Leaking Arbitrarily Many Secrets: Any-out-of-Many Proofs and Applications to RingCT Protocols

Tianyu Zheng
Department of Computing
The Hong Kong Polytechnic University
tian-yu.zheng@connect.polyu.hk

Shang Gao
Department of Computing
The Hong Kong Polytechnic University
shanggao@polyu.edu.hk

Yubo Song
School of Cyber Science and Engineering
Southeast University
songyubo@seu.edu.cn

Bin Xiao
Department of Computing
The Hong Kong Polytechnic University
csbxiao@polyu.edu.hk

Abstract

Ring Confidential Transaction (RingCT) protocol is an effective cryptographic component for preserving the privacy of cryptocurrencies. However, existing RingCT protocols are instantiated from one-out-of-many proofs with only one secret, leading to low efficiency and weak anonymity when handling transactions with multiple inputs. Additionally, current partial knowledge proofs with multiple secrets are neither secure nor efficient to be applied in a RingCT protocol.

In this paper, we propose a novel *any-out-of-many proof*, a logarithmic-sized zero-knowledge proof scheme for showing the knowledge of arbitrarily many secrets out of a public list. Unlike other partial knowledge proofs that have to reveal the number of secrets [ACF21], our approach proves the knowledge of multiple secrets without leaking the exact number of them. Furthermore, we improve the efficiency of our method with a generic inner-product transformation to adopt the Bulletproofs compression [BBB+18], which reduces the proof size to $2\lceil\log_2(N)\rceil+9$.

Based on our proposed proof scheme, we further construct a compact RingCT protocol for privacy cryptocurrencies, which can provide a logarithmic-sized communication complexity for transactions with multiple inputs. More importantly, as the only known RingCT protocol instantiated from the partial knowledge proofs, our protocol can achieve the highest anonymity level compared with other approaches like Omniring [LRR+19]. For other applications, such as multiple ring signatures, our protocol can also be applied with some modifications. We believe our techniques are also applicable in other privacy-preserving scenarios, such as multiple ring signatures and coin-mixing in the blockchain.

I. Introduction

Blockchain-based cryptocurrencies verify and record each transaction through a decentralized network. Specifically, a distributed ledger of all legal transactions is renewed and maintained by every user in the network. As a result, the correctness and immutability of the whole system will be guaranteed in an honest-majority situation. To allow all transactions to be verified easily, some cryptocurrencies record all transaction details in plaintext. For example, the balance and address of each account are accessible to anyone in the Bitcoin network [1]. Such public property inevitably hinders the original blockchain scheme from application in scenarios with privacy concerns. As a result, this deficiency has impelled the development of *private cryptocurrencies* [2], [3], which aim to provide confidentiality and anonymity meanwhile allowing public verification. Currently, there are two main types of privacy protection schemes for cryptocurrency, one is zk-SNARKs-based approach, represented by Zcash and Tornadocash [4]–[6], and the other is ring-signature-based approach, such as Monero [2]. Compared with the former, the ring-signature-based approach does not require trusted setups and is more friendly to nodes with low computational resources. However, it scales only to medium-size anonymity sets, leading to weaker anonymity.

RingCT. *Ring confidential transaction* (RingCT) protocol is a popular privacy-preserving solution applied in Monero [7]–[11]. Generally speaking, they focus on two main aspects: (1) *confidentiality*, hiding the amounts of money to be transferred; (2) *anonymity*, hiding the identities of users (spenders) in a transaction. For the confidentiality property, existing solutions [9]–[11] typically adopt balance proofs to ensure the sums of inputs and outputs are equal, as well as range proofs to ensure the amount of money of each account (i.e., account balance) lying in the valid range, e.g., $[0, 2^{64} - 1]$ in Monero [2]. For the anonymity property, Groth et al. propose the one-out-of-many proof [12], which enables a prover (spender) to prove a statement about *one* message (secret key) among a list of commitments (a set of public keys) in a secure way with logarithmic size. This technique can be used to construct a ring signature scheme that provides anonymity by hiding the identity of the signer and is further used in RingCT protocols.

However, the initial design of one-out-of-many proofs does not consider the demand of dealing with multiple messages, leaving this scheme hard to satisfy the requirements of high TPS (Transaction Per Second) and sophisticated functions in decentralized finance. Besides, the separate implementation of ring signatures and confidential transactions consumes extra computation resources of the users.

Challenges. To state the challenges clearly, we first consider an anonymous transaction with k real source accounts as inputs. This is a prevalent scenario in crytocurrency systems, where a spender may have multiple unspent accounts. A naive way to sign a transaction with one-out-of-many proofs is generating a proof (as the signature) for each real source account under an independently selected ring set. Specifically, k ring signatures are generated in total, and each one consists 1 real source account and $n-1$ decoy source accounts. Accordingly, the transaction size grows linearly with k because it cosists of k signatures and kn public keys. This property may lead to an uneconomic transaction fee, especially for transaction with many real inputs. Meanwhile, we observe that the transaction can only achieve 1-out-of-n anonymity with kn accounts. Besides, the scheme leaks the number of real source accounts (k), which further compromises the anonymity. Due to the inefficiency mentioned above, current anonymous cryptocurrencies can only use small ring sizes with weak anonymity. For example, Monero [2] uses the ring size of 11 and hence is vulnerable to the de-anonymization attacks as shown in [13], [14].

Existing work. To overcome the challenge above, several schemes are proposed [9]–[11], [15]–[17]. Generally speaking, these schemes try to overcome the above difficulties in two directions. One is to optimize the current RingCT structure based on one-out-of-many proofs. Another is to design new proofs with better efficiency for multiple secrets.

