From Σ-protocol based Signatures to Ring Signatures: General Construction and Applications

Xue Chen, *Student Member, IEEE,* Shang Gao, *Member, IEEE,* Shiyuan Xu, *Student Member, IEEE,* Liquan Chen, *Senior Member, IEEE,* Siu-Ming Yiu, *Member, IEEE,* and Bin Xiao, *Fellow, IEEE*

***Abstract*—Public Key Infrastructure (PKI) has gained widespread attention for ensuring the security and integrity of data communication. While existing PKI mainly supports digital signatures, it is lacking in crucial anonymity, leading to the leakage of a signer's identity information. To alleviate the issue, ring signatures are a suitable choice to provide anonymity as they allow users to create their own rings without the need for an administrator. Unfortunately, the utilization of ring signatures in PKI may present compatibility challenges within the system. Thus, proposing a general mechanism to convert a standardized Σ-based signature to a ring signature is far-reaching.**

In this paper, we propose a general construction for converting Σ-based signatures into ring signatures. To achieve this, we first introduce a Σ-based general model, providing a general 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 ring signatures leveraging on one-out-of-many proofs. Furthermore, to reduce the signature size, we employ the Bulletproofs folding technique, enabling the attainment of logarithmic size ring signatures. To demonstrate the wide applicability of our general construction, we present four prominent signatures as case studies. Ultimately, we conduct a rigorous security analysis and benchmark experimental evaluation. The signing and verification times are 0.44 to 0.97 times and 0.27 to 0.91 times compared to other state-of-the-art schemes, respectively. Additionally, we exhibit the lowest signature size to date.

***Index Terms*—Sigma Protocol, One-out-of-Many Proofs, Ring Signature, General Construction.**

I. Introduction

Digital signatures [1] are a vital component of modern cryptography [2]. They have been serving as an essential tool in ensuring computer security [3] and the authenticity, integrity, and unforgeability of transmitted data. Signatures possessing diverse properties, are prominently utilized in vehicular networks [4], [5], government announcements [6], electronic medical record sharing [7], financial sector [8], the blockchain-based system [9], [10], and other aspects. Nowadays, various Σ-based standard signatures are in the spotlight due to their succinct structure, e.g., Schnorr signatures [11], [12] and ECDSA signatures [13], etc.

Public Key Infrastructure (PKI) [14], [15] is popular for its robust ability to enhance security by providing strong authentication, data integrity, and confidentiality in digital communications. Existing PKI mainly supports digital signatures, such as Σ-based standard signatures [16], which are succinct and can guarantee data security, but without considering the crucial user anonymity. For example, for the anonymous e-voting scenario [17], under the PKI using existing digital signatures cannot fulfill the anonymity requirement, which cannot establish a credible and fair election process. Missing anonymity can lead to the leakage of users' private information and compromise the equitability of systems.

To tackle the concern about anonymity, scholars have conducted a large number of studies and found that the ring signature [18]–[20] with strong anonymity and unforgeability is a feasible choice [21], [22]. It ensures the anonymity of a signer within a dynamically formed ring and requires the signing process not to reveal the signatory's identity, in which each individual involved possesses a pair of keys, comprising a secret key (aka. signing key) and a public key (aka. verification key). In 2001, Rivest et al. [18] originally presented the concept of ring signature, which enables any member within a ring to sign messages on behalf of the entire ring while effectively concealing the identity. In 2004, Abe et al. [23] proposed a novel method for constructing a general ring signature, that allows for the use of any hash-and-sign signature or Σ-protocol based signature. The compelling attributes of ring signatures have garnered substantial scholarly interest in the field of data security and user privacy.

Consistent with this trend, numerous ring signatures have been proposed [24]–[26]. Unfortunately, if we replace digital signatures with these existing ring signatures, the existing PKI system may not be compatible and friendly to support. Also, existing ring signatures may lose advantages that the standard signature offered [11], [12], such as lower time and signature size. Moreover, 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 transformation from existing (standardized) signatures to

This work was supported in part by HK RGC GRF under Grants PolyU 15205624/QCFD, 15202123/QC1V, 15207522/Q93W, 15216721/Q86A, NSFC Youth 62302418/ZGJV, HKU-SCF FinTech Academy, Shenzhen-Hong Kong-Macao Science and Technology Plan Project (Category C Project: SGDX20210823103537030), and Theme-based Research Scheme T35-710/20-R. (*Corresponding author*: Dr. Shang Gao)

Xue Chen and Bin Xiao are with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong. (E-mail: xue-serena.chen@connect.polyu.hk, b.xiao@polyu.edu.hk).

Shang Gao is with the Department of Computing and the School of Accounting and Finance at the Hong Kong Polytechnic University, Hung Hom, Hong Kong. (Email: shang-jason.gao@polyu.edu.hk)

Xue Chen, Shiyuan Xu and Siu-Ming Yiu are with the Department of Computer Science, School of Computing and Data Science, The University of Hong Kong, Pok Fu Lam, Hong Kong. (E-mail: syxu2@cs.hku.hk, smyiu@cs.hku.hk).

Liquan Chen is with the School of Cyber Science and Engineering, Southeast University, China. (E-mail: lqchen@seu.edu.cn).

ring signatures. By employing this transformation technique, scholars can directly enhance the security level by converting a Σ-based signature into a ring signature, which is friendly and compatible with PKI. But to our best knowledge, there is no established general construction to realize this transformation till now.

A. Our Motivation

Although several signatures based on Σ-protocols have been proposed, there is no general model to summarize these protocols, leading the analysis of these signatures to be difficult and their extension generality to be reduced. Moreover, the current ring signatures are not standardized by large formal organizations, which increases the problem of adaptability in ring signature scenarios. Further, the existing ring signatures may lose some advantages of the Σ-based signatures, such as low time cost and signature size. More importantly, existing PKI mainly supports signatures, they are not suitable for use in some anonymous scenarios, such as anonymous voting or anonymous payment. In these cases, the use of existing ring signatures in PKI can lead to incompatibility and unsupported problems, which result in replacing the PKI or being unable to provide the necessary anonymity.

To solve the bottlenecks we mentioned above, we are mainly engaged in proposing a general ring signature construction that provides a transformation method from Σ-based signatures to ring signatures without reducing the security level and friendly compatible with the PKI. With our general transformation framework, scholars can directly transform the Σ-based signature into the ring signature. In this way, the system can friendly provide user anonymity in PKI and does not lose the advantages of the Σ-based signatures.

We further adopt the Bulletproofs folding technique to reduce the signature size of our proposed ring signature from $O(n)$ to $O(\log n)$. We emphasize that our major goal is to present a general transformation construction with better performance rather than construct the much more efficient ring signature.

We focus on researching ring signatures, rather than other primitives like group signatures. This is because ring signatures are suitable for most anonymity scenarios and provide higher anonymity than group signatures (the group signatures require the group manager). They allow users to create their own rings without the need for an administrator, making it impossible to trace the true identity of the signer.

Our proposed transformation framework is general, which can be applied to several different Σ-protocol based primitives by setting different parameters, as showcased in section VI.

B. Our Contributions

The primary objective of this paper is to propose a general construction that transforms Σ-protocol-based signatures into ring signatures. Meanwhile, the proposed scheme needs to ensure both efficiency and security. Specifically, we require 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 summarize our fivefold contributions as follows:

- 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 general 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 help of the one-out-of-many proofs and the Fiat-Shamir heuristic, we propose a generalized construction that transforms the existing signature schemes into ring signature schemes.
- To further enhance the signature size of our general ring signature, we adopt the Bulletproofs folding technique to our general ring signature, which reduces the size of the ring signature from $O(n)$ to $O(\log n)$, where n is the size of the ring.
- To demonstrate the high universality of our proposed 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 theoretical security analysis[1] is conducted to show the security of our construction. Furthermore, we open source the implementation code of our scheme and perform a comparison with other ring signatures in time cost and signature size. 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, our scheme's signing and verification times are 0.44 to 0.97 times and 0.27 to 0.91 times compared to other state-of-the-art schemes. Additionally, the signature size of our scheme is $2\log n$, outperforming existing ring signatures.

C. Comparison with DualRing Construction

In this paper, we propose a general construction, which can transfer the Σ-based signatures to ring signatures. In 2021, Yuen et al. [27] also proposed a general construction of ring signatures, which is applicable to several Σ-protocols[2]. While our work is similar to the work of Yuen et al. [27], our scheme differs from theirs in terms of motivation, construction, and technique, as discussed below.

a) Motivation: Our motivation is different from theirs. Our work is engaged in proposing a standard general ring signature construction that provides a transformation method from Σ-protocol based signatures to ring signatures. Whereas, Yuen et al. [27] dedicated to proposing an entirely novel, non-standard general ring signature construction.

[1]Following prior research [24]–[27], we have focused on theoretical security analysis rather than real-world security performance. Evaluating real-world security performance requires a large workload and necessitates support from systems and equipment, making it a potential direction for our future research and an open problem.

[2]The general construction of ring signature proposed by Yuen et al. [27] can be built from several Σ-protocols based schemes, such as Schnorr identification.

TABLE I
COMPARISON WITH CURRENT STATE-OF-THE-ART $O(\log n)$-SIZE RING SIGNATURE SCHEMES.

Schemes	Elements in Signature		Signature Size (Bytes)					
	$\mathbb{Z}_q$	$\mathbb{G}$	$n=4$	$n=8$	$n=16$	$n=32$	$n=64$	$n=1024$
Groth et al. [24]	$3\log n+1$	$4\log n$	488	716	944	1172	1400	2312
Bootle et al. [25]	$1.5\log n+6$	$\log n+12$	750	831	912	993	1074	1398
Yuen et al. [27]	3	$2\log n+1$	261	327	393	459	525	789
Our Scheme	1	$2\log n+1$	197	263	329	395	461	725

b) Construction: For the construction, in our scheme, a ring signature consists of one challenge c and n responses $\mathbf{f}$. In contrast, the DualRing scheme [27] with two rings structure (namely DualRing), consists of n challenges $\mathbf{c}$ and one response f.

c) Improvement Technique: The use of Bulletproofs folding for further compression is a common method, such as DualRing [27], and we also improved the signature size of the ring signature through this method. Another potential alternative to improve the signature size of ring signatures is to use a binary Kronecker's delta vector. However, this approach is not as efficient and succinct as the Bulletproofs folding technique (which will be discussed in the section II).

Although both our scheme and the DualRing [27] use the variations of Bulletproofs folding to improve the ring signature size, however, there are still many differences. In our scheme, the relation $\prod_{i=1}^{n} {pk_i}^{f_i} = R \cdot G^c$ (for Schnorr case) is utilized to realize the Bulletproofs folding, where pk_i is the public key of user i, f_i is the response, c is the challenge, R is the commitment and G is the generator. While in DualRing [27], the optimization requires satisfying the relations $\prod_{i=1}^{n} {pk_i}^{c_i} = R \cdot (G^z)^{-1}$ (for Schnorr case) and $H(\mu \| R \| pk_i) = \sum_{i=1}^{n} {c_i}$[3], where μ is a message, pk_i is the public key of user i, z is the response, c_i is the challenge, R is the commitment and G is the generator. More importantly, the signature size and time cost of our scheme are better than the DualRing [27].

