $(pp, s, P) \in \mathcal{R}$.

Definition 3 (n-**special Soundness**). *[33, 18] The tuple $(\mathscr{S}, \mathcal{P}, \mathcal{V})$ is n-special soundness if there exists an extractor $\mathcal{E}$, which has the ability to calculate the witness s by giving n accepting transcripts. Thus, for all adversaries $\mathcal{A}$*

$$\Pr\left[(pp,s,P)\in\mathcal{R}\;\middle|\;\begin{array}{c} pp\leftarrow\mathscr{S}(\lambda);\\ (P,\mu,c_1,f_1,\cdots,c_n,f_n)\leftarrow\mathcal{A}(pp);\\ \mathcal{V}(pp,P,\mu,c_i,f_i)=1,\ \forall\ i\in[1,n];\\ s\leftarrow\mathcal{E}(pp,P,\mu,c_1,f_1,\cdots,c_n,f_n);\end{array}\right]\approx 1-\mathsf{negl}, \tag{2.3}$$

where the negl *denotes as negligible, and we say that is n-special soundness.*

Definition 4 (**Honest Verifier Zero-Knowledge (HVZK)**). *[33, 18] The tuple* $(\mathscr{S},\mathcal{P},\mathcal{V})$ *is HVZK if there exists a simulator* $\mathcal{S}$*, for all adversaries* $\mathcal{A}$

$$\left|\Pr\left[\mathcal{A}(\mu,f)=1\;\middle|\;\begin{array}{c} pp\leftarrow\mathscr{S}(\lambda);\\ (s,P,c)\leftarrow\mathcal{A}(pp);\\ \mu\leftarrow\mathcal{P}(pp,s,P);\\ f\leftarrow\mathcal{P}(c);\end{array}\right]-\Pr\left[\mathcal{A}(\mu,f)=1\;\middle|\;\begin{array}{c} pp\leftarrow\mathscr{S}(\lambda);\\ (s,P,c)\leftarrow\mathcal{A}(pp);\\ (\mu,f)\leftarrow\mathcal{S}(pp,P,c);\end{array}\right]\right|\leq\mathsf{negl}, \tag{2.4}$$

where $\mathcal{A}$ *outputs* (s,P,c) *such that* $(pp,s,P)\in\mathcal{R}$ *and* $c\in\{0,1\}^{\lambda}$*. If* negl *is negligible, we say that is HVZK.*

2.1.4 Commitment Scheme

The commitment, which incorporates the properties of hiding and binding, enables the sender to generate a commitment $R=com(k)$ in the *commit* phase. In the *open* phase, the receiver can check the validity of the opening and whether the commitment R is constructed from a real value k.

Taking the Pedersen commitment as an example, in a general scenario, the group $\mathbb{G}$ and the generator G are obtained by *setup* phase. In *commit* phase, the sender calculates the commitment $R=com(k)=G^k$. In *open* phase, the receiver can perform validity verification. For signature, the public key P_i can be treated as a commitment concerning the secret key s_i.

Definition 5 (**Hiding**). *[33, 18] The commitment $(\mathscr{S}, com)$ is hiding if that commitment com exposes nothing about the real message value μ, and the commitment com is statistically independent. Hence, for all adversaries $\mathcal{A}$*

$$\Pr\left[\mathcal{A}(c) = b \;\middle|\; \begin{array}{c} pp \leftarrow \mathscr{S}(\lambda); \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(pp); \\ b \leftarrow \{0,1\}; \\ r \leftarrow \mathcal{R}_{pp}; \\ c \leftarrow com_{pp}(\mu_b); \end{array}\right] \approx \frac{1}{2}, \tag{2.5}$$

where the adversary $\mathcal{A}$ outputs the message $\mu_0, \mu_1 \in \mathcal{M}_{pp}$. If the probability of this equation is equal to $\frac{1}{2}$, we say that is hiding.

Definition 6 (**Binding**). *[33, 18] For all efficient adversaries $\mathcal{A}$, if the commitment $(\mathscr{S}, com)$ is opened only for one single value, such that*

$$\Pr\left[com_{pp}(\mu_0) = com_{pp}(\mu_1), \mu_0 \neq \mu_1; \;\middle|\; \begin{array}{c} pp \leftarrow \mathscr{S}(\lambda); \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(pp); \end{array}\right] \approx 0 \tag{2.6}$$

where the adversary $\mathcal{A}$ outputs $\mu_0, \mu_1 \in \mathcal{M}_{pp}$. If the probability of this equation is equal to 0, we say that is binding.

2.1.5 Bulletproofs Folding

Bulletproofs [15] is a spatially efficient and concise protocol, without a trusted setup phase. The core compression idea is dichotomy and recursion. We use it to fold our response value, serving as a stepping stone towards our objective.

In our scheme, the central folding process is the relation $\mathbf{P}^{\mathbf{f}}$. In particular, n must be a power of 2. We set $n' = \frac{n}{2}$, and then we divide $\mathbf{f}$ and $\mathbf{P}$ into two vectors respectively, i.e. $\mathbf{f}_L$, $\mathbf{f}_R$ and $\mathbf{P}_L$, $\mathbf{P}_R$. Then, we compute $L_B = {\mathbf{P}_L}^{\mathbf{f}_R}$ and $R_B = {\mathbf{P}_R}^{\mathbf{f}_L}$. The process of compressing the size of folding value $\mathbf{f}$ is calculated as $\mathbf{f}' = x \cdot \mathbf{f}_R + x^{-1} \cdot \mathbf{f}_L$, where x is sampled randomly. In each round, the prover $\mathcal{P}$ and verifier $\mathcal{V}$ computes a new

vector $\mathbf{P}' = \mathbf{P}_R^{x^{-1}} \circ \mathbf{P}_L^x$ to replace the original $\mathbf{P}$. The verification can be denoted as $com(\mathbf{f}') = g(com(\mathbf{k}), c, e, com(\mathbf{s})) \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}}$. We can minimize the response size by folding logarithmic number rounds.

2.1.6 One-out-of-Many Proofs

One-out-of-many proofs is a notable approach to constructing a ring signature. In this way, the prover $\mathcal{P}$ utilizes one-out-of-many proofs to prove that it possesses the secret key s_l with the public key P_l in the set $\mathbf{P} = (P_1, \cdots, P_l, \cdots, P_n)$. To be more specific, the signer's public key P_l is a commitment to $(0; s_l)$, i.e., $P_l = com(0; s_l)$. Kronecker's delta vector is represented as $\boldsymbol{\delta}_l = (\delta_{l,0}, \cdots, \delta_{l,l}, \cdots, \delta_{l,n-1})$, with $\delta_{l,i} = 1$ when $i = l$, and $\delta_{l,i} = 0$ when $i \neq l$. In this case, the ring signature can be transformed to prove it having $\boldsymbol{\delta}_l$. Thus, it is possible to address the problem of P_l leakage. By representing $\boldsymbol{\delta}_l$ as binary, the signature size can be thereby reduced.

2.2 Lattice-based Ring Signature

2.2.1 Notations

For any $N \in \mathbb{N}$, let $[N] := \{1, 2, \cdots, N\}$. Let $(a||b)$ denote the concatenation of a and b. We denote $a \xleftarrow{\$} S$ as randomly sampling a from the set S. For any PPT algorithm A, we denote $a \leftarrow \mathsf{A}(b)$ as running A on input b and assigning a the result. We write vectors in $\mathbb{Z}_q^n$ in bolded lowercase letters, e.g., $\mathbf{a} = (a_1, \cdots, a_n) \in \mathbb{Z}_q^n$. We write matrices in $\mathbb{Z}_q^{n \times m}$ in bolded uppercase letters, e.g., $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_m) \in \mathbb{Z}_q^{n \times m}$ and each $\mathbf{b}_i \in \mathbb{Z}_q^n$.

2.2.2 Lattice

Definition 7. *[3, 10] Given a matrix* $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\} \subset \mathbb{R}^m$ *consists of* n *linearly independent vectors, we define the* m*-dimensional lattice* Λ *with its basis* $\mathbf{B}$ *below.*

$$\Lambda = \{x_1 \cdot \mathbf{b}_1 + \cdots + x_n \cdot \mathbf{b}_n | x_i \in \mathbb{Z}\}. \tag{2.7}$$

Definition 8. *[56] Given three integers* n, m, q*, and a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$*, we define the* q*-ary lattice and its shifted cosets as:*

$$\Lambda_q(\mathbf{B}) := \{\mathbf{z} \in \mathbb{Z}_q^m | \exists \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{B}^\top \mathbf{s} = \mathbf{z} \bmod q\}. \tag{2.8}$$

$$\Lambda_q^\perp(\mathbf{B}) := \{\mathbf{z} \in \mathbb{Z}_q^m | \mathbf{B}\mathbf{z} = 0 \bmod q\}. \tag{2.9}$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{B}) := \{\mathbf{z} \in \mathbb{Z}_q^m | \mathbf{B}\mathbf{z} = \mathbf{u} \bmod q\}. \tag{2.10}$$

2.2.3 Lattice Basis Algorithms

Definition 9. *Given a positive integer* σ*, a vector* $\mathbf{v} \in \mathbb{Z}^m$ *and any vector* $\mathbf{s} \in \mathbb{Z}^m$*, we define* $\mathcal{D}_{\sigma,\mathbf{v}} = \rho_{\sigma,\mathbf{v}}(\mathbf{s})/\rho_{\sigma,\mathbf{v}}(\Lambda)$ *for* $\forall \mathbf{s} \in \Lambda$ *is the discrete Gaussian distribution over* Λ*:* $\rho_{\sigma,\mathbf{v}}(\mathbf{s}) = \exp(-\pi \frac{\|\mathbf{s}-\mathbf{v}\|^2}{\sigma^2})$*, where* $\mathbf{v}$ *is a center and* $\rho_{\sigma,\mathbf{v}}(\Lambda) = \sum_{\mathbf{s}\in\Lambda} \rho_{\sigma,\mathbf{v}}(\mathbf{s})$*.*

Lemma 1. *[52] Given three integers* n, m, q*, the* $\mathsf{TrapGen}(n, m, q)$ *algorithm returns full rank* $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ *and its trapdoor* $\mathbf{T_B} \in \mathbb{Z}_q^{m \times m}$*,* s.t. $\|\widetilde{\mathbf{T}}_\mathbf{B}\| = \mathcal{O}(\sqrt{n \log q})$*, where* $\mathbf{B}$ *is statistically close to being uniform.*

Lemma 2. *[31] For two integers* $m, n,$ s.t. $m \geq 2n\lceil \log q \rceil$*, a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ *and its trapdoor* $\mathbf{T_M} \in \mathbb{Z}_q^{m \times m}$*, a vector* $\mathbf{v} \in \mathbb{Z}_q^n$*, and a Gaussian parameter* $\sigma > \|\widetilde{\mathbf{T}}_\mathbf{M}\| \cdot \omega(\sqrt{\log(m)})$*, this* $\mathsf{SamplePre}(\mathbf{M}, \mathbf{T_M}, \mathbf{v}, \sigma)$ *algorithm outputs a vector* $\mathbf{e} \in \mathbb{Z}_q^m$ *where statistically distributed in* $\mathcal{D}_{\Lambda_q^v(\mathbf{M}),\sigma}$*.*

Lemma 3. *[17] Given four integers $n, k, q \geq 2$, and $m \geq 2n \log q$, a set $S \subset [k]$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times km}$, a basis $\mathbf{T}_S$ of $\Lambda_q^{\perp}(\mathbf{A}_S)$, a vector $\mathbf{v} \in \mathbb{Z}_q^n$, and $u \geq \|\widetilde{\mathbf{T}}_S\| \cdot \sqrt{km} \cdot \omega(\sqrt{\log km})$, the* $\mathsf{GenSamplePre}(\mathbf{A}, \mathbf{T}_S, S, \mathbf{v}, u)$ *algorithm returns a vector $\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_k] \in \mathbb{Z}^{km}$.*

Lemma 4. *[17] Given two matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$, and a basis $\mathbf{T}_A \in \mathbb{Z}_q^{m \times m}$ of $\Lambda_q^{\perp}(\mathbf{A})$, the* $\mathsf{ExtBasis}(\mathbf{C}, \mathbf{T}_A)$ *algorithm returns a basis $\mathbf{T}_C$ of $\Lambda_q^{\perp}(\mathbf{C}) \subseteq \mathbb{Z}_q^{m \times m''}$,* s.t. *$\|\tilde{\mathbf{T}}_A\| = \|\tilde{\mathbf{T}}_C\|$, $\mathbf{C} = \mathbf{A}||\mathbf{B}$, and $m'' = m + m'$.*

2.2.4 Short Integer Solution (SIS) Assumption

Definition 10. *Given uniformly random vectors $\mathbf{b}_1, \cdots, \mathbf{b}_m \in \mathbb{Z}_q^n$ and forming the matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, the Short Integer Solution (SIS) is to find a non-zero integer vector $\mathbf{x} \in \mathbb{Z}^m$ with $\|\mathbf{x}\| \leq \beta$* s.t. *$f_{\mathbf{B}}(\mathbf{x}) := \mathbf{B}\mathbf{x} = \mathbf{0} \in \mathbb{Z}_q^n$, where $\beta < q$.*

Chapter 3

DL-based Ring Signature Construction and Application

In this Chapter, we present a general construction for transforming Σ-based signatures into ring signatures. In order to showcase the broad applicability of our general construction, we also construct four prominent ring signatures as case studies by using our transformation method.

3.1 Overview

In this Section, we elaborate on our motivations and contributions.

3.1.1 Motivations

Although several signatures based on Σ-protocols have been proposed, there is no general model to summarize these protocols till now, resulting in the analysis of these signatures being difficult and their extensions generality reduced. Moreover, the current ring signatures are not standardized by large formal organizations, which

increases the problem of adaptability in ring signature scenarios. Further, there is no established general construction to realize the transformation from Σ-based signatures to ring signatures to date.

To solve the aforementioned bottlenecks in privacy-preserving systems, we are mainly engaged in proposing a generic ring signature construction that provides a conversion method from Σ-based signatures to ring signatures without reducing the security level. We further adopt the Bulletproofs folding technique to improve the efficiency of our proposed ring signature from $O(n)$ to $O(\log n)$. Our major goal is to present a generic construction with better performance rather than construct much more efficient ring signatures.

Difference from DualRing Construction. Yuen et al. [74] presented a general construction of ring signature in 2021, namely DualRing, which is also possible for several Σ-protocols. Our scheme is different from them, whether in motivation, construction, or technique.

Firstly, our motivation is different from them. As mentioned above, we are engaged in proposing a standard generic ring signature construction that provides a conversion method from Σ-protocol based signatures to ring signatures. Whereas, Yuen et al. [74] dedicated to proposing an entirely novel, non-standard generic ring signature construction.

After that, for our construction, a signature comprises one challenge c and n responses $\mathbf{f}$. In contrast, the DualRing signature with two rings structure, includes n challenges $\mathbf{c}$ and one response f.

Finally, we point out the differences in the technology used for optimizing efficiency. In our proposed general construction, the relation $\prod_{i=1}^{n} {pk_i}^{f_i}$ is utilized to realize the folding, where pk_i is the public key and f_i is the response value. While efficiency optimization in the DualRing requires it to fulfill the equation $\prod_{i=1}^{n} {pk_i}^{c_i}$, where pk_i is a public key, and c_i is the challenge.

More importantly, our scheme's performance is better than the DualRing scheme.

3.1.2 Contributions

Our primary objective is to propose a generic construction that transforms Σ-protocol-based signatures into ring signatures. Meanwhile, the proposed scheme needs to ensure both efficiency and security. Specifically, we demand the ring signature size to be logarithmic to the ring size, while also maintaining the security proof does not introduce extra assumptions than the original security assumptions of the signature schemes.

We present our five-fold contributions as:

- We propose a general model to generalize existing signature constructions, which can encompass widely used standardized signatures such as ECDSA, EdDSA, SM2, and Schnorr signatures. With this model, we can propose a generic transformation from existing signatures to the Σ-based signatures.
- Building on the aforementioned general Σ-protocol model, we additionally include a novel redesigned one-out-of-many relation that avoids revealing the signer's index. With the assistance of the one-out-of-many proofs and the Fiat-Shamir transformation, we propose a generalized construction that transforms the existing signatures into ring signatures.
- To further enhance the efficiency of our general ring signature, we adopt the Bulletproofs folding technique to our ring signatures, which reduces the size of the ring signatures from $O(n)$ to $O(\log n)$, where n is the size of the ring.
- To demonstrate the high universality of our general ring signature scheme, four case studies are given, which include the transformation of ECDSA, EdDSA, SM2, and Schnorr signatures, to the ring signatures.

- A formal security analysis is conducted to show the security of our construction. Furthermore, we implement our scheme and perform a comparison with other ring signatures in computation and communication overhead. The performance evaluation indicates that our construction only incurs a slight overhead in the transformation process when the ring size is relatively small ($n \leq 16$), which is acceptable on a common device implementation. Moreover, compared to other existing schemes, our scheme's signing and verifying times are 0.38 - 0.83 times and 0.23 - 0.65 times, respectively. Additionally, the communication cost of our scheme is $2 \log n$, outperforming existing ring signature schemes.

3.1.3 Overview of Performance Results

We comparatively evaluate the signature size overhead of our $O(\log n)$-size general ring signature with the logarithmic size schemes of Groth et al. [33], Bootle et al. [12] and Yuen et al. [74]. Our ring signature size is $2 \log n + 2$, which is significantly lower than the schemes of Groth et al. [33] and Bootle et al. [12], and slightly lower than Yuen et al.'s scheme [74]. Moreover, our experiments and comparative evaluations are performed for the time overheads, as detailed follow.

3.2 Technique Overview

In a ring signature, a signer has the ability to sign a message on behalf of the ring. With all public keys in the group, anyone can verify whether the signature is valid without knowing the identity of the real signer. This can be formalized into proving a one-out-of-many relation: a signer processes a secret key corresponding to a public key in the group.

Assume that there are n users in a group and each user i has secret-public keys (s_i, P_i), the relation $P_i = G^{s_i}$ which can be regarded as a Pedersen commitment of

s_i with the generator G. In the one-out-of-many relation, a signer l needs to prove two constraints: 1) it has a secret key s_l corresponding to a public key P_l; and 2) its public key P_l is in the public keys set $\mathbf{P} = (P_1, \cdots, P_l, \cdots, P_n)$. In summary, proving that the secret key s_l indeed exists can be accomplished by existing signature schemes, and proving that public key P_l is inside the public list can be achieved by an additional one-out-of-many proof.

The first constraint can be proved with existing signature schemes with some transformation. By regarding the secret s_l as a vector $\mathbf{v} = (v_1, \cdots, v_l, \cdots, v_n) = (0, \cdots, 0, s_l, 0, \cdots, 0)$, we have the relation $\prod P_i^{v_i} = P_l = G^{s_l}$. Note that $v_i = s_i$ when $i = l$ and $v_i = 0$ otherwise. For the public key P_l corresponding to the secret key s_l, a straightforward intuition is to regard the relation of secret key s_l and public key P_l as the Pedersen vector commitment between $\mathbf{v}$ and $\mathbf{P}$, and use a Σ-protocol to prove $com(\mathbf{v}) = \prod P_i^{v_i} = P_l$. This requires us to convert the existing signature scheme into Σ-protocol to facilitate subsequent constructions. Unfortunately, existing signatures use many special constructions. For example, in ECDSA and SM2 signatures take the point's x-coordinate during the signing process. In order to cover these special cases, we need to propose a more general model of the Σ-protocol-based signature schemes.

Regarding the second problem, there exists some technical difficulties. In the above constraint, P_l is regarded as the Pedersen vector commitment of the secret $\mathbf{v}$ on $\mathbf{P}$, which needs to be public to the verifier. Accordingly, the verifier knows the index (ID) information of the signer l (the message is signed by a signer with P_l). To tackle this challenge, we redesign the relationship between the public key and secret key as $P_i = G^{{s_i}^{-1}}$, ensuring the security of this variant is the same as the original one since it can be reduced to the original relation. We assume that a ring signature has n users, and we have the equation $\prod_{i=1}^{n} P_i^{v_i} = {P_1}^0 \cdots {P_l}^{s_l} \cdots {P_n}^0 = G^{{s_l}^{-1} \cdot s_l} = G$. As $com(\mathbf{v})$ becomes G instead of P_l, the verifier cannot know the identity information of signer l.

With the approach described above, we can obtain a ring signature without introducing additional security assumptions. Since we only use assumptions of commitments in proving the second constraint, which is the same as the security assumption of the public-secret key relation in the original signature, a new ring signature does not incur additional assumptions. However, the signature size of that is linear to the ring size, which is impractical for real-world applications.

To further improve the efficiency, we employ the idea of folding from Bulletproofs to compress the ring signature size to logarithmic to ring size. Since Bulletproofs is based on the same security assumption as the commitment, further improvement does not sacrifice the security of the original scheme.

3.3 Syntax and Security Models

3.3.1 Syntax

For our ring signature primitive, we formalize its syntax, including four algorithms, $\varphi = (\mathsf{RingSetup}, \mathsf{RingKeyGen}, \mathsf{RingSign}, \mathsf{RingVerify})$.

- $\mathsf{RingSetup}(\lambda)$: Given the security parameter λ, this algorithm returns the system public parameters pp.
- $\mathsf{RingKeyGen}(pp)$: Given the public parameters pp, this PPT algorithm returns the public-secret key (P_l, s_l) of signer l.
- $\mathsf{RingSign}(pp, \mathbf{P}, s_l, \mu)$: Given the public parameters pp, a secret key s_l of user l, a public keys set $\mathbf{P}$ (constitutes the ring), and a message μ, this PPT algorithm returns a signature σ.
- $\mathsf{RingVerify}(pp, \mathbf{P}, \mu, \sigma)$: Given the public parameters pp, a public keys set $\mathbf{P}$, a message μ, and a signature σ, this deterministic algorithm returns the 'accept'

or the 'reject'.

3.3.2 Security Models

The security models of our proposed ring signature primitive encompass anonymity and unforgeability. The formal definitions interacted with challenge $\mathcal{C}$ and adversary $\mathcal{A}$ are provided below.

Anonymity

This game is designed as follows:

- **Setup.** $\mathcal{C}$ executes RingSetup algorithm to procure the system public parameters pp and delivers pp to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded n queries adaptively:
  - Random Oracle $\mathcal{O}_H$: We set a hash function $\mathcal{H}$ in $\mathcal{O}_H$ and maintain the list L_H as empty initially. If the input has been queried before, $\mathcal{C}$ checks the list L_H and returns the corresponding result. For a new query, $\mathcal{C}$ calculates the results and records the related information. Finally, $\mathcal{C}$ returns it to $\mathcal{A}$.
  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key P_i for index $i \in [1, n]$. Then, $\mathcal{C}$ invokes RingKeyGen algorithm to get public-secret key (P_i, s_i) and outputs the public key P_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key s_i for index $i \in [1, n]$. $\mathcal{C}$ calculates its secret key s_i and then outputs it to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ_i for user $i \in [1, n]$. Then, $\mathcal{C}$ outputs σ_i to $\mathcal{A}$ by invoking RingSign algorithm.

- **Challenge.** $\mathcal{A}$ chooses a message μ, a public keys set $\mathbf{P}$, and send them to $\mathcal{C}$. Then, $\mathcal{C}$ randomly selects a $\varphi \in [1, n]$ to generate the signature σ_φ and sends it to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a guess $\varphi^* \in [1, n]$. If $\varphi^* = \varphi$, $\mathcal{A}$ wins.

The advantage for $\mathcal{A}$ in attacking the anonymity game is defined as

$$Adv_{\mathcal{A}}^{\text{Anony}}(\lambda) := |\Pr[\varphi^* = \varphi] - \frac{1}{n}|. \tag{3.1}$$

Definition 11 (**Anonymity**). *For any PPT adversary $\mathcal{A}$, if the advantage $Adv_{\mathcal{A}}^{\text{Anony}}(\lambda)$ is negligible, we say that our scheme fulfills the requirement for* anonymity.

Unforgeability

This game is designed as follows:

- **Setup.** $\mathcal{A}$ randomly picks a target index $i^* \in [1, n]$. $\mathcal{C}$ executes RingSetup algorithm to procure the system public parameters pp and delivers it to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded n queries adaptively:
  - Random Oracle $\mathcal{O}_H$: We set a hash function $\mathcal{H}$ in $\mathcal{O}_H$ and maintain the list L_H as empty initially. If the input has been queried before, $\mathcal{C}$ checks the list L_H and returns the corresponding result. For a new query, $\mathcal{C}$ calculates the results and records the related information. Finally, $\mathcal{C}$ returns the result to $\mathcal{A}$.
  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key P_i of a user $i \in [1, n]$. Then, $\mathcal{C}$ invokes RingKeyGen algorithm to get public-secret key (P_i, s_i) and outputs the public key P_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ quires the secret key s_i for a user $i \in [1, n]$. $\mathcal{C}$ returns s_i to $\mathcal{A}$ if $i \neq i^*$. Otherwise, the query is aborted and $\mathcal{C}$ returns $\perp$.

  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ_i for user $i \in [1, n]$. If $P_i \notin \mathbf{P}$, $\mathcal{C}$ returns $\perp$. Otherwise, it returns σ_i to $\mathcal{A}$ by invoking RingSign algorithm ($i \neq i^*$) or simulator $\mathcal{S}$ ($i = i^*$).

- **Forge.** $\mathcal{A}$ outputs a public keys set $\mathbf{P}^*$ including P_{i^*}, message μ^*, and σ_{i^*}. Then, $\mathcal{C}$ invokes RingVerify algorithm to verify σ_{i^*}. If the requirements listed below are fulfilled, we say that $\mathcal{A}$ wins.

  - The message μ^* of forgery signature σ_{i^*} has not queried in $\mathcal{O}_{SO}$.
  - Inputs σ_{i^*} in RingVerify algorithm and it returns 'accept'.
  - $\mathcal{A}$ does not possess any secret key related to the public keys in $\mathbf{P}^*$.

The advantage for $\mathcal{A}$ in attacking unforgeability game is defined as

$$Adv_{\mathcal{A}}^{Forge}(\lambda) := \mathsf{negl}(\lambda). \tag{3.2}$$

Definition 12 (**Unforgeability**). *For any PPT adversary $\mathcal{A}$, if the advantage $Adv_{\mathcal{A}}^{Forge}(\lambda)$ is negligible, we say that our ring signature scheme satisfies the requirement of* unforgeability.

3.4 From One-out-of-Many Proofs to Ring Signature

In this Section, we demonstrate how to transform existing signatures into ring signatures. Firstly, we present a general model from existing signatures to Σ-protocol. Subsequently, we convert this model into a ring signature with the help of one-out-of-many proofs and Fiat–Shamir heuristic. To further improve efficiency, we employ the Bulletproofs folding technique. Ultimately, we formally propose a logarithmic size ring signature in general.

Note that our general model and generic ring signature are not applicable to RSA-based schemes, and are compatible with the Σ-protocol-based schemes that have one response computation, such as Schnorr, ECDSA signature, etc.

3.4.1 General Model of Signature Schemes

Existing signature schemes consists of four algorithms, SigSetup, SigKeyGen, SigSign, and SigVerify. The SigSetup algorithm sets some initialization system parameters and SigKeyGen algorithm generates the public-secret key of the user. We require modeling the SigSign and SigVerify algorithms so that all existing signature algorithms can be transformed into a Σ-protocol form to facilitate the construction of subsequent ring signature schemes.

The relation $\mathcal{R}$ in Σ-protocol can effectively represent the relationship between a witness s and a statement P. Thus, in the general case, we denote an effective relation as $(s, P) \in \mathcal{R}$.