Optimizing RingCT with one-out-of-many proofs. RingCT 2.0 [9], RingCT 3.0 [10], Omniring [11], and Anonymous Zether [18] improve the performance of RingCT by rearranging the structure of multiple accounts and applying compression techniques correspondingly. However, the effect is limited since their underlying building block is still one-out-of-many relations.

New proofs for multiple secrets. Thomas et al. [17] present the first known logarithmic-sized partial knowledge proof with extended Bulletproofs compression techniques. This proof can prove the knowledge of k-subset among n public elements in logarithmic size. Although it seems to have a bunch of appealing merits for RingCT protocols, the use of bilinear pairings hinders efficient combinations with existing techniques such as range proofs, and leaking number k weakens the anonymity.

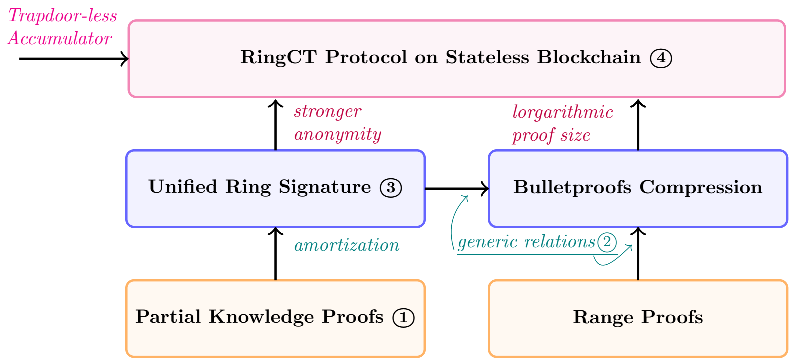

Fig. 1: Structural overview of techniques in this paper. Technique ① is introduced in Section II-B and instantiated in Section V. Technique ② is introduced in Section II-C, and formally shown in Section IV. Technique ③ is introduced in Section II-A. Technique ④ is introduced in Section II-D and formally presented in Section VI.

A. *Our Contributions*

In this paper, we follow the second direction and propose an efficient partial knowledge proof that can be used as a plug-and-play algorithm in RingCT to improve the performance. Specifically, the contributions are as follows:

Bulletproofs compression for generic relations. Inspired by previous work [15], [19], we first extend the current Bulletproofs compression technique to generic relations in Section IV. The generic compression model mainly captures multiple quadratic relations and presents the general transformation approach.

Logarithmic-sized any-out-of-many proofs. Based on the generic compression model, we propose an efficient zero-knowledge proof of knowledge (zkPoK) for partial knowledge relations without revealing the number of secrets. We name this scheme as *any-out-of-many proof* and give the detailed construction with security proofs in Section V.

Efficient instantiation of RingCT. We present a concrete instantiation of the RingCT protocol based on our *any-out-of-many* proofs (Section VI). As the first RingCT constructed with partial knowledge proofs, our scheme does not require a trusted setup or pairings, improves anonymity with a truly unified ring, and has a logarithmic spend proof size.

II. Technical Overview and Related Work

In this section, we briefly introduce some core techniques included in this paper, along with their related work. Figure 1 generally illustrates the building blocks of our RingCT protocol. We first evaluate the weak anonymity of the existing RingCT protocol and explain the motivation for designing a unified ring signature (technique ③) with multiple secret keys in Section II-A. To achieve this, we construct a partial knowledge proof (technique ①) for revealing multiple secrets. Section II-B presents the challenges of proposing an *any-out-of-many* relation and gives a high-level description of our solution. To further optimize our ring signature scheme, we accommodate the Bulletproofs compression technique [19] to our scheme. Accordingly, we propose a generic compression model (technique ②) for aggregating the signature with range proofs and compressing them into logarithmic size. Finally, we instantiate an efficient RingCT protocol based on the stateless blockchain structure with our new proofs (technique ④).

A. *Unified Ring with a Private k*

As mentioned above, current privacy cryptocurrency systems may frequently initiate transactions with multiple inputs [10], [18]. Unfortunately, existing RingCT protocols can scarcely handle these cases

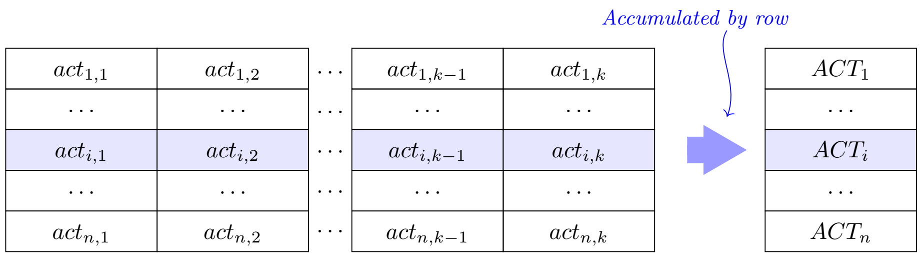

Fig. 2: Arranging k real source accounts $act_{i,1}, ..., act_{i,k}$ in $n \times k$ matrix in RingCT 2.0. Each real source account $act_{i,j}$ belongs to a ring containing $act_{1,j} \sim act_{n,j}$, $j = 1, ..., k$

efficiently. Since the one-out-of-many proof can only prove *one* opening out of n public commitments. As a result, each real source account needs to be hidden in an independent n-sized ring.