D. Technique Overview

In a ring signature, a signer can sign a message on behalf of a group of signers. With all public keys of the signers 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 a pair of 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 a one-out-of-many relation, a signer l needs to prove two constraints: 1) it has a secret key s_l which corresponds to a public key P_l; and 2) its public key P_l is in the public key list $\mathbf{P} = (P_1, \cdots, P_l, \cdots, P_n)$. 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 x-coordinate of the elliptic curve 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 exist 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 and secret keys 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 ($P_i = G^{s_i}$). 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 the assumption of commitment, e.g., Discrete Logarithm (DL) assumption for Pedersen commitment, 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 ring signature is linear to the ring size, which may be impractical for real-world applications.

To further improve our proposed ring signature, we employ the idea of folding from Bulletproofs to compress the ring signature size to logarithmic to ring size. Since Bulletproofs folding technique is highly efficient and is based on the same security assumption as the commitment scheme, further

[3]In Yuen et al.'s work [27], the relation is $H(\mu \| R \| pk_i) = \otimes_{i=1}^{n} c_i$ and $\otimes$ is the commutative group operation. In the DL setting, the group operation $\otimes$ means the modular addition.

improvement does not sacrifice the security of the original scheme.

E. Overview of Performance Results

We comparatively evaluate the signature size of our $O(\log n)$-size general ring signature with the logarithmic size schemes of Groth et al. [24], Bootle et al. [25] and Yuen et al. [27], shown in Table 1. The signature size of our scheme is significantly lower than the schemes of Groth et al. [24] and Bootle et al. [25], and slightly lower than Yuen et al.'s scheme [27]. Moreover, our experiments (with publicly available code) and comparative evaluations are performed for the time costs, as detailed in section VIII.

It is important to note that the improvement of our proposed ring signature scheme from $O(n)$ to $O(\log n)$ is attributed to the use of the Bulletproofs folding technique. Yuen et al. [27] also utilized the Bulletproofs folding technique to propose a $O(\log n)$ ring signature scheme. The slightly better size of our $O(\log n)$ ring signature compared to the $O(\log n)$ ring signature of Yuen et al. is a result of our framework design. In contrast to both Yuen et al. [27] and our approach, existing $O(\log n)$ schemes [24], [25], which are based on one-out-of-many proofs, utilize the binary Kronecker's delta vector to enhance the ring signatures.

Kronecker's delta vector and Bulletproofs are two methods (which are side-by-side) for achieving logarithmic size with $O(\log n)$ overhead to transform the relation (reduction of knowledge). Our scheme and Yuen et al. et al.'s scheme [27] use the Bulletproofs method, and Groth et al. [24], Bootle et al. [25] use Kronecker's delta vector. After using one of these two methods, it is not possible to use the other one, because this transformation already incurs $O(\log n)$ overhead.

II. Related Work

This section provides an overview of the existing literature and related works in the field.

A. Signature Schemes

From 1976 to 1986, several notable cryptographers [28]–[32] conducted research on signature schemes, formally providing definitions of signature schemes and primitives such as trapdoor functions, elliptic curve encryption, etc. Building on this foundation, researchers used the works mentioned above as the foundation to proceed study and construct a variety of signature schemes according to meet different requirements. Group signature [33], attribute-based signature [34], and ring signature are examples of such schemes. Furthermore, numerous existing succinct standard signatures are designed based on the Σ-protocols, such as ECDSA signature [13], Schnorr signature [11], [12], EdDSA signature [35] and SM2 signature [36].

B. Ring Signature Schemes

Since the seminal birth of the notion of ring signature [18], it has been widely studied by scholars due to its no administrator required and high anonymity features, which can be mainly classified into ring signatures based on trapdoor functions and ring signatures based on Σ-protocols. A general construction of a trapdoor-based ring signature was proposed by Rivest et al. [18] in 2001, using a trapdoor function that is easy to compute in one direction but very difficult to compute in the reverse direction. Similar schemes are also proposed by Lu et al. [37] in 2019, the raptor scheme, etc. While the ring signature based on Σ-protocol was originally introduced by Abe et al. [23] to construct the basic framework of one-out-of-many signature, which can use different types of keys to construct signatures based on integer decomposition and discrete logarithm problems for three move type schemes, such as schnorr signature, etc. For this structure, the DualRing scheme proposed by Yuen et al. [27] also used the three-move type to propose a double-ring structure.

In recent years, there has been a growing scholarly interest in ring signatures. In 2015, Groth et al. [24] introduced one-out-of-many proofs, utilizing the binary Kronecker's delta vector to present the ring signature scheme with $O(\log n)$ ring signature size and $O(n \log n)$ exponentiation costs. Subsequently, Bootle et al. [25] conducted optimizations on Groth et al.'s scheme [24] and introduced an accountable ring signature that exhibits similar performance to the aforementioned scheme. In 2021, Yuen et al. [27] used a variation of the Bulletproofs technique to present a ring signature with $O(\log n)$ size and $O(n)$ exponentiation cost, which is regarded as the optimal scheme till now. In 2024, Fang et al. [26] proposed an efficient SM2-based ring signature scheme with logarithmic size. The scheme of Feng et al. [26] can be regarded as a case study of Yuen et al.'s scheme [27].

C. Limitations of Existing Schemes

Despite the availability of various constructions for ring signatures, there is no standardized framework established by institutions, e.g., the National Natural Science Foundation of China (NSFC). Moreover, it is worth noting that existing signatures based on Σ-protocols supported by PKI unfortunately fail to provide user anonymity. The existing ring signatures may not be compatible with PKI and may lead to the system not providing the required anonymity in anonymous scenarios. Further, since ring signatures are not being standardized, both parties are still required to negotiate and communicate regarding certain realistic details. It leads to fewer general implementations and presents bottlenecks in the pursuit of enhancement to provide the required anonymity. Therefore, a viable solution to address these is to propose a transformation from existing signatures in various standards to ring signatures without reducing the security level.

III. Preliminaries

We now introduce the notions in our work and the fundamental knowledge toward to our design.

A. Notions

Let λ be a security parameter. Let $\mathbb{G}$ is the cyclic group order q, $\mathbb{Z}_q$ denotes the ring of integers modulo q, and $\mathbb{Z}_q^*$ denotes the $\mathbb{Z}_q \backslash \{0\}$. The uppercase letter $A \in \mathbb{G}$ and lowercase letter $a \in \mathbb{Z}_q$. The vectors can be present as bold letters $\mathbf{a}$ or $\mathbf{A}$. The $\mathbf{P^a}$ means $\prod P_i^{a_i}$. The Hadamard product $\mathbf{a} \circ \mathbf{b}$ denotes as $(a_1 \cdot b_1, \cdots, a_n \cdot b_n)$. The vector

$\mathbf{f}$ can be write as $\mathbf{f} = (f_1, \cdots, f_n)$ and the $\mathbf{f}^{-1}$ denotes as $\mathbf{f}^{-1} = ({f_1}^{-1}, \cdots, {f_n}^{-1})$. The vector division $\frac{\mathbf{a}}{\mathbf{b}}$ is expressed as $(\frac{a_1}{b_1}, \cdots, \frac{a_n}{b_n})$. Given a set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ means randomly sample the notation x form a set $\mathcal{S}$.

B. Σ-protocol

Σ-protocol is a 3-phase (commitment, challenge, and response) interactive protocol, used by a prover $\mathcal{P}$ to convince the verifier $\mathcal{V}$ that some statement is true without revealing its witness. It satisfies the properties of completeness, soundness, and Honest Verifier Zero-Knowledge (HVZK).

Specifically, the Σ-protocol is one of the zero-knowledge proofs between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, and also existing a probabilistic polynomial time setup algorithm $\mathcal{G}$. The tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is defined for the polynomial-time decidable relation $\mathcal{R}$, and for $(s, P) \subseteq \mathcal{R}$, s is referred to as a witness for P. The Σ protocol satisfies three properties, i.e. completeness, soundness, and Honest Verifier Zero-Knowledg (HVZK).

*Definition 1 (**Perfect completeness**):* The tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called the Σ-protocol, with the perfect completeness, if for all probabilistic polynomial time adversaries $\mathcal{A}$

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

where $\mathcal{A}$ outputs the (s, P) such that $(pp, s, P) \in \mathcal{R}$.

*Definition 2 (n-**special soundness**):* The tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is n-special soundness if there exists an efficient probabilistic polynomial time extractor $\mathcal{E}$ that has the ability to compute the witness s by given n accepting transcripts with the same initial message. Thus, for all probabilistic polynomial time adversaries $\mathcal{A}$

$$\Pr\left[(pp, s, P) \in \mathcal{R} \left| \begin{array}{c} pp \leftarrow \mathcal{G}(1^\lambda); \\ (P, \mu, c_1, f_1, \cdots, \\ c_n, f_n) \leftarrow \mathcal{A}(pp); \\ \mathcal{V}(pp, P, \mu, c_i, f_i) = 1, \\ \forall\ i \in [1, n]; \\ s \leftarrow \mathcal{E}(pp, P, \mu, \\ c_1, f_1, \cdots, c_n, f_n); \end{array} \right.\right] \approx 1 - \mathsf{negl}, \tag{2}$$

where the negl is negligible, and we say that the protocol is n-special soundness.

*Definition 3 (**Honest verifier zero-knowledge (HVZK)**):* The tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is an honest verifier zero-knowledge if there exists an efficient probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive probabilistic polynomial time adversaries $\mathcal{A}$

$$\left| \Pr\left[\mathcal{A}(\mu, f) = 1 \left| \begin{array}{c} pp \leftarrow \mathcal{G}(1^\lambda); \\ (s, P, c) \leftarrow \mathcal{A}(pp); \\ \mu \leftarrow \mathcal{P}(pp, s, P); \\ f \leftarrow \mathcal{P}(c); \end{array} \right.\right] \right.$$

$$\left. - \Pr\left[\mathcal{A}(\mu, f) = 1 \left| \begin{array}{c} pp \leftarrow \mathcal{G}(1^\lambda); \\ (s, P, c) \leftarrow \mathcal{A}(pp); \\ (\mu, f) \leftarrow \mathcal{S}(pp, P, c); \end{array} \right.\right] \right| \leq \mathsf{negl}, \tag{3}$$

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 the protocol is an honest verifier zero-knowledge.

C. Commitment Scheme

A commitment scheme is a protocol with the properties of hiding and binding, involving the sender and the receiver that allows a sender to construct a commitment $R = com(k)$ in the *commit* phase. In the *open* phase, the receiver can verify the opening and check whether the commitment R is constructed from a real value k. Taking Pedersen commitment as an example, in a general scenario, the group of large prime order q and the generator element G are obtained in the *setup* phase. In the *commit* phase, the sender computes the commitment $R = com(k) = G^k$. In the *open* phase, a receiver can perform validity verification. For signature, the public key P_i can be treated as a commitment concerning the secret key s_i.