In the existing Σ-protocol based signatures, a prover $\mathcal{P}$ owns a witness s and a statement P, and verifier $\mathcal{V}$ only owns a statement P. In real situations, the response f has various expressions in different signatures. For this, we define $g(\cdot)$ to represent the detailed calculation method of the response f, and model the SigSign and SigVerify algorithms as the interactive process as follows.

- $\mathcal{P}$ randomly samples $k \xleftarrow{\$} \mathbb{Z}_q^*$.
- $\mathcal{P}$ calculates the commitment $R \leftarrow com(k)$, then sends R to $\mathcal{V}$.
- $\mathcal{V}$ randomly samples a challenge $c \xleftarrow{\$} \mathbb{Z}_q^*$ and returns it to $\mathcal{P}$.
- $\mathcal{P}$ generates a response $f \leftarrow g(s, k, c)$ and sends f to $\mathcal{V}$.
- $\mathcal{V}$ checks f upon received it, and outputs 'accept' or 'reject'.

Through the Fiat–Shamir transformation technique, we can transform the above protocol into a signature. As we intend to cover all existing signature schemes based on the Σ-protocol, we hereby need to construct a generalized protocol. For example, in the ECDSA signature, we have to take x coordinate of the commitment point R as one of the input values of the response f. However, in the above interactive process, there is no space for this operation.

Thus, we are required to modify the computation of the response f to propose a general model. Following the interactive process of the existing signatures, we model the signature generation and verification process as a general case of the form based on the Σ-protocol. More importantly, it is crucial to generalize the computation of the response value f, as the expression for f varies considerably. By our proposed method, we are able to obtain the Σ-protocol for a variety of signature schemes by setting different values of $t_0, a, t_1, t_2, t_0', a', t_1', t_2'$ in the general model, where a, a' are predefined values. Consequently, the calculation equation of f can be generalized. The general model is depicted in Figure 3.1.

Remark that our general model of Σ-protocol form requires that it must satisfy the homomorphism property of commitment over s and k, i.e. the equation $com(g(k, c, e, s)) = g(com(k), c, e, com(s))$ holds. Specifically, the function g varies in different signatures, and we set the parameters $t_0, a, t_1, t_2, t_0', a', t_1', t_2'$ in the original model.

For special constructions in signatures, we generalize them to a function $F(\cdot)$, which is able to transform the commitment R to obtain the part information $e \leftarrow F(R)$, such as a hash function, taking the point's x-coordinate, or a mapping function, etc. The computation method of the function $F(\cdot)$ is publicly available and we require it to satisfy the properties of one-wayness and uniqueness, i.e. we can easily compute the output value of the function from its input, but it is difficult to extrapolate the input value from the output value; the same input will always produce the same output.

Figure 3.1: Our Proposed General Model of Existing Signatures.

3.4.2 General Model with One-out-of-Many Proofs Relation

Till now, we have converted an existing signature into a general model, and then we continue to transform it into a ring signature with the assistance of one-out-of-many proofs.

Firstly, we review the role of the one-out-of-many proofs. By utilizing that, the prover $\mathcal{P}$ is able to certify that it knows the opening of some public key P_l in the public keys set $\mathbf{P} = \{P_1, \cdots, P_n\}$, which proves that $\mathcal{P}$ owns the secret key s_l related to the public key P_l. For the ring setting, we extend the secret key s_l of the user l to a vector $\mathbf{v}$ with the v_l term set to s_l and the rest of the terms set to 0. Thus, we have the secret keys $\mathbf{s} = (0, \cdots, s_l, \cdots, 0)$ and the public key list $\mathbf{P} = \{P_1, \cdots, P_l, \cdots, P_n\}$ in a ring signature.

In our scheme, we need to add one-out-of-many relation to the general model to transform it into a ring signature, while avoiding revealing the information about the signer l. In particular, the prover $\mathcal{P}$ randomly generates s_l as its secret key. Then, the prover $\mathcal{P}$ computes the commitment value $P_l = com(s_l)$ on the secret key s_l and regards that commitment P_l as its public key. For the ring setting, we extend the secret key s_l of the user l to a vector $\mathbf{v}$ with the v_l term set to s_l and the rest of the terms set to 0. Thus, we have the secret keys $\mathbf{s} = (0, \cdots, s_l, \cdots, 0)$ and the public key list $\mathbf{P} = \{P_1, \cdots, P_l, \cdots, P_n\}$ in a ring signature.

The straightforward method directly transfers the relation of s_l and P_l to the vector $\mathbf{v}$ and $\mathbf{P}$, regarding as the Pedersen vector commitment of the vector $\mathbf{v}$ on $\mathbf{P}$. However, that method would reveal the signer's index value l, which leads to the privacy of the scheme being compromised. Therefore, we devise a relation to $\prod_{i=1}^{n} {P_i}^{v_i} = {P_1}^0 \cdots {P_l}^{s_l} \cdots {P_n}^0 = G^{{s_l}^{-1} \cdot s_l} = G$. In this way, we successfully hide the signer's identity information. Based on the above relation, we can transform the Σ-based general model to a ring signature. Concretely, the relation of our secret-public keys is $P_i = G^{{s_i}^{-1}}$. The secret keys $\mathbf{s} = (0, \cdots, 0, s_l, 0, \cdots, 0)$ and the public keys $\mathbf{P}$

satisfy the relation $\mathbf{P^s} = G$, without revealing any information about l.

We show the interactive process of the general model adding our redesigned relation of one-out-of-many proofs as in Figure 3.2. Note that a, a' are the predefined value, and parameters t_0, t'_0, t_2, t'_2 cannot be zero simultaneously. Besides, taking the vector to its inverse means pointwise inversion.

Theorem 1. *In our scheme, s_l and ${s_l}^{-1}$ can be transformed to each other and deduced from each other.*

By the Fiat-Shamir transformation and Theorem 1, we obtain a general ring signature scheme, consisting of four algorithms, namely RingSetup, RingKeyGen, RingSign and RingVerify, as shown in Algorithms 1, 2, 3, 4, respectively.

Algorithm 1 RingSetup(λ)

1: Chooses the parameters q, n;
2: Chooses the generator G;
3: Defines a hash function $\mathcal{H} : \{0,1\}^*$;
4: Chooses the parameters $t_0, a, t_1, t_2, t'_0, a', t'_1, t'_2$;
5: **return** pp.

Algorithm 2 RingKeyGen(pp)

1: Samples an arbitrary number $s_l \xleftarrow{\$} \mathbb{Z}_q^*$;
2: Computes $P_l = com({s_l}^{-1})$;
3: **return** (P_l, s_l).

3.4.3 General Ring Signature with Folding from Bulletproofs

Further, we can improve the efficiency of our proposed general ring signature by the folding idea from the Bulletproofs [15], to achieve a logarithmic ring signature scheme.

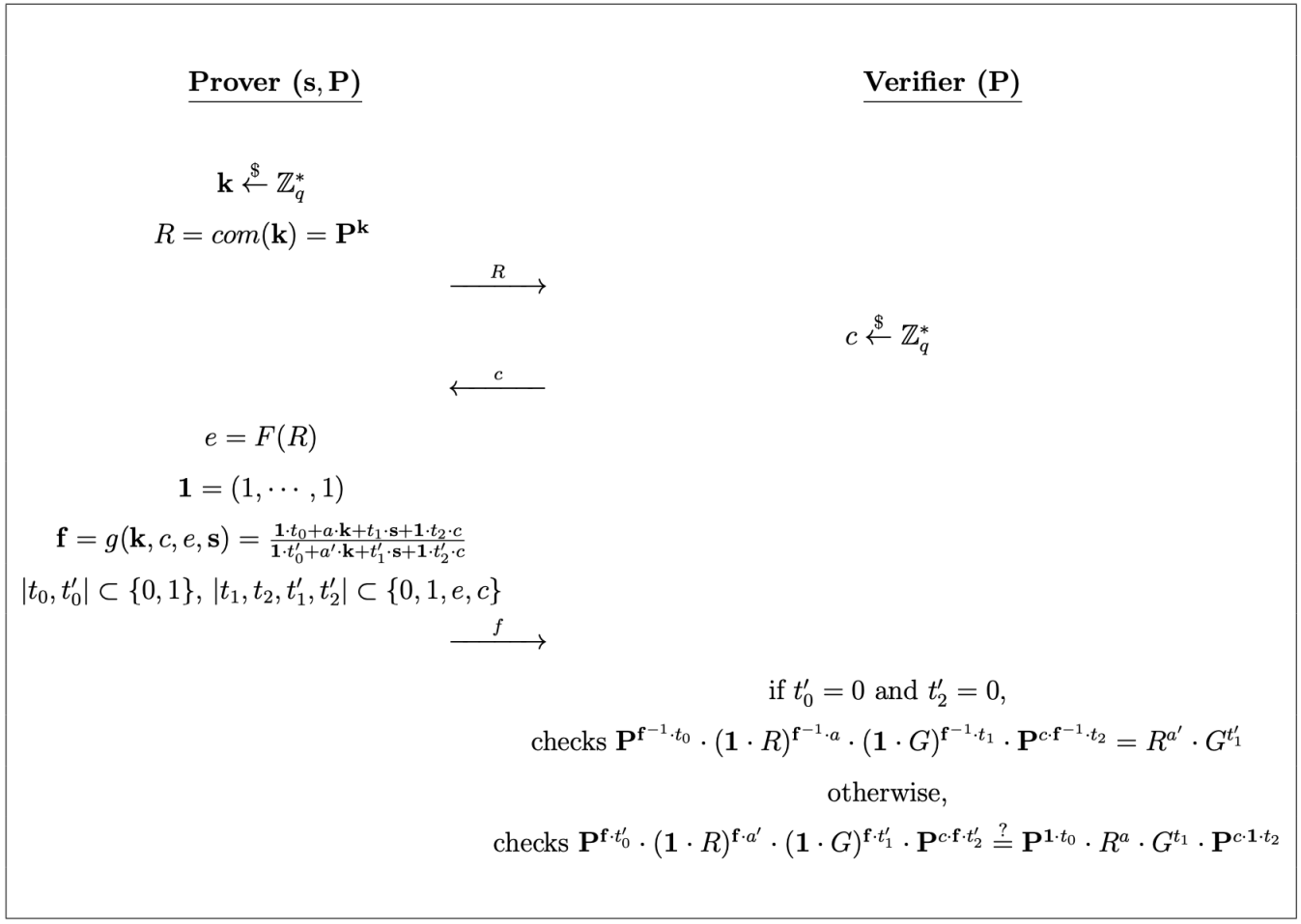


Figure 3.2: Our Proposed General Model with One-out-of-Many Proofs.

Algorithm 3 $\mathsf{RingSign}(pp, \mathbf{P}, \mu, s_l, a, a')$

Computes $t_0, t_1, t_2, t'_0, t'_1, t'_2$;
for all $i \in [n]$ **do**
    Randomly chooses $k_i \xleftarrow{\$} \mathbb{Z}_q^*$;
end for
Computes $R = com(\mathbf{k}) = \mathbf{P}^{\mathbf{k}}$;
Computes $e \leftarrow F(R)$;
Sets $c \leftarrow \mathcal{H}(\mu \| R \| \mathbf{P})$;
for all $i \in [n], i \neq l$ **do**
    Sets $f_i \leftarrow g(k, c, e) = \frac{t_0 + a \cdot k_i + t_2 \cdot c}{t'_0 + a' \cdot k_i + t'_2 \cdot c}$;
end for
for $i \in [n], i = l$ **do**
    Computes $f_l \leftarrow g(k, c, e, s) = \frac{t_0 + a \cdot k_l + t_1 \cdot s_l + t_2 \cdot c}{t'_0 + a' \cdot k_l + t'_1 \cdot s_l + t'_2 \cdot c}$;
end for
return $\sigma = (R, \mathbf{f})$.

Algorithm 4 $\mathsf{RingVerify}(pp, \mathbf{P}, \mu, a, a', \sigma)$

Computes $t_0, t_1, t_2, t'_0, t'_1, t'_2$;
Computes $e \leftarrow F(R)$;
Computes $c \leftarrow \mathcal{H}(\mu \| R \| \mathbf{P})$;
if $t'_0 = 0$ and $t'_2 = 0$ **then**
    Checks $\mathbf{P}^{\mathbf{f}^{-1} \cdot t_0} \cdot (\mathbf{1} \cdot R)^{\mathbf{f}^{-1} \cdot a} \cdot (\mathbf{1} \cdot G)^{\mathbf{f}^{-1} \cdot t_1} \cdot \mathbf{P}^{c \cdot \mathbf{f}^{-1} \cdot t_2} \stackrel{?}{=} R^{a'} \cdot G^{t'_1}$;
else if then
    Checks $\mathbf{P}^{\mathbf{f} \cdot t'_0} \cdot (\mathbf{1} \cdot R)^{\mathbf{f} \cdot a'} \cdot (\mathbf{1} \cdot G)^{\mathbf{f} \cdot t'_1} \cdot \mathbf{P}^{c \cdot \mathbf{f} \cdot t'_2} \stackrel{?}{=} \mathbf{P}^{\mathbf{1} \cdot t_0} \cdot R^{a} \cdot G^{t_1} \cdot \mathbf{P}^{c \cdot \mathbf{1} \cdot t_2}$;
end if
return 'accept' or 'reject'.

Note that, the response $\mathbf{f}$ is compressed using our folding methodology in our ring signature. We set that $n' = \frac{n}{2}$ and the prover $\mathcal{P}$ computes (divides) the $\mathbf{f} = \{f_1, \cdots, f_n\}$ as:

$$\mathbf{f}_L = (f_1, \cdots, f_{n'}),\ \mathbf{f}_R = (f_{n'+1}, \cdots, f_n). \tag{3.3}$$

Similar with the above, the prover $\mathcal{P}$ computes (divides) the $\mathbf{P} = \{P_1, \cdots, P_n\}$ as:

$$\mathbf{P}_L = (P_1, \cdots, P_{n'}),\ \mathbf{P}_R = (P_{n'+1}, \cdots, P_n). \tag{3.4}$$

We then introduce the value L_B and R_B as:

$$L_B = {\mathbf{P}_L}^{\mathbf{f}_R} \text{ and } R_B = {\mathbf{P}_R}^{\mathbf{f}_L}. \tag{3.5}$$

Then, the verifier $\mathcal{V}$ randomly samples $x \xleftarrow{\$} \mathbb{Z}_q^*$.

For the folding of the response value $\mathbf{f}$, the calculation process is as follows:

$$\mathbf{P}' = {\mathbf{P}_R}^{x^{-1}} \circ {\mathbf{P}_L}^{x} \text{ and } \mathbf{f}' = \mathbf{f}_R \cdot x + \mathbf{f}_L \cdot x^{-1}. \tag{3.6}$$

The relation of the verification is $com(\mathbf{f}') \stackrel{?}{=} g(com(\mathbf{k}), c, e, com(\mathbf{s})) \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}}$. The verifier $\mathcal{V}$ checks whether the equation $\mathbf{P}^{f' \cdot \mathbf{s} \cdot t'_0} \cdot (\mathbf{1} \cdot R)^{f' \cdot \mathbf{s} \cdot a'} \cdot (\mathbf{1} \cdot G)^{f' \cdot \mathbf{s} \cdot t'_1} \cdot \mathbf{P}^{c \cdot f' \cdot \mathbf{s} \cdot t'_2} \stackrel{?}{=} \mathbf{P}^{\mathbf{1} \cdot t_0} \cdot R^a \cdot G^{t_1} \cdot \mathbf{P}^{c \cdot \mathbf{1} \cdot t_2} \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}}$ is hold, where $s_i = \sum_{j=1}^{\log n} {x_j}^{t(i,j)}$.

After one round of folding, we can obtain the folding value $\mathbf{f}'$. In each subsequent round of folding, we replace $\mathbf{P}$ with $\mathbf{P}'$ and $\mathbf{f}$ with $\mathbf{f}'$. Thus, we achieve the $O(\log n)$ signature size by calling $\log n$ times folding.

3.4.4 Formal Construction of Logarithmic Size General Ring Signature

Through the above work, i.e. the general model of existing signature schemes, the one-out-of-many proofs technique, the folding idea from Bulletproofs, and the assistance of Fiat-Shamir transformation techniques, we now present the formal construction of our proposed $O(\log n)$ ring signature, as described below:

- For the $\mathsf{RingBPSetup}(\lambda)$, we follow the existing signature and take a security parameter λ as input, to get a parameter q, a hash function $\mathcal{H} : \{0,1\}^*$, a function F to replace special construction, a generator G of the group, the parameters $t_0, a, t_1, t_2, t'_0, a', t'_1, t'_2$, and then output public parameters pp.

- For the $\mathsf{RingBPKeyGen}(pp)$, the prover $\mathcal{P}$ randomly samples $s_l \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $P_l = G^{{s_l}^{-1}}$ for user l. The public and secret keys satisfy the relation ${P_l}^{s_l} = G$. The output of this algorithm is the public-secret key (P_l, s_l) of user l.

- For the $\mathsf{RingBPSign}(pp, \mathbf{P}, \mu, s_l, a, a')$, it inputs the public parameters pp, a secret key s_l of user l, a public key list $\mathbf{P}$ including the public key P_l, a message μ, and predefined values a, a'. The prover $\mathcal{P}$ generates a ring signature for message μ as follows:

  - Computes the parameters $t_0, t_1, t_2, t'_0, t'_1, t'_2$.
  - For all $i \in [n]$, randomly selects the $k_i \xleftarrow{\$} \mathbb{Z}_q^*$.
  - Computes the commitment $R = com(\mathbf{k}) = \mathbf{P}^{\mathbf{k}}$.
  - Computes the challenge $c \leftarrow \mathcal{H}(\mathbf{P} \| R \| \mu)$.
  - Computes $e \leftarrow F(R)$.
  - Sets the vector $\mathbf{1} = (1, \cdots, 1)$.
  - Computes the response value $\mathbf{f} = g(\mathbf{k}, c, e, \mathbf{s}) = \frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c}$.

- Sets $n' = n/2$ to begin folding.
- Computes the $\mathbf{f_L} = (f_1, \cdots, f_{n'})$.
- Computes the $\mathbf{f_R} = (f_{n'+1}, \cdots, f_n)$.
- Computes the $\mathbf{P_L} = (P_1, \cdots, P_{n'})$.
- Computes the $\mathbf{P_R} = (P_{n'+1}, \cdots, P_n)$.
- Computes the $L_B = \mathbf{P_L}^{\mathbf{f_R}}$, $R_B = \mathbf{P_R}^{\mathbf{f_L}}$.
- Computes the $x \leftarrow \mathcal{H}(L_B \| R_B)$.
- Folding the response value $\mathbf{f}$ by $\mathbf{f}' = \mathbf{f_R} \cdot x + \mathbf{f_L} \cdot x^{-1}$ and $\mathbf{P}' = \mathbf{P}_R{}^{x^{-1}} \circ \mathbf{P}_L{}^{x}$.
- Repeats above folding process $\log n$ rounds and each time replaces the $\mathbf{P}, \mathbf{f}$ by $\mathbf{P}', \mathbf{f}'$.

Finally, The prover $\mathcal{P}$ outputs the ring signature $\sigma = (R, f', \mathbf{L}_B, \mathbf{R}_B)$.

- For the $\mathsf{RingBPVerify}(pp, \mathbf{P}, \mu, a, a', \sigma)$, it takes the public parameters pp, a public keys set $\mathbf{P}$, a message μ, predefined values a, a', the setting parameters $t_0, t_1, t_2, t_0', t_1', t_2'$ and the signature $\sigma = (R, f', \mathbf{L}_B, \mathbf{R}_B)$ as input. The c and e can be computed by $\mathcal{V}$. The x_j for $j \in [\log n]$ can be computed by $x_j \leftarrow \mathcal{H}(L_{Bj} \| R_{Bj})$. Besides, for all $i \in [n]$, $\mathcal{V}$ computes $s_i = \sum_{j=1}^{\log n} x_j{}^{t(i,j)}$. If $(i-1)$'s j-th is 0, we set $t(i,j) = 1$; else if, $t(i,j) = -1$. Then, $\mathcal{V}$ checks the validity of the signature. If t_0' and t_2' are both equal to 0, the $\mathcal{V}$ checks whether $\mathbf{P}^{\cdot f'^{-1} \cdot \mathbf{s} \cdot t_0} \cdot (\mathbf{1} \cdot R)^{f'^{-1} \cdot \mathbf{s} \cdot a} \cdot (\mathbf{1} \cdot G)^{f'^{-1} \cdot \mathbf{s} \cdot t_1} \cdot \mathbf{P}^{c \cdot f'^{-1} \cdot \mathbf{s} \cdot t_2} \stackrel{?}{=} R^{a'} \cdot G^{t_1'} \cdot \mathbf{L}_B{}^{\mathbf{x}^2} \cdot \mathbf{R}_B{}^{\mathbf{x}^{-2}}$ is valid. Otherwise, the $\mathcal{V}$ checks the equation $\mathbf{P}^{f' \cdot \mathbf{s} \cdot t_0'} \cdot (\mathbf{1} \cdot R)^{f' \cdot \mathbf{s} \cdot a'} \cdot (\mathbf{1} \cdot G)^{f' \cdot \mathbf{s} \cdot t_1'} \cdot \mathbf{P}^{c \cdot f' \cdot \mathbf{s} \cdot t_2'} \stackrel{?}{=} \mathbf{P}^{\mathbf{1} \cdot t_0} \cdot R^a \cdot G^{t_1} \cdot \mathbf{P}^{c \cdot \mathbf{1} \cdot t_2} \cdot \mathbf{L}_B{}^{\mathbf{x}^2} \cdot \mathbf{R}_B{}^{\mathbf{x}^{-2}}$.

Therefore, the formal construction of our proposed general ring signature for logarithmic size is shown in Algorithms 5, 6, 7 and 8, namely as $\mathsf{RingBPSetup}$, $\mathsf{RingBPKeyGen}$, $\mathsf{RingBPSign}$ and $\mathsf{RingBPVerify}$, respectively.

Algorithm 5 RingBPSetup(λ)

1: Chooses the parameters q, n;

2: Chooses the generator G;

3: Defines a hash function $\mathcal{H} : \{0,1\}^*$;

4: Chooses the parameters $t_0, a, t_1, t_2, t'_0, a', t'_1, t'_2$;

5: **return** pp.

Algorithm 6 RingBPKeyGen(pp)

1: Samples an arbitrary number $s_l \xleftarrow{\$} \mathbb{Z}_q^*$;

2: Computes $P_l = com({s_l}^{-1})$;

3: **return** (P_l, s_l).

3.5 Case Studies

We perform four case studies to show how to transform a Σ-protocol based signature into a ring signature in this Section, including the Schnorr, ECDSA, EdDSA, and SM2 cases.

3.5.1 Schnorr Case

In SchnorrRingSetup algorithm, we set $t_0 = 0$, $a = 1$, $t_1 = c$, $t_2 = 0$, $t'_0 = 1$, $a' = 0$, $t'_1 = 0$, $t'_2 = 0$. The SchnorrRingKeyGen algorithm is the same as RingKeyGen algorithm. In SchnorrRingSign algorithm, the response value is computed as $\mathbf{f} \leftarrow \mathbf{k} + c \cdot \mathbf{s}$. In SchnorrRingVerify algorithm, $\mathcal{V}$ checks if the formula $\mathbf{P}^{\mathbf{f}} = G^{\mathcal{H}(\mu \| R \| \mathbf{P})} \cdot R$ holds.

3.5.2 ECDSA Case

For ECDSA case, we set $t_0 = 0$, $a = 0$, $t_1 = e$, $t_2 = 1$, $t'_0 = 0$, $a' = 1$, $t'_1 = 0$, $t'_2 = 0$ in ECDSARingSetup algorithm. The ECDSARingKeyGen algorithm follows the RingKeyGen algorithm. The response value in ECDSARingSign algorithm is computed

Algorithm 7 $\mathsf{RingBPSign}(pp, \mathbf{P}, \mu, s_l, a, a')$

Computes $t_0, t_1, t_2, t'_0, t'_1, t'_2$;
for all $i \in [n]$ **do**
    Randomly chooses $k_i \xleftarrow{\$} \mathbb{Z}_q^*$;
end for
Computes $R = com(\mathbf{k}) = \mathbf{P}^{\mathbf{k}}$;
Computes $e \leftarrow F(R)$;
Sets $c \leftarrow \mathcal{H}(\mu \| R \| \mathbf{P})$;
for all $i \in [n], i \neq l$ **do**
    Sets $f_i \leftarrow g(k, c, e) = \frac{t_0 + a \cdot k_i + t_2 \cdot c}{t'_0 + a' \cdot k_i + t'_2 \cdot c}$;
end for
for $i \in [n], i = l$ **do**
    Computes $f_l \leftarrow g(k, c, e, s) = \frac{t_0 + a \cdot k_l + t_1 \cdot s_l + t_2 \cdot c}{t'_0 + a' \cdot k_l + t'_1 \cdot s_l + t'_2 \cdot c}$;
end for
if $n = 1$ **then**
    Sets $f' = f$;
else if **then**
    Sets $n' = \frac{n}{2}$;
    Computes $\mathbf{f_L} = (f_1, \cdots, f_{n'})$;
    Computes $\mathbf{f_R} = (f_{n'+1}, \cdots, f_n)$;
    Computes $\mathbf{P_L} = (P_1, \cdots, P_{n'})$;
    Computes $\mathbf{P_R} = (P_{n'+1}, \cdots, P_n)$;
    Computes $L_B = \mathbf{P_L}^{\mathbf{f_R}}$ and $R_B = \mathbf{P_R}^{\mathbf{f_L}}$;
    Computes $x \leftarrow \mathcal{H}(L_B \| R_B)$;
    Computes $\mathbf{P}' = \mathbf{P}_R{}^{x^{-1}} \circ \mathbf{P}_L{}^{x}$;
    Folding the response value $\mathbf{f}' = \mathbf{f_R} \cdot x + \mathbf{f_L} \cdot x^{-1}$;
    Sets $n = n'$, $\mathbf{f} = \mathbf{f}'$ and $\mathbf{P} = \mathbf{P}'$;
    Recursively execution on the **if** loop;
end if
return $\sigma = (R, f', \mathbf{L}_B, \mathbf{R}_B)$.

Algorithm 8 RingBPVerify$(pp, \mathbf{P}, \mu, a, a', \sigma)$

Computes $t_0, t_1, t_2, t'_0, t'_1, t'_2$;

Computes $e \leftarrow F(R)$;

Computes $c \leftarrow \mathcal{H}(\mu \| R \| \mathbf{P})$;

for all $j \in [\log n]$ **do**

    Computes $x_j = \mathcal{H}(L_{Bj} \| R_{Bj})$;

end for

for all $i \in [n]$ **do**

    Computes $s_i = \sum_{j=1}^{\log n} {x_j}^{t(i,j)}$, where $t(i,j) = 1$ if $(i-1)$'s j-th is 0, else if $t(i,j) = -1$;

end for

if $t'_0 = 0$ and $t'_2 = 0$ **then**

    Checks $\mathbf{P}^{\cdot {f'}^{-1} \cdot \mathbf{s} \cdot t_0} \cdot (\mathbf{1} \cdot R)^{{f'}^{-1} \cdot \mathbf{s} \cdot a} \cdot (\mathbf{1} \cdot G)^{{f'}^{-1} \cdot \mathbf{s} \cdot t_1} \cdot \mathbf{P}^{c \cdot {f'}^{-1} \cdot \mathbf{s} \cdot t_2} \stackrel{?}{=} R^{a'} \cdot G^{t'_1} \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}}$;

else if then

    Checks $\mathbf{P}^{f' \cdot \mathbf{s} \cdot t'_0} \cdot (\mathbf{1} \cdot R)^{f' \cdot \mathbf{s} \cdot a'} \cdot (\mathbf{1} \cdot G)^{f' \cdot \mathbf{s} \cdot t'_1} \cdot \mathbf{P}^{c \cdot f' \cdot \mathbf{s} \cdot t'_2} \stackrel{?}{=} \mathbf{P}^{\mathbf{1} \cdot t_0} \cdot R^a \cdot G^{t_1} \cdot \mathbf{P}^{c \cdot \mathbf{1} \cdot t_2} \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}}$;

end if

return 'accept' or 'reject'.

as $\mathbf{f} \leftarrow (c + e \cdot \mathbf{s}) \cdot \mathbf{k}^{-1}$. In ECDSARingVerify algorithm, $\mathcal{V}$ checks $\mathbf{P}^{\mathcal{H}(\mu \| \mathbf{P}) \cdot \mathbf{f}^{-1}} \cdot (\mathbf{1} \cdot G)^{F(R) \cdot \mathbf{f}^{-1}} = R$.