Figure 2 shows a typical account arrangement in RingCT 2.0 [9]. Real source accounts in purple boxes consist of public keys and valid coins (non-negative account balances). A spender (prover) generates a signature for each real account, e.g., $act_{i,k}$, and takes other decoy source accounts, e.g., $act_{1,k}, ..., act_{i-1,k}, act_{i+1,k}, ..., act_{n,k}$ in the same column to build one-out-of-many proofs. We use an $n \times k$ matrix to represent the public keys of the transaction corresponding to the accounts in Figure 2. Although RingCT 2.0 presents a practical way to accumulate accounts in the same row into $ACT_1, ..., ACT_n$ to reduce the signature size, the anonymity levele of this scheme remains 1-out-of-n. An adversary who de-anonymizes one real source account can immediately know other real accounts by referring to the same row of the public key matrix in the transaction data.

Several researchers are aware of this problem and propose different solutions, such as many-out-of-many proofs [18], RingCT 3.0 [10], and Omniring [11] as shown in Figure 3. However, none of them realizes a real partial knowledge proof. In many-out-of-many proofs, Diamond et al. propose a public permutation to compress multiple signatures but hardly improve anonymity. RingCT 3.0 and Omniring protocols rearrange the positions of real source accounts in different rows to decrease the relevance between them. Such property is called *anonymity against insider attacks* which can improve the anonymity level. Unfortunately, although the above ideas of overcoming the defects of one-out-of-many proofs are pretty impressive, they can not essentially solve the problems.

Prior to illustrating our idea of improving the performance of existing RingCT schemes, we first conclude two properties that a highly anonymous RingCT needs to satisfy: (1) *a unified ring* and (2) *a private number* k with corresponding justifications as follows.

Unified ring. Lai et al. [11] first proposed the concept of the "unified ring" in Omniring. In this paper, we strengthen this property to provide a higher anonymous level by requiring all source accounts of a single transaction need to share one ring. Concretely, we require that all k real source accounts are distributed uniformly in a unified ring of $N = kn$ size at random. Different from the structure in Figure 3, the unified ring signature can still have $(k-1)$-out-of-$(N-1)$ anonymity when a real source account is de-anonymized. Consider an example of $k = 8$ and $n = 16$ in Figure 2. We denote the positions of real source accounts among the unified ring with a 128-length binary vector $\boldsymbol{b}$ containing 8 bits of "1". The possible value of $\boldsymbol{b}$ is $\binom{128}{8} \approx 2^{40}$ in a unified ring (we denote it as *anonymity space* in the following) since the 8 bits of "1" is uniformly distributed in the vector at random. However, the RingCT 2.0 can only reach $16 = 2^4$ and Omniring and RingCT 3.0 can reach $16^8 = 2^{32}$. We also want to point out that Omniring [11] uses a different notion of the ring size n, i.e., the number of columns in Figure 3. Therefore, in our strengthened definition, their scheme is not a truly unified ring. To the best of our knowledge, the k-out-of-N partial knowledge proof [17] is the only known scheme that can provide a

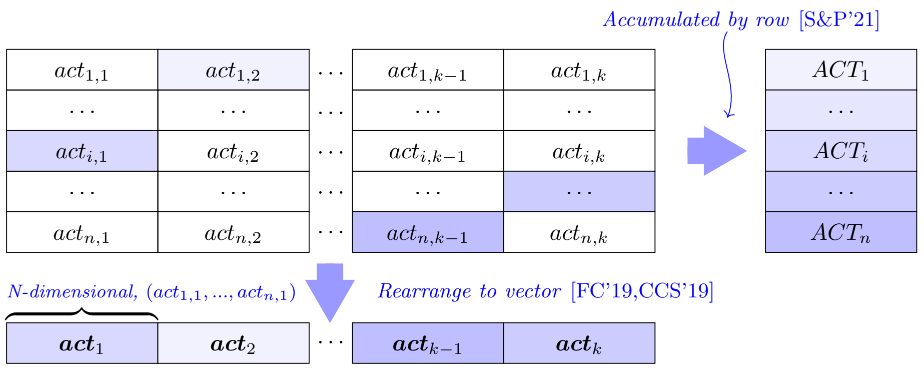

Fig. 3: Arranging k real source accounts $act_{i,1}, act_{1,2}, act_{n,k-1}$ and others in an $n \times k$ matrix in RingCT 3.0 [FC'19], Omniring [CCS'19] and many-out-of-many proofs [S&P'21]. Each real account $act_{*,j}$ belongs to a ring containing $act_{1,j} \sim act_{n,j}$

unified ring and achieve the anonymity space of about 2^{40}.

Private k. we notice that the anonymity space of the unified ring with k-out-of-N proofs is still far less than the ideal upper bound. Since the number of secrets is also unnecessary in some applications such as RingCT, the prover can keep it privately and only prove $\boldsymbol{b}$ is a binary vector in the space of $\{0,1\}^{128}$. Here we claim that $\boldsymbol{b}$ being a zero vector is also legal in anonymous cryptocurrencies, while this case should be excluded in some other applications (more detailed discussions in Section V). Accordingly, the anonymity space achieves 2^{128}. More precisely, we can regard the number k as auxiliary information that is highly correlated to the positions of secrets. Therefore, leaking k increases the probability of correctly guessing the indices of real source accounts. If k is hidden, the signature has $(*-1)$-out-of-$(N-1)$ anonymity after a real source account is de-anonymized ($*$ means "unknown number"), while the $(*-1)$ and $*$ have no difference from the view of adversary's and hence it does no help. As a result, hiding k can significantly improve the anonymity of RingCT protocols. The formal analysis of this point is given in Appendix C.