D. Bulletproofs Folding

Bulletproofs [38] is a spatially efficient and succinct protocol, without a trusted setup phase. The core compression idea is dichotomy and recursion. We use it to fold our response value, which serves as a stepping stone toward our goal.

In our scheme, the central folding process is the relation $\mathbf{P}^{\mathbf{f}}$. In particular, we set n to be a power of 2 and $n' = \frac{n}{2}$. Then, we divide $\mathbf{f} = (f_1, \cdots, f_n)$ and $\mathbf{P} = (P_1, \cdots, P_n)$ into two vectors respectively, i.e. $\mathbf{f}_L = (f_1, \cdots, f_{n'})$, $\mathbf{f}_R = (f_{n'+1}, \cdots, f_n)$ and $\mathbf{P}_L = (P_1, \cdots, P_{n'})$, $\mathbf{P}_R = (P_{n'+1}, \cdots, P_n)$. Then, we compute two additional parameters $L_B = {\mathbf{P}_L}^{\mathbf{f}_R}$ and $R_B = {\mathbf{P}_R}^{\mathbf{f}_L}$. The process of compressing the size of folding value $\mathbf{f}$ can be computed 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 compressing logarithmic number rounds.

E. One-out-of-Many Proofs

One-out-of-many proofs is a notable approach to constructing a ring signature. In this, the prover $\mathcal{P}$ utilizes one-out-of-many proofs to prove that it possesses the secret key s_l corresponding to one of the public keys 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 reduced. However, this approach cannot perform further improvement by the Bulletproofs folding and the efficiency is lower than our method.

IV. Syntax and Security Models

This section presents syntax and security models of our ring signature, which are designed based on existing ring signatures.

A. Syntax

We formalize the syntax of our ring signature primitive, consisting of four algorithms, $\Pi = (\mathsf{RingSetup}, \mathsf{RingKeyGen}, \mathsf{RingSign}, \mathsf{RingVerify})$.

- $\mathsf{RingSetup}(\lambda)$: Given the security parameter λ, this algorithm outputs the public parameters pp.
- $\mathsf{RingKeyGen}(pp)$: Given the public parameters pp, this probabilistic polynomial time algorithm outputs a public-secret key pair (P_l, s_l) of signer l.
- $\mathsf{RingSign}(pp, \mathbf{P}, s_l, \mu)$: Given the secret key s_l of user l, a set of public keys $\mathbf{P}$ (constitutes the ring), and the message μ to be signed, this probabilistic polynomial time algorithm returns a signature σ on the message μ by the signer l.
- $\mathsf{RingVerify}(pp, \mathbf{P}, \mu, \sigma)$: Given the set of public keys $\mathbf{P}$, the message μ, and the signature σ, this deterministic polynomial time algorithm returns the 'accept' or the 'reject' by the verification process.

B. Security Models

The security models of our ring signature primitive encompass anonymity and unforgeability. The formal definitions with challenger and adversary are provided below.

a) Anonymity: We say that a ring signature $\Pi = (\mathsf{RingSetup}, \mathsf{RingKeyGen}, \mathsf{RingSign}, \mathsf{RingVerify})$ is anonymous, if for every polynomial n and PPT adversary $\mathcal{A}$, it holds that $\mathcal{A}$ has at most negligible advantage in the anonymity game. We use a challenger $\mathcal{C}$ to define our game as follows: $\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A})$:

1) **Setup.** $\mathcal{C}$ executes $\mathsf{RingSetup}$ algorithm to generate the public parameters pp and then delivers pp to $\mathcal{A}$.
2) **Queries.** $\mathcal{A}$ performs a polynomial bounded number n queries in an adaptive manner:
   - Random Oracle $\mathcal{O}_H$: We model the hash function H as the oracle and maintains 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 $\mathcal{A}$.
   - Random Oracle $\mathcal{O}_F$: We model the function F as the oracle and maintains the list L_F as empty initially. If the input has been queried before, $\mathcal{C}$ checks the list L_F, 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 user $i \in [1, n]$, and the $\mathcal{O}_{RO}$ can only be queried once of each i. Then, $\mathcal{C}$ invokes $\mathsf{RingKeyGen}$ algorithm to get public-secret keys (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 of user $i \in [1, n]$. $\mathcal{C}$ 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 $\mathsf{RingSign}$ algorithm.
3) **Challenge.** In this phase, $\mathcal{A}$ chooses a message μ, a set of public keys $\mathbf{P}$, and send that to $\mathcal{C}$. Note that each element in $\mathbf{P}$ must be chosen from the public keys generated by registration queries. Then, $\mathcal{C}$ randomly selects a $\pi \in [1, n]$ to generate the signature σ_π and sends it to $\mathcal{A}$.
4) **Guess.** $\mathcal{A}$ outputs a guess $\pi^* \in [1, n]$. If $\pi^* = \pi$, $\mathcal{C}$ outputs 1; otherwise outputs 0.

The advantage for $\mathcal{A}$ in attacking the anonymity game is defined as $\mathsf{Adv}_{\mathrm{Anony}}(\mathcal{A}) := |\Pr[\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A}) = 1] - \frac{1}{n}|$.

b) Unforgeability: We say that a ring signature $\Pi = (\mathsf{RingSetup}, \mathsf{RingKeyGen}, \mathsf{RingSign}, \mathsf{RingVerify})$ is unforgeable, if for every polynomial n and PPT adversary $\mathcal{A}$, it holds that $\mathcal{A}$ has at most negligible advantage in the unforgeability game. We use a challenger $\mathcal{C}$ to define our game as follows: $\mathsf{Game}_{\mathrm{Forge}}(\mathcal{A})$:

1) **Setup.** $\mathcal{C}$ picks a target index $i^* \in [1, n]$. Then, $\mathcal{C}$ sends the public parameters pp to $\mathcal{A}$.
2) **Queries.** $\mathcal{A}$ performs a polynomial bounded number n queries in an adaptive manner:
   - Random Oracle $\mathcal{O}_H$: Similar to $\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A})$.
   - Random Oracle $\mathcal{O}_F$: Similar to $\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A})$.
   - Registration Oracle $\mathcal{O}_{RO}$: Similar to $\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A})$.
   - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ quires the secret key s_i for user $i \in [1, n], i \neq i^*$ (if $i = i^*$, the query is aborted), $\mathcal{C}$ returns s_i to $\mathcal{A}$.
   - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the signature σ for user $i \in [1, n]$. If $P_i \notin \mathbf{P}$, the query is aborted. Otherwise, $\mathcal{C}$ returns σ to $\mathcal{A}$ by invoking $\mathsf{RingSign}$ algorithm (for $i \neq i^*$) or simulator $\mathcal{S}$ (for $i = i^*$).
3) **Forge.** $\mathcal{A}$ outputs a forgery signature σ^*, and the message μ^*, the public keys set $\mathbf{P}^*$ including P_{i^*}. Note that each element in $\mathbf{P}^*$ must be chosen from the public keys generated by registration queries. If it holds the following requirements, $\mathcal{C}$ outputs 1; otherwise outputs 0.
   - The message μ^* of forgery signature σ^* has not queried in $\mathcal{O}_{SO}$;
   - Inputs σ^* into $\mathsf{RingVerify}$ algorithm, which returns 'accept';
   - $\mathcal{A}$ does not make $\mathcal{O}_{CO}$ queries corresponding to any public keys in $\mathbf{P}^*$;

The advantage for $\mathcal{A}$ in attacking unforgeability game is defined as $\mathsf{Adv}_{\mathrm{Forge}}(\mathcal{A}) := \Pr[\mathsf{Game}_{\mathrm{Forge}}(\mathcal{A}) = 1]$.

V. 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 scheme with the help of one-out-of-many proofs and Fiat–Shamir heuristic. To further reduce the signature size, we employ the Bulletproofs folding technique. Ultimately, we formally propose a ring signature scheme with a logarithmic size in general.

Note that our general model and general 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.

A. General Model of Signature Scheme

Existing signature schemes consists of four algorithms, i.e. 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.

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

Through the Fiat–Shamir heuristic technique, we can transform the above protocol into a signature. However, as we intend to cover all existing signature schemes based on the Σ-protocol, we 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 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 shown in Fig. 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 on the construction of the signature, and the setting of 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 that to a function $F(\cdot)$, which is able to transform the commitment R to obtain the part information e as $e \leftarrow F(R)$. The function F can be a hash function, taking the x-coordinate of a point in an elliptic curve, and so on. The computation method of the function F is publicly available, and we require it to be a collision-resistant function.

B. General Model with One-out-of-Many Proofs Relation

Firstly, we review the role of the one-out-of-many proofs. By using 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 corresponding to the public key P_l. Specifically, for 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 into a ring signature scheme. Concretely, the relation of the secret key and public key 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 $\mathbf{P}^{\mathbf{s}} = G$, where no information of l.

We show the interactive process of the general model adding our redesigned relation of one-out-of-many proofs as in Fig. 2. Note that a, a' are the predefined value, and parameters t_0, t_0', t_2, t_2' cannot be 0 simultaneously. Besides, taking the vector to its inverse means pointwise inversion.

Theorem 1: In our scheme, the knowledge of s_l and ${s_l}^{-1}$ can be reduced from each other[4], which means:

- Given s_l, it can compute $s_l \cdot {s_l}^{-1} = 1$;
- Given ${s_l}^{-1}$, it can compute $({s_l}^{-1})^{-1} = s_l$.

By the Fiat-Shamir heuristic 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, and 4, respectively.

Algorithm 1 RingSetup(λ)

1: Chooses the parameters q, n;
2: Chooses the generator G;
3: Defines a hash function $H : \{0,1\}^*$;
4: Chooses the parameters $t_0, a, t_1, t_2, t_0', a', t_1', t_2'$;
5: **return** pp;

[4]This means that if we have s_l, we can compute ${s_l}^{-1}$ and vice versa.

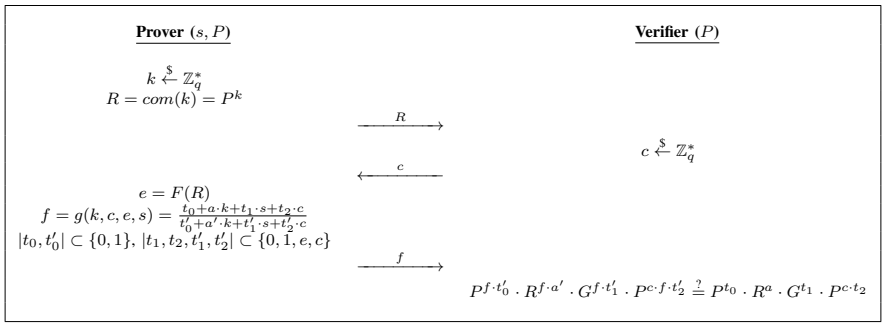


Fig. 1. Our Proposed General Model of Existing Signatures.