3.5.3 EdDSA Case

We transfer EdDSA case by setting $t_0 = 0$, $a = 1$, $t_1 = c$, $t_2 = 0$, $t'_0 = 1$, $a' = 0$, $t'_1 = 0$, $t'_2 = 0$ in EdDSARingSetup algorithm. The EdDSARingKeyGen algorithm follows the RingKeyGen algorithm. In EdDSARingSign algorithm, the response $\mathbf{f}$ is performed by $\mathbf{f} \leftarrow \mathbf{k} + c \cdot \mathbf{s}$. In EdDSARingVerify algorithm, $\mathcal{V}$ checks the formula $\mathbf{P}^{\mathbf{f}} = G^{\mathcal{H}(R \| \mathbf{P} \| \mu)} \cdot R$.

3.5.4 SM2 Case

For SM2 case, we set $t_0 = 0$, $a = 1$, $t_1 = -e$, $t_2 = 0$, $t'_0 = 1$, $a' = 0$, $t'_1 = 1$, $t'_2 = 0$ in SM2RingSetup algorithm. The SM2RingKeyGen algorithm is same as RingKeyGen algorithm. In the SM2RingSign algorithm, the response is calculated by $\mathbf{f} \leftarrow (1 + \mathbf{s})^{-1} \cdot (\mathbf{k} - e \cdot \mathbf{s})$ and $\mathcal{V}$ checks if $\mathbf{P}^{\mathbf{f}} \cdot (\mathbf{1} \cdot G)^{\mathbf{f}} \cdot G^{\mathcal{H}(R) + \mathcal{H}(ZA \| \mu)} = R$ holds in SM2RingVerify algorithm.

3.6 Security Analysis

Our general ring signature satisfies the correctness, anonymity, and unforgeability.

3.6.1 Correctness

Theorem 2 (**Correctness**). *Our general ring signature is converted from the Σ-protocol-based signature to satisfy the correctness (completeness) property.*

Proof. In our construction, commitment satisfies the homomorphism property for $\mathbf{k}$,

$\mathbf{s}$, the relation $com(\mathbf{f}) = com(g(\mathbf{k}, c, e, \mathbf{s})) = g(com(\mathbf{k}), c, e, com(\mathbf{s}))$ holds for $\mathbf{f} = g(\mathbf{k}, c, e, \mathbf{s})$. To explain with the most classic Schnorr example, response $\mathbf{f} = \mathbf{k} + c \cdot \mathbf{s}$. Then $com(\mathbf{f}) = \mathbf{P}^{\mathbf{f}}$, and $com(g(\mathbf{k}, c, e, \mathbf{s})) = \mathbf{P}^{(\mathbf{k}+c\cdot\mathbf{s})} = \mathbf{P}^{\mathbf{k}} \cdot \mathbf{P}^{(c\cdot\mathbf{s})}$. Further derivation of this formula gives us $R \cdot G^c$, which is the result of $g(com(\mathbf{k}), c, e, com(\mathbf{s}))$. In summary, $com(\mathbf{f}) = com(g(\mathbf{k}, c, e, \mathbf{s})) = g(com(\mathbf{k}), c, e, com(\mathbf{s}))$ for Schnorr case is $\mathbf{P}^{\mathbf{f}} = \mathbf{P}^{(\mathbf{k}+c\cdot\mathbf{s})} = R \cdot G^c$.

For our $O(\log n)$-size scheme, the verification process follows the Bulletproofs [15] based on our $O(n)$-size scheme. $\mathcal{V}$ will check if the following equation

$$com(f') = com(\mathbf{f}) \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}} = g(com(\mathbf{k}), c, e, com(\mathbf{s})) \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}} \tag{3.7}$$

holds.

For our general construction, we have

$$\mathbf{f} = \frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c}, \tag{3.8}$$

and we denote it as

$$\mathbf{P}^{\mathbf{f}} = \mathbf{P}^{\frac{\mathbf{1}\cdot t_0 + a\cdot\mathbf{k} + t_1\cdot\mathbf{s} + \mathbf{1}\cdot t_2\cdot c}{\mathbf{1}\cdot t'_0 + a'\cdot\mathbf{k} + t'_1\cdot\mathbf{s} + \mathbf{1}\cdot t'_2\cdot c}}. \tag{3.9}$$

Expanding the above equation, we obtain

$$\mathbf{P}^{\mathbf{f}(\mathbf{1}\cdot t'_0 + a'\cdot\mathbf{k} + t'_1\cdot\mathbf{s} + \mathbf{1}\cdot t'_2\cdot c)} = \mathbf{P}^{(\mathbf{1}\cdot t_0 + a\cdot\mathbf{k} + t_1\cdot\mathbf{s} + \mathbf{1}\cdot t_2\cdot c)}. \tag{3.10}$$

Thus, we can derive the verification formula as

$$\mathbf{P}^{\mathbf{f}\cdot t'_0} \cdot (\mathbf{1} \cdot R)^{\mathbf{f}\cdot a'} \cdot (\mathbf{1} \cdot G)^{\mathbf{f}\cdot t'_1} \cdot \mathbf{P}^{c\cdot\mathbf{f}\cdot t'_2} = \mathbf{P}^{\mathbf{1}\cdot t_0} \cdot R^a \cdot G^{t_1} \cdot \mathbf{P}^{c\cdot\mathbf{1}\cdot t_2}. \tag{3.11}$$

For the situation of both $t'_0 = 0$ and $t'_2 = 0$, the verification equation is

$$\mathbf{P}^{\mathbf{f}^{-1}} = \mathbf{P}^{(\frac{\mathbf{1}\cdot t_0 + a\cdot\mathbf{k} + t_1\cdot\mathbf{s} + \mathbf{1}\cdot t_2\cdot c}{\mathbf{1}\cdot t'_0 + a'\cdot\mathbf{k} + t'_1\cdot\mathbf{s} + \mathbf{1}\cdot t'_2\cdot c})^{-1}}. \tag{3.12}$$

In this way, we have

$$\mathbf{P}^{\mathbf{f}^{-1}\cdot(\mathbf{1}\cdot t_0 + a\cdot\mathbf{k} + t_1\cdot\mathbf{s} + \mathbf{1}\cdot t_2\cdot c)} = \mathbf{P}^{(\mathbf{1}\cdot t'_0 + a'\cdot\mathbf{k} + t'_1\cdot\mathbf{s} + \mathbf{1}\cdot t'_2\cdot c)}. \tag{3.13}$$

Expanding the above form, we have the final verification equation

$$\mathbf{P}^{\mathbf{f}^{-1}\cdot t_0} \cdot (\mathbf{1}\cdot R)^{\mathbf{f}^{-1}\cdot a} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^{-1}\cdot t_1} \cdot \mathbf{P}^{c\cdot\mathbf{f}^{-1}\cdot t_2} = \mathbf{P}^{\mathbf{1}\cdot t'_0} \cdot R^{a'} \cdot G^{t'_1} \cdot \mathbf{P}^{c\cdot\mathbf{1}\cdot t'_2}. \tag{3.14}$$

Hence, our scheme satisfies the property of completeness (correctness). □

3.6.2 Anonymity

Theorem 3 (**Anonymity**). *Our general ring signature based on Σ-protocol is anonymous, with the witness $\mathbf{s}$ and the statement $\mathbf{P} = com(\mathbf{s})$. We say it is HVZK, if there is a PPT algorithm $\mathcal{S}$, called the simulator, which has the ability that take the statement $\mathbf{P}$ as the input and always outputs accepted transcripts $(R, c, e, \mathbf{f})$ for $\mathcal{P}$ with the same distribution as a real transcript generate between $\mathcal{P}(\mathbf{s}, \mathbf{P})$ and $\mathcal{V}(\mathbf{P})$.*

Proof. The anonymity game can be described below, referred to as $Game_{Anony}$, involving an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- **Setup**: $\mathcal{C}$ inputs the security parameter λ to execute RingSetup algorithm to procure public parameters *pp*, and returns it to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded n queries adaptively:

- Random Oracle $\mathcal{O}_H$: We set a hash function $\mathcal{H}$ in $\mathcal{O}_H$. The list L_H : $\{\mu, \mathbf{P}, R, c\}$ is initially empty. Upon received a query on $(\mu, \mathbf{P}, R)$, $\mathcal{C}$ first looks up in L_H. If the input has been queried before, the corresponding value c is output. Otherwise, $\mathcal{C}$ calculates correspond c, adds a tuple $\{\mu, \mathbf{P}, R, c\}$ in L_H and returns it to $\mathcal{A}$.
- Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, n]$, $\mathcal{A}$ queries the public key P_i. Then, $\mathcal{C}$ invokes RingKeyGen algorithm to get the key pair (P_i, s_i). Finally, $\mathcal{C}$ returns public key P_i to $\mathcal{A}$.
- Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key s_i for index i. Then, $\mathcal{C}$ responds the answer to $\mathcal{A}$.
- Signing Oracle $\mathcal{O}_{SO}$: The $\mathcal{A}$ inputs a public keys list $\mathbf{P}$ include a public key P_i, a secret key sk_i, and a message μ, $\mathcal{C}$ calls the RingSign algorithm, and returns σ to $\mathcal{A}$.

- **Challenge**: $\mathcal{A}$ selects a public keys set $\mathbf{P}$, a message μ, and delivers them to $\mathcal{C}$ to request a signature. $\mathcal{C}$ randomly chooses $\varphi \in [1, n]$ and then calls the RingSign algorithm to get the signature σ_φ. Then, $\mathcal{C}$ sends σ_φ to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ returns a guess $\varphi^* \in [1, n]$.
- **Analysis**: If $\varphi^* = \varphi$, $\mathcal{A}$ wins $Game_{Anony}$. The advantage of $\mathcal{A}$ is defined by

$$Adv_{\mathcal{A}}^{Anony} = |\Pr[\varphi^* = \varphi] - \frac{1}{n}|, \tag{3.15}$$

  which is negligible.

The anonymity property of our ring signature also can be derived directly from the HVZK in Σ-protocol. To generate the accepting transcript without the witness $\mathbf{s}$, we have to utilize the simulator $\mathcal{S}$. In $\mathcal{S}$, we need to change the order of the parameters generation, which is different from the transcripts generated between $\mathcal{P}(\mathbf{s}, \mathbf{P})$ and $\mathcal{V}(\mathbf{P})$. To be precise, we take statement $\mathbf{P}$ as input and perform as:

1. Randomly chooses $\mathbf{f} \xleftarrow{\$} \mathbb{Z}_q^*$;

2. Randomly chooses $e, c \xleftarrow{\$} \mathbb{Z}_q^*$;

3. Computes the value of R to satisfy the verification equation.

Thus, we can output the transcript $(R, c, e, \mathbf{f})$ by the simulator $\mathcal{S}$. Indeed, the transcript has the right distribution. The parameters c and $\mathbf{f}$ are independent and R is calculated by the real relation of the verification, fulfilling the correctness check requirement. Hence, the transcript obtained from the simulator $\mathcal{S}$ is the same as the real one. Thus, our ring signature is anonymous. □

3.6.3 Unforgeability

Theorem 4 (**Unforgeability**). *Our general ring signature based on Σ-protocol is unforgeable, providing knowledge soundness if there exists an efficient deterministic witness extractor algorithm $\mathcal{E}$, satisfies: For witness $\mathbf{s}$, we have a statement $\mathbf{P} = com(\mathbf{s})$ with relation $\mathbf{P^s} = G$. Given two accepting conversations $(\mathbf{k}, c_a, e, \mathbf{f}_a)$ and $(\mathbf{k}, c_b, e, \mathbf{f}_b)$, where $c_a \neq c_b$, the algorithm $\mathcal{E}$ always outputs the witness $\mathbf{s}$ such that $(\mathbf{s}, \mathbf{P}) \in \mathcal{R}$.*

Proof. For our generic ring signature, without a secret key, it is not possible to generate a valid signature. In particular, the unforgeability can be given by the game shown below, referred to $Game_{Forge}$, including adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

- **Setup**: $\mathcal{A}$ randomly picks a target index $i^* \in [1, n]$. $\mathcal{C}$ runs RingSetup algorithm to procure the system public parameters pp by input security parameter λ, and returns it to $\mathcal{A}$.

- **Oracle Query**: $\mathcal{A}$ performs a polynomial bounded n queries adaptively:

- Random Oracle $\mathcal{O}_H$: We set a hash function $\mathcal{H}$ in $\mathcal{O}_H$. The list L_H : $\{\mu, \mathbf{P}, R, c\}$ is initially empty. Upon received a query on $(\mu, \mathbf{P}, R)$, $\mathcal{C}$ firstly looks up in L_H. If the input has been queried before, the corresponding value c is the output. Otherwise, $\mathcal{C}$ calculates correspond c, adds a tuple $\{\mu, \mathbf{P}, R, c\}$ in L_H and returns it to $\mathcal{A}$.
- Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, n]$, $\mathcal{A}$ queries the public key P_i. Then, $\mathcal{C}$ invokes RingKeyGen algorithm to get the key pair (P_i, s_i). Finally, $\mathcal{C}$ returns public key P_i to $\mathcal{A}$.
- Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ quires the secret key s_i for index $i \in [1, n]$. $\mathcal{C}$ returns s_i to $\mathcal{A}$ if $i \neq i^*$. Otherwise, the query is aborted and $\mathcal{C}$ returns $\perp$.
- Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ_i for user $i \in [1, n]$. The signer's public key P_i has to include in the public key sets $\mathbf{P}$, otherwise, if $P_i \notin \mathbf{P}$, $\mathcal{C}$ returns $\perp$. If $i \neq i^*$, $\mathcal{C}$ returns σ_i to $\mathcal{A}$ by invoking the RingSign algorithm. If $i = i^*$, the query cannot be aborted. In this case, $\mathcal{C}$ also calls the RingSign algorithm to obtain the signature σ_i. However, in this process, the secret key of signer s_i is missing, so $\mathcal{C}$ uses the ability of the simulator $\mathcal{S}$ (has been described above).

- **Forge**: The adversary $\mathcal{A}$ outputs a forgery signature σ_{i^*}, including the message μ^*, a public keys set $\mathbf{P}^*$, and the signer's public key $P_{i^*} \in \mathbf{P}^*$. After that, $\mathcal{C}$ invokes RingVerify algorithm to check the signature σ_{i^*}. We say that $\mathcal{A}$ wins the $Game_{Forge}$ game, if:
  - The message μ^* of forgery signature σ_{i^*} has not queried in $\mathcal{O}_{SO}$.
  - Inputs σ_{i^*} to RingVerify algorithm, which returns 'accept'.
  - $\mathcal{A}$ does not have any secret key related to public keys set $\mathbf{P}^*$.
- **Analysis**: The advantage of $\mathcal{A}$ in $Game_{Forge}$ game is defined as:

$$Adv_{\mathcal{A}}^{Forge} = |\Pr[\mathcal{A}\ wins\ Game_{Forge}]|, \tag{3.16}$$

which is negligible.

In addition, our general ring signature is designed based on the Σ-protocol, which is knowledge soundness. If the signature is forged by an adversary $\mathcal{A}$, the extractor $\mathcal{E}$ cannot extract the wittiness value through the rewind technique. In particular, we rewind the prover $\mathcal{P}$ two times by the extractor $\mathcal{E}$ to get two accepting conversations $(\mathbf{k}, c_a, e, \mathbf{f}_a)$ and $(\mathbf{k}, c_b, e, \mathbf{f}_b)$ for P with $c_a \neq c_b$. Thus, we have two equations

$$\mathbf{f}_a = \frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_{1a} \cdot \mathbf{s} + \mathbf{1} \cdot t_{2a} \cdot c_a}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_{1a} \cdot \mathbf{s} + \mathbf{1} \cdot t'_{2a} \cdot c_a}, \tag{3.17}$$

and

$$\mathbf{f}_b = \frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_{1b} \cdot \mathbf{s} + \mathbf{1} \cdot t_{2b} \cdot c_b}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_{1b} \cdot \mathbf{s} + \mathbf{1} \cdot t'_{2b} \cdot c_b}. \tag{3.18}$$

Extending the formula, we extract two different expressions about $\mathbf{k}$ as

$$\mathbf{k} = \frac{\mathbf{f}_a \cdot t'_0 + \mathbf{f}_a \circ \mathbf{s} \cdot t'_{1a} + \mathbf{f}_a \cdot t'_{2a} \cdot c_a - \mathbf{1} \cdot t_0 - t_{1a} \cdot \mathbf{s} - \mathbf{t_2}_a \cdot c_a}{\mathbf{1} \cdot a - \mathbf{f}_a \cdot a'}, \tag{3.19}$$

and

$$\mathbf{k} = \frac{\mathbf{f}_b \cdot t'_0 + \mathbf{f}_b \circ \mathbf{s} \cdot t'_{1b} + \mathbf{f}_b \cdot t'_{2b} \cdot c_b - \mathbf{1} \cdot t_0 - t_{1b} \cdot \mathbf{s} - \mathbf{t_2}_b \cdot c_b}{\mathbf{1} \cdot a - \mathbf{f}_b \cdot a'}. \tag{3.20}$$

Through a series of mathematical transformations (e.g., cross-multiplication, elimination/combination of like terms), we successfully extract the witness value s, as:

$$\mathbf{s} = \frac{\begin{array}{c} a \cdot t'_0 \cdot (\mathbf{f}_a - \mathbf{f}_b) + a \cdot (\mathbf{f}_a \cdot t'_{2a} \cdot c_a - \mathbf{f}_b \cdot t'_{2b} \cdot c_b) - \mathbf{1} \cdot a \cdot \\ (t_{2a} \cdot c_a - t_{2b} \cdot c_b) - a' \cdot \mathbf{f}_a \circ \mathbf{f}_b \cdot (t'_{2a} \cdot c_a - t'_{2b} \cdot c_b) - \\ a' \cdot t_0 \cdot (\mathbf{f}_a - \mathbf{f}_b) + a' \cdot (\mathbf{f}_b \cdot t_{2a} \cdot c_a - \mathbf{f}_a \cdot t_{2b} \cdot c_b) \end{array}}{\begin{array}{c} a \cdot (\mathbf{f}_b \cdot t'_{1b} - \mathbf{f}_a \cdot t'_{1a}) - \mathbf{1} \cdot a \cdot (t_{1b} - t_{1a}) - \\ a' \cdot \mathbf{f}_a \circ \mathbf{f}_b \cdot (t'_{1b} - t'_{1a}) + a' \cdot (\mathbf{f}_a \cdot t_{1b} - \mathbf{f}_b \cdot t_{1a}) \end{array}} \tag{3.21}$$

Therefore, our general ring signature satisfies the property of unforgeability.

□

3.7 Performance Evaluation and Comparison

To evaluate our proposed ring signature, we show the implementation of our four case studies in Java language based on the JPBC library, and we use the 256-bit elliptic curve (secp256k1) to evaluate performance. All experiments were conducted in a system environment featuring an Apple M2 processor and 16.0 GB of memory. Moreover, we conduct a comparative analysis of our ring signature with others [33, 12, 74] in computation and communication overhead.

Note that our main goal is to propose a generic construction that can convert existing signatures to ring signatures, enabling scholars to enhance the anonymity of privacy-preserving systems more concisely and retain the original properties. Therefore, in the comparative analysis, we focus on evaluating whether the overhead of our generic construction is acceptable for real-world applications, rather than aiming for a significant performance improvement.

3.7.1 Our Experiment Results

We first evaluate the computation and communication cost of our proposed $O(n)$-size and $O(\log n)$-size general ring signature.

Computation Overhead

Firstly, we show the running time cost of our four case studies. The time overhead of the Schnorr ring signature, ECDSA ring signature, EdDSA ring signature, and SM2 ring signature are shown in Figure 3.3(a), 3.3(b), 3.3(c), 3.3(d), respectively. In our experiments, the time required to generate an original Σ-based signature is around 6 ms, which is the same as when there is only one ring member in our $O(n)$-size ring signature. When we transform it into a ring signature, the security level improves, and the time costs increase smoothly as the ring size grows.

In Figure 3.3, the time cost increases slightly and does not introduce a large additional overhead when we convert the Σ-based signature to our $O(n)$-size ring signature. For our $O(\log n)$-size ring signature, as ring size n increases, the time cost grows linearly. Compared with our $O(n)$-size ring signature, although the time overhead becomes larger, the size of the signature is substantially reduced, which is a trade-off between time overhead and signature size, and it is acceptable.

In detail, when the ring member is 8, the signature and verification time are around 41.56 ms to 47.83 ms, 33.6 ms to 38.84 ms for the linear-size scheme, and 169.12 ms to 174.38 ms, 83.22 ms to 87.43 ms for the logarithmic-size scheme, respectively.

Communication Overhead

From the perspective of our $O(n)$-size scheme, the signature only consists of R and $\mathbf{f}$, where $R \in \mathbb{G}$, $f_i \in \mathbb{Z}_q$. So, the communication cost is $1|\mathbb{G}| + n|\mathbb{Z}_q|$. In our $O(\log n)$-size scheme, the signature is $\sigma = (R, f', \mathbf{L}_B, \mathbf{R}_B)$, where $R \in \mathbb{G}$, $f' \in \mathbb{Z}_q$,

$\mathbf{L}_B, \mathbf{R}_B \in \mathbb{G}$. Thus, the communication cost is $(1 + 2\log n)|\mathbb{G}| + 1|\mathbb{Z}_q|$. The specific overhead is related to the selection of an elliptic curve. Note that the parameter in $\mathbb{G}$ is represented by 264 bits, and the parameter in $\mathbb{Z}_q$ is represented by 256 bits.

When we convert a Σ-based signature to a ring signature, the size is related to the ring size n. When the ring size $n = 64$, the size of $\mathbf{f}$ in our $O(n)$-size scheme is $64 \times 256 = 16384$ bits, so the overall size is $264 + 16384 = 16648$ bits. While when we perform six $(\log 64)$ rounds of Bulletproofs folding on $\mathbf{f}$, the size of $\mathbf{f}$ will be reduced from $64 \times 256 = 16384$ bits to $1 \times 256 = 256$ bits. However, the folding processes two list additional values $\mathbf{L}_B$ and $\mathbf{R}_B$, which respectively have a size of $\log 64 \times 264 = 1584$ bits. Thus, after Bulletproofs folding, the communication costs are $256 + 264 + 2 \times 1584 = 3688$ bits, which is 0.22 times the $O(n)$-size general ring signature. Therefore, in our $O(\log n)$-size scheme, the size of the ring signature grows pretty smoothly as the number of ring members increases. The comprehensive communication cost analysis of our $O(\log n)$-size general ring signature is shown in Table 3.1.

3.7.2 Comparison with Prior Arts

For the sake of showing the practicality of our general ring signature construction, we compare it with other ring signatures. Our $O(\log n)$-size generic ring signature performance is implementation-acceptable on common devices and has a better computation and communication overhead than other ring signature schemes [33, 12, 74]. Thus, it is a suitable solution for privacy-preserving systems using various Σ-based signatures to employ our generic construction, converting existing signatures to ring signatures to provide user anonymity.

Computation Overhead

In the following, we perform a comparison analysis with other existing schemes [33, 12, 74] in the context of signing and verifying time overhead, and signature size, using our Schnorr ring signature as a representative. Note that the performance of the Feng et al. scheme [29] is consistent with that of Yuen et al. [74] and is thereby not repeated below.

As shown in Figure 3.4(a), the signing time of schemes [33] and [12] is higher than others, since their schemes have $O(n \log n)$ exponentiation costs. Our $O(\log n)$-size scheme has a slightly lower signing time overhead than the $O(\log n)$-size scheme of Yuen et al [74]. Similarly, our $O(n)$-size scheme has a slightly lower signing time overhead than the $O(n)$-size scheme of Yuen et al [74]. In short, in signing time overhead performance, both our $O(\log n)$-size scheme and $O(n)$-size scheme have the best performance.

A comparison of verifying time overhead is illustrated in Figure 3.4(b). As same as the signing time cost, the verifying time of schemes [33, 12] is higher than other ring signatures. The overhead of Yuen et al.'s $O(\log n)$-size scheme [74] is slightly higher than our $O(\log n)$-size scheme. Moreover, Yuen et al.'s $O(n)$-size scheme [74] is slightly higher than our $O(n)$-size scheme. Therefore, our scheme performs best among other existing schemes [33, 12, 74] in verification time overhead.

In particular, the signing and verifying times of our scheme are 0.38 - 0.83 times and 0.23 - 0.65 times compared to other existing schemes, respectively.

Communication Overhead

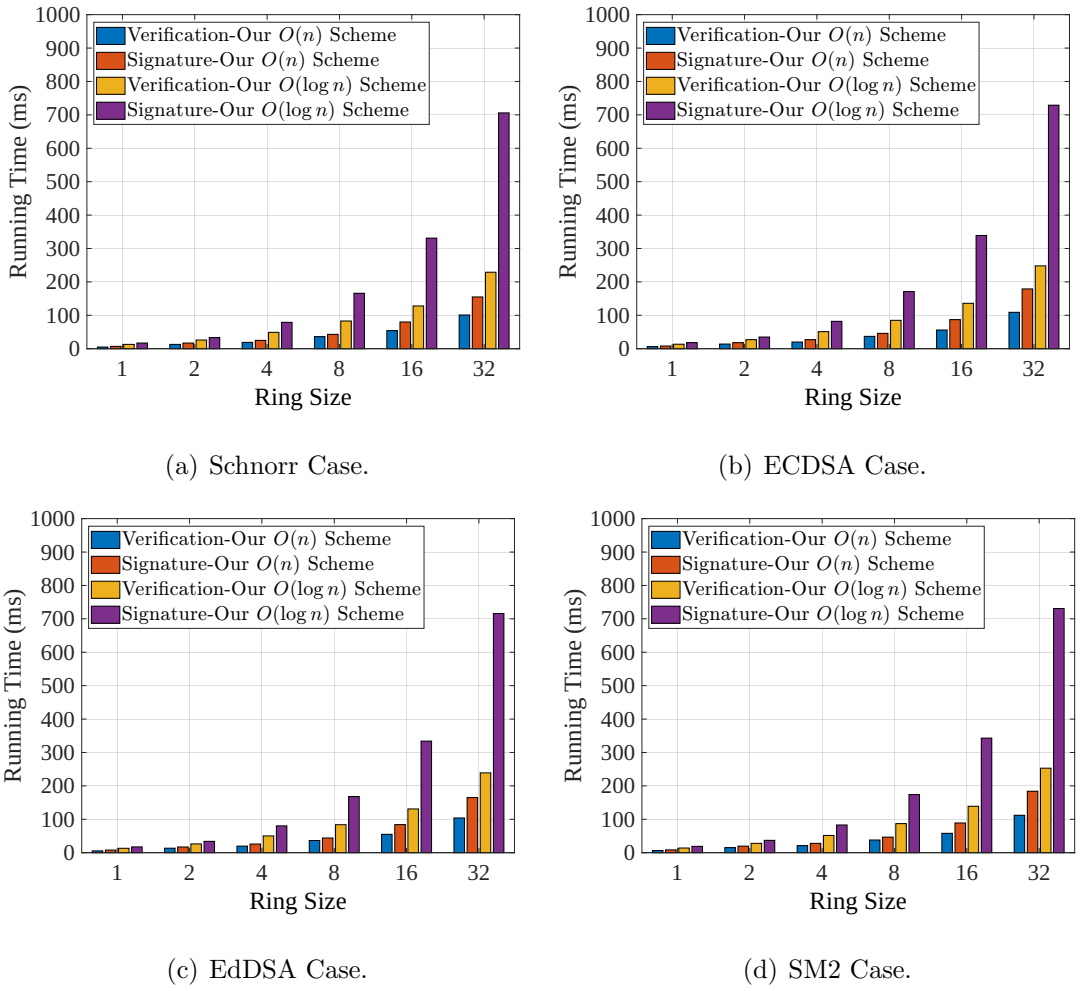
Table 3.1 shows the theoretical communication cost analysis. In scheme [33], the ring signature size is $(3 \log n + 1)|\mathbb{Z}_q| + (4 \log n)|\mathbb{G}|$ and the size of scheme [12] is equal to $(1.5 \log n + 6)|\mathbb{Z}_q| + (\log n + 12)|\mathbb{G}|$, with n being the number of ring members. In our

scheme, the size of the ring signature is $1|\mathbb{Z}_q| + (2\log n + 1)|\mathbb{G}|$, which is obviously better than the above mentioned two schemes [33, 12] and is slightly better than the size $3|\mathbb{Z}_q| + (2\log n + 1)|\mathbb{G}|$ in scheme [74].