To achieve the above two properties, redesigning partial knowledge proofs for RingCT protocols seems to be a better way, which gives us an explicit goal for the following work.

B. Efficient Partial Knowledge Proofs without Pairing

Although partial knowledge proofs with randomly distributed secrets can improve the anonymity of RingCT protocols (or ring signatures), designing a logarithmic-sized proof scheme still comes with several technical problems. Among them, transforming the secret key verification into a "compression-friendly" form is a major challenge.

To state it clearly, we use vectors $\boldsymbol{s} \in \mathbb{Z}_q^k$ and $\boldsymbol{P} \in \mathbb{G}^N$ to denote k secrets and N public commitments, and i_j links the index i in $\boldsymbol{P}$ to the index j in $\boldsymbol{s}$, e.g., $P_i = P_{i_j} = g^{s_j}$. Let $\boldsymbol{b} = \{b_i\}_{i=1}^N$ be an N-dimensional binary vector with "1" at the i_j-th position and "0" everywhere else for $j = 1, ..., k$. For instance, the following $\boldsymbol{b}$ indicates that the commitments $(P_2, P_4) = (P_{2_1}, P_{4_2}) = (g^{s_1}, g^{s_2})$ are the commitments to the secrets s_1, s_2 which prover wants to prove.

$$\begin{aligned} \boldsymbol{b} &= (0, && 1, && 0, && 1, && 0, && 0, && 0, && 0), \\ \boldsymbol{P} &= (P_1, && [P_2], && P_3, && [P_4], && P_5, && P_6, && P_7, && P_8). \end{aligned}$$

We further use the binary vector $\boldsymbol{b}$ to show the relation between $\boldsymbol{P}$ and $\boldsymbol{s}$. As shown below, we can compute the exponentiation of each commitment P_i with a bit $b_i \in \boldsymbol{b}$ as the exponent,

$$\boldsymbol{P}^{\boldsymbol{b}} := \prod_{i=1}^{N} P_i^{b_i} = \prod_{j=1}^{k} P_{i_j}^{1} = \prod_{j=1}^{k} g^{s_j}, \tag{1}$$

where the commitments with an exponent of 0 are canceled out from the products.

A direct approach to prove Equation (1) holds is instantiating extra zkPoKs for $\boldsymbol{b}$ and $\boldsymbol{s}$, and checking Equation (1) correspondingly with the encoding of $\boldsymbol{b}$ and $\boldsymbol{s}$. However, the proof size of this approach is linear, which is inefficient and leaks the number of k. One may consider to use Bulletproofs compression to reduce the proof size, but it incurs two additional problems: (i) the Discrete Logarithm (DL) relation may not be secure between elements in $\boldsymbol{P}$; (ii) the relation in Equation (1) can not be transformed into an inner-product form of $\boldsymbol{b}||\boldsymbol{s}$. We think these are the main reasons for RingCT 3.0 [10] and Omnirnig [11] to use only one "1"-bit in each vector $\boldsymbol{b}$.

Thomas et al. introduce a feasible compression approach to Equation (1) [17]. They construct a polynomial with its evaluations to indicate vector $\boldsymbol{b}$. Additionally, they can concatenate $\boldsymbol{b}$ and the polynomial coefficient vector into one and compress them with an extended Bulletproofs compression mechanism for homomorphic relations. This approach is the only known logarithmic-sized partial knowledge proof, but it still has some unwanted properties for RingCT. First, the approach uses a bilinear pairing mechanism to reduce the proof size, which makes it inefficient when combining this approach with range proofs in RingCT constructions such as Omniring [11]. Second, this approach does not consider the "public k" problem as discussed in Section II-A, which leads to a weak anonymity level.

Based on the discussions above, we aim to propose a novel logarithmic-sized partial knowledge proof for a hidden k without pairings. Our main idea is to leverage the amortization technique [17] (in Section III-C) to compress the openings of secret key vector $\boldsymbol{s}$ into a single f_s. Briefly speaking, we use a random challenge $y \in \mathbb{Z}_q$ to encode the secret keys into a field element $f_s = \sum_{j=1}^{k} y^{i_j} \cdot s_j$. Note that the masking value in f_s is omitted for simplicity. Accordingly, Equation (1) can be rewritten as:

$$\boldsymbol{P}^{\boldsymbol{y}^N \circ \boldsymbol{b}} = \prod_{j=1}^{k} P_{i_j}{}^{y^{i_j} \cdot 1} = g^{f_s}. \tag{2}$$

where $\boldsymbol{y}^N \circ \boldsymbol{b} = (yb_1, y^2 b_2, ..., y^N b_N)$.

Since the amortized equation only has *one* field element f_s, it follows the same form as the *one-out-of-many* relation, which enables us to further use Bulletproofs compressions. Perhaps surprisingly, our new partial knowledge proofs have nearly the same proof size as the one-out-of-many proofs. More importantly, since there is no need to restrict the number k of "1" bits in $\boldsymbol{b}$ nor to include k in the relation. Our protocol can be used to prove multiple secrets. The RingCT instantiated from it can achieve both anonymous properties of *unified rings* and *hidden k* described in Section II-A.

We also want to highlight that our partial knowledge proofs have a wider application range beyond the RingCT protocol. For the case that $k = N$, the proof can be used as a Coin-Mixing proof without a trusted setup. We can also instantiate a multi-party ring signature while an extra proof of $k \neq 0$. More detailed discussions are given in Section VIII-B.