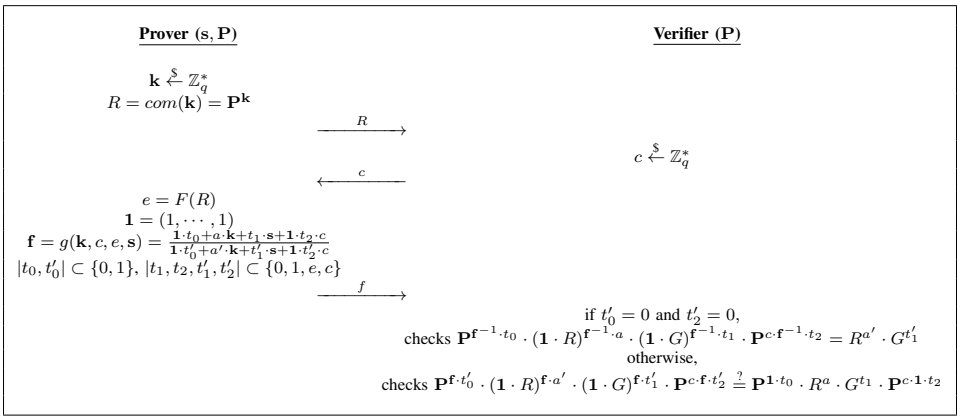


Fig. 2. Our Proposed General Model with One-out-of-Many Proofs.

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);

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

1: Computes $t_0, t_1, t_2, t'_0, t'_1, t'_2$;
2: **for all** $i \in [n]$ **do**
3: Randomly chooses $k_i \xleftarrow{\$} \mathbb{Z}_q^*$;
4: **end for**
5: Computes $R = com(\mathbf{k}) = \mathbf{P}^{\mathbf{k}}$;
6: Computes $e \leftarrow F(R)$;
7: Sets $c \leftarrow H(\mu \| R \| \mathbf{P})$;
8: **for all** $i \in [n], i \neq l$ **do**
9: Sets $f_i \leftarrow g(k, c, e) = \frac{t_0 + a \cdot k_i + t_2 \cdot c}{t'_0 + a' \cdot k_i + t'_2 \cdot c}$;
10: **end for**
11: **for** $i \in [n], i = l$ **do**
12: Computes $f_l \leftarrow g(k, c, e, s) = \frac{t_0 + a \cdot k_l + t_1 \cdot s_l + t_2 \cdot c}{t'_0 + a' \cdot k_l + t'_1 \cdot s_l + t'_2 \cdot c}$;
13: **end for**
14: **return** $\sigma = (R, \mathbf{f})$;

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

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

C. General Ring Signature Model with Folding from Bulletproofs

Further, we can reduce the signature size of our proposed general ring signature by the folding idea from the Bulletproofs [38], to achieve a logarithmic ring signature scheme.

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

$$\mathbf{f}_L = (f_1, \cdots, f_{n'}),\ \mathbf{f}_R = (f_{n'+1}, \cdots, f_n), \tag{4}$$

$$\mathbf{P}_L = (P_1, \cdots, P_{n'}),\ \mathbf{P}_R = (P_{n'+1}, \cdots, P_n). \tag{5}$$

We then introduce the value L_B and R_B as:

$$L_B = {\mathbf{P}_L}^{\mathbf{f}_R} \text{ and } R_B = {\mathbf{P}_R}^{\mathbf{f}_L}. \tag{6}$$

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:

$$\mathbf{P}' = \mathbf{P}_R{}^{x^{-1}} \circ \mathbf{P}_L{}^{x} \text{ and } \mathbf{f}' = \mathbf{f}_R \cdot x + \mathbf{f}_L \cdot x^{-1}. \quad (7)$$

The relation of the verification is $com(\mathbf{f}') \stackrel{?}{=} g(com(\mathbf{k}), c, e, com(\mathbf{s})) \cdot \mathbf{L}_B{}^{\mathbf{x}^2} \cdot \mathbf{R}_B{}^{\mathbf{x}^{-2}}$. Before the check process, the verifier $\mathcal{V}$ have to compute $s_i = \sum_{j=1}^{\log n} x_j{}^{f(i,j)}$ for all $i \in [n]$, where $f(i,j) = 1$ if 's j-th is 0, otherwise $f(i,j) = -1$. Then, 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.

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.

D. Formal Construction of General Ring Signature Scheme

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 heuristic techniques, we now present the formal construction of our proposed $O(\log n)$ ring signature, as described below:

- For the RingBPSetup(λ), we follow the existing signature and input a security parameter λ, the parameter q, the hash function $H : \{0,1\}^*$, the function F to replace special construction, the generator G of the group, the parameters $t_0, a, t_1, t_2, t'_0, a', t'_1, t'_2$, and output the public parameters pp.
- For the RingBPKeyGen(pp), the prover $\mathcal{P}$ randomly samples $s_l \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ 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 is a public-secret key pair (P_l, s_l) of user l.
- For the 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 of user l, a message μ, and predefined values a, a'. The prover $\mathcal{P}$ generates a ring signature as follows:
  - Computes the parameters $t_0, t_1, t_2, t'_0, t'_1, t'_2$.
  - For all $i \in [n]$, randomly chooses the $k_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ and computes the commitment $R = com(\mathbf{k}) = \mathbf{P}^{\mathbf{k}}$.
  - Computes the challenge $c \leftarrow H(\mathbf{P} \| R \| \mu)$.
  - 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 $x \leftarrow H(L_B \| R_B)$.
  - 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 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 $x_j \leftarrow H(L_{Bj} \| R_{Bj})$. In addition, $\mathcal{V}$ have to compute $s_i = \sum_{j=1}^{\log n} x_j{}^{f(i,j)}$ for all $i \in [n]$, where $f(i,j) = 1$ if $(i-1)$'s j-th is 0, otherwise $f(i,j) = -1$. 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 scheme for logarithmic size is shown in Algorithms 5, 6, 7, and 8, namely as RingBPSetup, RingBPKeyGen, RingBPSign and RingBPVerify, respectively.

Algorithm 5 RingBPSetup(λ)

1: Chooses the parameters q, n;
2: Chooses the generator G;
3: Defines a hash function $H : \{0,1\}^*$;
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 $s_l \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$;
2: Computes $P_l = com(s_l{}^{-1})$;
3: **return** (P_l, s_l);

VI. Case Study

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

A. From Schnorr Signature to Ring Signature

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 $\mathbf{P}^{\mathbf{f}} = G^{H(\mu \| R \| \mathbf{P})} \cdot R$ holds.

B. From ECDSA Signature to Ring Signature

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

Algorithm 7 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 $c \leftarrow H(\mu \| R \| \mathbf{P})$;
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 $x \leftarrow H(L_B \| R_B)$;
  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 $c \leftarrow H(\mu \| R \| \mathbf{P})$;
for all $j \in [\log n]$ **do**
  Computes $x_j = H(L_{Bj} \| R_{Bj})$;
end for
for all $i \in [n]$ **do**
  Computes $s_i = \sum_{j=1}^{\log n} {x_j}^{f(i,j)}$, where $f(i,j) = 1$ if $(i-1)$'s j-th is 0, otherwise $f(i,j) = -1$;
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';

C. From EdDSA Signature to Ring Signature

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 $\mathbf{P}^{\mathbf{f}} = G^{H(R \| \mathbf{P} \| \mu)} \cdot R$.

D. From SM2 Signature to Ring Signature

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 $\mathbf{P}^{\mathbf{f}} \cdot (\mathbf{1} \cdot G)^{\mathbf{f}} \cdot G^{H(R) + H(ZA \| \mu)} = R$ holds in SM2RingVerify algorithm.

VII. Security Analysis

In this section, we provide a security analysis demonstrating that our general ring signature scheme satisfies the properties of correctness, anonymity, and unforgeability.

In our security analysis, for the function F in our general ring signature, we refer to the assumption of Pointcheval et al. [39], Brickell et al. [40] in the security proof, where the function F is modeled as a random oracle.

A. Correctness

*Theorem 2 (**Correctness.**):* Our ring signature is converted from the Σ-protocol-based signature to satisfy the correctness property, which can be derived directly from the completeness of Σ-protocol.

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 Bulletproofs [38] based on our $O(n)$-size scheme. $\mathcal{V}$ will check if equation $com(f') = com(\mathbf{f}) \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}} = g(com(\mathbf{k}), c, e, com(\mathbf{s})) \cdot {\mathbf{L}_B}^{\mathbf{x}^2} \cdot {\mathbf{R}_B}^{\mathbf{x}^{-2}}$ holds.

For our general construction, we have $\mathbf{f} = \frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c}$ and we write it as $\mathbf{P}^{\mathbf{f}} = \mathbf{P}^{\frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c}}$. Expanding the above equation, we obtain $\mathbf{P}^{\mathbf{f}(\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c)} = \mathbf{P}^{(\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c)}$. Thus, we can derive the verification formula as $\mathbf{P}^{\mathbf{f} \cdot t'_0} \cdot (\mathbf{1} \cdot R)^{\mathbf{f} \cdot a'} \cdot (\mathbf{1} \cdot G)^{\mathbf{f} \cdot t'_1} \cdot \mathbf{P}^{c \cdot \mathbf{f} \cdot t'_2} = \mathbf{P}^{\mathbf{1} \cdot t_0} \cdot R^a \cdot G^{t_1} \cdot \mathbf{P}^{c \cdot \mathbf{1} \cdot t_2}$.

For the situation of both $t'_0 = 0$ and $t'_2 = 0$, the verification equation is $\mathbf{P}^{\mathbf{f}^{-1}} = \mathbf{P}^{(\frac{\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c}{\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c})^{-1}}$. In this way, we have $\mathbf{P}^{\mathbf{f}^{-1} \cdot (\mathbf{1} \cdot t_0 + a \cdot \mathbf{k} + t_1 \cdot \mathbf{s} + \mathbf{1} \cdot t_2 \cdot c)} = \mathbf{P}^{(\mathbf{1} \cdot t'_0 + a' \cdot \mathbf{k} + t'_1 \cdot \mathbf{s} + \mathbf{1} \cdot t'_2 \cdot c)}$. Expanding the above form, we have the final verification equation $\mathbf{P}^{\mathbf{f}^{-1} \cdot t_0} \cdot (\mathbf{1} \cdot R)^{\mathbf{f}^{-1} \cdot a} \cdot (\mathbf{1} \cdot G)^{\mathbf{f}^{-1} \cdot t_1} \cdot \mathbf{P}^{c \cdot \mathbf{f}^{-1} \cdot t_2} = \mathbf{P}^{\mathbf{1} \cdot t'_0} \cdot R^{a'} \cdot G^{t'_1} \cdot \mathbf{P}^{c \cdot \mathbf{1} \cdot t'_2}$.