The comparison of our $O(\log n)$-size ring signature with others $O(\log n)$-size schemes [33, 12, 74] is also shown in Figure 3.5. We can observe that when the ring size $n \leq 16$, the size of Groth et al.'s [33] is lower than that of Bootle et al.'s [12], whereas the size of Groth et al.'s [33] grows substantially with increasing ring size. As ring size increases, the communication costs of Bootle et al.'s [12] increase rate less than the scheme [33]. More importantly, the communication costs of ours are slightly lower than Yuen et al's [74] and are substantially lower than that of scheme [33, 12], which is the best performance scheme till now.

Table 3.1: Comparison the Signature Size with Existing $O(\log n)$-Size Ring Signatures.

Schemes	Consisting Elements		Signature Size (Bytes)				
	$\mathbb{Z}_q$	$\mathbb{G}$	$n = 4$	$n = 8$	$n = 16$	$n = 32$	$n = 64$	$n = 1024$
Groth et al. [33]	$3\log n + 1$	$4\log n$	488	716	944	1172	1400	2312
Bootle et al. [12]	$1.5\log n + 6$	$\log n + 12$	750	831	912	993	1074	1398
Yuen et al. [74]	3	$2\log n + 1$	261	327	393	459	525	789
Our Scheme	1	$2\log n + 1$	197	263	329	395	461	725



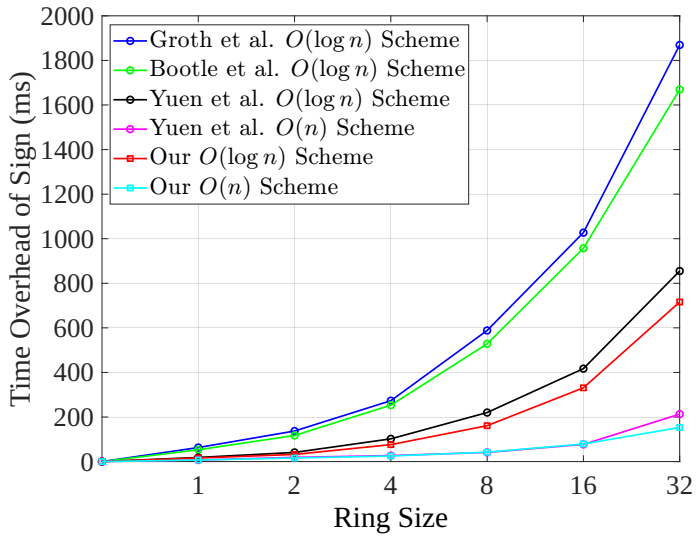
(a) Schnorr Case.

(b) ECDSA Case.

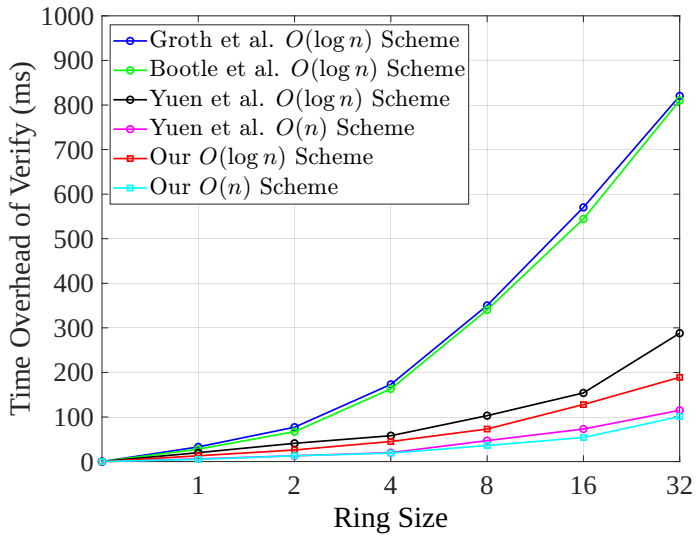
(c) EdDSA Case.

(d) SM2 Case.

Figure 3.3: Time Overhead Associated with Several Algorithms of Our $O(n)$ and $O(\log n)$ Ring Signature Schemes.



(a) Signing Time.



(b) Verifying Time.

Figure 3.4: Comparison of Computation Overhead between Our $O(n)$ and $O(\log n)$ Schemes and Current State-of-the-Art Ring Signature Primitives [33], [12], [74].

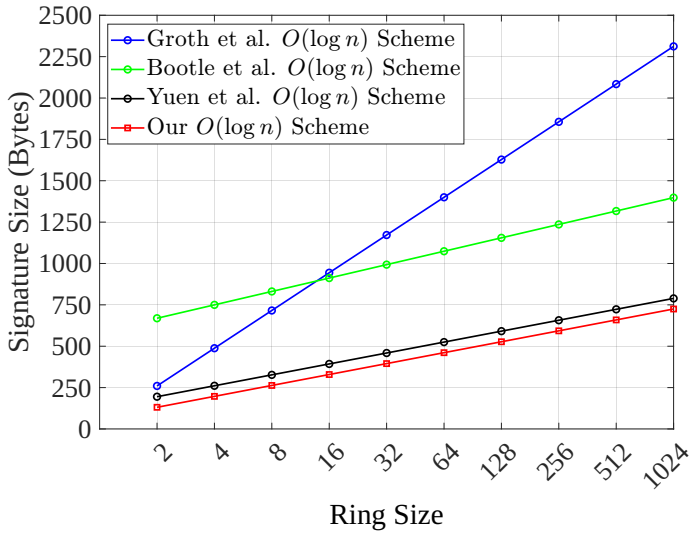


Figure 3.5: Comparison of Communication Cost between Our $O(\log n)$ Scheme and Current State-of-the-Art $O(\log n)$ Ring Signature Primitives [33], [12], [74].

Chapter 4

Lattice-based Ring Signature Construction and Application

In this Chapter, we first propose an efficient lattice-based linkable ring signature (LLRS) to guarantee patient privacy and EMRs security during data sharing in the cloud through anonymity and unforgeability. Our scheme also allows checking the linkability for multiple signatures. We then present an enhancement of our LLRS protocol, a lattice-based linkable ring signature with forward security, called FS-LLRS, additionally providing forward security of EMRs. It allows users to periodically update the secret keys, and thereby ensuring the security of prior ring signatures in the event of a leakage of the current key.

4.1 Overview

In this Section, we elaborate on our motivations and contributions.

4.1.1 Motivations

Following the trend of e-health, widely used EMRs sharing solves the bottleneck of information silos problem. Nevertheless, sensitive EMRs are susceptible to forgery or attack, and patient's personal information is vulnerable to leakage, resulting in critical privacy and security risks. Hence, we are devoted to proposing a secure and efficient EMRs sharing scheme.

First and foremost, we desire to propose an efficient lattice-based linkable ring signature that safeguards the integrity and unforgeability of EMRs and user anonymity. In addition, for multiple EMRs, the linkability feature enables the correlation among them to be verified.

Furthermore, we also need to present an enhancement (FS-LLRS) in order to mitigate the risk of secret key disclosure. In this way, we can provide forward-secure unforgeability and linkability in a quantum setting.

More centrally, we introduce a cloud-assisted EMRs sharing framework that applies the proposed LLRS and FS-LLRS schemes to assure the security of EMRs data, the privacy of users, and the reliability of healthcare services.

4.1.2 Contributions

We summarize our five-fold contributions to the work below.

- **Efficient LLRS Primitive.** We propose an efficient lattice-based linkable ring signature in the standard model, called LLRS, permitting verifiers to check whether multi-signatures are generated by the same user. Our primitive not only satisfies anonymity, unforgeability, and linkability, but also achieves superior efficiency in a quantum setting.

- **Novel FS-LLRS Primitive.** We introduce a novel lattice-based linkable ring

signature with forward security construction in the standard model, called FS-LLRS, which is an enhancement of our proposed LLRS scheme. We formally define the FS-LLRS primitive and its security models, i.e., anonymity, unforgeability with forward security, and linkability. Our solution addresses the vulnerability of ring signatures to secret key exposure attacks while enjoying linkability and quantum safety.

- **Cloud-assisted EMRs Sharing Framework.** We then propose a secure cloud-assisted EMRs sharing framework. Combined with our proposed LLRS primitive, we guarantee the unforgeability of EMRs, and the privacy of patients. When doctors are confronted with several EMRs of the same patient from clouds, they can check the integrity and correlation of these EMRs through linkability easily. After that, in conjunction with our FS-LLRS primitive, we can guarantee the forward security of users' secret keys, so that even if the current secret key has been compromised, the security of previous EMRs can also be guaranteed.

- **Security Analysis.** The rigorous security analysis demonstrates that our LLRS scheme offers anonymity, unforgeability and linkability in the standard model, which can be reduced to the hardness of the Short Integer Solution (SIS) problem. Our FS-LLRS scheme additionally provides the unforgeability with forward security in the standard model.

- **Performance Evaluation.** The comprehensive performance comparison indicates that our LLRS scheme outperforms prior arts [8], [38], [4], [68], [20] in communication and computation overhead. Notably, our LLRS scheme's communication overhead is only about 0.08 times of existing lattice-based ring signatures. The signing and verifying overheads of our LLRS are just 0.17 - 0.23 times and 0.32 - 0.67 times compared to others, respectively. Besides, our enhanced FS-LLRS scheme, which has the utmost security level and function-

ality, outperforms most existing schemes [8], [38], [4], [68], [20] in computation overhead. Specifically, its signing and verifying overheads are 0.79 - 1.01 times and 0.34 - 0.71 times compared to the others, respectively.

4.2 Technical Overview

In this Chapter, we address three main challenges as below.

The first challenge is how to enhance the efficiency in designing a lattice-based linkable ring signature in the standard model. To reduce the computational cost, we design the signature generation and verification process with the assistance of the concise Sigma protocol. We assume that the ring size is N and the signer $l \in [N]$ has two constraints: it has a secret key sk_l and a public key pk_l with the relation $pk_l \cdot sk_l = p$, where p is publicly available; its public key pk_l is in the public keys set. By regarding the secret key sk_l as a set $(0, \cdots, sk_l, \cdots, 0)$, we have the relation $\sum_{i=1}^{N} pk_i \cdot sk_i = 0 + \cdots + pk_l \cdot sk_l + \cdots + 0 = p$ in the verification process. To reduce the communication cost, we redesign the dimension of the parameters. In particular, the signature consists of n responses and one commitment. Our intuition is to reduce the dimension of the n responses, thereby reducing the communication overhead. Further, we replace the Fiat-Shamir heuristic in the traditional scheme with a method that selects the public matrices depending on each bit of the message to compute the challenge value, to enable our scheme without the Random Oracle Model (ROM).

The second challenge is how to design a lattice-based ring signature that provides both linkability and forward security. In existing research, forward security is mainly studied in the field of encryption [71]. Inspired by [73], [16], and [45], we design a secret key evolution mechanism for the lattice-based ring signature. Specifically, we use a l-level binary tree to divide the 2^l time period. Cooperating with the user's identity tag, we generate a matrix and its basis as the user's root public key and

root secret key. For the secret key update, we utilize the ExtBasis algorithm with the representation of leaf nodes, the user's root public key, and the public matrices selected based on time t to derive the user's secret key at time t. The updated secret key can be obtained from either the ancestor basis or the root secret key. To achieve both forward security and linkability, we need to redesign the computation of the link tag. Concretely, we require the user to retain its root secret key and compute the link tag using the root secret key and the public matrix. To further reduce the storage overhead and be applicable to EMRs sharing system, we introduce the identity-based property by our designed method in our proposed FS-LLRS scheme.

The third challenge is how to design a cloud-assisted EMRs sharing system that achieves EMRs integrity and unforgeability, protects users' information privacy, prevents users from exaggerating or downplaying medical conditions, ensures the reliability of medical services, and provides forward security and resistance to quantum attacks. The detailed framework will be described later.

4.3 Framework Description

4.3.1 System Model

The system model for our cloud-assisted EMRs sharing system is illustrated in Figure 4.1. It involves six entities described below.

- Patients: The patient gets healthcare and signs the EMR.
- Doctors: The doctor provides healthcare services, and signs the EMR, and finally uploads it to the cloud server.
- Requesters: The requester downloads the EMRs from the cloud server, and checks their validity and linkability.

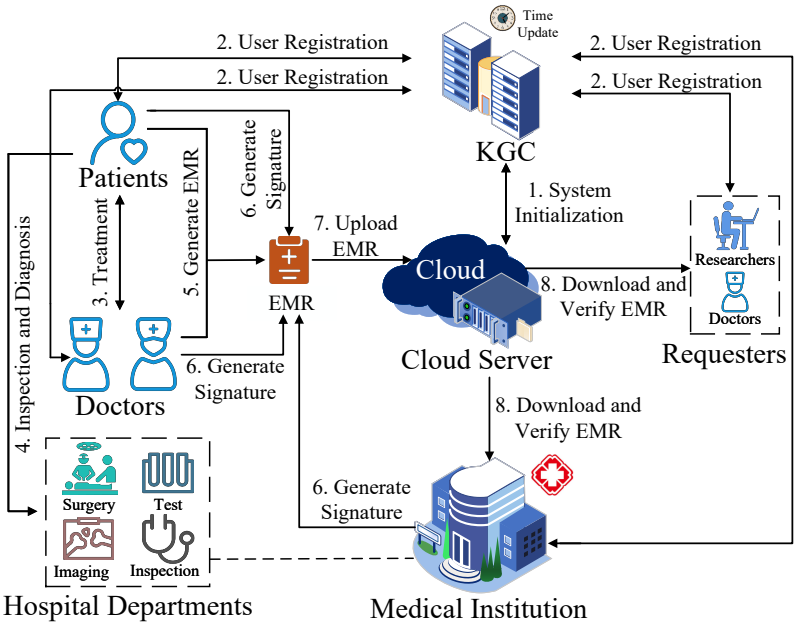


Figure 4.1: System Model of Cloud-assisted EMRs Sharing Framework.

- Key Generation Center (KGC): The KGC generates key pairs for users and updates them periodically.
- Medical Institutions: This entity signs the EMRs.
- Cloud Server: The cloud server stores EMRs and signatures. It also returns EMRs and signatures upon request.

4.3.2 Design Goals

We aim to establish a secure and efficient cloud-assisted EMRs sharing system using our proposed LLRS and FS-LLRS primitives. Therefore, the following objectives should be considered.

1. EMRs Integrity and Unforgeability: EMRs must be reliable and secure to ensure the accuracy of the service.

2. Users' Information Privacy: User anonymity must be ensured to protect personal information privacy.

3. Preventing Users from Exaggerating or Downplaying Medical Conditions: This prevents misuse for personal gain, such as falsely claiming disability benefits or misrepresenting mental health for employment purposes.

4. Reliability of Medical Services: It is essential to reconfirm the requested EMRs belong to the same user to ensure the services' reliability and prevent misdiagnosis.

5. Forward Security: It ensures that even if the current secret key has been compromised, the previously generated EMRs still remain secure.

6. Resistance to Quantum Attacks: Our system requires the ability to resist quantum computing attacks in order to achieve a higher level of security.

4.3.3 System Procedure

The basic procedures of our cloud-assisted EMRs sharing framework are illustrated in Figure 4.1, including five phases as follows. With the help of our proposed LLRS and FS-LLRS primitives, we have implemented a secure and efficient cloud-assisted EMRs sharing system. Since our proposed primitives are based on the lattice assumption, our system can resist the quantum attack (also meet the sixth goal).

System Initialization

As the Step. 1 in Figure 4.1, the KGC executes the Setup algorithm to initialize the system by inputting a security parameter λ to generate public parameters.

New User Registration

As the Step. 2 in Figure 4.1, the patients, doctors, and medical institutions can request their public-secret keys by invoking the KeyExtract algorithm from the KGC. In this phase, we simply note them as users.

Generate and Upload EMRs

As the Step. 3, 4, 5 in Figure 4.1, EMRs are created during patient-doctor interactions. Each EMR record undergoes a checking process involving multiple signatures before being uploaded to cloud storage by the doctor, as outlined below.

- EMR is generated during the treatment process, as the Step. 3, 4, 5 in Figure 4.1.
- Doctor, patient, and medical institution generate signatures using the Sign algorithm for the EMR, as the Step. 6 in Figure 4.1.
- The doctor uploads the EMR with its signatures to the cloud server, as the Step. 7 in Figure 4.1.

The use of ring signatures for EMRs achieves the first two design goals, that is, the integrity and unforgeability of EMRs and the users' anonymity.

To meet the third design goal, each EMR is required to be confirmed and signed by all parties involved (the patient, doctor, and medical institution), before being uploaded to the cloud for storage. This process prevents users from manipulating or misrepresenting the extent of their medical conditions.

Download and Verify EMRs

As the Step. 8 in Figure 4.1, when a patient undergoes follow-up medical treatment or a researcher conducts a medical study, the EMR requester can download EMRs from the cloud and validate each EMR as:

- Requester downloads the EMRs with their signatures, as the Step. 8 in Figure 4.1.
- Requester verifies signatures using the Verify algorithm.
- Requester checks if the patient's signatures are linked by calling the Link algorithm.

The requester downloads multiple EMRs for the same patient and verifies the signatures' validity. However, valid signatures can only confirm the authenticity of EMRs without proving whether they belong to the same patient. In the case of a disorganized or damaged network, these EMRs may not be from the same patient. In this way, the EMR requester needs to further check the linkability, fulfilling the fourth design goal.

User Key Update

With regard to the forward security, each user calls the KeyUpdate algorithm to update its secret key when the time period t is updated, thereby fulfilling the fifth design goal.

4.4 Our Proposed LLRS Scheme

4.4.1 Formal Definitions

We formalize the syntax of LLRS primitive, consisting of five algorithms, $\Pi_{\text{LLRS}} = (\text{LLRS} \cdot \mathsf{Setup}, \text{LLRS} \cdot \mathsf{KeyExtract}, \text{LLRS} \cdot \mathsf{Sign}, \text{LLRS} \cdot \mathsf{Verify}, \text{LLRS} \cdot \mathsf{Link})$.

- $pp \leftarrow \textbf{LLRS} \cdot \textbf{Setup}(\lambda)$: Given a security parameter λ, this algorithm outputs the public parameters pp.

- $(pk_i, sk_i) \leftarrow \textbf{LLRS} \cdot \textbf{KeyExtract}(pp, \tau_i)$: Given the public parameters pp and identity tag τ_i, this PPT algorithm outputs the i-th user's public-secret keys (pk_i, sk_i).

- $\sigma \leftarrow \textbf{LLRS} \cdot \textbf{Sign}(pp, \mu, sk_l, \mathcal{L}_{pk})$: Given the public parameters pp, a public keys set $\mathcal{L}_{pk}$, a secret key sk_l of user l, and a message μ, this PPT algorithm outputs the signature σ including the link tag tag.

- 'accept'/'reject' $\leftarrow \textbf{LLRS} \cdot \textbf{Verify}(pp, \mu, \sigma, \mathcal{L}_{pk})$: Given the public parameters pp, a public keys set $\mathcal{L}_{pk}$, a signature σ, and a message μ, this deterministic algorithm outputs 'accept'/'reject'.

- 'link'/'unlink' $\leftarrow \textbf{LLRS} \cdot \textbf{Link}(pp, \mu_1, \mu_2, \sigma_1, \sigma_2, \mathcal{L}_{pk_1}, \mathcal{L}_{pk_2})$: Given the public parameters pp, two public keys sets $\mathcal{L}_{pk_1}, \mathcal{L}_{pk_2}$, two signatures σ_1, σ_2 including two link tags tag_1, tag_2, and two messages μ_1, μ_2, this deterministic algorithm outputs 'link'/'unlink'.

Definition 13 (Verification Correctness of LLRS)**.** *For a valid signature σ, the probability of the LLRS $\cdot$ $\mathsf{Verify}(pp, \mu, \sigma, \mathcal{L}_{pk})$ algorithm outputting 'reject' is negligible, as*

$$\Pr\left[\begin{array}{c|c} \text{`reject'} \leftarrow LLRS\cdot \\ \mathsf{Verify}(pp,\mu,\sigma,\mathcal{L}_{pk}) & \begin{array}{c} pp \leftarrow LLRS\cdot\mathsf{Setup}(\lambda); \\ (pk_i,sk_i)\leftarrow LLRS\cdot\mathsf{KeyExtract}(pp,\tau_i); \\ \sigma\leftarrow LLRS\cdot\mathsf{Sign}(pp,\mu,sk_i,\mathcal{L}_{pk}); \end{array} \end{array}\right] \leq \mathsf{negl}. \tag{4.1}$$

Definition 14 (Linking Correctness of LLRS)**.** *If two valid signatures σ_1, σ_2 are generated by the same user, the probability of the $LLRS\cdot\mathsf{Link}(pp,\mu_1,\mu_2,\sigma_1,\sigma_2,\mathcal{L}_{pk_1},\mathcal{L}_{pk_2})$ algorithm outputting 'unlink' is negligible, as*

$$\Pr\left[\begin{array}{c|c} \text{`unlink'} \leftarrow LLRS\cdot\mathsf{Link}(pp, & pp \leftarrow LLRS\cdot\mathsf{Setup}(\lambda); \\ \mu_1,\mu_2,\sigma_1,\sigma_2,\mathcal{L}_{pk_1},\mathcal{L}_{pk_2}) & (pk_i,sk_i)\leftarrow LLRS\cdot\mathsf{KeyExtract}(pp,\tau_i); \\ & \sigma_1\leftarrow LLRS\cdot\mathsf{Sign}(pp,\mu_1,sk_i,\mathcal{L}_{pk_1}); \\ & \sigma_2\leftarrow LLRS\cdot\mathsf{Sign}(pp,\mu_2,sk_i,\mathcal{L}_{pk_2}); \end{array}\right] \leq \mathsf{negl}. \tag{4.2}$$

4.4.2 Security Models

The security models of the LLRS primitive encompass anonymity, unforgeability, and linkability. The formal definitions interacted with a challenge $\mathcal{C}$ and an adversary $\mathcal{A}$ are provided below.

Anonymity of LLRS

This game is designed as:

- **Setup.** $\mathcal{C}$ executes the LLRS·Setup algorithm to generate the public parameters pp and send it to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:

  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key pk_i of the user $i \in [1, N]$. Then, $\mathcal{C}$ invokes the LLRS $\cdot$ KeyExtract algorithm to get public-secret keys (pk_i, sk_i) and returns the public key pk_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key sk_i of the user $i \in [1, N]$. $\mathcal{C}$ calculates its secret key sk_i and then sends it to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ_i for the user $i \in [1, N]$. Then, $\mathcal{C}$ returns σ_i to $\mathcal{A}$ by invoking the LLRS $\cdot$ Sign algorithm.

- **Challenge.** $\mathcal{A}$ chooses a message μ^*, a public keys set $\mathcal{L}^*_{pk}$, public keys $pk^*_0, pk^*_1 \in \mathcal{L}^*_{pk}$, and send them to $\mathcal{C}$. Then, $\mathcal{C}$ randomly selects a bit $b \in \{0, 1\}$ to generate the signature σ^* and sends it to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b = b'$, $\mathcal{A}$ wins.

The advantage for $\mathcal{A}$ to attack the anonymity game is defined as $Adv_{\mathcal{A}}^{\text{Anonymity}_{\text{LLRS}}}(\lambda) := |\Pr[b = b'] - \frac{1}{2}|$.

Definition 15 (**Anonymity of LLRS**). *For any* PPT *adversary* $\mathcal{A}$*, if the advantage* $Adv_{\mathcal{A}}^{\text{Anonymity}_{\text{LLRS}}}(\lambda)$ *is negligible, we say that our* LLRS *scheme satisfies the* anonymity.

Unforgeability of LLRS

This game is designed as:

- **Setup.** $\mathcal{A}$ randomly picks a target index $i^* \in [1, N]$. $\mathcal{C}$ executes the LLRS$\cdot$Setup algorithm to generate the public parameters pp and send it to $\mathcal{A}$.

- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:

  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key pk_i of the user $i \in [1, N]$. Then, $\mathcal{C}$ invokes the LLRS $\cdot$ KeyExtract algorithm to get public-secret keys (pk_i, sk_i) and returns the public key pk_i to $\mathcal{A}$.

  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key sk_i for the user $i \in [1, N]$. $\mathcal{C}$ sends sk_i to $\mathcal{A}$ if $i \neq i^*$. Otherwise, this query is aborted and $\mathcal{C}$ returns $\perp$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ_i for the user $i \in [1, N]$. If $pk_i \notin \mathcal{L}_{pk}$, $\mathcal{C}$ returns $\perp$. Otherwise, it returns σ_i to $\mathcal{A}$ by invoking the LLRS $\cdot$ Sign algorithm ($i \neq i^*$) or a simulator $\mathcal{S}$ ($i = i^*$).

- **Forge.** $\mathcal{A}$ outputs a public keys set ${\mathcal{L}_{pk}}^*$ including pk_{i^*}, and the forgery signature ${\sigma_i}^*$. If the following requirements are fulfilled, we say that $\mathcal{A}$ wins.
  - 'accept' $\leftarrow$ LLRS $\cdot$ Verify$(pp, \mu^*, {\mathcal{L}_{pk}}^*, {\sigma_i}^*)$.
  - $\mathcal{A}$ has not queried $({\mu_i}^*, {\mathcal{L}_{pk}}^*)$ in $\mathcal{O}_{SO}$.
  - $\mathcal{A}$ has not queried any secret key corresponding to the public keys in ${\mathcal{L}_{pk}}^*$.

The advantage for $\mathcal{A}$ to attack unforgeability game is defined as $Adv_{\mathcal{A}}^{\text{Unforgeability}_{\text{LLRS}}}(\lambda) :=$ [$\mathcal{A}$ wins the game].