C. Generic Compression Model for ZKP of Vectors

For real-world applications, we need to further instantiate the above proof relation as a concrete proof scheme. A straightforward idea is to write the relation into arithmetic circuits and prove it with generic zk-SNARKs such as Groth16 and Sonic [20], [21]. However, this solution has two main problems. First, running a zk-SNARKs system requires a lot of resources from the node, which deteriorates the performance of the cryptocurrency. Second, although existing generic zk-SNARKs systems capture arithmetic circuits satisfiability, it is inefficient to prove the exponentiation constraint in Equation (2). Actually, most zk-SNARKs-based systems use Merkle tree for partial knowledge proofs (which requires a large circuit) instead of Σ-protocols, as mentioned by Attema et al. [17].

As a result, we aim to find an approach to transform our any-out-of-many proofs into a compression-friendly form for Bulletproofs technique [19]. The target scheme is a specific Σ-protocol that can be executed directly by nodes without a trusted setup. Furthermore, we extend the above transformation approach slightly to obtain a more general inner-product transformation technique for quadratic relations. We also introduce feasible aggregation and combination techniques for multiple relations in Section IV, which allows us to aggregate the partial knowledge proofs and combine it with range proofs.

D. *RingCT Protocol based on Stateless Blockchain*

As a direct application of our *any-out-of-many proofs*, we mainly construct the RingCT protocol based on the formalized model proposed by Lai in [11]. By applying a combination technique for two inner-product relations of partial knowledge proofs and range proofs, our RingCT protocol can achieve the same proof size of $\mathcal{O}(Nk + md)$ as Omniring with better anonymity under the exact ring size, where N, k, m represents the size of the ring, the number of real source accounts and target (output) accounts respectively, and 2^d is the maximum currency amount that can be sent in a single transaction.

Besides, to alleviate the burdensome ledger state data requirements in most of the existing RingCT protocols, we improve the original transaction validation mechanism with a trapdoor-less accumulator [22]. As a result, we build a RingCT protocol based on a stateless blockchain, allowing the verifier to validate each transaction efficiently with only a commitment to the ledger state.

III. Security Definitions and Building Blocks

A. *Notation*

Let λ be a security parameter. A commitment key ck is generated with a setup algorithm $ck \leftarrow \mathrm{Setup}(1^\lambda)$, which further indicates a cyclic group $\mathbb{G}$ determined by ck. Let g, h, u, v be generators of $\mathbb{G}$, conforming to the DL assumption. $\mathbb{Z}_q$ denotes the ring of integers modulo q. Let $\mathbb{G}^n$ and $\mathbb{Z}_q^n$ be vector spaces of dimension n over $\mathbb{G}$ and $\mathbb{Z}_q$ respectively, $x \leftarrow\$ \mathbb{Z}_q$ denotes sampling x from $\mathbb{Z}_q$ uniformly at random.

We use bold-type lower-case letters to denote vectors, e.g., $\boldsymbol{a} \in \mathbb{F}^n$ is a vector of elements $a_1, ..., a_n \in \mathbb{F}$, where $\mathbb{F}$ can be $\mathbb{Z}_q$ or $\mathbb{G}$. Let $c\boldsymbol{a} = (c{\cdot}a_1, c{\cdot}a_2, ..., c{\cdot}a_n)$ denotes the scalar product of scalar c and vector $\boldsymbol{a}$; $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{n} a_i b_i \in \mathbb{F}$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$; $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 b_1, ..., a_n b_n) \in \mathbb{F}^n$ denotes the Hadamard product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$; and $\boldsymbol{a}^{\boldsymbol{b}} = \prod_{i=1}^{n} {a_i}^{b_i} \in \mathbb{F}$ denotes the multi-exponentiation of two vectors. For $k \in \mathbb{Z}_q$, we define $\boldsymbol{k}^n$ as $(1, k, k^2, ..., k^{n-1}) \in \mathbb{Z}_q^n$. $|\boldsymbol{a}|$ represents the number of non-zero elements in vector $\boldsymbol{a}$. Let $\boldsymbol{a} \parallel \boldsymbol{b}$ denotes the concatenation of two vectors $\boldsymbol{a} \in \mathbb{Z}_q^n$ and $\boldsymbol{b} \in \mathbb{Z}_q^m$, then $\boldsymbol{a} \parallel \boldsymbol{b} \in \mathbb{Z}_q^{m+n}$.

B. *Commitment Scheme*

A non-interactive commitment scheme (over a commitment key ck) can output a commitment $c \in \mathbb{G}$ based on input secret message $m \in \mathcal{M}$ ($\mathcal{M}$ is the message space) and randomness $r \in \mathcal{R}$ (randomness space) in the commitment stage. In the opening stage, encoding of m and r are sent to allow anyone to verify that c is indeed a commitment to m. We use Com_{ck} to denote a commitment and require that a commitment scheme satisfies the hiding and binding properties: (1) The hiding property requires the commitment not to reveal the message. (2) The binding property requires the commitment can only be opened to one message. Formal definitions are given in Appendix A.

In this paper, we instantiate our zkPoK with the Pedersen commitments [23] and its extended version of Pedersen vector commitments. For $m, r \in \mathbb{Z}_q$, we define $\mathrm{Com}_{ck}(m; r) = g^m h^r \in \mathbb{G}$, and for $\boldsymbol{m} \in \mathbb{Z}_q^n, r \in \mathbb{Z}_q$, define $\mathrm{Com}_{ck}(\boldsymbol{m}; r) = \boldsymbol{g}^{\boldsymbol{m}} h^r = (\prod_{i=1}^{n} g_i^{m_i}) h^r \in \mathbb{G}$.