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

B. Anonymity

*Theorem 3 (**Anonymity.**):* Our general ring signature is anonymous, which is based on Σ-protocol for the witness $\mathbf{s}$ and the statement $\mathbf{P} = com(\mathbf{s})$. We say it is HVZK, if there is an efficient polynomial probabilistic time algorithm $\mathcal{S}$, called the simulator, 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 Denote $\mathcal{A}$ as the adversary breaking the anonymity of our proposed ring signature. The anonymity game $\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A})$ runs between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ as follows:

- **Setup**: First, $\mathcal{C}$ sends the public parameters pp to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ performs a polynomial bounded number n oracle queries in an adaptive manner, and $\mathcal{C}$ responds to these queries as follows:
  - Random Oracle $\mathcal{O}_H$: We model the hash function H to the random oracle. The list $L_H : \{\mu, \mathbf{P}, R, c\}$ is initially set empty. After receiving a query on $(\mu, \mathbf{P}, R)$, $\mathcal{C}$ checks the lists L_H. If the input has been queried before, $\mathcal{C}$ outputs the corresponding value c. Otherwise, $\mathcal{C}$ calculates correspond c, adds the tuple $\{\mu, \mathbf{P}, R, c\}$ to lists L_H and returns c to $\mathcal{A}$.
  - Random Oracle $\mathcal{O}_F$: We model the function F to the random oracle. The list $L_F : \{R, e\}$ is initially set empty. After receiving a query on R, $\mathcal{C}$ checks the list L_F. If it has been queried before, $\mathcal{C}$ outputs the corresponding value e. Otherwise, $\mathcal{C}$ calculates correspond e, adds the tuple $\{R, e\}$ to list L_F and returns e to $\mathcal{A}$.
  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key P_i of user $i \in [1, n]$. $\mathcal{C}$ invokes $\mathsf{RingKeyGen}$ algorithm to get (P_i, s_i). This also means that a new user i is added to the system. Finally, $\mathcal{C}$ outputs the public key P_i to $\mathcal{A}$.
  - Corruption Oracle $\mathcal{O}_{CO}$: $\mathcal{A}$ quires the secret key s_i for user $i \in [1, n]$. Then, $\mathcal{C}$ returns the secret key s_i to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: The $\mathcal{A}$ inputs a public key list $\mathbf{P}$ include a public key P_i of user i and a message μ, $\mathcal{C}$ runs the $\mathsf{RingSign}$ algorithm and returns a valid signature σ to $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ picks a public keys set $\mathbf{P}$ and a message μ and sends that to $\mathcal{C}$ to request a signature. Note that the set of public keys $\mathbf{P}$ is arbitrarily chosen by $\mathcal{A}$ and each public key in $\mathbf{P}$ need to be generated by the $\mathcal{O}_{RO}$. $\mathcal{C}$ randomly picks $\pi \in [1, n]$ and invokes the $\mathcal{O}_{SO}$ to obtain the signature σ_π. Then, $\mathcal{C}$ sends σ_π to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a guess $\pi^* \in [1, n]$.
  If $\pi^* = \pi$, $\mathcal{C}$ outputs 1; otherwise outputs 0. The advantage of $\mathcal{A}$ which defined by $\mathsf{Adv}_{\mathrm{Anony}}(\mathcal{A}) := |\Pr[\mathsf{Game}_{\mathrm{Anony}}(\mathcal{A}) = 1] - \frac{1}{n}|$ is negligible.
  The anonymity property of our ring signature scheme can also be derived directly from the property of HVZK in Σ-protocol. To generate the accepting transcript without the witness $\mathbf{s}$, we have to utilize the simulator $\mathcal{S}$. In the simulator $\mathcal{S}$, 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 the following operations in order:
  1) choose random $\mathbf{f} \xleftarrow{\$} \mathbb{Z}_q^*$;
  2) choose random $e, c \xleftarrow{\$} \mathbb{Z}_q^*$;
  3) compute $R \leftarrow (\frac{\mathbf{P}^{\mathbf{f}\cdot t_0'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}\cdot t_1'} \cdot \mathbf{P}^{c\cdot\mathbf{f}\cdot t_2'}}{\mathbf{P}^{\mathbf{1}\cdot t_0} \cdot G^{t_1} \cdot \mathbf{P}^{\mathbf{1}\cdot c\cdot t_2}})^{\frac{1}{(a - \mathbf{1}^\top \cdot \mathbf{f}\cdot a')}}$;
  4) program $H(\mu||R||\mathbf{P})$ to be c in $\mathcal{O}_H$ and $F(R)$ to be e in $\mathcal{O}_F$.

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

□

C. Unforgeability

*Theorem 4 (**Unforgeability.**):* If the Discrete Logarithm (DL) problem is hard, then we say our general ring signature is unforgeable.

Proof

Denote $\mathcal{A}$ as the adversary breaking the unforgeability of our proposed ring signature. We build an adversary $\mathcal{B}$ that runs the adversary $\mathcal{A}$ to solve the DL problem.

Suppose $\mathcal{B}$ is given the public parameters pp and a public key P' (computed by $P' = G^{s'}$) from its challenger $\mathcal{C}$. The behavior of $\mathcal{B}$ consists of two stages in the following.

In the first stage, $\mathcal{B}$ plays the role of challenger $\mathcal{C}$ to $\mathcal{A}$ in the unforgeability game $\mathsf{Game}_{\mathrm{Forge}}(\mathcal{A})$, as follows:

- **Setup**: $\mathcal{B}$ picks a target index $i^* \in [1, n]$ and sets $P_{i^*} = P'$. Then, $\mathcal{B}$ sends the public parameters pp to $\mathcal{A}$.
- **Oracle Query**: $\mathcal{A}$ performs a polynomial bounded number n oracle queries in an adaptive manner, and $\mathcal{B}$ responds to these queries as follows:
  - Random Oracle $\mathcal{O}_H$: We model the hash function H to the random oracle. The list $L_H : \{\mu, \mathbf{P}, R, c\}$ is initially set empty. After receiving a query on $(\mu, \mathbf{P}, R)$, $\mathcal{B}$ checks the lists L_H. If the input has been queried before, $\mathcal{B}$ outputs the corresponding value c. Otherwise, $\mathcal{B}$ calculates correspond c, adds the tuple $\{\mu, \mathbf{P}, R, c\}$ to lists L_H and returns c to $\mathcal{A}$.
  - Random Oracle $\mathcal{O}_F$: We model the function F to the random oracle. The list $L_F : \{R, e\}$ is initially set empty. After receiving a query on R, $\mathcal{B}$ checks the list L_F. If it has been queried before, $\mathcal{B}$ outputs the corresponding value e. Otherwise, $\mathcal{B}$ calculates correspond e, adds the tuple $\{R, e\}$ to list L_F and returns e to $\mathcal{A}$.
  - Registration Oracle $\mathcal{O}_{RO}$: $\mathcal{A}$ queries the public key P_i of user $i \in [1, n], i \neq i^*$. $\mathcal{B}$ invokes $\mathsf{RingKeyGen}$ algorithm to get (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 user $i \in [1, n]$. If $i = i^*$, the query is aborted and $\mathcal{B}$ returns $\perp$. If $i \neq i^*$, $\mathcal{B}$ returns the secret key s_i to $\mathcal{A}$.
  - Signing Oracle $\mathcal{O}_{SO}$: $\mathcal{A}$ queries the ring signature for user $i \in [1, n]$. If the signer's public key P_i is not in the public keys set $\mathbf{P}$, $\mathcal{B}$ returns $\perp$. If $i \neq i^*$, $\mathcal{B}$ returns the ring signature σ to $\mathcal{A}$ using the $\mathsf{RingSign}$ algorithm.

Otherwise, if $i = i^*$, the query cannot be aborted. In this case, the signer's secret key s_i is missing, so $\mathcal{B}$ uses the simulator $\mathcal{S}$ (as described earlier) to generate and return ring signature σ.

- **Forge**: $\mathcal{A}$ returns a forgery ring signature σ^*, along with the message μ^* and a set of public keys $\mathbf{P}^*$ that the i^*-th element is the signer's public key P_{i^*}. The signature σ^* is verified using the RingVerify algorithm. $\mathcal{A}$ has not queried the ring signature about the message μ^* and $\mathbf{P}^*$ in $\mathcal{O}_{SO}$, and has not queried $\mathcal{O}_{CO}$ regarding any public keys in $\mathbf{P}^*$. Each public key in $\mathbf{P}^*$ is generated from $\mathcal{O}_{RO}$.
The goal of $\mathcal{B}$ in this step is to compute two accepting transcripts with different challenges. For the same message μ^* and public keys set $\mathbf{P}^*$, if $\mathcal{A}$ successfully forges a ring signature σ^* which not queried $\mathcal{O}_{SO}$, according to the forking lemma in [41], $\mathcal{B}$ rewinds to the point where R^* is generated and returns a different value $c^\diamond$ by the random oracle $\mathcal{O}_H$ instead, such that $c^* \neq c^\diamond$. Subsequently, $\mathcal{A}$ returns another ring signature $\sigma^\diamond$. Due to the rewinding capability of $\mathcal{B}$, R^* remains the same for both valid signatures σ^* and $\sigma^\diamond$. Consequently, there exist two accepting transcripts $(R^*, c^*, e^*, \mathbf{f}^*)$ and $(R^*, c^\diamond, e^*, \mathbf{f}^\diamond)$ with $c^* \neq c^\diamond$ and $\mathbf{f}^* \neq \mathbf{f}^\diamond$, the equations are as follows:

$$\begin{aligned}&\mathbf{P}^{*\mathbf{f}^*\cdot t_0'} \cdot (\mathbf{1}\cdot R^*)^{\mathbf{f}^*\cdot a'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^*\cdot t_1'} \cdot \mathbf{P}^{*c^*\cdot\mathbf{f}^*\cdot t_2'} \\ &= \mathbf{P}^{*\mathbf{1}\cdot t_0} \cdot R^{*a} \cdot G^{t_1} \cdot \mathbf{P}^{*c^*\cdot\mathbf{1}\cdot t_2},\end{aligned} \tag{8}$$

and

$$\begin{aligned}&\mathbf{P}^{*\mathbf{f}^\diamond\cdot t_0'} \cdot (\mathbf{1}\cdot R^*)^{\mathbf{f}^\diamond\cdot a'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^\diamond\cdot t_1'} \cdot \mathbf{P}^{*c^\diamond\cdot\mathbf{f}^\diamond\cdot t_2'} \\ &= \mathbf{P}^{*\mathbf{1}\cdot t_0} \cdot R^{*a} \cdot G^{t_1} \cdot \mathbf{P}^{*c^\diamond\cdot\mathbf{1}\cdot t_2}.\end{aligned} \tag{9}$$