Definition 16 (**Unforgeability of LLRS**). *For any* PPT *adversary* $\mathcal{A}$*, if the advantage* $Adv_{\mathcal{A}}^{\text{Unforgeability}_{\text{LLRS}}}(\lambda)$ *is negligible, we say that our* LLRS *scheme satisfies the* unforgeability.

Linkability of LLRS

This game is designed as:

- **Setup.** $\mathcal{C}$ executes the LLRS$\cdot$Setup algorithm to generate the public parameters pp and send it to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:
  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key pk_i of the user $i \in [1, N]$. Then, $\mathcal{C}$ invokes the LLRS $\cdot$ KeyExtract algorithm to get public-secret keys (pk_i, sk_i) and returns the public key pk_i to $\mathcal{A}$.

  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key sk_i of the user $i \in [1, N]$. $\mathcal{C}$ calculates its secret key sk_i and then sends it to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ_i for the user $i \in [1, N]$. Then, $\mathcal{C}$ returns σ_i to $\mathcal{A}$ by invoking the LLRS $\cdot$ Sign algorithm.

- **Unlink.** $\mathcal{A}$ outputs two tuples $(\mathcal{L}_{pk_1}^*, \mu_1^*, \sigma_1^*)$ and $(\mathcal{L}_{pk_2}^*, \mu_2^*, \sigma_2^*)$, where $\mathcal{L}_{pk_i}^*$ is the public keys set including user i^*'s public key, μ_i^* is the signed message, σ_i^* is the signature including link tag tag_i^*. If the following requirements are fulfilled, we say that $\mathcal{A}$ wins.
  - 'accept' $\leftarrow$ LLRS $\cdot$ Verify$(pp, \mu_1^*, \mathcal{L}_{pk_1}^*, {\sigma_1}^*)$.
  - 'accept" $\leftarrow$ LLRS $\cdot$ Verify$(pp, \mu_2^*, \mathcal{L}_{pk_2}^*, {\sigma_2}^*)$.
  - 'unlink' $\leftarrow$ LLRS $\cdot$ Link$(pp, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*, \mathcal{L}_{pk_1}^*, \mathcal{L}_{pk_2}^*)$
  - $\mathcal{A}$ has not queried $({\mu_1}^*, \mathcal{L}_{pk_1}^*)$ and $({\mu_2}^*, \mathcal{L}_{pk_2}^*)$ in $\mathcal{O}_{SO}$.
  - $\mathcal{A}$ owns at most one secret key corresponding to the public keys in ${\mathcal{L}_{pk}}^*$.

The advantage for $\mathcal{A}$ to attack linkability game is defined as $Adv_{\mathcal{A}}^{\text{Linkability}_{\text{LLRS}}}(\lambda) :=$ [$\mathcal{A}$ wins the game].

Definition 17 (**Linkability of LLRS**). *For any* PPT *adversary* $\mathcal{A}$*, if the advantage* $Adv_{\mathcal{A}}^{\text{Linkability}_{\text{LLRS}}}(\lambda)$ *is negligible, we say that our* LLRS *scheme satisfies the* linkability.

4.4.3 The Concrete Construction

- **LLRS** $\cdot$ **Setup**(λ): Given a security parameter λ, this algorithm performs as follows:
  1. Sets a prime $q \geq 2$, several integers n, $m \geq 2n\lceil \log q \rceil$, the length of a signed message d, and a vector $\mathbf{p} \in \mathbb{Z}_q^n$.
  2. Randomly selects $d + 1$ vectors $\mathbf{c}_0, \cdots, \mathbf{c}_d \in \mathbb{Z}_q^n$.

3. Sets integer k as the length of a user's identity tag. Note that $\frac{n}{k}$ has to be an integer.
4. Generates the public matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$.
5. Returns the public parameters $pp = (n, m, q, d, k, \mathbf{p}, \mathbf{c}_0, \cdots, \mathbf{c}_d, \mathbf{A}_0)$.

- **LLRS** $\cdot$ **KeyExtract**(pp, τ_i): Given the public parameters pp, the identity tag τ_i of user i, this algorithm performs as follows:

1. Parses the tag τ_i as $\tau_i[1], \cdots, \tau_i[k]$, where $\tau_i[j]$ denotes the j-th bit of the tag τ_i.
2. Randomly selects $2k$ matrices $\mathbf{\Gamma}_1^{(0)}, \mathbf{\Gamma}_1^{(1)}, \cdots, \mathbf{\Gamma}_k^{(0)}, \mathbf{\Gamma}_k^{(1)} \in \mathbb{Z}_q^{n \times \frac{n}{k}}$.
3. Computes a secret key $\mathbf{T}_i = (\mathbf{\Gamma}_1^{\tau_1} \| \cdots \| \mathbf{\Gamma}_k^{\tau_k}) \in \mathbb{Z}_q^{n \times n}$.
4. Computes a public key $\mathbf{B}_i = \mathbf{A}_0^\top \cdot \mathbf{T}_i^{-1} \in \mathbb{Z}_q^{m \times n}$.
5. Checks the relation between the public key and secret key as $\mathbf{B}_i \cdot \mathbf{T}_i = \mathbf{A}_0^\top$.
6. Returns public-secret keys $(\mathbf{B}_i, \mathbf{T}_i)$ for a user i.

- **LLRS**$\cdot$**Sign**$(pp, \mu, sk_l, \mathcal{L}_{pk})$: Given the public parameters pp, the signed message μ[1] $\in \{0\} \times \{0,1\}^d$, the secret key $sk_l = \mathbf{T}_l$ of user l and a ring with public keys set $\mathcal{L}_{pk} = \{pk_1, \cdots, pk_N\}$, this algorithm performs as:

1. Parses message μ as $\mu[0], \mu[1], \cdots, \mu[d]$, where $\mu[j]$ is the j-th bit of the message μ.
2. Computes $\mathbf{c}_\mu = \sum_{j=0}^{d} (-1)^{\mu[j]} \mathbf{c}_j = (-1)^{\mu[0]} \mathbf{c}_0 + (-1)^{\mu[1]} \mathbf{c}_1 + \cdots + (-1)^{\mu[d]} \mathbf{c}_d \in \mathbb{Z}_q^n$.
3. For all $i \in [N]$, randomly selects $\mathbf{k}_i \in \mathbb{Z}_q^n$.
4. Computes $\mathbf{r} = \sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{k}_i \in \mathbb{Z}_q^m$.
5. For all $i \in [N]$, $i = l$, computes $\mathbf{e}_l = \mathbf{k}_l + \mathbf{T}_l \cdot \mathbf{c}_\mu \in \mathbb{Z}_q^n$.

[1] μ is a $(d+1)$-bit message where the first bit is set to 0 and the remaining d bits can be 0 or 1.

6. For all $i \in [N]$, $i \neq l$, computes $\mathbf{e}_i = \mathbf{k}_i + 0 \cdot \mathbf{c}_\mu \in \mathbb{Z}_q^n$.
7. Computes $\boldsymbol{\upsilon} = \mathbf{p} \cdot \mathbf{T}_i \in \mathbb{Z}_q^n$.
8. Returns a signature $\sigma = \{\mathbf{r}, \{\mathbf{e}_i\}_{i\in[N]}\}$ including a link tag $tag = \boldsymbol{\upsilon}$.

- **LLRS** $\cdot$ **Verify**$(pp, \mu, \sigma, \mathcal{L}_{pk})$: Given the public parameters pp, the signed message μ, the signature σ of message μ, and a ring with public keys $\mathcal{L}_{pk} = \{pk_1, \cdots, pk_N\}$, this algorithm performs as:

1. Parses message μ as $\mu[0], \mu[1], \cdots, \mu[d]$.
2. Computes $\mathbf{c}_\mu = \sum_{i=1}^{d} (-1)^{\mu[j]} \mathbf{c}_j$.
3. Checks $\sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{e}_i = \mathbf{r} + \mathbf{A}_0^\top \cdot \mathbf{c}_\mu$.
4. Returns 'accept' or 'reject'.

- **LLRS** $\cdot$ **Link**$(pp, \mu_1, \mu_2, \sigma_1, \sigma_2, \mathcal{L}_{pk_1}, \mathcal{L}_{pk_2})$: Given the public parameters pp, two messages μ_1, μ_2, two signatures σ_1, σ_2 including two link tags tag_1, tag_2, and two ring with public keys $\mathcal{L}_{pk_1}, \mathcal{L}_{pk_2}$, this algorithm performs as follows:

1. Checks if 'accept' $\leftarrow$ LLRS $\cdot$ Verify$(pp, \mu_1, \sigma_1, \mathcal{L}_{pk_1})$.
2. Checks if 'accept' $\leftarrow$ LLRS $\cdot$ Verify$(pp, \mu_2, \sigma_2, \mathcal{L}_{pk_2})$.
3. Checks if two link tags $\boldsymbol{\upsilon}_1 = \boldsymbol{\upsilon}_2$.
4. Returns 'link' or 'unlink'.

4.4.4 Correctness

In the verification, we calculate that $\sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{e}_i = \mathbf{r} + \mathbf{A}_0^\top \cdot \mathbf{c}_\mu$, which is correct as:

$$\sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{e}_i = \sum_{i=1}^{N} \mathbf{B}_i \cdot (\mathbf{k}_i + \mathbf{T}_i \cdot \mathbf{c}_\mu) \tag{4.3}$$

$$= \sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{k}_i + \sum_{i=1}^{N} \mathbf{B}_i \cdot (\mathbf{T}_i \cdot \mathbf{c}_\mu) \tag{4.4}$$

$$= \mathbf{r} + \mathbf{B}_1 \cdot \mathbf{T}_1 \cdot \mathbf{c}_\mu + \cdots + \mathbf{B}_N \cdot \mathbf{T}_N \cdot \mathbf{c}_\mu \tag{4.5}$$

$$= \mathbf{r} + 0 + \cdots + \mathbf{B}_l \cdot \mathbf{T}_l \cdot \mathbf{c}_\mu + \cdots + 0 \tag{4.6}$$

$$= \mathbf{r} + \mathbf{B}_l \cdot \mathbf{T}_l \cdot \mathbf{c}_\mu \tag{4.7}$$

$$= \mathbf{r} + \mathbf{A}_0^\top \cdot \mathbf{c}_\mu \tag{4.8}$$

For linking correctness, the user i signs two messages μ_1 and μ_2 using the same secret key $\mathbf{T}_i$ to generate two signatures σ_1 and σ_2 containing the link tags $\boldsymbol{\upsilon}_1$ and $\boldsymbol{\upsilon}_2$, respectively. The link tag of the signed message μ_1 is $\boldsymbol{\upsilon}_1 = \mathbf{p} \cdot \mathbf{T}_i$, and the link tag of the signed message μ_2 is $\boldsymbol{\upsilon}_2 = \mathbf{p} \cdot \mathbf{T}_i$. Since $\boldsymbol{\upsilon}_1$, $\boldsymbol{\upsilon}_2$ are generated with the same randomly chosen matrix $\mathbf{p}$, if the user signs the messages μ_1, μ_2 with the same secret key $\mathbf{T}_i$, then it must be the case that $\boldsymbol{\upsilon}_1 = \boldsymbol{\upsilon}_2$.

4.4.5 Security Analysis

Anonymity

Lemma 5. *Let $(pk_0, pk_1, \mathcal{L}_{pk}, \mu)$ be a tuple such that μ is a message to be signed with the ring $\mathcal{L}_{pk}$, pk_0 and pk_1 are both includes in the $\mathcal{L}_{pk}$. If $SIS_{q,n,\delta}$ is hard, $\sigma_0 \leftarrow LLRS \cdot \mathsf{Sign}(pp, \mu, sk_0, \mathcal{L}_{pk})$ and $\sigma_1 \leftarrow LLRS \cdot \mathsf{Sign}(pp, \mu, sk_1, \mathcal{L}_{pk})$ are computationally indistinguishable.*

Analysis. In the LLRS $\cdot$ Sign algorithm, the ring signatures σ_0 and σ_1 have the same distribution, which implies that two signatures σ_0 and σ_1 are computationally indistinguishable.

Theorem 5 (Anonymity of LLRS.)**.** *The probability of an adversary $\mathcal{A}$ to win the*

anonymity game with the polynomial time is negligible. Therefore, the proposed LLRS *scheme satisfies the anonymity in the standard model.*

Proof. We assume that there exists an adaptive adversary $\mathcal{A}$ who tries to break the anonymity of the LLRS scheme and a challenger $\mathcal{C}$ is able to respond to queries of $\mathcal{A}$.

- **Setup.** $\mathcal{C}$ initially executes LLRS $\cdot$ Setup algorithm to generate the public parameters $pp = (n, m, q, d, k, \mathbf{p} \in \mathbb{Z}_q^n, \{\mathbf{c}_0, \cdots, \mathbf{c}_d\} \in \mathbb{Z}_q^n, \mathbf{A}_0 \in \mathbb{Z}_q^{n \times m})$.
- **Queries.** $\mathcal{A}$ picks an index $i \in [N]$ and performs a polynomial bounded number N queries adaptively:
  - Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, N]$, $\mathcal{A}$ queries the public key pk_i of a user i. Then, $\mathcal{C}$ invokes LLRS $\cdot$ KeyExtract algorithm to get the key pair (pk_i, sk_i). Finally, $\mathcal{C}$ returns public key pk_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key sk_i of a user i. Then, $\mathcal{C}$ responds the answer to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ inputs a public keys set $\mathcal{L}_{pk}$, an i-th user's public key $pk_i \in \mathcal{L}_{pk}$ and a message μ. $\mathcal{C}$ then invokes the LLRS $\cdot$ Sign algorithm and returns a valid signature σ_i to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ selects a message $\mu^* \in \{0\} \times \{0,1\}^d$, a public keys set $\mathcal{L}_{pk}^*$, two public keys $pk_0^*, pk_1^* \in \mathcal{L}_{pk}^*$ to $\mathcal{C}$. Then, $\mathcal{A}$ sends $(pk_0^*, pk_1^*, \mu^*, \mathcal{L}_{pk}^*)$ to $\mathcal{C}$ to request a signature. Then, $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ and queries $\mathcal{O}_{SO}$. Finally, $\mathcal{C}$ outputs the signature σ^* to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b' = b$, we say that $\mathcal{A}$ wins the game.

Analysis. The signatures have the same distribution, which are computationally indistinguishable. If $\mathcal{A}$ successfully distinguishes signatures with non-negligible probability, it contradicts Lemma 5. Hence, we claim that the advantage of $\mathcal{A}$ to win the game is negligible and our LLRS scheme is anonymous. □

Unforgeability

Theorem 6 (Unforgeability of LLRS.)**.** *The proposed* LLRS *scheme satisfies the unforgeability in the standard model under the $SIS_{q,2n,\delta}$ hardness assumption.*

Proof. We assume that there exists an adaptive adversary $\mathcal{A}$ who tries to break the unforgeability of the LLRS scheme and a challenger $\mathcal{C}$ is able to solve $SIS_{q,2n,\delta}$ problem.

- **Setup.** $\mathcal{C}$ initially executes the LLRS $\cdot$ Setup algorithm to obtain and send the public parameters $pp = (n, m, q, d, k, \mathbf{p}, \mathbf{c}_0, \cdots, \mathbf{c}_d, \mathbf{A}_0)$ to $\mathcal{A}$. Suppose $\mathcal{A}$ is also given a uniformly sampled $\mathbf{U} \in \mathbb{Z}_q^{m\times n}$, and $\hat{\mathbf{U}} = [\mathbf{A}_0^\top \| \mathbf{U}] \in \mathbb{Z}_q^{m\times 2n}$ as the SIS matrix. $\mathcal{A}$ picks an index $i^* \in [N]$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:
  - Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, N]$, $\mathcal{A}$ queries the public key pk_i of a user i. Then, $\mathcal{C}$ invokes the LLRS $\cdot$ KeyExtract algorithm to get the key pair (pk_i, sk_i). For $i = i^*$, the pk_{i^*} can be computed as $pk_{i^*} = \mathbf{A}_0^\top \cdot \mathbf{X} + \mathbf{U}$ for $\mathbf{X} \in \mathbb{Z}_q^{n\times n}$. Finally, $\mathcal{C}$ keeps the secret key as confidential and returns public key to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key sk_i of a user i. If $i \neq i^*$, $\mathcal{C}$ responds the answer to $\mathcal{A}$; otherwise, $\mathcal{C}$ aborts the query and returns failure.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ inputs a public keys set $\mathcal{L}_{pk}$, an i-th user's public key pk_i and a message μ to query a vaild signature. If $pk_i \notin \mathcal{L}_{pk}$, $\mathcal{C}$ returns $\perp$. If $i \neq i^*$, $\mathcal{C}$ returns the signature σ_i to $\mathcal{A}$ by invoking the LLRS $\cdot$ Sign algorithm. Otherwise, if $i = i^*$, the query cannot be aborted. In this case, the secret key is missing, thereby $\mathcal{C}$ has to use a simulator $\mathcal{S}$ to obtain signature σ_i as:

1. Picks a random matrix $\mathbf{c}_\mu \in \mathbb{Z}_q^n$.
2. Picks random vectors $\mathbf{e}_1, \cdots, \mathbf{e}_N \in \mathbb{Z}_q^n$.
3. Computes $\mathbf{r} = \sum_{i=1}^{N} pk_i \cdot \mathbf{e}_i - \mathbf{A}_0^\top \cdot \mathbf{c}_\mu \in \mathbb{Z}_q^m$.

Finally, $\mathcal{C}$ returns the signature σ_i to $\mathcal{A}$.

- **Forge.** $\mathcal{A}$ outputs a public keys set $\mathcal{L}_{pk}^*$ including a user i^*'s public key, a message μ^* and a forgery signature σ_i^*.

Analysis. By the ability of the extractor algorithm $\mathcal{E}$, we have two transcripts $(\mathbf{e}_1^*, \mathbf{r}_1^*, \mathbf{c}_{\mu 1}^*)$ and $(\mathbf{e}_2^*, \mathbf{r}_2^*, \mathbf{c}_{\mu 2}^*)$, where $\mathbf{r}_1^* = \mathbf{r}_2^*$ and $\mathbf{c}_{\mu 1}^* \neq \mathbf{c}_{\mu 2}^*$. Hence, we have

$$\sum_{i=1}^{N} \mathcal{L}_{pk_i} \cdot \mathbf{e}_{1_i} - \mathbf{A}_0^\top \cdot \mathbf{c}_{\mu_1} = \sum_{i=1}^{N} \mathcal{L}_{pk_2} \cdot \mathbf{e}_{2_i} - \mathbf{A}_0^\top \cdot \mathbf{c}_{\mu_2}. \tag{4.9}$$

Transferring the above equation, we obtain:

$$pk_{i^*} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}) = \mathbf{A}_0^\top \cdot (\mathbf{c}_{\mu_1} - \mathbf{c}_{\mu_2}) - \sum_{i=1, i\neq i^*}^{N} pk_i \cdot (\mathbf{e}_{1i} - \mathbf{e}_{2i}). \tag{4.10}$$

Then, we have:

$$pk_{i^*} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}) = \mathbf{A}_0^\top \cdot (\mathbf{c}_{\mu_1} - \mathbf{c}_{\mu_2}) - \sum_{i=1, i\neq i^*}^{N} pk_i \cdot (\mathbf{e}_{1i} - \mathbf{e}_{2i}) \tag{4.11}$$

$$= \mathbf{A}_0^\top \cdot \left[(\mathbf{c}_{\mu_1} - \mathbf{c}_{\mu_2}) - \sum_{i=1, i\neq i^*}^{N} \mathbf{X}_i \cdot (\mathbf{e}_{1i} - \mathbf{e}_{2i}) \right] \tag{4.12}$$

$$= \hat{\mathbf{U}} \cdot \left[\begin{pmatrix} \mathbf{c}_{\mu_1} - \mathbf{c}_{\mu_2} \\ 0 \end{pmatrix} - \sum_{i=1, i\neq i^*}^{N} \begin{pmatrix} \mathbf{X}_i \\ 0 \end{pmatrix} (\mathbf{e}_{1i} - \mathbf{e}_{2i}) \right] \tag{4.13}$$

Furthermore, by $pk_{i^*} = \mathbf{A}_0^\top \cdot \mathbf{X} + \mathbf{U}$, we have:

$$\sum_{i=1}^{N} \mathcal{L}_{pk_i} \cdot (\mathbf{e}_{1i} - \mathbf{e}_{2i}) = \mathbf{A}_0^{\top} \cdot \mathbf{X} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}) + \mathbf{U} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}) \tag{4.14}$$

$$= \hat{\mathbf{U}} \cdot \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}) \tag{4.15}$$

Therefore, we can say:

$$\hat{\mathbf{U}} \cdot \left[\begin{pmatrix} \mathbf{c}_{\mu_1} - \mathbf{c}_{\mu_2} \\ 0 \end{pmatrix} - \sum_{i=1, i \neq i^*}^{N} \begin{pmatrix} \mathbf{X}_i \\ 0 \end{pmatrix} (\mathbf{e}_{1i} - \mathbf{e}_{2i}) \right] = \hat{\mathbf{U}} \cdot \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}). \tag{4.16}$$

For $\hat{\mathbf{U}} \cdot \mathbf{s} = 0$, $\mathbf{s}$ cannot be zero, and we have:

$$\mathbf{s} = \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \cdot (\mathbf{e}_{1i^*} - \mathbf{e}_{2i^*}) - \left[\begin{pmatrix} \mathbf{c}_{\mu_1} - \mathbf{c}_{\mu_2} \\ 0 \end{pmatrix} - \sum_{i=1, i \neq i^*}^{N} \begin{pmatrix} \mathbf{X}_i \\ 0 \end{pmatrix} (\mathbf{e}_{1i} - \mathbf{e}_{2i}) \right]. \tag{4.17}$$

The vector $\mathbf{s}$ gives a solution to SIS problem. Thus, we prove that our LLRS scheme satisfies the unforgeability. □

Linkability

Theorem 7 (Linkability of LLRS.)**.** *The proposed* LLRS *scheme satisfies linkability in the standard model if the* LLRS *scheme satisfies the unforgeability.*

Proof. We assume that there exists an adaptive adversary $\mathcal{A}$ who tries to break the linkability of the LLRS scheme and a challenger $\mathcal{C}$ is able to respond to queries of $\mathcal{A}$.

- **Setup.** $\mathcal{C}$ initially executes LLRS $\cdot$ Setup algorithm to generate the public parameters $pp = (n, m, q, d, k, \mathbf{p} \in \mathbb{Z}_q^n, \{\mathbf{c}_0, \cdots, \mathbf{c}_d\} \in \mathbb{Z}_q^n, \mathbf{A}_0 \in \mathbb{Z}_q^{n \times m})$.

- **Queries.** $\mathcal{A}$ picks an index $i \in [N]$ and performs a polynomial bounded number N queries adaptively:

  - Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, N]$, $\mathcal{A}$ queries the public key pk_i of a user i. Then, $\mathcal{C}$ invokes the LLRS $\cdot$ KeyExtract algorithm to get the key pair (pk_i, sk_i). Finally, $\mathcal{C}$ returns public key pk_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key sk_i of a user i. Then, $\mathcal{C}$ responds the answer to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ inputs a public keys set $\mathcal{L}_{pk}$, an i-th user's public key $pk_i \in \mathcal{L}_{pk}$ and a message μ. $\mathcal{C}$ then invokes the LLRS $\cdot$ Sign algorithm and returns a valid signature σ_i to $\mathcal{A}$.

- **Unlink.** $\mathcal{A}$ outputs two tuples $(\mathcal{L}_{pk_1}^*, \mu_1^*, \sigma_1^*)$ and $(\mathcal{L}_{pk_2}^*, \mu_2^*, \sigma_2^*)$, where $\mathcal{L}_{pk_i}^*$ is a public keys set including a user i^*'s public key, μ_i^* is a signed message, σ_i^* is a signature including a link tag $\boldsymbol{\upsilon}_i^*$.

Analysis. We assume that $\mathcal{A}$ generates two ring signatures σ_1^*, σ_2^* with non-negligible probability while holding only one secret key, and both LLRS $\cdot$ Verify$(pp, \mathcal{L}_{pk_1}^*, \mu_1^*, \sigma_1^*)$ and LLRS $\cdot$ Verify$(pp, \mathcal{L}_{pk_2}^*, \mu_2^*, \sigma_2^*)$ algorithm outputs 'accept'. Since our LLRS scheme satisfies the unforgeability, these two signatures can pass the verification algorithm only if the $\mathcal{A}$ honestly generates the signatures σ_1, σ_2.

When $\mathcal{A}$ generates the two signatures, we have two link tags $\boldsymbol{\upsilon}_1^* = \mathbf{p}^* \cdot sk_1^*$, $\boldsymbol{\upsilon}_2^* = \mathbf{p}^* \cdot sk_2^*$ respectively. Since $\mathcal{A}$ only has one secret key, then $sk_1^* = sk_2^*$. Moreover, as the public matrix $\mathbf{p}^*$ is the same, we get $\boldsymbol{\upsilon}_1^* = \boldsymbol{\upsilon}_2^*$. It shows that the two tuples of $\mathcal{A}$ verified by the LLRS $\cdot$ Link$(pp, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*, \mathcal{L}_{pk_1}^*, \mathcal{L}_{pk_2}^*)$ algorithm will return 'link', which contradicts the assumption of the linkability game. Therefore, the advantage of $\mathcal{A}$ is negligible and our LLRS scheme is linkable.

□

4.5 Our Proposed FS-LLRS Scheme

4.5.1 Formal Definitions

We formalize the syntax of enhanced FS-LLRS primitive, consisting of six algorithms, $\Pi_{\text{FS-LLRS}} = (\text{FS-LLRS} \cdot \mathsf{Setup}, \text{FS-LLRS} \cdot \mathsf{KeyExtract}, \text{FS-LLRS} \cdot \mathsf{KeyUpdate}, \text{FS-LLRS} \cdot \mathsf{Sign}, \text{FS-LLRS} \cdot \mathsf{Verify}, \text{FS-LLRS} \cdot \mathsf{Link})$.

- $pp \leftarrow$ **FS-LLRS** $\cdot$ **Setup**(λ): Given a security parameter λ, this algorithm returns the public parameters pp.
- $(pk_i, sk_i) \leftarrow$ **FS-LLRS** $\cdot$ **KeyExtract**(pp, τ_i): Given the public parameters pp, and user's identity tag τ_i, this PPT algorithm returns the i-th user's public-secret keys (pk_i, sk_i).
- $sk_{i,t+1} \leftarrow$ **FS-LLRS** $\cdot$ **KeyUpdate**$(pp, t, sk_{i,t}, \tau_i)$: Given the public parameters pp, the current time period t, an identity tag τ_i, a secret key $sk_{i,t}$, this PPT algorithm updates the secret key from $sk_{i,t}$ to $sk_{i,t+1}$ and then discards the $sk_{i,t}$.
- $\sigma \leftarrow$ **FS-LLRS** $\cdot$ **Sign**$(pp, \mu, sk_{l,t}, \mathcal{L}_\tau, t)$: Given the public parameters pp, the time period t, a users' identity tags set $\mathcal{L}_\tau$, a secret key sk_l of the user l, and a message μ, this PPT algorithm returns the signature σ.
- 'accept'/'reject' $\leftarrow$ **FS-LLRS** $\cdot$ **Verify**$(pp, \mu, \sigma, \mathcal{L}_\tau, t)$: Given the public parameters pp, the time period t, a users' identity tags set $\mathcal{L}_\tau$, a signature σ, a message μ, this deterministic algorithm returns 'accept'/'reject'.
- 'link'/'unlink' $\leftarrow$ **FS-LLRS** $\cdot$ **Link**$(pp, \mu_1, \mu_2, \sigma_1, \sigma_2, \mathcal{L}_{\tau 1}, \mathcal{L}_{\tau 2}, t_1, t_2)$: Given the public parameters pp, two users' identity tags sets $\mathcal{L}_{\tau 1}, \mathcal{L}_{\tau 2}$, two signatures σ_1, σ_2 generated on time period t_1, t_2 including two link tags tag_1, tag_2, and two messages μ_1, μ_2, this deterministic algorithm returns 'link'/'unlink'.