C. Σ-Protocol and Amortization Technique

A Σ-protocol is a 3-move zero-knowledge protocol between a prover and a verifier in which a binary relation $(x, y) \in \mathcal{R}$ of witness x and statement y is proved. The prover generates and sends a commitment t first, and then the verifier chooses a random public coin c as a challenge. According to the challenge, the prover computes the response z. Finally, the verifier checks the validity of statement y based on the transcript. Typically, a Σ-protocol must satisfy three properties: perfect completeness, special soundness, and HVZK (honest verifier perfect zero-knowledge). The formal definitions are given in Appendix A.

1) Σ-protocol for Opening Homomorphisms: Σ-protocols can be instantiated with different commitment schemes for different relations, providing a basis for the design of zero-knowledge proofs. This paper only focuses on the Σ-protocol for opening homomorphic relations proposed by Thomas et al. [17], built on the Pedersen commitment scheme. Assume a prover wants to prove the knowledge of a value $x \in \mathbb{Z}_q$ and its constraint with a public element $y \in \mathbb{G}$ as $f(x) = y$ (without revealing x), where $f \in \mathrm{Hom}(\mathbb{Z}_q, \mathbb{G})$ is a homomorphism. The relation for the proof of argument can be written as:

$$R_f = \{(A \in \mathbb{G}, y \in \mathbb{G}; x \in \mathbb{Z}_q) : A = g^x, y = f(x)\}.$$

Based on the relation R_f, a zkPoK scheme Π_0 can be constructed with the Pedersen commitment as below:

```
$\Pi_0 : \langle \mathcal{P}(A, y; x), \mathcal{V}(A, y) \rangle$
$\mathcal{P}$ :
  (1) $r \leftarrow\$ \mathbb{Z}_q$
  (2) $B := g^r$
  (3) $t := f(r)$
$\mathcal{P} \rightarrow \mathcal{V} : B, t$
$\mathcal{V}$ :
  (4) $c \leftarrow\$ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V} : c$
$\mathcal{P}$ :
  (5) $z := cx + r$
$\mathcal{P} \rightarrow \mathcal{V} : z$
$\mathcal{V}$ : Check if the following equations hold:
  (6) $g^z \stackrel{?}{=} A^c B$
  (7) $f(z) \stackrel{?}{=} cy + t$
```

According to the results in [17], Π_0 satisfies perfect completeness, special soundness, and special HVZK.

2) Amortization Techniques: we consider a scenario where the prover wants to proves knowledge of multiple secrets $x_1, ..., x_n$ conformed with the same homomorphism f as follows:

$$R_{\mathrm{nf}} = \left\{ \begin{matrix} (A_1, ..., A_n \in \mathbb{G}^n, y_1, ..., y_n \in \mathbb{G}^n; x_1, ..., x_n \in \mathbb{Z}_q^n) : \\ A_1 = g^{x_1}, y_1 = f(x_1), ..., A_n = g^{x_n}, y_n = f(x_n). \end{matrix} \right\}.$$

It is practical to simply run Π_0 n times, while this method will introduce a high communication complexity with both linear-sized commitments $B_1, ..., B_n$ and responses $z_1, ..., z_n$. We use the amortization technique already adopted in a lot of previous work such as Bulletproofs [19] and compressed Σ-protocols [24]. By compressing the n responses $z_1, ..., z_n$ with a challenge c to $z = \sum_{i=1}^{n} c^i x_i + r$, the verifier can check the following equations instead as long as the DL assumption holds.

$$g^z = A_1^c A_2^{c^2} \cdots A_n^{c^n} B \tag{3}$$

$$f(z) = t + cy_1 + c^2 y_2 + \cdots + c^n y_n \tag{4}$$

Note that instead sending n commitments $B_1, ..., B_n$, only a new commitment B of randomness $r \leftarrow\!\!\$\ \mathbb{Z}_q$ needs to be sent.

D. *Bulletproofs Compression Technique*

Bulletproofs protocol is an interactive approach to compress n-size vectors to scalars with $\log(n)$ times of iterations [19]. This scheme is mainly based on the inner-product argument [15], which aims to prove that the prover knows the openings of two Pedersen vector commitments that satisfy an inner-product relation.

We define a commitment scheme with commitment key ck, with cyclic group $\mathbb{G}$ determined by ck and $g, h \in \mathbb{G}$. The inner-product relation guanratees that the prover knows two secret vectors $\boldsymbol{a}, \boldsymbol{b}$ to the commitment B, which satisfying inner-product relation $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = t$. We give the formal definition of relation R_{ip} as

$$R_{\text{ip}} = \left\{ \begin{matrix} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, B \in \mathbb{G}, t \in \mathbb{Z}_q; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^n) : \\ B = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} \wedge \langle \boldsymbol{a}, \boldsymbol{b} \rangle = t. \end{matrix} \right\}.$$