So far, $\mathcal{B}$ simulates the challenger $\mathcal{C}$ perfectly. After that, in the second stage, $\mathcal{B}$ feeds these two accepting transcripts into the extractor $\mathcal{E}$ to compute the ${s_{i^*}}^{-1}$, as follows:
First, deriving the above equations further yields two expressions about R^*:

$$R^* \leftarrow \left(\frac{\mathbf{P}^{*\mathbf{f}^*\cdot t_0'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^*\cdot t_1'} \cdot \mathbf{P}^{*c^*\cdot\mathbf{f}^*\cdot t_2'}}{\mathbf{P}^{*\mathbf{1}\cdot t_0} \cdot G^{t_1} \cdot \mathbf{P}^{*\mathbf{1}\cdot c^*\cdot t_2}}\right)^{\frac{1}{(a-\mathbf{1}^\top\cdot\mathbf{f}^*\cdot a')}}, \tag{10}$$

and

$$R^* \leftarrow \left(\frac{\mathbf{P}^{*\mathbf{f}^\diamond\cdot t_0'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^\diamond\cdot t_1'} \cdot \mathbf{P}^{*c^\diamond\cdot\mathbf{f}^\diamond\cdot t_2'}}{\mathbf{P}^{*\mathbf{1}\cdot t_0} \cdot G^{t_1} \cdot \mathbf{P}^{*\mathbf{1}\cdot c^\diamond\cdot t_2}}\right)^{\frac{1}{(a-\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot a')}}. \tag{11}$$

As R^* is the same for these two transcripts, the right sides of the above two formulas are equivalent, i.e.,

$$\begin{aligned}&\left(\frac{\mathbf{P}^{*\mathbf{f}^*\cdot t_0'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^*\cdot t_1'} \cdot \mathbf{P}^{*c^*\cdot\mathbf{f}^*\cdot t_2'}}{\mathbf{P}^{*\mathbf{1}\cdot t_0} \cdot G^{t_1} \cdot \mathbf{P}^{*\mathbf{1}\cdot c^*\cdot t_2}}\right)^{\frac{1}{(a-\mathbf{1}^\top\cdot\mathbf{f}^*\cdot a')}} \\ &= \left(\frac{\mathbf{P}^{*\mathbf{f}^\diamond\cdot t_0'} \cdot (\mathbf{1}\cdot G)^{\mathbf{f}^\diamond\cdot t_1'} \cdot \mathbf{P}^{*c^\diamond\cdot\mathbf{f}^\diamond\cdot t_2'}}{\mathbf{P}^{*\mathbf{1}\cdot t_0} \cdot G^{t_1} \cdot \mathbf{P}^{*\mathbf{1}\cdot c^\diamond\cdot t_2}}\right)^{\frac{1}{(a-\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot a')}}.\end{aligned} \tag{12}$$

Through a series of mathematical transformations, it gets:

$$G = \mathbf{P}^{*\frac{\frac{(\mathbf{f}^*\cdot t_0' + c^*\cdot\mathbf{f}^*\cdot t_2' - \mathbf{1}\cdot t_0 - \mathbf{1}\cdot c^*\cdot t_2)}{(a-\mathbf{1}^\top\cdot\mathbf{f}^*\cdot a')} + \frac{(\mathbf{1}\cdot t_0 + \mathbf{1}\cdot c^\diamond\cdot t_2 - \mathbf{f}^\diamond\cdot t_0' - c^\diamond\cdot\mathbf{f}^\diamond\cdot t_2')}{(a-\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot a')}}{\frac{t_1 - \mathbf{1}^\top\mathbf{f}^*\cdot t_1'}{(a-\mathbf{1}^\top\cdot\mathbf{f}^*\cdot a')} + \frac{\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot t_1' - t_1}{(a-\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot a')}}} \tag{13}$$

From Eq. (13), the extractor $\mathcal{E}$ can output $\mathbf{s}^*$ with respect to $\mathbf{P}^*$ as:

$$\mathbf{s}^* = \frac{\frac{(\mathbf{f}^*\cdot t_0' + c^*\cdot\mathbf{f}^*\cdot t_2' - \mathbf{1}\cdot t_0 - \mathbf{1}\cdot c^*\cdot t_2)}{(a-\mathbf{1}^\top\cdot\mathbf{f}^*\cdot a')} + \frac{(\mathbf{1}\cdot t_0 + \mathbf{1}\cdot c^\diamond\cdot t_2 - \mathbf{f}^\diamond\cdot t_0' - c^\diamond\cdot\mathbf{f}^\diamond\cdot t_2')}{(a-\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot a')}}{\frac{t_1 - \mathbf{1}^\top\mathbf{f}^*\cdot t_1'}{(a-\mathbf{1}^\top\cdot\mathbf{f}^*\cdot a')} + \frac{\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot t_1' - t_1}{(a-\mathbf{1}^\top\cdot\mathbf{f}^\diamond\cdot a')}} \tag{14}$$

From Eq. (14), the adversary $\mathcal{B}$ obtains the i^*-th element s_{i^*} in $\mathbf{s}^*$ and outputs the ${s_{i^*}}^{-1}$ as a DL solution. To this point, $\mathcal{B}$ successfully extract the ${s_{i^*}}^{-1}$.
Finally, we prove that the adversary $\mathcal{B}$ wins the game below, i.e., prove the output is valid ($P_{i^*} = G^{{s_{i^*}}^{-1}}$). Proving the $P_{i^*} = G^{{s_{i^*}}^{-1}}$ is equal to proving the $G = {P_{i^*}}^{s_{i^*}}$.
For the Eq. (13) and Eq. (14), we can write that as $G = \mathbf{P}^{*\mathbf{s}^*} = {P_1}^{s_1} \cdots {P_{i^*}}^{s_{i^*}} \cdots {P_n}^{s_n}$. Thus, to prove the $G = {P_{i^*}}^{s_{i^*}}$, we just need to prove that all s_i for $i \neq i^*$ are equal to 0, i.e. $G = {P_1}^0 \cdots {P_{i^*}}^{s_{i^*}} \cdots {P_n}^0 = {P_{i^*}}^{s_{i^*}}$.
According Eq. (14), we write s_i as:

$$s_i = \frac{\frac{(f_i^*\cdot t_0' + c^*\cdot f_i^*\cdot t_2' - t_0 - c^*\cdot t_2)}{(a-f_i^*\cdot a')} + \frac{(t_0 + c^\diamond\cdot t_2 - f_i^\diamond\cdot t_0' - c^\diamond\cdot f_i^\diamond\cdot t_2')}{(a-f_i^\diamond\cdot a')}}{\frac{t_1 - f_i^*\cdot t_1'}{(a-f_i^*\cdot a')} + \frac{f_i^\diamond\cdot t_1' - t_1}{(a-f_i^\diamond\cdot a')}} \tag{15}$$

Since the R^* and $\mathbf{P}^*$ are the same for two accepting transcripts and $R^* = \mathbf{P}^{*\mathbf{k}^*}$, each element k_i in $\mathbf{k}^*$ is also the same for two accepting transcripts. In our setting, $f_i \leftarrow \frac{t_0 + a\cdot k_i + t_2\cdot c}{t_0' + a'\cdot k_i + t_2'\cdot c}$ for each $i \neq i^*$, and the parameters $a, t_0, t_1, t_2, a', t_0', t_1', t_2'$ are the same for two accepting transcripts. Therefore, for $i \neq i^*$, each element $f_i^* = f_i^\diamond$, and for $i = i^*$, $f_i^* \neq f_i^\diamond$. So, we replace $f_i^\diamond$ with f_i^* to compute s_i as:

$$\begin{aligned}s_i &= \frac{\frac{f_i^*\cdot t_0' - f_i^*\cdot t_0' + c^*\cdot f_i^*\cdot t_2' - c^\diamond\cdot f_i^*\cdot t_2' - t_0 + t_0 - c^*\cdot t_2 + c^\diamond\cdot t_2}{a-f_i^*\cdot a'}}{\frac{t_1 - t_1 - f_i^*\cdot t_1' + f_i^*\cdot t_1'}{a-f_i^*\cdot a'}} \\ &= \frac{\frac{c^*\cdot f_i^*\cdot t_2' - c^\diamond\cdot f_i^*\cdot t_2' - c^*\cdot t_2 + c^\diamond\cdot t_2}{a-f_i^*\cdot a'}}{\frac{t_1 - t_1 - f_i^*\cdot t_1' + f_i^*\cdot t_1'}{a-f_i^*\cdot a'}} = 0\end{aligned} \tag{16}$$

According $G = \mathbf{P}^{*\mathbf{s}^*} = {P_1}^{s_1} \cdots {P_{i^*}}^{s_{i^*}} \cdots {P_n}^{s_n}$ and $s_i = 0$ for any $i \neq i^*$, we prove that $G = {P_{i^*}}^{s_{i^*}}$. Thus, the ${s_{i^*}}^{-1}$ is a valid DL solution such that $P_{i^*} = G^{{s_{i^*}}^{-1}}$.
Since the adversary $\mathcal{A}$ performs a polynomial bounded number of $\mathcal{O}_{RO}$ and $\mathcal{O}_{SO}$ queries, the probability of $\mathcal{B}$ winning this game is non-negligible. To this point, we prove that our general ring signature satisfies the property of unforgeability.

□

VIII. Performance Evaluation

In this section, we show the implementation of our four case studies in Java language based on the JPBC library [5], and we use the 256-bit elliptic curve secp256k1 (which is 256-bit security) to evaluate performance. All experiments were conducted in a system environment with a processor of Apple M1 and memory of 16.0 GB. Moreover, we perform a comparative analysis of our scheme with others [24], [25], [27] in time cost and signature size. In the time cost measurement,

[5]The source code is published at https://github.com/BatchClayderman/GRS.

we do not measure the time cost for the KeyGen algorithm. When measuring the time cost of the sign algorithm and verify algorithm, we calculate the average time cost of 30 benchmark runs, with each benchmark run consisting of 1000 iterations.

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

A. Our Experiment Result

In this subsection, we evaluate the time cost and signature size of our proposed $O(n)$-size and $O(\log n)$-size general ring signature.

a) Time Cost: Firstly, we show the signing time and verification time of our four case studies. The time cost of the Schnorr ring signature, ECDSA ring signature, EdDSA ring signature, and SM2 ring signature are shown in Fig. 3(a), 3(b), 3(c), 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 of the scheme improves, and the time costs increase smoothly as the ring size grows.

In Fig. 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 the number of ring members n increases, the signature generation and verification time increase linearly. Compared with our $O(n)$-size ring signature, although the time cost becomes larger, the size of the signature is substantially reduced, which is a trade-off between time cost and signature size, and it is acceptable.

In detail, when the ring member is 8, the signing time and verification time are around 48.19 ms to 50.79 ms, 47.42 ms to 49.42 ms for the linear-size scheme, and 194.68 ms to 201.68 ms, 85.19 ms to 90.19 ms for the logarithmic-size scheme, respectively.

b) Signature Size: 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 signature size 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 signature size is $(1 + 2\log n)|\mathbb{G}| + 1|\mathbb{Z}_q|$. The specific overhead is related to the selection of an elliptic curve. In our implementation, an element in $\mathbb{G}$ is denoted by 264 bits, and an element in $\mathbb{Z}_q$ is denoted by 256 bits.

When we convert a Σ-based signature to a ring signature, the size is related to the number of ring members n. When the number of ring members $n = 64$, 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 signature sizes 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 signature size analysis of our $O(\log n)$-size general ring signature is shown in Table 1.