Definition 18 (Verification Correctness of FS-LLRS)**.** *For a valid signature σ, the probability of the FS-LLRS $\cdot$ $\mathsf{Verify}(pp, \mu, \sigma, \mathcal{L}_\tau, t)$ algorithm outputting 'reject' is negligible, as*

$$\Pr\left[\begin{array}{c|c} \text{'reject'} \leftarrow & pp \leftarrow \text{FS-LLRS} \cdot \mathsf{Setup}(\lambda); \\ \text{FS-LLRS} \cdot \mathsf{Verify} & (pk_i, sk_i) \leftarrow \text{FS-LLRS} \cdot \mathsf{KeyExtract}(pp, \tau_i); \\ (pp, \mu, \sigma, \mathcal{L}_\tau, t) & \sigma \leftarrow \text{FS-LLRS} \cdot \mathsf{Sign}(pp, \mu, sk_{i,t}, \mathcal{L}_\tau, t); \end{array}\right] \leq \mathsf{negl}. \quad (4.18)$$

Definition 19 (Linking Correctness of FS-LLRS)**.** *If two valid signatures σ_1, σ_2 are generated by the same user, the probability of the FS-LLRS$\cdot\mathsf{Link}(pp, \mu_1, \mu_2, \mathcal{L}_{\tau 1}, \mathcal{L}_{\tau 2}, \sigma_1, \sigma_2, t_1, t_2)$ algorithm outputting 'unlink' is negligible, as*

$$\Pr\left[\begin{array}{c|c} \text{'unlink'} \leftarrow & pp \leftarrow \text{FS-LLRS} \cdot \mathsf{Setup}(\lambda); \\ \text{FS-LLRS} \cdot \mathsf{Link} & (pk_i, sk_i) \leftarrow \text{FS-LLRS} \cdot \mathsf{KeyExtract}(pp, \tau_i); \\ (pp, \mu_1, \mu_2, \mathcal{L}_{\tau 1}, & \sigma_1 \leftarrow \text{FS-LLRS} \cdot \mathsf{Sign}(pp, \mu_1, sk_{i,t}, \mathcal{L}_{\tau 1}, t_1); \\ \mathcal{L}_{\tau 2}, \sigma_1, \sigma_2, t_1, t_2) & \sigma_2 \leftarrow \text{FS-LLRS} \cdot \mathsf{Sign}(pp, \mu_2, sk_{i,t}, \mathcal{L}_{\tau 2}, t_2); \end{array}\right] \leq \mathsf{negl}. \quad (4.19)$$

4.5.2 Security Models

The security models of the FS-LLRS primitive are threefold: anonymity, unforgeability with forward security, and linkability. The formal definitions interacted with a challenge $\mathcal{C}$ and an adversary $\mathcal{A}$ are provided below.

Anonymity of FS-LLRS

This game is designed as:

- **Setup.** $\mathcal{C}$ executes the FS-LLRS $\cdot$ Setup algorithm to generate the public parameters pp and send it to $\mathcal{A}$.

- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:
  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key of a user $i \in [1, N]$. Then, $\mathcal{C}$ invokes the FS-LLRS $\cdot$ KeyExtract algorithm to get public-secret keys (pk_i, sk_i) and returns the public key pk_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key $sk_{i,t}$ of a user $i \in [1, N]$ on time t. $\mathcal{C}$ calculates its secret key $sk_{i,t}$ and then sends it to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature $\sigma_{i,t}$ for the user $i \in [1, N]$ on time period t. Then, $\mathcal{C}$ returns $\sigma_{i,t}$ to $\mathcal{A}$ by invoking the FS-LLRS $\cdot$ Sign algorithm.
- **Challenge.** $\mathcal{A}$ chooses a time period t^*, a message μ^*, a public keys set $\mathcal{L}_\tau^*$, two users' identity tags $\tau_0^*, \tau_1^* \in \mathcal{L}_\tau^*$, and sends them to $\mathcal{C}$. Then, $\mathcal{C}$ randomly selects a bit $b \in \{0, 1\}$ to generate the signature $\sigma_{i,t}^*$ and sends it to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b = b'$, $\mathcal{A}$ wins.

The advantage for $\mathcal{A}$ to attack the anonymity game is defined as $Adv_{\mathcal{A}}^{\text{Anonymity}_{\text{FS-LLRS}}}(\lambda) := |\Pr[b = b'] - \frac{1}{2}|$.

Definition 20 (**Anonymity of FS-LLRS**). *For any* PPT *adversary* $\mathcal{A}$*, if the advantage* $Adv_{\mathcal{A}}^{\text{Anonymity}_{\text{FS-LLRS}}}(\lambda)$ *is negligible, we say that our* FS-LLRS *scheme satisfies the* anonymity.

Unforgeability with Forward Security of FS-LLRS

This game is designed as:

- **Setup.** $\mathcal{A}$ randomly picks a target index $i^* \in [1, N]$. $\mathcal{C}$ executes the FS-LLRS $\cdot$ Setup algorithm to generate the public parameters pp and send it to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:

- Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key of a user $i \in [1, N]$. Then, $\mathcal{C}$ invokes the FS-LLRS $\cdot$ KeyExtract algorithm to get public-secret keys (pk_i, sk_i) and returns the public key pk_i to $\mathcal{A}$.
- Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key $sk_{i,t}$ for a user $i \in [1, N]$ on time t. $\mathcal{C}$ sends $sk_{i,t}$ to $\mathcal{A}$ if $i \neq i^*$. Otherwise, this query is aborted, $\mathcal{C}$ returns $\perp$.
- Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature $\sigma_{i,t}$ for the user $i \in [1, N]$ on time t. If $\tau_i \notin \mathcal{L}_\tau$, $\mathcal{C}$ returns $\perp$. Otherwise, it returns $\sigma_{i,t}$ to $\mathcal{A}$ by invoking the FS-LLRS $\cdot$ Sign algorithm ($i \neq i^*$) or a simulator $\mathcal{S}$ ($i = i^*$).

- **Forge.** $\mathcal{A}$ outputs a users' identity tags set ${\mathcal{L}_\tau}^*$ including τ_{i^*}, a time period t^* and the forgery signature ${\sigma_{i,t}}^*$. If the following requirements are fulfilled, we say that $\mathcal{A}$ wins.
  - 'accept' $\leftarrow$ FS-LLRS $\cdot$ Verify$(pp, \mu^*, {\mathcal{L}_\tau}^*, {\sigma_{i,t}}^*, t^*)$.
  - $\mathcal{A}$ has not queried $({\mu_i}^*, {\mathcal{L}_\tau}^*)$ on time t^* in $\mathcal{O}_{SO}$.
  - $\mathcal{A}$ has not queried any secret key corresponding to the identity tags in ${\mathcal{L}_\tau}^*$.
  - For all $\tau_i^* \in {\mathcal{L}_\tau}^*$, there is no $\mathcal{O}_{CO}(\tau_i^*, t')$ query with time period $t' \leq t^*$.

The advantage for $\mathcal{A}$ to attack the unforgeability with forward security game is defined as $Adv_{\mathcal{A}}^{\text{FS-Unforgeability}_{\text{FS-LLRS}}}(\lambda) := [\mathcal{A} \text{ wins the game}]$.

Definition 21 (**Unforgeability with Forward Security of FS-LLRS**). *For any* PPT *adversary* $\mathcal{A}$*, if the advantage* $Adv_{\mathcal{A}}^{\text{FS-Unforgeability}_{\text{FS-LLRS}}}(\lambda)$ *is negligible, we say that our* FS-LLRS *scheme satisfies the* unforgeability with forward security*.*

Linkability of FS-LLRS

This game is designed as:

- **Setup.** $\mathcal{C}$ executes the FS-LLRS $\cdot$ Setup algorithm to generate the public parameters pp and send it to $\mathcal{A}$.

- **Queries.** $\mathcal{A}$ performs a polynomial bounded number N queries adaptively:

  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key of a user $i \in [1, N]$. Then, $\mathcal{C}$ invokes the FS-LLRS $\cdot$ KeyExtract algorithm to get public-secret keys (pk_i, sk_i) and returns the public key pk_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key $sk_{i,t}$ of a user $i \in [1, N]$ on time t. $\mathcal{C}$ calculates its secret key $sk_{i,t}$ and then sends it to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature $\sigma_{i,t}$ for the user $i \in [1, N]$ on time period t. Then, $\mathcal{C}$ returns $\sigma_{i,t}$ to $\mathcal{A}$ by invoking the FS-LLRS $\cdot$ Sign algorithm.

- **Unlink.** $\mathcal{A}$ outputs two tuples $(\mathcal{L}_{\tau 1}^*, \mu_1^*, \sigma_{1,t}^*, t_1^*)$ and $(\mathcal{L}_{\tau 2}^*, \mu_2^*, \sigma_{2,t}^*, t_2^*)$, where $\mathcal{L}_{\tau i}^*$ is the users' identity tags set including user i^*'s identity tag, μ_i^* is a signed message, $\sigma_{i,t}^*$ is a signature including a link tag tag_i^*, and t_i^* is the time period. If the following requirements are fulfilled, we say that $\mathcal{A}$ wins.

  - 'accept' $\leftarrow$ FS-LLRS $\cdot$ Verify$(pp, \mu_1^*, \mathcal{L}_{\tau 1}^*, {\sigma_{1,t}}^*, t_1^*)$.
  - 'accept' $\leftarrow$ FS-LLRS $\cdot$ Verify$(pp, \mu_2^*, \mathcal{L}_{\tau 2}^*, {\sigma_{2,t}}^*, t_2^*)$.
  - 'unlink' $\leftarrow$ FS-LLRS $\cdot$ Link$(pp, \mu_1^*, \mu_2^*, \sigma_{1,t}^*, \sigma_{2,t}^*, \mathcal{L}_{\tau 1}^*, \mathcal{L}_{\tau 2}^*, t_1^*, t_2^*)$
  - $\mathcal{A}$ has not queried $({\mu_1}^*, {\mathcal{L}_{\tau 1}}^*)$ and $({\mu_2}^*, {\mathcal{L}_{\tau 2}}^*)$ in $\mathcal{O}_{SO}$.
  - For all $\tau_i^* \in {\mathbf{L}_\tau}^*$, there is no $\mathcal{O}_{CO}(\tau_i^*, t')$ query with time period $t' \leq t^*$.
  - $\mathcal{A}$ owns at most one secret key corresponding to the user identity tag in $\mathcal{L}_{\tau i}^*$.

The advantage for $\mathcal{A}$ to attack linkability game is defined as $Adv_{\mathcal{A}}^{\text{Linkability}_{\text{FS-LLRS}}}(\lambda) :=$ [$\mathcal{A}$ wins the game].

Definition 22 (**Linkability of FS-LLRS**). *For any* PPT *adversary* $\mathcal{A}$*, if the advantage* $Adv_{\mathcal{A}}^{\text{Linkability}_{\text{FS-LLRS}}}(\lambda)$ *is negligible, we say that our* FS-LLRS *scheme satisfies the* linkability.

4.5.3 The Concrete Construction

- **FS-LLRS** $\cdot$ **Setup**(λ): Given a security parameter λ, this algorithm performs as follows:

  1. Sets a prime $q \geq 2$, several integers n, $m \geq 2n\lceil \log q \rceil$, s, the length of a signed message d, a sampling parameter δ, and a matrix $\mathbf{P} \in \mathbb{Z}_q^{n \times m}$.
  2. Selects l as the depth of a binary tree and the total number of time periods as $T = 2^l$.
  3. Randomly selects $2l$ matrices $\mathbf{B}_1^{(0)}, \mathbf{B}_1^{(1)}, \cdots, \mathbf{B}_l^{(0)}, \mathbf{B}_l^{(1)} \in \mathbb{Z}_q^{n \times m}$.
  4. Randomly selects $d+1$ matrices $\mathbf{C}_0, \cdots, \mathbf{C}_d \in \mathbb{Z}_q^{n \times m}$.
  5. Sets integer k as the length of a user's identity tag. Note that $\frac{m}{2k}$ has to be an integer.
  6. Randomly selects $2k$ 0-1 matrices $\mathbf{\Gamma}_1^{(0)}, \mathbf{\Gamma}_1^{(1)}, \cdots, \mathbf{\Gamma}_k^{(0)}, \mathbf{\Gamma}_k^{(1)} \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2k}}$.
  7. Generates $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times \frac{m}{2}}$ as the master public key and its basis $\mathbf{T}_{A_0} \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2}}$ as the master secret key, by TrapGen algorithm.
  8. Sets $\mathbf{g}_k = [2^0, 2^1, \cdots, 2^{(\frac{m}{2n}-1)}] \in \mathbb{Z}_q^{\frac{m}{2n}}$.
  9. Sets an identity matrix $\mathbf{I} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} \in \mathbb{Z}_q^{n \times n}$.
  10. Computes $\mathbf{G}_k = \mathbf{I} \otimes \mathbf{g}_k \in \mathbb{Z}_q^{n \times \frac{m}{2}}$.
  11. Returns the public parameters $pp = (n, m, q, d, \delta, s, l, T, \mathbf{P}, \mathbf{B}_1^{(0)}, \mathbf{B}_1^{(1)}, \cdots, \mathbf{B}_l^{(0)}, \mathbf{B}_l^{(1)}, \mathbf{C}_0, \cdots, \mathbf{C}_d, \mathbf{\Gamma}_1^{(0)}, \mathbf{\Gamma}_1^{(1)}, \cdots, \mathbf{\Gamma}_k^{(0)}, \mathbf{\Gamma}_k^{(1)}, \mathbf{A}_0, \mathbf{G}_k)$.

- **FS-LLRS** · **KeyExtract**(pp, τ_i): Given the public parameters pp, the identity tag τ_i of a user i, this algorithm performs as follows:

  1. Parses the tag τ_i as $\tau_i[1], \cdots, \tau_i[k]$, where $\tau_i[j]$ denotes the j-th bit of the tag τ_i.
  2. Computes $\mathbf{F}_{\tau,i} = (\mathbf{\Gamma}_1^{\tau_i[1]} \| \cdots \| \mathbf{\Gamma}_k^{\tau_i[k]}) \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2}}$.
  3. Computes the public key $\mathbf{B}_{0,i} \in \mathbb{Z}_q^{n \times m}$ of a user i as $\mathbf{B}_{0,i} = [\mathbf{A}_0 || (\mathbf{A}_0 \cdot \mathbf{F}_{\tau,i} + \mathbf{G}_k)]$.
  4. Computes $\mathbf{T}_{B_{0,i}} \in \mathbb{Z}_q^{m \times m}$ as the basis of $\Lambda^{\perp}(\mathbf{B}_{0,i})$.
  5. Returns public-secret keys $(\mathbf{B}_{0,i}, \mathbf{T}_{B_{0,i}})$ for a user i.

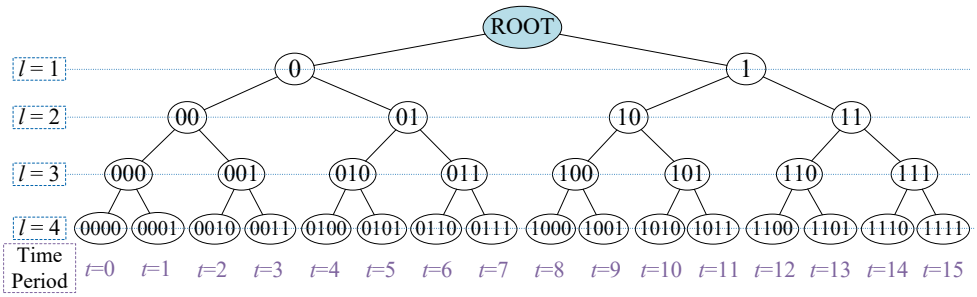


Figure 4.2: Binary Tree ($l = 4$) for Time Period Expression.

- **FS-LLRS** · **KeyUpdate**$(pp, t, sk_{i,t}, \tau_i)$: Given the public parameters pp, a time period t, a user index i, a secret key $sk_{i,t}$, and an identity tag τ_i used to compute $pk_i = \mathbf{B}_{0,i}$. The secret key update is achieved by combining the ExtBasis algorithm and a binary tree. This algorithm performs as follows:

  1. By the minimal cover set technique, we can present all nodes in a binary tree (with depth l). The time period t is represented in binary as $t = (t_1 t_2 \cdots t_i)$, where $t_i \in \{0, 1\}$ and $i \in \{1, 2, \cdots, l\}$. Let $\Gamma^{(i)} := (\gamma_1 \gamma_2 \cdots \gamma_i) \in \texttt{Leaf}(t)$, where $\gamma_i \in \{0, 1\}$. For a binary tree's leaf node t, the minimal cover set $\texttt{Leaf}(t)$ includes the ancestor of all leaves in $\{t, \cdots, 2^l - 1\}$, while excluding any ancestors of leaves in $\{0, \cdots, t - 1\}$.

2. In Figure 4.2, we display a binary tree with depth $l = 4$, containing a total of $2^l = 16$ time periods. The examples of $\texttt{Leaf}(t)$ are as: $t = 0 : \{\texttt{root}\}$; $t = 1 : \{0001, 001, 01, 1\}$; $t = 2 : \{001, 01, 1\}$; $t = 3 : \{0011, 01, 1\}$; $t = 4 : \{01, 1\}$; $t = 5 : \{0101, 011, 1\}$; $t = 6 : \{011, 1\}$; $t = 7 : \{0111, 1\}$; $t = 8 : \{1\}$; $t = 9 : \{1001, 101, 11\}$; $t = 10 : \{101, 11\}$; $t = 11 : \{1011, 11\}$; $t = 12 : \{11\}$; $t = 13 : \{1101, 111\}$; $t = 14 : \{111\}$; $t = 15 : \{1111\}$.

3. Let $\mathbf{M}_{\Gamma^{(i)}} := [\mathbf{B}_{0,i} \| \mathbf{B}_1^{(\gamma_1)} \| \mathbf{B}_2^{(\gamma_2)} \| \cdots \| \mathbf{B}_i^{(\gamma_i)}] \in \mathbb{Z}^{n \times (l+1)m}$ as $\Gamma^{(i)}$'s corresponding matrix. Each node in a binary tree possesses a distinct secret key. As for $\texttt{Leaf}(001)$ at level 3, its secret key $sk_{i,001}$ is the basis of lattice $\Lambda_q^{\perp}(\mathbf{M}_{\Gamma^{(001)}})$. For the initial secret key, we have $sk_{i,0} = \{\mathbf{T}_{B_{0,i}}\}$. When time period $t = 1$, we update the $sk_{i,0}$ to $sk_{i,1} = \{\mathbf{N}_{0001}, \mathbf{N}_{001}, \mathbf{N}_{01}, \mathbf{N}_1\}$, where $\mathbf{N}_{0001}, \mathbf{N}_{001}, \mathbf{N}_{01}, \mathbf{N}_1$ are the corresponding basis for matrices $\mathbf{M}_{i,0001} = [\mathbf{B}_{0,i} \| \mathbf{B}_1^{(0)} \| \mathbf{B}_2^{(0)} \| \mathbf{B}_3^{(0)} \| \mathbf{B}_4^{(1)}]$, $\mathbf{M}_{i,001} = [\mathbf{B}_{0,i} \| \mathbf{B}_1^{(0)} \| \mathbf{B}_2^{(0)} \| \mathbf{B}_3^{(1)}]$, $\mathbf{M}_{i,01} = [\mathbf{B}_{0,i} \| \mathbf{B}_1^{(0)} \| \mathbf{B}_2^{(1)}]$ and $\mathbf{M}_{i,1} = [\mathbf{B}_{0,i} \| \mathbf{B}_1^{(1)}]$, respectively. Note that the $sk_{i,t}$ includes the basis of all nodes in set $\texttt{Leaf}(t)$.

4. Let $\mathbf{N}_{\Gamma^{(i)}}$ be the basis of node $\Gamma^{(i)}$ in binary tree. Then, it leverages the ExtBasis algorithm to update its basis via two different methods as below:
   - Through its any ancestor's basis: Given any ancestor's basis $\mathbf{N}_{\Gamma^{(j)}}$, the authority invokes the $\mathsf{ExtBasis}(\mathbf{M}_{\Gamma^{(i)}}, \mathbf{N}_{\Gamma^{(j)}})$ algorithm to calculate $\mathbf{N}_{\Gamma^{(i)}}$ as the basis on the time period i, where $\Gamma^{(i)} = (\gamma_1 \gamma_2 \cdots \gamma_j \gamma_{j+1} \cdots \gamma_i)$.
   - Through the root secret key $\mathbf{T}_{\mathbf{B}_{0,i}}$: The authority executes the $\mathbf{N}_{\Gamma^{(i)}} \leftarrow \mathsf{ExtBasis}(\mathbf{M}_{\Gamma^{(i)}}, \mathbf{T}_{\mathbf{B}_{0,i}})$ algorithm to obtain $\mathbf{N}_{\Gamma^{(i)}}$ directly.

5. For the update from $sk_{i,t}$ to sk_{i+1}, it firstly determines the minimal cover set $\texttt{Leaf}(t+1)$. Following, it computes all the basis of nodes that are in $\texttt{Leaf}(t+1) \backslash \texttt{Leaf}(t)$ (shown in Step. 4). Finally, it discards the basis of nodes that are in $\texttt{Leaf}(t) \backslash \texttt{Leaf}(t+1)$.

6. Returns the secret key $sk_{i,t+1}$ and discards former $sk_{i,t}$.

- **FS-LLRS** · **Sign**$(pp, \mu, sk_{l,t}, \mathcal{L}_\tau, t)$: Given the public parameters pp, a signed message $\mu \in \{0\} \times \{0,1\}^d$, the secret key $sk_{l,t} = \mathbf{N}_{\Gamma^{(t)}}$ of a user l, a ring of N users with identity tags set $\mathcal{L}_\tau = \{\tau_1, \cdots, \tau_N\}$, and the time period t, this algorithm performs as:

  1. Parses the message μ as $\mu[0], \mu[1], \cdots, \mu[d]$, where $\mu[j]$ is the j-th bit of the message μ.
  2. Computes $\mathbf{C}_\mu = \sum_{j=0}^{d} (-1)^{\mu[j]} \mathbf{C}_j = (-1)^{\mu[0]}\mathbf{C}_0 + (-1)^{\mu[1]}\mathbf{C}_1 + \cdots + (-1)^{\mu[d]}\mathbf{C}_d \in \mathbb{Z}_q^{n \times m}$.
  3. Parses the tag τ_i as $\tau_i[1], \cdots, \tau_i[k]$.
  4. Computes $\mathbf{B}_{0,i} = [\mathbf{A}_0 || (\mathbf{A}_0 \cdot \mathbf{F}_{\tau,i} + \mathbf{G}_k)]$, where $\mathbf{F}_{\tau,i} = (\mathbf{\Gamma}_1^{\tau_i[1]} \| \cdots \| \mathbf{\Gamma}_k^{\tau_i[k]}) \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2}}$.
  5. Sets $\mathbf{M}_{i,t} = [\mathbf{B}_{0,i} \| \mathbf{B}_1^{(t_1)} \| \cdots \| \mathbf{B}_l^{(t_l)}] \in \mathbb{Z}_q^{n \times (l+1)m}$.
  6. Sets $\mathbf{B}_{R,t} = [\mathbf{M}_{1,t} \| \cdots \| \mathbf{M}_{N,t} \| \mathbf{C}_\mu] \in \mathbb{Z}_q^{n \times (N(l+1)+1)m}$.
  7. Invokes the GenSamplePre algorithm to generate a signature $\mathbf{e} \leftarrow \mathsf{GenSamplePre}(\mathbf{B}_{R,t}, \mathbf{M}_{l,t}, \mathbf{N}_{\Gamma^{(t)}}, 0, \delta) \in \mathbb{Z}_q^{(N(l+1)+1)m}$, which distributed according to $\mathcal{D}_{\Lambda^\perp \mathbf{B}_{R,t}, \delta}$.
  8. Computes $\boldsymbol{\upsilon} = \mathbf{P} \cdot \mathbf{T}_{B_{0,i}} \in \mathbb{Z}_q^{n \times m}$.
  9. Returns the signature $\sigma = \{\mathbf{e}\}$ including link tag $tag = \boldsymbol{\upsilon}$.

- **FS-LLRS** · **Verify**$(pp, \mu, \sigma_t, \mathcal{L}_\tau, t)$: Given the public parameters pp, a signed message μ, the signature σ_t of a message μ, and a ring of N users with identity tags set $\mathcal{L}_\tau = \{\tau_1, \cdots, \tau_N\}$, and the time period t, this algorithm performs as follows:

  1. Parses the message μ as $\mu[0], \mu[1], \cdots, \mu[d]$.
  2. Computes $\mathbf{C}_\mu = \sum_{i=1}^{d} (-1)^{\mu[j]} \mathbf{C}_j \in \mathbb{Z}_q^{n \times m}$.
  3. Checks if $0 \leq \|\mathbf{e}\| \leq \delta\sqrt{(N(l+1)+1)m}$.
  4. Parses the tag τ_i as $\tau_i[1], \cdots, \tau_i[k]$.

5. Computes $\mathbf{B}_{0,i} = [\mathbf{A}_0||(\mathbf{A}_0 \cdot \mathbf{F}_{\tau,i} + \mathbf{G}_k)]$, where $\mathbf{F}_{\tau,i} = (\mathbf{\Gamma}_1^{\tau_i[1]}\| \cdots \|\mathbf{\Gamma}_k^{\tau_i[k]}) \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2}}$.
6. Computes $\mathbf{M}_{i,t} = [\mathbf{B}_{0,i}\|\mathbf{B}_1^{(t_1)}\| \cdots \|\mathbf{B}_l^{(t_l)}]$.
7. Checks if $[\mathbf{M}_{1,t}\| \cdots \|\mathbf{M}_{N,t}\|\mathbf{C}_\mu] \cdot \mathbf{e} = 0 \mod q$.
8. Returns 'accept' or 'reject'.

- **FS-LLRS** $\cdot$ **Link**$(pp, \mu_1, \mu_2, \sigma_1, \sigma_2, \mathcal{L}_{\tau 1}, \mathcal{L}_{\tau 2}, t_1, t_2)$: Given the public parameters pp, two messages μ_1, μ_2, two signatures σ_1, σ_2 including two link tags tag_1, tag_2, and two ring of N users with identity tags sets $\mathcal{L}_{\tau 1}$, $\mathcal{L}_{\tau 2}$, and the time period t_1, t_2, this algorithm performs as follows:

  1. Checks if 'accept' $\leftarrow$ Verify$(pp, \mu_1, \sigma_1, \mathcal{L}_{\tau 1}, t_1)$.
  2. Checks if 'accept' $\leftarrow$ Verify$(pp, \mu_2, \sigma_2, \mathcal{L}_{\tau 2}, t_2)$.
  3. Checks if two link tags $\boldsymbol{\upsilon}_1 = \boldsymbol{\upsilon}_2$.
  4. Returns 'link' or 'unlink'.

4.5.4 Correctness

In our FS-LLRS scheme, the GenSamplePre algorithm samples a vector $\mathbf{e}$, where $\|\mathbf{e}\| \leq \delta\sqrt{(N(l+1)+1)m}$ from the distribution within the negligible statistical distance of $\mathcal{D}_{\Lambda_q^\perp \mathbf{B}_{R,t},\delta}$. It satisfies the one-wayness equation $\mathbf{B}_{R,t} \cdot \mathbf{e} = 0 \mod q$ with overwhelming probability.