In most cases the inner-product is also embedded into B by using another generator $u \in \mathbb{G}$, and we can obtained a compact commitment $C = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} \cdot u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}$. To reduce this size sent in the opening stage from linear to logarithmic, an elegant compression mechanism is given in [19]: assume that n is an even number and $n' = n/2$, we define the slices of vectors:

$$\begin{aligned} \boldsymbol{a}_L &= (a_0, ..., a_{n'-1}) \in \mathbb{Z}_q^{n'}, \\ \boldsymbol{a}_R &= (a_{n'}, ..., a_{n-1}) \in \mathbb{Z}_q^{n'}, \end{aligned} \tag{5}$$

so do $\boldsymbol{b}_L, \boldsymbol{b}_R, \boldsymbol{g}_L, \boldsymbol{g}_R, \boldsymbol{h}_L, \boldsymbol{h}_R$. Therefore, we can compute compressed vectors $\boldsymbol{a'}, \boldsymbol{b'}$ and corresponding group vectors $\boldsymbol{g'}, \boldsymbol{h'}$ with a random challenge $x \leftarrow\!\!\$\ \mathbb{Z}_q$ as follows:

$$\begin{aligned} \boldsymbol{a'} &= x\boldsymbol{a}_L + \boldsymbol{a}_R, \quad & \boldsymbol{b'} &= \boldsymbol{b}_L + x\boldsymbol{b}_R, \\ \boldsymbol{g'} &= \boldsymbol{g}_L \circ \boldsymbol{g}_R^x, \quad & \boldsymbol{h'} &= \boldsymbol{h}_L^x \circ \boldsymbol{h}_R, \end{aligned} \tag{6}$$

which indicates the following relation holds:

$$C' \quad = (\boldsymbol{g'})^{\boldsymbol{a'}} (\boldsymbol{h'})^{\boldsymbol{b'}} u^{\langle \boldsymbol{a'}, \boldsymbol{b'} \rangle} = L C^x R^{x^2}, \tag{7}$$

where $L = \boldsymbol{g}_L^{\boldsymbol{a}_R} \boldsymbol{h}_L^{\boldsymbol{b}_R} u^{\langle \boldsymbol{a}_R, \boldsymbol{b}_L \rangle}$ and $R = \boldsymbol{g}_R^{\boldsymbol{a}_L} \boldsymbol{h}_R^{\boldsymbol{b}_L} u^{\langle \boldsymbol{a}_L, \boldsymbol{b}_R \rangle}$. The new relation in Equation (7) shows that the original relation R_{ip} still holds on the new elements $\boldsymbol{a'}, \boldsymbol{b'}, \boldsymbol{g'}, \boldsymbol{h'}, L, R$. Therefore, only half of the origin vectors $\boldsymbol{a'}$ and $\boldsymbol{b'}$ and extra two group elements L and R need to be sent, and finally the original vector of length n can be reduced to 1 after $\lceil \log_2(n) \rceil$ rounds of iterations.

IV. Generic Inner-product Transformation

According to the discussion in Section IV, our main task in designing a generic compression model is to generalize and extend the transforming process from quadratic relations to inner-product forms. We start from a quadratic relation and further extend to multiple inner-product relations.

A. Quadratic Relation

We consider the problem of proving the knowledge of a secret vector $\boldsymbol{b} \in \mathbb{Z}_q^n$ with the following relation:

$$R_{\text{quad}} = \left\{(\boldsymbol{g} \in \mathbb{G}^n, B \in \mathbb{G}; \boldsymbol{b} \in \mathbb{Z}_q^n) : B = \boldsymbol{g}^{\boldsymbol{b}}, f(\boldsymbol{b}) = \boldsymbol{0}^n\right\},$$

where f is a quadratic relation in general form as $f(\boldsymbol{b}) = \boldsymbol{\alpha} \circ (\boldsymbol{b} \circ \boldsymbol{b}) + \boldsymbol{\beta} \circ \boldsymbol{b} + \boldsymbol{\gamma} = \boldsymbol{b} \circ (\boldsymbol{\alpha} \circ \boldsymbol{b} + \boldsymbol{\beta}) + \boldsymbol{\gamma}$ with public coefficients $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$. A linear relation can be regarded as a special case of the quadratic relation where $\boldsymbol{\alpha} = \boldsymbol{0}^n$. Here we omit the randomness for hiding because it can be simply represented by concatenation $\boldsymbol{b}' = (\boldsymbol{b}||r)$ and $\boldsymbol{g}' = (\boldsymbol{g}||h)$.

To ensure that $f(\boldsymbol{b}) = \boldsymbol{0}^n$ holds, it is equivalent to show the following equations hold by writing $\boldsymbol{b}$ as $\boldsymbol{b}_0$ and $\boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta}$ as $\boldsymbol{b}_1$ satisfying:

$$\boldsymbol{b}_0 \circ \boldsymbol{b}_1 = -\boldsymbol{\gamma} \ \wedge \ \boldsymbol{b}_1 = \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta}. \tag{8}$$

With a challenge $y \in \mathbb{Z}_q$, relation R_{quad} can be further transformed into inner-product forms:

$$\langle \boldsymbol{y}^n, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \rangle = -\langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle \ \wedge \ \langle \boldsymbol{y}^n, \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta} - \boldsymbol{b}_1 \rangle = 0. \tag{9}$$

We can write the above two equations into one inner-product with another challenge $z \in \mathbb{Z}_q$ as:

$$\begin{aligned} & \langle \boldsymbol{y}^n, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \rangle + z \langle \boldsymbol{y}^n, \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta} - \boldsymbol{b}_1 \rangle \\ \iff \ & \langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle = \delta, \end{aligned} \tag{10}$$

where $\zeta(\boldsymbol{b}_0) = \boldsymbol{b}_0 - z\boldsymbol{1}^n$, $\eta(\boldsymbol{b}_1) = (z\boldsymbol{\alpha} + \boldsymbol{b}_1) \circ \boldsymbol{y}^n$, and $\delta = -\langle \boldsymbol{y}^n, z^2\boldsymbol{\alpha} + z\boldsymbol{\beta} \rangle - \langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle$. With the steps above, we can transform a generic quadratic relation into inner-product forms. We formally conclude this process as Lemma 1.