B. Comparison with Prior Art

For the sake of showing the practicality of our general ring signature construction, we compare it with other ring signatures [24], [25], [27]. Our $O(\log n)$-size general ring signature performance is implementation-acceptable on common devices and has a better time cost and signature size than other ring signature schemes [24], [25], [27]. Thus, the scholars use our general transformation method to transfer various Σ-based signatures to ring signatures, which is practical if they need user anonymity.

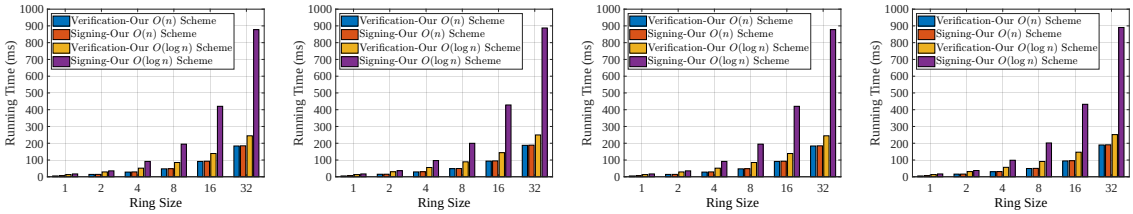
a) Time Cost: Now, we perform a comparison analysis with other existing schemes [24], [25], [27] in signing and verification time cost, and signature size, using our Schnorr ring signature as a representative. Note that the performance of the Feng et al. scheme [26] is consistent with that of Yuen et al. [27] and is thereby not repeated below.

The comparison of signing time cost is shown in Fig. 4(a) and Fig. 5(a). In Fig. 4(a), the signing time of schemes [24] and [25] are higher than others since their schemes have $O(n \log n)$ exponentiation costs. It can be seen that our $O(\log n)$-size scheme has a slightly lower signing time cost than the $O(\log n)$-size scheme of Yuen et al [27]. Similarly, our $O(n)$-size scheme has a slightly lower signing time cost than the $O(n)$-size scheme of Yuen et al [27] in Fig. 5(a). In short, in signing time cost performance, our $O(\log n)$-size scheme and $O(n)$-size scheme have the best performance.

A comparison of verification time cost is illustrated in Fig. 4(b) and Fig. 5(b). In Fig. 4(b), as same as the signing time cost, the verification time of schemes [24], [25] is higher than other ring signatures. The overhead of Yuen et al.'s $O(\log n)$-size scheme [27] is slightly higher than our $O(\log n)$-size scheme. Moreover, in Fig. 5(b), Yuen et al.'s $O(n)$-size scheme [27] is slightly higher than our $O(n)$-size scheme. Therefore, our scheme performs best among other existing schemes [24], [25], [27] in verification time cost.

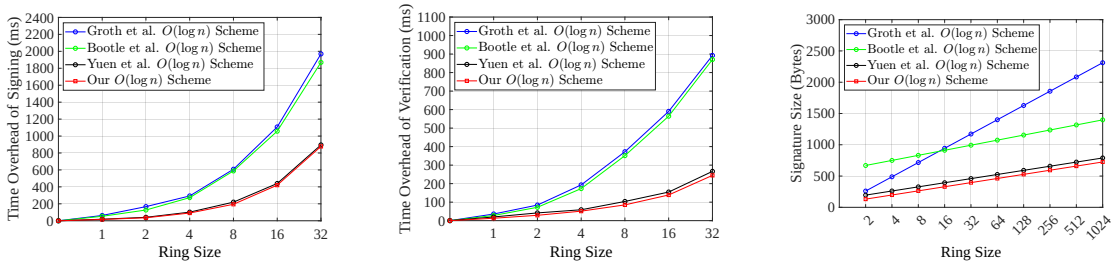
In particular, the signing and verification times of our ring signature scheme are 0.44 to 0.97 times and 0.27 to 0.91 times compared to other state-of-the-art schemes, respectively.

b) Signature Size: For the signature size, the comparison of our $O(\log n)$-size ring signature with others $O(\log n)$-size schemes [24], [25], [27] is shown in Fig. 4(c). In scheme [24], the ring signature size is $(3\log n + 1)|\mathbb{Z}_q| + (4\log n)|\mathbb{G}|$ and the size of scheme [25] is equal to $(1.5\log n + 6)|\mathbb{Z}_q| + (\log n + 12)|\mathbb{G}|$, with n being the number of ring members. We can observe that when the number of ring members is less than 16, the size of Groth et al.'s scheme [24] is lower than that of Bootle et al.'s scheme [25], whereas the size of



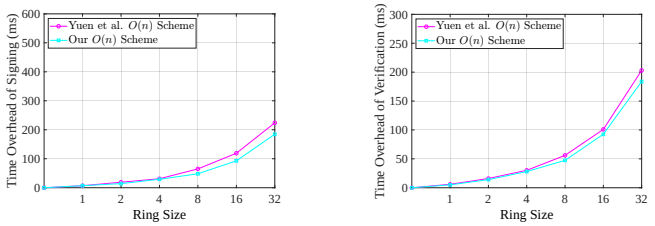
(a) Schnorr Case. (b) ECDSA Case. (c) EdDSA Case. (d) SM2 Case.

Fig. 3. Time Cost of Our $O(n)$ and $O(\log n)$ Ring Signatures.



(a) Signing Time of $O(\log n)$ Schemes. (b) Verification Time of $O(\log n)$ Schemes. (c) Signature Size of $O(\log n)$ Schemes.

Fig. 4. Comparison of Time Cost and Signature Size of $O(\log n)$ Ring Signatures.



(a) Signing Time of $O(n)$ Schemes. (b) Verification Time of $O(n)$ Schemes.

Fig. 5. Comparison of Time Cost of $O(n)$ Ring Signatures.

Groth et al.'s scheme [24] grows substantially with increasing the number of ring members. As the number of ring members grows, the signature sizes of scheme [25] increase rate less than the scheme [24]. More importantly, the signature size of our scheme is slightly lower than Yuen et al.'s scheme [27] and is substantially lower than that of scheme [24], [25], which shows the best performance till now.

C. Further Performance Analysis between Ours and DualRing

In this paper, we first propose a $O(n)$-size general ring signature, and then we further use Bulletproofs folding technique to reduce the signature size to propose a $O(\log n)$-size general ring signature. Similar to our work, Yuen et al. [27] also proposed a $O(n)$-size ring signature and a $O(\log n)$-size ring signature. The ring signatures proposed by Groth et al. [24] and Bootle et al. [25] are $O(\log n)$-size.

By employing Bulletproofs folding, both Yuen et al. [27] and ours have introduced additional steps to minimize the signature size in $O(\log n)$ scheme. However, these extra steps also result in an increase in time cost. Thus, the time cost comparison should be made between ring signatures of the same size[6]. For $O(n)$-size ring signatures, we compare our proposed $O(n)$-size ring signature with the $O(n)$-size ring signature proposed by Yuen et al. [27] in Fig. 5. For the $O(\log n)$-size ring signature, we analyze our proposed $O(\log n)$-size ring signature in comparison with the $O(\log n)$-size ring signatures proposed by Yuen et al. [27], Groth et al. [24], and Bootle et al. [25] in Fig. 4. As can be seen from Fig. 4(a) and Fig. 4(b), the blue and green lines are the $O(\log n)$-size ring signatures proposed by Groth et al. [24] and Bootle et al. [25] respectively, which have a much higher time cost than the other four lines, and are therefore no longer analyzed.

Then we analyze our scheme and the scheme proposed by Yuen et al. [27] in detail. From Fig. 5(a) and Fig. 5(b), we can observe that for the $O(n)$-size ring signature, the pink line represents the $O(n)$-size ring signature proposed by Yuen et al. [27]. The cyan line represents the $O(n)$-size ring signature proposed by us, where the time cost of our $O(n)$-size ring signature slightly outperforms that proposed by Yuen et al. [27]. The comparison of $O(\log n)$-size ring signature is shown in Fig. 4(a) and Fig. 4(b), the black line is the $O(\log n)$-size ring signature proposed by Yuen et al. [27]. The red line is on our proposed $O(\log n)$-size ring signature, which shows that the overhead of our $O(\log n)$-size ring signature outperforms that of Yuen et al. [27]. In summary, the time cost of our scheme outperforms that of Yuen et al. [27] for both $O(n)$-size ring signatures and $O(\log n)$-size ring signatures.

D. Discussion on Choices of $O(n)$ and $O(\log n)$ Schemes

The choice of $O(n)$-size ring signature and $O(\log n)$-size ring signature should be based on actual requirements. The $O(\log n)$-size ring signature has low signature size and high time cost, while the $O(n)$-size ring signature has high signature size and low time cost. In short, if the system focuses on saving signature size, it should choose our proposed $O(\log n)$-size ring signature, while if the system needs to generate the signature and complete the verification as soon as possible, it should choose our proposed $O(n)$-size ring signature.

[6]Specifically, the $O(\log n)$-size ring signature should only be compared to other $O(\log n)$-size ring signatures [24], [25], [27] (in Fig. 4), and the $O(n)$-size ring signature should only be compared to other $O(n)$-size ring signature [27] (in Fig. 5). Comparisons between $O(n)$-size and $O(\log n)$-size ring signatures are not meaningful, as the complexities are different.

IX. Conclusion

In this paper, we propose a general ring signature construction that provides a transformation method from Σ-based signatures to ring signatures without reducing the security level. 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 propose a general construction technique that can transform the existing signatures to ring signatures, utilizing our redesigned one-out-of-many relation and Fiat-Shamir heuristic technique. Further, to enhance the ring signature scheme, we implement the Bulletproofs folding technique on our ring signature scheme to achieve an efficient logarithmic-size ring signature.

To showcase the practicality of our general scheme, we conduct four case studies using our general construction technique to convert the Schnorr signature, ECDSA signature, EdDSA signature, and SM2 signature into ring signature schemes respectively. Finally, we perform security analysis and performance evaluation, showing that our scheme satisfies correctness, anonymity, and unforgeability, with a minor additional overhead when we construct it to a ring signature. Comparative analysis shows that our scheme is better than most existing schemes in time cost and outperforms other existing ring signatures in signature size.

References

[1] S. Goldwasser and M. Bellare, "Lecture notes on cryptography," *Summer course "Cryptography and computer security" at MIT*, vol. 1999, p. 1999, 1996.

[2] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2020.

[3] B. A. Forouzan and D. Mukhopadhyay, *Cryptography and network security*. Mc Graw Hill Education (India) Private Limited New York, NY, USA:, 2015, vol. 12.

[4] S. Xu, X. Chen, C. Wang, Y. He, K. Xiao, and Y. Cao, "A lattice-based ring signature scheme to secure automated valet parking," in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2021, pp. 70–83.

[5] J. Ren, Y. Cheng, and S. Xu, "Edppa: An efficient distance-based privacy preserving authentication protocol in vanet," *Peer-to-Peer Networking and Applications*, vol. 15, no. 3, pp. 1385–1397, 2022.