For linking correctness, the user i signs two messages μ_1 and μ_2 using the same secret key $\mathbf{T}_{B_{0,i}}$ without the affect of time to generate two signatures σ_1 and σ_2 containing the link tags $\boldsymbol{\upsilon}_1 = \mathbf{P} \cdot \mathbf{T}_{B_{0,i}}$ and $\boldsymbol{\upsilon}_2 = \mathbf{P} \cdot \mathbf{T}_{B_{0,i}}$, respectively. Since $\boldsymbol{\upsilon}_1$, $\boldsymbol{\upsilon}_2$ are generated with the same randomly chosen matrix $\mathbf{P}$, if the user signs the messages μ_1, μ_2 with the same secret key $\mathbf{T}_{B_{0,i}}$, then it must be the case that $\boldsymbol{\upsilon}_1 = \boldsymbol{\upsilon}_2$.

4.5.5 Security Analysis

Anonymity

Lemma 6. *Let $(\tau_0, \tau_1, \mathcal{L}_\tau, \mu, t)$ be a tuple such that t is the time period, μ is a message to be signed with the ring $\mathcal{L}_\tau$, including τ_0 and τ_1. For the $SIS_{q,(N(l+1)+1)m,\delta}$ hardness problem, $\sigma_0 \leftarrow$ FS-LLRS $\cdot$ $\mathsf{Sign}(pp, \mu, sk_{0,t}, \mathcal{L}_\tau, t)$ and $\sigma_1 \leftarrow$ FS-LLRS $\cdot$ $\mathsf{Sign}(pp, \mu, sk_{1,t}, \mathcal{L}_\tau, t)$ are computationally indistinguishable.*

Analysis. Through the Lemma 3, the $\mathsf{GenSamplePre}$ algorithm samples a vector $\mathbf{e} \in \mathbb{Z}^{(N(l+1)+1)m}$ within negligible statical distance of $\mathcal{D}_{\Lambda_q^\perp(\mathbf{B}_{R,t}),\delta}$, which satisfied $\mathbf{B}_{R,t} \cdot \mathbf{e} = 0$. Inverting the above equation is equivalent to solve the $SIS_{q,(N(l+1)+1)m,\delta}$ hardness problem.

In the FS-LLRS $\cdot$ Sign algorithm, two ring signatures σ_0 and σ_1 have the same distribution, which implies that σ_0 and σ_1 are computationally indistinguishable.

Theorem 8 (Anonymity of FS-LLRS.)**.** *The probability of an adversary $\mathcal{A}$ to win the anonymity game with the polynomial time is negligible. Therefore, the proposed* FS-LLRS *scheme satisfies the anonymity in the standard model.*

Proof. We assume that there exists an adaptive adversary $\mathcal{A}$ who tries to break the anonymity of the FS-LLRS scheme and a challenger $\mathcal{C}$ is able to respond to queries of $\mathcal{A}$.

- **Setup.** $\mathcal{C}$ executes the FS-LLRS $\cdot$ Setup algorithm to generate the public parameters $pp = (n, m, q, d, \delta, s, l, T, \mathbf{P} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{B}_1^{(0)}, \mathbf{B}_1^{(1)}, \cdots, \mathbf{B}_l^{(0)}, \mathbf{B}_l^{(1)}\} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{C}_0, \cdots, \mathbf{C}_d\} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{\Gamma}_1^{(0)}, \mathbf{\Gamma}_1^{(1)}, \cdots, \mathbf{\Gamma}_k^{(0)}, \mathbf{\Gamma}_k^{(1)}\} \in \mathbb{Z}_q^{\frac{m}{2}\times\frac{m}{2k}}, \mathbf{A}_0 \in \mathbb{Z}_q^{n\times\frac{m}{2}}, \mathbf{G}_k \in \mathbb{Z}_q^{n\times\frac{m}{2}})$, and sends it to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ picks an index $i \in [N]$ and performs a polynomial bounded number N queries adaptively:

  - Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, N]$, $\mathcal{A}$ queries the public key pk_i of a user i. Then, $\mathcal{C}$ invokes the FS-LLRS $\cdot$ KeyExtract algorithm to get the key pair (pk_i, sk_i). Finally, $\mathcal{C}$ returns the public key pk_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key $sk_{i,t}$ of a user i on period time t. Then, $\mathcal{C}$ responds the answer to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ inputs a time period t, a users' identity tags set $\mathcal{L}_\tau$, i-th user's identity tag $\tau_i \in \mathcal{L}_\tau$ and a message μ. $\mathcal{C}$ then invokes the FS-LLRS $\cdot$ Sign algorithm and outputs a signature $\sigma_{i,t}$ to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ selects a time period t^*, a message $\mu^* \in \{0\} \times \{0,1\}^d$, a users' identity tags set $\mathcal{L}_\tau^*$, two identity tags $\tau_0^*, \tau_1^* \in \mathcal{L}_\tau^*$ to $\mathcal{C}$. Then, $\mathcal{A}$ sends $(\tau_0^*, \tau_1^*, \mu^*, \mathcal{L}_\tau^*, t^*)$ to $\mathcal{C}$ to request a signature. Finally, $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ and sends $\sigma_{i,t}^*$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b' = b$, we say that $\mathcal{A}$ wins the game.

Analysis. The signatures have the same distribution within a negligible statistical distance of $\mathcal{D}_{\Lambda^\perp \mathbf{B}_{R,t},\delta}$, which are computationally indistinguishable. If $\mathcal{A}$ successfully distinguishes signatures with non-negligible probability, it contradicts Lemma 6. Hence, we claim that the advantage of $\mathcal{A}$ to win the game is negligible and our FS-LLRS scheme is anonymous. □

Unforgeability with Forward Security

Theorem 9 (Unforgeability with Forward Security of FS-LLRS.)**.** *The proposed* FS-LLRS *scheme satisfies the unforgeability with forward security in the standard model under the* $SIS_{q,N(1+2l)m,\delta}$ *hardness assumption.*

Proof. We assume that there exists an adaptive adversary $\mathcal{A}$ who tries to break the unforgeability with forward security of the FS-LLRS scheme and a challenger $\mathcal{C}$ is able to solve the SIS problem.

- **Setup.** $\mathcal{C}$ generates and sends the public parameters pp to $\mathcal{A}$ with the following parameters generation procedures.

  - Initially executes the FS-LLRS $\cdot$ Setup algorithm to obtain the public parameters, including the integers n, m, q, δ, s, the length of message d, a public matrix $\mathbf{P} \in \mathbb{Z}_q^{n \times m}$, a binary tree with divided $T = 2^l$ time period, $2k$ 0-1 matrices $\{\mathbf{\Gamma}_1^{(0)}, \mathbf{\Gamma}_1^{(1)}, \cdots, \mathbf{\Gamma}_k^{(0)}, \mathbf{\Gamma}_k^{(1)}\} \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2k}}$, master public key $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times \frac{m}{2}}$, a matrix $\mathbf{G}_k \in \mathbb{Z}_q^{n \times \frac{m}{2}}$.
  - Randomly selects $2l$ matrices $\mathbf{V}_1^{(0)}, \mathbf{V}_1^{(1)} \cdots \mathbf{V}_l^{(0)}, \mathbf{V}_l^{(1)} \in \mathbb{Z}_q^{n \times m}$.
  - For $i^* \in [N]$, sets $\mathbf{B}_{i^*} \in \mathbb{Z}_q^{n \times m}$ by τ_{i^*} and computes $\mathbf{M}_{i^*} = [\mathbf{B}_{i^*} \| \mathbf{V}_1^{(0)} \| \mathbf{V}_1^{(1)} \| \cdots \| \mathbf{V}_l^{(0)} \| \mathbf{V}_l^{(1)}] \in \mathbb{Z}_q^{n \times (1+2l)m}$.
  - Sets $\mathbf{M} = [\mathbf{M}_1 \| \cdots \| \mathbf{M}_N] \in \mathbb{Z}_q^{n \times N(1+2l)m}$ as SIS matrix.
  - Sets a ring $R_{\mathbf{v}} = \{v_1, \cdots, v_N\}$.
  - Invokes the TrapGen algorithm to get a matrix $\mathbf{S} \in \mathbb{Z}_q^{n \times m}$ and its short basis $\mathbf{T}_S \in \mathbb{Z}_q^{m \times m}$.
  - For $i \in [l]$, sets $\mathbf{B}_i^{(0)} = \mathbf{V}_i^{(0)}$, $\mathbf{B}_i^{(1)} = \mathbf{V}_i^{(1)}$.
  - Randomly chooses $d+1$ matrices $\mathbf{U}_0, \cdots, \mathbf{U}_{d+1} \in \mathbb{Z}_q^{N(l+1)m \times m}$.
  - Sets $m_0 = 1$ and randomly chooses d uniformly scalars $m_1, \cdots m_d \in \mathbb{Z}_q^*$.
  - Guesses target attacking time period $t^* = (t_1^*, \cdots, t_l^*) \in \{0, \cdots, T-1\}$.
  - For $i \in [l]$, sets $\mathbf{B}_i^{(t_i^*)} = \mathbf{V}_i^{(t_i^*)}$

- **Queries.** $\mathcal{A}$ picks an index $i \in [N]$ and performs a polynomial bounded number N queries adaptively:

  - Registration Oracle $\mathcal{O}_{RO}$: Constructs a list $\mathcal{L}$ and sets it empty initially. For an index $i \in [1, N]$, $\mathcal{A}$ queries the public key pk_i of a user i. If $i = v_{i^*} \in R_{\mathbf{v}}$, $\mathcal{C}$ sets $\mathbf{B}_i = \mathbf{B}_{i^*}$; otherwise, if $i \notin R_{\mathbf{v}}$, $\mathcal{C}$ invokes the FS-LLRS $\cdot$ KeyExtract algorithm to generate a matrix $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$ and its

short basis $\mathbf{T}_{B_0} \in \mathbb{Z}_q^{m \times m}$ and then stores the $(\tau_i, \mathbf{B}_i, \mathbf{T}_{B_0}, i)$ in $\mathcal{L}$. Finally, $\mathcal{C}$ returns public key $pk_i = \mathbf{B}_i$ to $\mathcal{A}$.

– Corruption Oracle $\mathcal{O}_{CO}$: For a user $i \in [N]$ and $t = (t_1, \cdots, t_l)$, the public key is $\mathbf{B}_i$. If $pk_i \notin R_{\mathbf{v}}$ or $t \leq t^*$, $\mathcal{C}$ outputs $\perp$. We assume that $k < l$, $t_k \neq t_k^*$, and then $\mathcal{C}$ calculates $\mathbf{T}_{t_k}$ by $\mathsf{ExtBasis}$ algorithm. The returned secret key $sk_{i,t}$ is obtained through FS-LLRS $\cdot$ $\mathsf{KeyUpdate}$ algorithm.

– Signing Oracle $\mathcal{O}_{SO}$: Inputs an identity tag τ_i, a ring R, a time period t and a message μ to query a signature (without consideration of link tag). Firstly, for a ring $R_{\mathbf{v}}$, $\mathcal{C}$ computes $\mathbf{B}_{R_{\mathbf{v}},t^*} = [\mathbf{M}_1^* \| \cdots \| \mathbf{M}_N^*]$, where $\mathbf{M}_i^* = [\mathbf{B}_i \| \mathbf{B}_1^{(t_1^*)} \| \cdots \| \mathbf{B}_l^{(t_l^*)}]$ for $i \in [N]$. Then, $\mathcal{C}$ computes $\mathbf{C}_i = \mathbf{B}_{R_{\mathbf{v}},t^*}\mathbf{U}_i + m_i\mathbf{S}$ for $i \in \{0, \cdots, d\}$. Finally, it samples the vector e in three different situations and returns it to $\mathcal{A}$, executing as follows:

1. For the ring $R = R_{\mathbf{v}}$, $\mathcal{C}$ computes the matrix $\mathbf{B}_{R,t} = [\mathbf{M}_1 \| \cdots \| \mathbf{M}_N]$, where $\mathbf{M}_i = [\mathbf{B}_i \| \mathbf{B}_1^{(t_1)} \| \cdots \| \mathbf{B}_l^{(t_l)}]$ for $i \in [N]$. Then, $\mathcal{C}$ calculates $\mathbf{U}_\mu = \sum_{i=1}^{d} (-1)^{\mu[i]} \mathbf{U}_i$ and $m_\mu = \sum_{i=1}^{d} (-1)^{\mu[i]} m_i \neq 0$. Finally, $\mathcal{C}$ constructs the matrix $\mathbf{M}' = [\mathbf{B}_{R,t} \| \mathbf{B}_{R,t}\mathbf{U}_\mu + m_\mu \mathbf{S}]$ and finds a short $\mathbf{e} \in \Lambda_q^\perp(\mathbf{M}')$ by $\mathbf{T}_S$.
2. Otherwise, if the tuple $(\tau_i, \mathbf{B}_i, \mathbf{T}_{B_i}, i)$ includes in $\mathcal{L}$, $\mathcal{C}$ constructs the matrix $\mathbf{M}'' = [\mathbf{B}_{R,t} \| \sum_{i=1}^{d} (-1)^{\mu[i]} \mathbf{C}_i]$. Then, $\mathcal{C}$ samples a vector $\mathbf{e}$ as $\mathbf{e} \leftarrow \mathsf{GenSamplePre}(\mathbf{M}'', \mathbf{B}_i, \mathbf{T}_{B_i}, 0, \delta)$.
3. Otherwise, if a user $k \in R_j$ such that $(\tau_k, \mathbf{B}_k, \mathbf{T}_{B_k}, k)$ contains in list $\mathcal{L}$, $\mathcal{C}$ samples $\mathbf{e}$ as $\mathbf{e} \leftarrow \mathsf{GenSamplePre}(\mathbf{M}'', \mathbf{B}_k, \mathbf{T}_{B_k}, 0, \delta)$.

- **Forge.** $\mathcal{A}$ outputs a forgery tuple $(\tau_i^*, \mu^*, \sigma_{i,t}^*, R^*, t^*)$, where $\sigma_{i,t}^*$ is the forgery signature for the user i^*. If $R^* \neq R_t$, $\mathcal{C}$ returns abort; otherwise, $\mathcal{C}$ performs as:

– Computes $\mathbf{U}_{\mu^*} = \sum_{i=1}^{d} (-1)^{\mu^*[i]} \mathbf{U}_i$.

– Computes $m_{\mu^*} = \sum_{i=1}^{d} (-1)^{\mu^*[i]} m_i \neq 0$.

– Separates σ^* as $\begin{pmatrix}\sigma_1^*\\ \sigma_2^*\end{pmatrix}$.

– Computes $\mathbf{e}^* \in \mathbb{Z}^{N(l+1)m}$ as $\mathbf{e}^* = \sigma_1^* + \mathbf{U}_{\mu^*}\sigma_2^*$.

Analysis. Let $\mathbf{C}_{\mu^*} = \sum_{i=1}^{d}(-1)^{\mu^*[i]}\mathbf{C}_i = \sum_{i=1}^{d}(-1)^{\mu^*[i]}(\mathbf{B}_{R^*,t^*}\mathbf{U}_i + m_i\mathbf{S})$ and we have $\mathbf{C}_{\mu^*} = \mathbf{B}_{R^*,t^*}\mathbf{U}_{\mu^*}$. Then, we can express as $\mathbf{B}_{R^*,t^*}\mathbf{e}^* = \mathbf{B}_{R^*,t^*}(\sigma_1^* + \mathbf{U}_{\mu^*}\sigma_2^*) = [\mathbf{B}_{R^*,t^*}|\mathbf{B}_{R^*,t^*}\mathbf{U}_{\mu^*}]\begin{pmatrix}\sigma_1^*\\ \sigma_2^*\end{pmatrix} = [\mathbf{B}_{R^*,t^*}|\mathbf{C}_{\mu^*}]\mathbf{e}^* = [\mathbf{B}_{R_\mathbf{v},t^*}|\mathbf{C}_{\mu^*}]\mathbf{e}^* = 0$, and $\mathbf{M}$ can be obtained from $\mathbf{B}_{R_\mathbf{v},t^*}$. By employing a method similar with [13], we can find that short non-zero $\mathbf{e}^*$ is the solution to the given SIS matrix in high probability. Thus, we prove that our FS-LLRS scheme satisfies the unforgeability with forward security. □

Linkability

Theorem 10 (Linkability of FS-LLRS.)**.** *The proposed* FS-LLRS *scheme satisfies the linkability in the standard model if the* FS-LLRS *scheme satisfies the unforgeability with forward security.*

Proof. We assume that there exists an adaptive adversary $\mathcal{A}$ who tries to break the linkability of the FS-LLRS scheme and a challenger $\mathcal{C}$ is able to respond to queries of $\mathcal{A}$.

- **Setup.** $\mathcal{C}$ executes the FS-LLRS $\cdot$ Setup algorithm to generate the public parameters $pp = (n, m, q, d, \delta, s, l, T, \mathbf{P} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{B}_1^{(0)}, \mathbf{B}_1^{(1)}, \cdots, \mathbf{B}_l^{(0)}, \mathbf{B}_l^{(1)}\} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{C}_0, \cdots, \mathbf{C}_d\} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{\Gamma}_1^{(0)}, \mathbf{\Gamma}_1^{(1)}, \cdots, \mathbf{\Gamma}_k^{(0)}, \mathbf{\Gamma}_k^{(1)}\} \in \mathbb{Z}_q^{\frac{m}{2}\times\frac{m}{2k}}, \mathbf{A}_0 \in \mathbb{Z}_q^{n\times\frac{m}{2}}, \mathbf{G}_k \in \mathbb{Z}_q^{n\times\frac{m}{2}})$, and sends it to $\mathcal{A}$.

- **Queries.** $\mathcal{A}$ picks an index $i \in [N]$ and performs a polynomial bounded number N queries adaptively:

– Registration Oracle $\mathcal{O}_{RO}$: For an index $i \in [1, N]$, $\mathcal{A}$ queries the public key pk_i of a user i. Then, $\mathcal{C}$ invokes the FS-LLRS $\cdot$ KeyExtract algorithm to get the key pair (pk_i, sk_i). Finally, $\mathcal{C}$ returns public key pk_i to $\mathcal{A}$.

– Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ queries the secret key $sk_{i,t}$ of a user i on time period t. Then, $\mathcal{C}$ responds the answer to $\mathcal{A}$.

– Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ inputs a time period t, a users' identity tags set $\mathcal{L}_\tau$, i-th user's identity tag $\tau_i \in \mathcal{L}_\tau$ and a message μ. $\mathcal{C}$ then invokes the FS-LLRS $\cdot$ Sign algorithm and outputs a signature $\sigma_{i,t}$ to $\mathcal{A}$.

- **Unlink.** $\mathcal{A}$ outputs two tuples $(\mathcal{L}_{\tau 1}^*, \mu_1^*, \sigma_{1,t}^*, t_1^*)$, $(\mathcal{L}_{\tau 2}^*, \mu_2^*, \sigma_{2,t}^*, t_2^*)$, where t_i^* is the time period, $\mathcal{L}_{\tau i}^*$ is the users' identity tags set including a user i^*'s identity tag, μ_i^* is a signed message, and $\sigma_{i,t}^*$ is a signature including the link tag $\boldsymbol{\upsilon}_i^*$.

Analysis. We assume that $\mathcal{A}$ generates two ring signatures $\sigma_{1,t}^*$, $\sigma_{2,t}^*$ with non-negligible probability while only owns one secret key. Besides, both FS-LLRS $\cdot$ Verify(pp, $\mathcal{L}_{\tau 1}^*, \mu_1^*, \sigma_{1,t}^*, t_1^*$) and FS-LLRS $\cdot$ Verify$(pp, \mathcal{L}_{\tau 2}^*, \mu_2^*, \sigma_{2,t}^*, t_2^*)$ algorithm outputs 'accept'. Since our FS-LLRS scheme satisfies the unforgeability with forward security, these two signatures can pass the FS-LLRS $\cdot$ Verify algorithm only if the $\mathcal{A}$ honestly generates two signatures $\sigma_{1,t}^*$, $\sigma_{2,t}^*$.

When $\mathcal{A}$ generates the two signatures, we have two link tags $\boldsymbol{\upsilon}_1^* = \mathbf{P}^* \cdot sk_1^*$ and $\boldsymbol{\upsilon}_2^* = \mathbf{P}^* \cdot sk_2^*$, respectively. Since $\mathcal{A}$ only has one secret key, then $sk_1^* = sk_2^*$. Moreover, since the public matrix $\mathbf{P}^*$ is the same, we get $\boldsymbol{\upsilon}_1^* = \boldsymbol{\upsilon}_2^*$. It shows that the two tuples of $\mathcal{A}$ verified by the FS-LLRS $\cdot$ Link$(pp, \mu_1^*, \mu_2^*, \sigma_{1,t}^*, \sigma_{2,t}^*, \mathcal{L}_{\tau 1}^*, \mathcal{L}_{\tau 2}^*, t_1^*, t_2^*)$ algorithm will return 'link', which contradicts the assumption of the linkability game. Hence, the advantage of $\mathcal{A}$ is negligible and our FS-LLRS scheme is linkable. □

4.6 Performance Evaluation and Comparison

We perform a comparative evaluation of our LLRS and FS-LLRS scheme with other lattice-based ring signatures (those can used in e-health system) [8, 38, 4, 68, 20] in computational and communication overhead. Specifically, we implemented our scheme in Python language and all simulation experiments were conducted in a system environment featuring an Apple M2 processor and 16.0 GB of memory. In general, we evaluate the overhead in different parameter setting environments shown in Table 4.1.

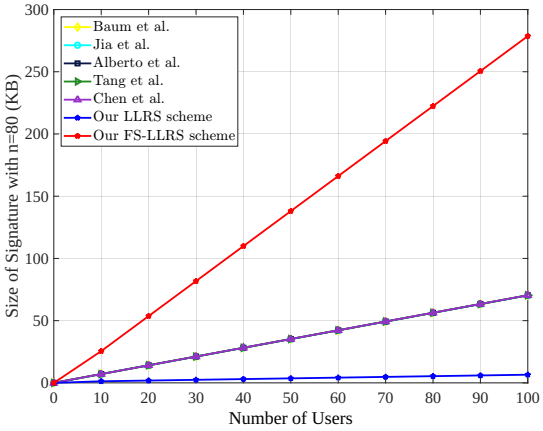
Table 4.1: Parameters Settings.

Recommendation Choice	n	m	q	d	l	k
Type I	80	960	80	10	3	4
Type II	128	1792	128	10	3	4
Type III	256	2048	256	10	3	4

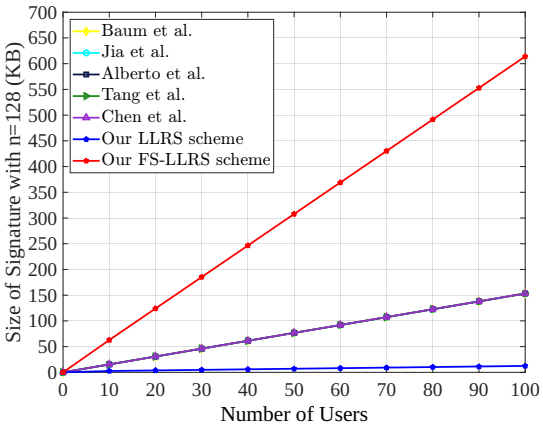
4.6.1 Communication Overhead Comparison

In this part, we focus on comparing the ring signature size. Specifically, in Figure 4.3(a), 4.3(b), 4.3(c), we show the comparative experimental results with schemes [8, 38, 4, 68, 20] at parameter settings of $n = 80$, $n = 128$ and $n = 256$, respectively. It is easy to notice that under the three types of parameter settings, our LLRS scheme has the lowest cost, while the FS-LLRS scheme has the highest overhead. However, our FS-LLRS scheme is the first scheme to guarantee both forward security and linkability in a quantum setting. Hence, the slightly higher communication overhead is an acceptable trade-off with security.

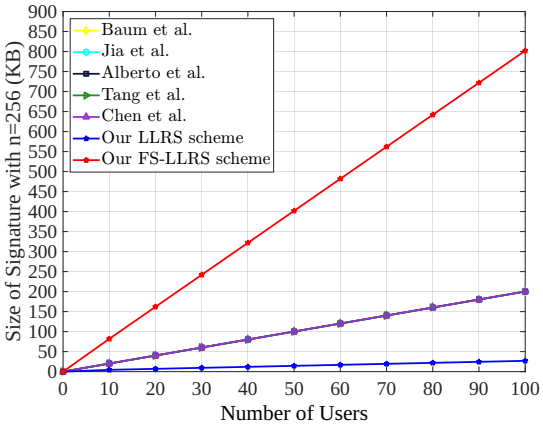
Specifically, we provide the communication overhead values for each scheme under different parameter settings when the ring size is 100. When we select the parameter



(a) Overhead with $n = 80$.



(b) Overhead with $n = 128$.



(c) Overhead with $n = 256$.

Figure 4.3: Comparison of Communication Cost between Our LLRS and FS-LLRS Schemes, and Current State-of-the-Art Ring Signature Primitives [8], [38], [4], [68], [20] with Different Parameters Settings.

setting of type I ($n = 80$), the communication overhead of our FS-LLRS scheme, our LLRS scheme, and others are 278.6KB, 6.5KB, and approximately 70.4KB, respectively. When the parameter setting is type II ($n = 128$), the communication cost of our FS-LLRS scheme, our LLRS scheme, and others are 614.0KB, 12.4KB, and approximately 153.3KB, respectively. When the parameter setting is type III ($n = 256$), the communication cost of our FS-LLRS scheme, our LLRS scheme, and others are 802.1KB, 27.0KB, and approximately 200.3KB, respectively.

In Table 4.2, we show the theoretical comparative analysis of user public key, user secret key, signature size and link tag.

4.6.2 Computational Overhead Comparison

We initially assessed the running time of our LLRS and FS-LLRS schemes to consider computational overhead. The evaluation involved examining the running time of each algorithm by setting the ring size to 10 in Figure 4.4(a), 50 in Figure 4.4(b), and 100 in Figure 4.4(c), respectively. For Link algorithm, two verification operations and one operation to check the link are required. The overhead of each algorithm increases with the growth of ring size. For the size is stationary, the larger the value of n, the more overhead the algorithms execute.

Furthermore, we analyze our LLRS and FS-LLRS schemes in comparison with others [8], [38], [4], [68], [20]. The comparative analysis of the computational overhead for three types of parameter settings is shown in Figure 4.5, 4.6, and 4.7, respectively. It is clear that the overhead increases as the value of n grows.

In our analysis, we use $n = 128$ as an example. The trends for $n = 80$ and $n = 256$ closely resemble those for $n = 128$, differing primarily in time cost. The overhead of KeyGen algorithm is depicted in Figure 4.6(a), showing that the time cost for a signing user to generate its secret-public key pair remains constant as the ring size increases. Figure 4.6(b) and 4.6(c) illustrate that the overhead of Sign and Verify

Table 4.2: Comparison of Communication Overhead with Existing Lattice-based Ring Signatures..