[6] Z. Na and X. G. Xi, "The application of a scheme of digital signature in electronic government," in *2008 International Conference on Computer Science and Software Engineering*, vol. 3. IEEE, 2008, pp. 618–621.

[7] X. Chen, S. Xu, Y. He, Y. Cui, J. He, and S. Gao, "Lfs-as: lightweight forward secure aggregate signature for e-health scenarios," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1239–1244.

[8] R. Kumar, "Cryptanalytic performance appraisal of improved hll, kuochen, gengvrf, fengvrf secure signature with tkip digital workspaces: for financial cryptography," *Wireless Personal Communications*, vol. 115, no. 2, pp. 1541–1563, 2020.

[9] Y. Shi, J. Liang, M. Li, T. Ma, G. Ye, J. Li, and Q. Zhao, "Threshold eddsa signature for blockchain-based decentralized finance applications," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 2022, pp. 129–142.

[10] G. Xu, Y. Cao, S. Xu, K. Xiao, X. Liu, X. Chen, and M. Dong, "A novel post-quantum blind signature for log system in blockchain." *Comput. Syst. Sci. Eng.*, vol. 41, no. 3, pp. 945–958, 2022.

[11] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1996, pp. 387–398.

[12] Y. 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*. Springer, 2012, pp. 554–571.

[13] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[14] J. Weise, "Public key infrastructure overview," *Sun BluePrints OnLine, August*, pp. 1–27, 2001.

[15] D. R. Kuhn, V. C. Hu, W. T. Polk, and S.-J. Chang, "Introduction to public key technology and the federal pki infrastructure," Citeseer, Tech. Rep., 2001.

[16] A. Imghoure, A. El-Yahyaoui, and F. Omary, "Ecdsa-based certificateless conditional privacy-preserving authentication scheme in vehicular ad hoc network," *Vehicular Communications*, vol. 37, p. 100504, 2022.

[17] K. Kim, J. Kim, B. Lee, and G. Ahn, "Experimental design of worldwide internet voting system using pki," in *SSGRR2001*. SSGRR, 2001, pp. 0–0.

[18] R. L. Rivest, A. Shamir, and Y. 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*. Springer, 2001, pp. 552–565.

[19] S. S. Chow, S.-M. Yiu, and L. C. 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*. Springer, 2005, pp. 499–512.

[20] A. Bender, J. Katz, and R. 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*. Springer, 2006, pp. 60–79.

[21] X. Chen, S. Xu, Y. Cao, Y. He, and K. Xiao, "Aqrs: Anti-quantum ring signature scheme for secure epidemic control with blockchain," *Computer Networks*, vol. 224, p. 109595, 2023.

[22] S. Zeng, Y. Huang, and X. Liu, "Privacy-preserving communication for vanets with conditionally anonymous ring signature," *International Journal of Network Security*, vol. 17, no. 2, pp. 135–141, 2015.

[23] M. Abe, M. Ohkubo, and K. Suzuki, "1-out-of-n signatures from a variety of keys," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 87, no. 1, pp. 131–140, 2004.

[24] J. Groth and M. 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*. Springer, 2015, pp. 253–280.

[25] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit, "Short accountable ring signatures based on ddh," in *European Symposium on Research in Computer Security*. Springer, 2015, pp. 243–265.

[26] M. Feng, C. Lin, W. Wu, and D. He, "Sm2-dualring: Efficient sm2-based ring signature schemes with logarithmic size," *Computer Standards & Interfaces*, vol. 87, p. 103763, 2024.

[27] T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding, "Dualring: generic construction of ring signatures with efficient instantiations," in *Annual International Cryptology Conference*. Springer, 2021, pp. 251–281.

[28] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on information theory*, vol. 22, no. 6, pp. 644–654, 1976.

[29] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[30] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[31] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[32] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.

[33] D. Chaum and E. Van Heyst, "Group signatures," in *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*. Springer, 1991, pp. 257–265.

[34] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Cryptographers' track at the RSA conference*. Springer, 2011, pp. 376–392.

[35] S. Josefsson and I. Liusvaara, "Edwards-curve digital signature algorithm (eddsa)," Tech. Rep., 2017.

[36] L. Bai, Y. Zhang, and G. Yang, "Sm2 cryptographic algorithm based on discrete logarithm problem and prospect," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. IEEE, 2012, pp. 1294–1297.

[37] X. Lu, M. H. Au, and Z. Zhang, "Raptor: a practical lattice-based (linkable) ring signature," in *Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17*. Springer, 2019, pp. 110–130.
[38] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 315–334.
[39] D. Pointcheval and S. Vaudenay, "On provable security for digital signature algorithms," 1996.
[40] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung, "Design validations for discrete logarithm based signature schemes," in *International Workshop on Public Key Cryptography*. Springer, 2000, pp. 276–292.
[41] J. Herranz and G. Sáez, "Forking lemmas for ring signature schemes," in *International Conference on Cryptology in India*. Springer, 2003, pp. 266–279.

Xue Chen (Graduate Student Member, IEEE) is currently a Ph.D. student in the Department of Computer Science at the University of Hong Kong. She received an M.Phil. degree in the Department of Computing at the Hong Kong Polytechnic University in 2024. Her research interests include Applied Cryptography, Lattice, Signature, and Data Security. Miss Xue has published 15+ academic papers in refereed international conferences and journals including IEEE TIFS, IEEE TC, Computer Networks, ACM CIKM, Inscrypt, IEEE ICC, etc. She served as a PC Member of IEEE MSN, and a Reviewer of CRYPTO, ASIACRYPT, IEEE TIFS, IEEE TCC, ACM WWW, INFORMATION SCIENCES, ELSEVIER JISA, etc. She won the Excellent Paper Award of TSINGHUA SCIENCE AND TECHNOLOGY in 2022. She is a Graduate Student Member of IEEE.

Shang Gao (Member, IEEE) is currently an Assistant Professor with a joint appointment in the Department of Computing and the School of Accounting and Finance at the Hong Kong Polytechnic University. He received a B.Sc. degree from Hangzhou Dianzi University in 2010, an M.E. degree from Southeast University in 2014, and a Ph.D. degree from Hong Kong Polytechnic University in 2019. After graduation, he worked at Microsoft China for one year. His research interests include Information Security, Applied Cryptography, Blockchain Security, and Zero-knowledge Proofs.
Dr. Gao has published 60+ papers in top-tier conferences and journals, including IEEE S&P, ACM CCS, IEEE INFOCOM, IACR PKC, IEEE ICDCS, ACM AsiaCCS, IEEE/ACM IWQoS, IEEE Globecom, IEEE ICC, IEEE TDSC, IEEE/ACM TON, IEEE TNSE, IEEE TNSM, IEEE IOTJ, etc. He served as a PC Member of ICDCS 2022, SecureCom 2023 & 2024, and a Reviewer of CRYPTO 2024, ASIACRYPT 2024.

Shiyuan Xu (Graduate Student Member, IEEE) is currently a Ph.D. candidate in the Department of Computer Science at the University of Hong Kong. His research interests include Applied Cryptography, Lattice, Searchable Encryption, and Data Security.
Mr. Xu has published 20+ academic articles in refereed conferences and journals, including IEEE TIFS, IEEE TC, IEEE IoTJ, Computer Networks, ACM CIKM, Inscrypt, IEEE ICC. He served as a PC Member of CRYPTO, ASIACRYPT, IEEE MSN, and a Reviewer of IEEE TDSC, IEEE TIFS, IEEE TITS, IEEE JSAC, ACM WWW, ACM CIKM, DASFAA. He won the Excellent Paper Award of Tsinghua Science and Technology in 2022. He is a Graduate Student Member of IEEE.

Liquan Chen (Senior Member, IEEE) received the Ph.D. degree from Southeast University, Nanjing, China, in 2005. He was a Post-Doctoral Researcher with Southeast University from 2005 to 2007, where he was an Associate Professor from 2008 to 2018. He was a Visiting Scholar with the National University of Singapore, Singapore, from 2011 to 2012. He is currently a Professor with the School of Cyber Science and Engineering, Southeast University. His research interests include information security, cryptography, and network security protocol. Professor Chen has published several papers in top-tier conferences and journals, including IEEE ToTJ, IEEE TNSM, IEEE TSC, IEEE TIFS, IEEE TII, IEEE TMC. He is a Senior Member of IEEE.

Siu-Ming Yiu (Member, IEEE) is currently a Full Professor and the Associate Director of School of Computing and Data Science at the University of Hong Kong. He received a B.Sc. degree from the Chinese University of Hong Kong in 1988, an M.S. degree from Temple University in 1992, and a Ph.D. degree from The University of Hong Kong in 1996. His research interests are Cryptography and Information Security.
Professor Yiu has published more than 500+ papers in top-tier conferences and refereed journals, including NATURE, SCIENCE, EUROCRYPT, ACM SIGMOD, IEEE ICDE, IEEE TIFS, IEEE TDSC, IEEE TPDS, IEEE TC, IEEE TKDE, IEEE TSC, etc. His work has been cited over 30000 times on Google Scholar. He has served as a Steering Committee Member of ASIACRYPT and a General Chair in ASIACRYPT 2017, ICICS 2014, ISC 2014, ProvSec 2014. He was named on the list of 'Highly Cited Researchers' from Clarivate Analytics as the most influential in the world in 2016, 2017, and 2019 and has been one of the Top 1% of HKU scholars for 12 consecutive years (2011-2022). He is also listed as the World's Top 2% Scientists for career-long impact by Stanford University in 2022. He is a Member of IEEE.

Bin Xiao (Fellow, IEEE) is currently a Full Professor in the Department of Computing, at the Hong Kong Polytechnic University. He was the Dept. Research Committee Chair (DRC Chair) from 2018 to 2021. He received a B.Sc. and M.Sc. degrees from Fudan University, China, and a Ph.D. degree from the University of Texas at Dallas, U.S.A. in 2003. His research interests include AI and Network security, Data Privacy, and Blockchain.
Professor Xiao has published more than 200 technical papers in top-tier conferences and journals, including IEEE S&P, ACM CCS, IEEE INFOCOM, IEEE/CVF CVPR, ACM MM, IEEE TIFS, IEEE TDSC, IEEE JSAC, IEEE TC, IEEE TNSE. Currently, he serves as the Associate Editor of IEEE TCC and IEEE TNSE. He served as the Associate Editor for Elsevier JPDC from 2016 to 2021 and IEEE IoTJ from 2020 to 2023. He is the Chair of the IEEE ComSoc CISTC Committee and IEEE ComSoc distinguished lecturer. He has been the Track Co-Chair of IEEE ICDCS 2022, the Symposium Track Co-Chair of IEEE ICC 2020, ICC 2018, GLOBECOM 2024, GLOBECOM 2017, and the General Chair of IEEE SECON 2018. He is the Chair of the IEEE ComSoc CISTC Committee and IEEE ComSoc distinguished lecturer. He is a Fellow of IEEE, a Fellow of AAIA, and a Member of ACM and CCF.