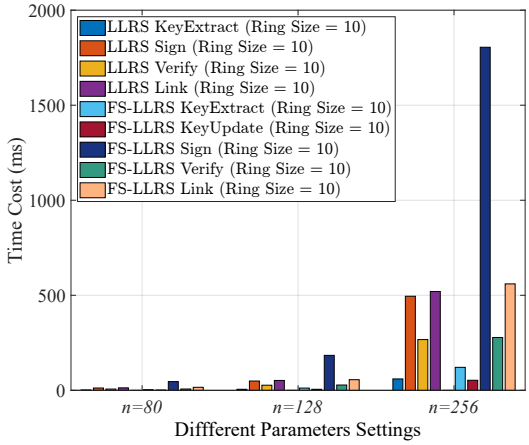
Schemes	User Key Size		Communication Size	
	Public Key	Secret Key	Signature	Link Tag
Baum et al. [8]	$n \log q$	$m \log q$	$(mN + n) \log q$	$n \log q$
Jia et al. [38]	$nm \log q$	$mk \log q$	$(mN + w) \log q$	-
Alberto et al. [4]	$n \log q$	$(m-1) \log q$	$(mN + n) \log q$	$1 \log q$
Tang et al. [68]	$n \log q$	$m \log q$	$((m+1)N + n) \log q$	$1 \log q$
Chen et al. [20]	$n \log q$	$m \log q$	$((m+1)N + n) \log q$	-
Our LLRS Scheme	$nm \log q$	$nn \log q$	$(nN + m) \log q$	$n \log q$
Our FS-LLRS Scheme	$nm \log q$	$mm \log q$	$(((l+1)N + 1)m) \log q$	$nm \log q$

algorithms increases as the ring size grows from 1 to 100, respectively. Our LLRS scheme's Sign algorithm demonstrates the lowest computational overhead, while the Sign algorithm of our FS-LLRS scheme exhibits similar overhead to Jia et al. [38] and still lower than most existing schemes [8], [4], [68], [20]. As for the Verify algorithm, the cost of our LLRS scheme is marginally lower than our FS-LLRS scheme, and both of them are lower than others [8], [38], [4], [68], [20].

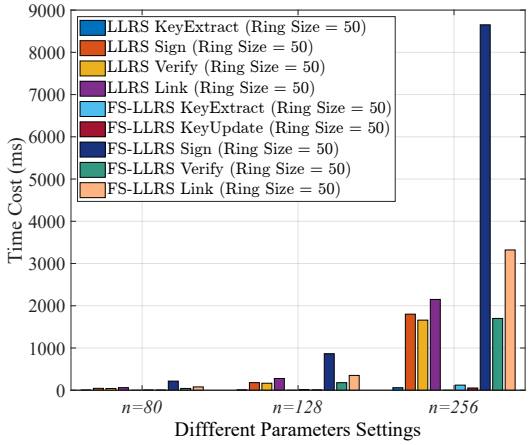
Additionally, we provide the time cost of the Sign and Verify algorithms when there are 100 ring users. When the parameter setting is the type I in Figure 4.5, the running times of Sign algorithm for our LLRS and our FS-LLRS scheme are 91.6ms and 426.1ms, respectively, while for other schemes are approximately 415.9ms to 513.75ms. The running times of the Verify algorithm for our LLRS and our FS-LLRS scheme are 85.5ms and 86.75ms, respectively, while for other schemes are approximately 125.74ms to 262.5ms.

Under the parameter setting of type II in Figure 4.6, our LLRS and FS-LLRS schemes have running times of 367.7ms, and 1707.3ms, respectively, while other schemes range from 1657.9ms to 2151.8ms, for the Sign algorithm. For the Verify algorithm, our LLRS and FS-LLRS schemes have running times of 342.1ms, and 357.6ms, respectively, while other schemes are approximately 503.2ms to 1050.1ms.

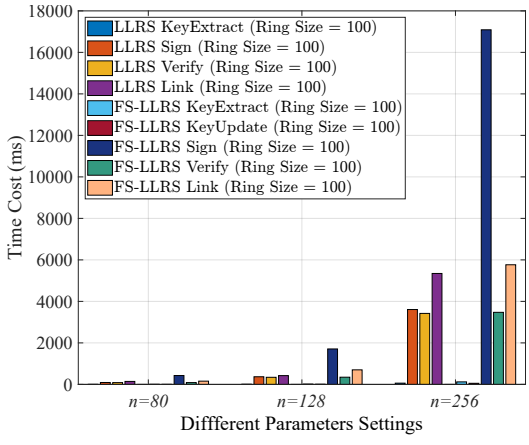
For type III in Figure 4.7, our LLRS and FS-LLRS schemes exhibit running times of 3606.9ms and 17088.0ms, respectively, for the Sign algorithm, while other schemes range from 16577.9ms to 21516.7ms. With regard to Verify algorithm, our LLRS and FS-LLRS schemes demonstrate running times of 3411.3ms and 3493.2ms, respectively, while other schemes range from 5013.1ms to 10509.4ms.



(a) Overhead of Ring Size is 10.

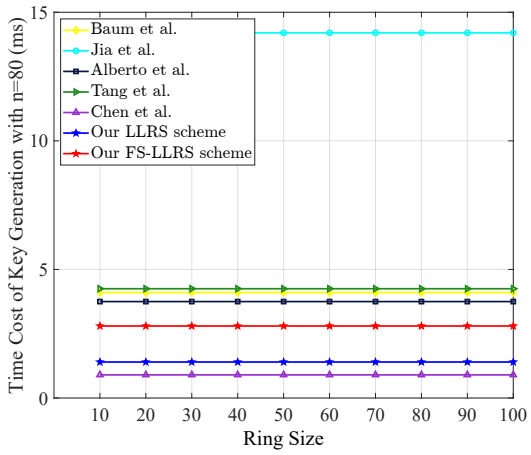


(b) Overhead of Ring Size is 50.

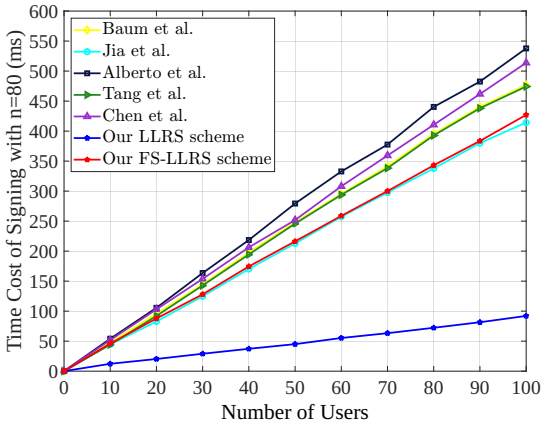


(c) Overhead of Ring Size is 100.

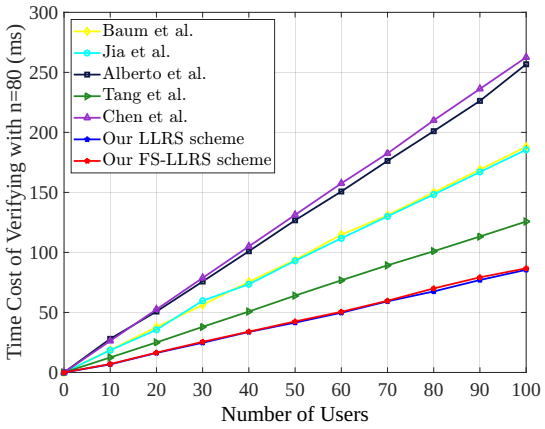
Figure 4.4: The Computation Overhead Associated with Several Algorithms of Our LLRS and FS-LLRS Schemes with Different Parameters Settings.



(a) Overhead of **KeyExtract**.

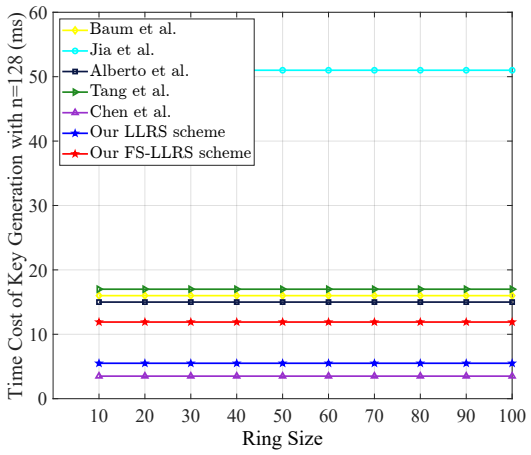


(b) Overhead of **Sign**.

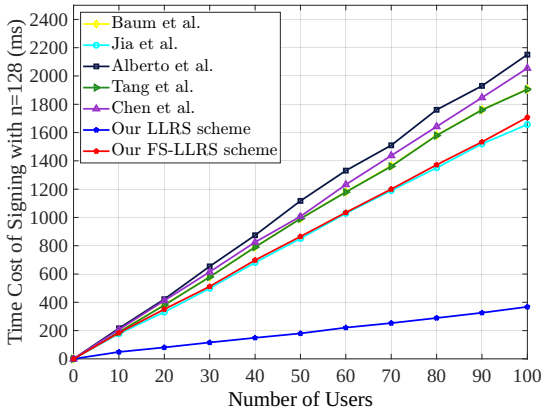


(c) Overhead of **Verify**.

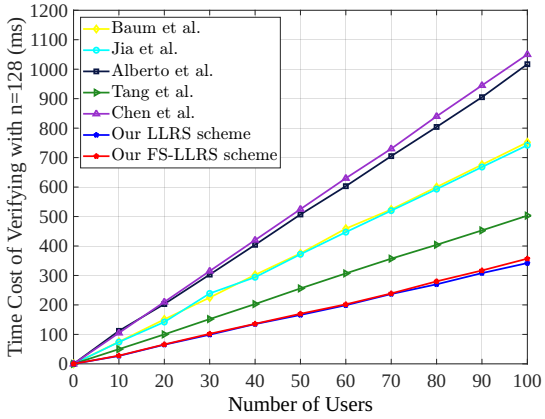
Figure 4.5: Comparison of Computation Overhead Associated with Several Algorithms between Our LLRS and FS-LLRS Schemes, and Current State-of-the-Art Ring Signature Primitives [8], [38], [4], [68], [20] with $n = 80$.



(a) Overhead of **KeyExtract**.

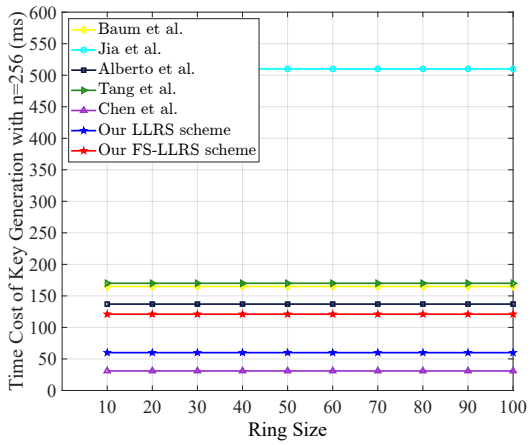


(b) Overhead of **Sign**.

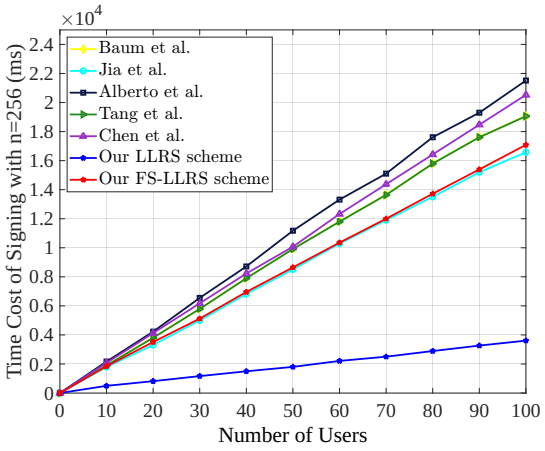


(c) Overhead of **Verify**.

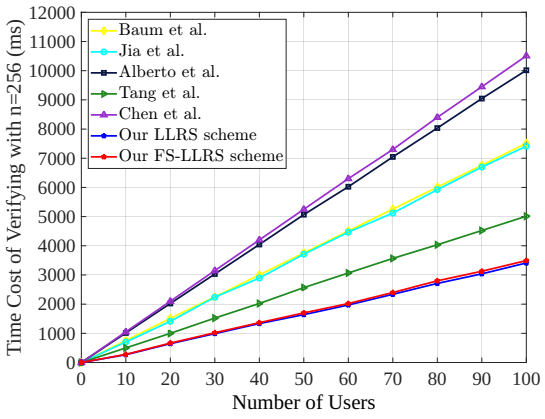
Figure 4.6: Comparison of Computation Overhead Associated with Several Algorithms between Our LLRS and FS-LLRS Schemes, and Current State-of-the-Art Ring Signature Primitives [8], [38], [4], [68], [20] with $n = 128$.



(a) Overhead of **KeyExtract**.



(b) Overhead of **Sign**.



(c) Overhead of **Verify**.

Figure 4.7: Comparison of Computation Overhead Associated with Several Algorithms between Our LLRS and FS-LLRS Schemes, and Current State-of-the-Art Ring Signature Primitives [8], [38], [4], [68], [20] with $n = 256$.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, our first contribution is to propose a general ring signature construction that provides a conversion method from Σ-based signatures to ring signatures without reducing the security level, shown in Chapter 3. To begin with, we present a general model for generalizing the existing signature schemes in the form of a Σ protocol. Leveraging this general model, we then propose a generic construction that can transform the existing signatures to ring signatures, utilizing our redesigned one-out-of-many relation and Fiat-Shamir transformation technique. Further, to enhance the efficiency of the ring signature, we implement the Bulletproofs folding technique on our ring signature to achieve the efficient logarithmic-size ring signature.

To showcase the practicality of our generic scheme, we conduct four case studies using our generic construction technique to convert the Schnorr signature, ECDSA signature, EdDSA signature, and SM2 signature into ring signature schemes, respectively. Further, we perform security analysis for our construction, showing that it satisfies correctness, anonymity, and unforgeability. The performance evaluation demonstrates that our scheme incurs only a minor additional overhead when constructing the orig-

inal signature into a ring signature. The comparative analysis indicates that our scheme is superior to most existing schemes in computation overhead and outperforms other existing ring signatures in communication overhead.

The second contribution is shown in Chapter 4, where we propose two lattice-based ring signature schemes for secure cloud-assisted EMRs. We first introduce an efficient lattice-based linkable ring signature (LLRS) to assure patient privacy and EMR security, and check whether multiple signatures are from the same signer. Then, we propose the FS-LLRS scheme, an enhancement of LLRS, incorporating periodically secret key updates for forward-secure unforgeability. In our system, EMRs created by patients and doctors undergo a signing process by multiple parties using either LLRS or FS-LLRS schemes. These signed EMRs are then securely stored in the cloud, ensuring EMRs unforgeability and user anonymity. Requesters can download the EMRs from the cloud to verify their validity and linkability. Compared to the existing solutions, our LLRS scheme offers superior efficiency, while our FS-LLRS provides enhanced security.

5.2 Future Work

In this thesis, we first propose a general construction based on DL, converting the Σ-protocol based signatures to ring signatures. The scheme achieves optimal storage overhead until now and relatively efficient computational costs. However, our scheme is based on the Random Oracle Model (ROM). More importantly, it relies on the DL assumption and is not resistant to quantum attacks, which means the security level is not ideal. Therefore, in future work, our goal is to improve the proposed general ring signature from the ROM to the standard model. Furthermore, we aim to propose a lattice-based general construction to transform Σ-protocol based signatures into lattice-based ring signatures.

For the second work, we present a lattice-based linkable ring signature with forward security for cloud-assisted EMR sharing. The scheme is multi-functional, meeting the requirements of anonymity, linkability, and unforgeability with forward security, while also demonstrating resistance against quantum attacks. However, there is still space for improvement the efficiency. Specifically, our scheme outperforms and is on par with some existing schemes' computational overhead. Moreover, our FS-LLRS scheme has higher storage overhead compared to other schemes, due to the additional implementation of forward security. In future work, we wish to achieve forward security in lattice-based ring signatures without using the GenSamplePre lattice basis algorithm and binary tree structure, which will require the design of a completely novel method for implementing forward security in lattice-based ring signatures.

References

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 87(1):131–140, 2004.

[2] Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, et al. Quantum algorithm implementations for beginners. *arXiv preprint arXiv:1804.03719*, 2018.

[3] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

[4] Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1. 0). In *Information Security and Privacy: 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings 23*. Springer, 2018.

[5] Liantao Bai, Yuegong Zhang, and Guoqiang Yang. Sm2 cryptographic algorithm based on discrete logarithm problem and prospect. In *2012 2nd International*

Conference on Consumer Electronics, Communications and Networks (CECNet), pages 1294–1297. IEEE, 2012.

[6] Yangyang Bao, Weidong Qiu, Peng Tang, and Xiaochun Cheng. Efficient, revocable, and privacy-preserving fine-grained data sharing with keyword search for the cloud-assisted medical iot system. *IEEE Journal of Biomedical and Health Informatics*, 26(5):2041–2051, 2021.

[7] Zijian Bao, Debiao He, Huaqun Wang, Min Luo, and Cong Peng. A group signature scheme with selective linkability and traceability for blockchain-based data sharing systems in e-health services. *IEEE Internet of Things Journal*, 2023.

[8] Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. In *International Conference on Information and Communications Security*, pages 303–322. Springer, 2018.

[9] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, USA, March 4-7, 2006. Proceedings 3*, pages 60–79. Springer, 2006.

[10] Rikke Bendlin. Lattice-based cryptography: Threshold protocols and multiparty computation. 2013.

[11] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.5*, 2020.

[12] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on ddh. In *European Symposium on Research in Computer Security*, pages 243–265. Springer, 2015.

[13] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *International workshop on public key cryptography*, pages 499–517. Springer, 2010.

[14] Xavier Boyen and Thomas Haines. Forward-secure linkable ring signatures. In *Information Security and Privacy: 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings 23*, pages 245–264. Springer, 2018.

[15] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[16] Yibo Cao, Shiyuan Xu, Xue Chen, Yunhua He, and Shuo Jiang. A forward-secure and efficient authentication protocol through lattice-based group signature in vanets scenarios. *Computer Networks*, 214:109149, 2022.

[17] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25, 2012.

[18] Pavlos Ioannis Pyrros Chaidos. *Zero Knowledge Protocols and Applications*. PhD thesis, UCL (University College London), 2017.

[19] Hsuan-Yu Chen, Zhen-Yu Wu, Tzer-Long Chen, Yao-Min Huang, and Chia-Hui Liu. Security privacy and policy for cryptographic based electronic medical information system. *Sensors*, 21(3):713, 2021.

[20] Xue Chen, Shiyuan Xu, Yibo Cao, Yunhua He, and Ke Xiao. Aqrs: Anti-quantum ring signature scheme for secure epidemic control with blockchain. *Computer Networks*, 224:109595, 2023.

[21] Xue Chen, Shiyuan Xu, Yunhua He, Yu Cui, Jiahuan He, and Shang Gao. Lfs-as: lightweight forward secure aggregate signature for e-health scenarios. In

ICC 2022-IEEE International Conference on Communications, pages 1239–1244. IEEE, 2022.

[22] Xue Chen, Shiyuan Xu, Tao Qin, Yu Cui, Shang Gao, and Weimin Kong. Aq–abs: Anti-quantum attribute-based signature for emrs sharing with blockchain. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1176–1181. IEEE, 2022.

[23] Sherman SM Chow, Victor K Wei, Joseph K Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 297–302, 2006.

[24] Sherman SM Chow, Siu-Ming Yiu, and Lucas CK Hui. Efficient identity based ring signature. In *Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings 3*, pages 499–512. Springer, 2005.

[25] Isaac L Chuang, Lieven MK Vandersypen, Xinlan Zhou, Debbie W Leung, and Seth Lloyd. Experimental realization of a quantum algorithm. *Nature*, 393(6681):143–146, 1998.

[26] Narendra K Dewangan, Preeti Chandrakar, Saru Kumari, and Joel JPC Rodrigues. Enhanced privacy-preserving in student certificate management in blockchain and interplanetary file system. *Multimedia Tools and Applications*, 82(8):12595–12614, 2023.

[27] Odai Enaizan, Ahmad A Zaidan, NHM Alwi, Bulat B Zaidan, Mohammed Assim Alsalem, OS Albahri, and AS Albahri. Electronic medical record systems: Decision support examination framework for individual, security and privacy concerns using multi-perspective analysis. *Health and Technology*, 10:795–822, 2020.

[28] Liu Feng, Yang Jie, Kong Deli, and Qi Jiayin. A secure multi-party computation protocol combines pederson commitment with schnorr signature for blockchain. In *2020 IEEE 20th International Conference on Communication Technology (ICCT)*, pages 57–63. IEEE, 2020.

[29] Mengqi Feng, Chao Lin, Wei Wu, and Debiao He. Sm2-dualring: Efficient sm2-based ring signature schemes with logarithmic size. *Computer Standards & Interfaces*, 87:103763, 2024.

[30] Behrouz A Forouzan and Debdeep Mukhopadhyay. *Cryptography and network security*, volume 12. Mc Graw Hill Education (India) Private Limited New York, NY, USA:, 2015.

[31] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[32] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. *Summer course "Cryptography and computer security" at MIT*, 1999:1999, 1996.

[33] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015.

[34] Bharti Grover and Dileep Kumar Kushwaha. Authorization and privacy preservation in cloud-based distributed ehr system using blockchain technology and anonymous digital ring signature. *Health Services and Outcomes Research Methodology*, 23(2):227–240, 2023.

[35] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[36] Mark Horowitz and Emily Grumbling. Quantum computing: progress and prospects. 2019.

[37] Abdelkrim Imghoure, Ahmed El-Yahyaoui, and Fouzia Omary. Ecdsa-based certificateless conditional privacy-preserving authentication scheme in vehicular ad hoc network. *Vehicular Communications*, 37:100504, 2022.

[38] X Jia, D He, Z Xu, and Q Liu. An efficient identity-based ring signature scheme over a lattice. *Journal of Cryptologic Research*, 4(04):392–404, 2017.

[39] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001.

[40] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (eddsa). Technical report, 2017.

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

[42] I Keshta and A Odeh. Security and privacy of electronic health records: Concerns and challenges. egyptian informatics journal, 22 (2), 177-183, 2021.

[43] Raman Kumar. Cryptanalytic performance appraisal of improved hll, kuochen, gengvrf, fengvrf secure signature with tkip digital workspaces: for financial cryptography. *Wireless Personal Communications*, 115(2):1541–1563, 2020.

[44] Chengzhe Lai, Zhe Ma, Rui Guo, and Dong Zheng. Secure medical data sharing scheme based on traceable ring signature and blockchain. *Peer-to-Peer Networking and Applications*, 15(3):1562–1576, 2022.

[45] Huy Quoc Le, Dung Hoang Duong, Willy Susilo, Ha Thanh Nguyen Tran, Viet Cuong Trinh, Josef Pieprzyk, and Thomas Plantard. Lattice blind signatures with forward security. In *Information Security and Privacy: 25th*

Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30–December 2, 2020, Proceedings 25, pages 3–22. Springer, 2020.

[46] Nan Li, Yingjiu Li, Atsuko Miyaji, Yangguang Tian, and Tsz Hon Yuen. A practical forward-secure dualring. In *International Conference on Cryptology and Network Security*, pages 516–537. Springer, 2023.

[47] Han Liu, Dezhi Han, Mingming Cui, Kuan-Ching Li, Alireza Souri, and Mohammad Shojafar. Idenmultisig: identity-based decentralized multi-signature in internet of things. *IEEE Transactions on Computational Social Systems*, 2023.

[48] Joseph K Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):157–165, 2013.

[49] Joseph K Liu, Tsz Hon Yuen, and Jianying Zhou. Forward secure ring signature without random oracles. In *Information and Communications Security: 13th International Conference, ICICS 2011, Beijing, China, November 23-26, 2011. Proceedings 13*, pages 1–14. Springer, 2011.

[50] Aoran Lu, Weihai Li, Yuanzhi Yao, and Nenghai Yu. Tcabrs: An efficient traceable constant-size attribute-based ring signature scheme for electronic health record system. In *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*, pages 106–113. IEEE, 2021.

[51] Nishay Madhani, Vikrant Gajria, and Pratik Kanani. Distributed and anonymous e-voting using blockchain and ring signatures. In *Communication and Intelligent Systems: Proceedings of ICCIS 2020*, pages 839–854. Springer, 2021.

[52] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Eurocrypt*, volume 7237. Springer, 2012.

[53] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[54] Zhu Na and Xiao Guo Xi. The application of a scheme of digital signature in electronic government. In *2008 International Conference on Computer Science and Software Engineering*, volume 3, pages 618–621. IEEE, 2008.

[55] Senshan Ouyang, Xiang Liu, Lei Liu, Shangchao Wang, Baichuan Shao, and Yang Zhao. An efficient and provably secure sm2 key-insulated signature scheme for industrial internet of things. *CMES-Computer Modeling in Engineering & Sciences*, 138(1), 2024.

[56] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In *Annual Cryptology Conference*, pages 80–97. Springer, 2010.

[57] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International conference on the theory and applications of cryptographic techniques*, pages 387–398. Springer, 1996.

[58] Kalkundri Ravi and S. A. Kulkarni. A secure message authentication scheme for vanet using ecdsa. In *Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–6, 2013.

[59] Jin Ren, Yuyang Cheng, and Shiyuan Xu. Edppa: An efficient distance-based privacy preserving authentication protocol in vanet. *Peer-to-Peer Networking and Applications*, 15(3):1385–1397, 2022.

[60] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 552–565. Springer, 2001.

[61] Yannick Seurin. On the exact security of schnorr-type signatures in the random oracle model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 554–571. Springer, 2012.

[62] Yang Shi, Junqing Liang, Mianhong Li, Tianchen Ma, Guodong Ye, Jiangfeng Li, and Qinpei Zhao. Threshold eddsa signature for blockchain-based decentralized finance applications. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 129–142, 2022.

[63] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[64] Shor, Peter W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[65] Edward H Shortliffe. The evolution of electronic medical records. *Academic Medicine*, 74(4):414–9, 1999.

[66] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, 1998.

[67] Yingnan Sun, Frank P-W Lo, and Benny Lo. Security and privacy for the internet of medical things enabled healthcare systems: A survey. *IEEE Access*, 7:183339–183355, 2019.

[68] Yongli Tang, Feifei Xia, Qing Ye, Yongjun Wang, and Xiaohang Zhang. Identity-based linkable ring signature on lattice. *Journal of Cryptologic Research*, 8(2):232–247, 2021.

[69] C-K Wang. Security and privacy of personal health record, electronic medical record and health information. *Problems and perspectives in management*, (13, Iss. 4):19–26, 2015.

[70] Gang Xu, Yibo Cao, Shiyuan Xu, Ke Xiao, Xin Liu, Xiubo Chen, and Mianxiong Dong. A novel post-quantum blind signature for log system in blockchain. *Comput. Syst. Sci. Eng.*, 41(3):945–958, 2022.

[71] Shiyuan Xu, Yibo Cao, Xue Chen, Yanmin Zhao, and Siu-Ming Yiu. Post-quantum public-key authenticated searchable encryption with forward security: General construction, and applications. In *International Conference on Information Security and Cryptology*, pages 274–298. Springer, 2023.

[72] Shiyuan Xu, Xue Chen, Chao Wang, Yunhua He, Ke Xiao, and Yibo Cao. A lattice-based ring signature scheme to secure automated valet parking. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 70–83. Springer, 2021.

[73] Xiaoling Yu and Yuntao Wang. Forward secure lattice-based ring signature scheme in the standard model. In *International Conference on Information and Communications Security*, pages 146–158. Springer, 2023.

[74] Tsz Hon Yuen, Muhammed F Esgin, Joseph K Liu, Man Ho Au, and Zhimin Ding. Dualring: generic construction of ring signatures with efficient instantiations. In *Annual International Cryptology Conference*, pages 251–281. Springer, 2021.

[75] Shengke Zeng, Yuan Huang, and Xingwei Liu. Privacy-preserving communication for vanets with conditionally anonymous ring signature. *International Journal of Network Security*, 17(2):135–141, 2015.