Lemma 1: Any quadratic relation R_{quad} of a vector $\boldsymbol{b}$ can be transformed into a compression-friendly form of inner-product relation R_{ip} with $\langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle = \delta$, by regarding $\boldsymbol{b}_0 = \boldsymbol{b}$ and $\boldsymbol{b}_1 = \boldsymbol{\alpha} \circ \boldsymbol{b} + \boldsymbol{\beta}$.

Proof: Following the process in Equation (10), we can get expected forms of $\zeta(\boldsymbol{b}_0)$ and $\eta(\boldsymbol{b}_1)$. ■

B. Multiple Inner-product Relations of Same Pairs

With Lemma 1, we can transform any quadratic relation into inner-product forms. Here we further consider the following relation of $\boldsymbol{b}_0, \boldsymbol{b}_1$ with k independent inner-product relations:

$$R_{\text{mip-1}} = \left\{ \begin{aligned} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, B \in \mathbb{G}; \boldsymbol{b}_0, \boldsymbol{b}_1 \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n) : \\ B = \boldsymbol{g}^{\boldsymbol{b}_0} \boldsymbol{h}^{\boldsymbol{b}_1} \wedge \text{IP}_0 \wedge, ..., \wedge \text{IP}_{k-1}. \end{aligned} \right\},$$

where IP_j denotes $\langle \zeta_j(\boldsymbol{b}_0), \eta_j(\boldsymbol{b}_1) \rangle = \delta_j$. $\zeta_j(\boldsymbol{b}_0) = \boldsymbol{\zeta}_j \circ \boldsymbol{b}_0 + \boldsymbol{\mu}_j$ and $\eta_i(\boldsymbol{b}_1) = \boldsymbol{\eta}_j \circ \boldsymbol{b}_1 + \boldsymbol{\nu}_j$ are linear relations for $j = 0, ..., k-1$.

We can also transform the above multiple inner-product relations into one inner product. We formally conclude this process as Lemma 2. We claim that the relation $R_{\text{mip-1}}$ satisfies the following lemma:

Lemma 2: For $R_{\text{mip-1}}$ of a single pair $(\boldsymbol{b}_0, \boldsymbol{b}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$ with independent inner-product relations $\text{IP}_0, ..., \text{IP}_{k-1}$, it can be transformed into R_{ip} of a single pair $(\boldsymbol{\tau}, \boldsymbol{\omega}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$ with an inner-product relation IP.

To show the idea above, we first consider two equations, $\langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle = \delta_0$ and $\langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1) \rangle = \delta_1$ in relation R_{ips}. we can rewrite the two inner-products with a challenge z as follows:

$$\langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle + z \langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1) \rangle = \delta_0 + z\delta_1. \tag{11}$$

The Equation (11) can also be rewritten as one inner-product relation:

$$\langle \tau(\boldsymbol{b}_0), \omega(\boldsymbol{b}_1)\rangle = \delta, \tag{12}$$

where $\tau(\boldsymbol{b}_0) = \boldsymbol{\tau} \circ \boldsymbol{b}_0 + \boldsymbol{\kappa}_0$, $\omega(\boldsymbol{b}_1) = \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}$, and $\delta = \delta_0 + z\delta_1 - \kappa_1 + \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle$.

$\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\kappa}_0, \boldsymbol{\kappa}_1$ are coefficients computed by constraints $\eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z\eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1 = \boldsymbol{\tau} \circ \boldsymbol{b}_1 + \boldsymbol{\omega}$, $\boldsymbol{\kappa}_0 = \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0 + z\boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1$ and $\kappa_1 = \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle + \langle z\boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle$.

We can further iteratively conduct the above approach to transform two inner-product relations into a single inner-product form, and finally convert k-many inner-product relations into one.

When dealing with multiple quadratic relations, $f_i(\boldsymbol{b}) = \boldsymbol{\alpha}_i \circ \boldsymbol{b} + \boldsymbol{\beta}_i = \boldsymbol{0}^n$, we need an additional step as the $\boldsymbol{b}_1$'s in Equation (8) can be different due to distinct $\boldsymbol{\alpha}_i$'s and $\boldsymbol{\beta}_i$'s, i.e., $\boldsymbol{b}_{1,i} = \boldsymbol{\alpha}_i \circ \boldsymbol{b}_0 + \boldsymbol{\beta}_i$. Nevertheless, we have $\boldsymbol{b}_{1,i} = \boldsymbol{\alpha}_i \circ \boldsymbol{\alpha}_1^{-1} \circ (\boldsymbol{b}_{1,1} - \boldsymbol{\beta}_1) + \boldsymbol{\beta}_i$, which indicates a linear relation between $\boldsymbol{b}_{1,i}$ and $\boldsymbol{b}_{1,1}$. Therefore, we can set $\boldsymbol{b}_1 = \boldsymbol{b}_{1,0}$ and convert $\langle \zeta_i(\boldsymbol{b}_0), \eta_i(\boldsymbol{b}_{1,i})\rangle = \delta_0$ to $\langle \zeta_i(\boldsymbol{b}_0), \rho_i(\boldsymbol{b}_1)\rangle = \delta_0$ for all $0 \leq i < n$, where $\rho_i(\cdot)$ is a linear function. By adopting the multiple inner-product transformation, we can rewrite them into one inner-product form.

C. *Multiple Inner-product Relations of Different Pairs*

Another special case is dealing with multiple secrets, $(\boldsymbol{a}_0, \boldsymbol{a}_1)$ and $(\boldsymbol{b}_0, \boldsymbol{b}_1)$ such that $\langle \boldsymbol{a}_0, \boldsymbol{a}_1\rangle = \delta_a$ and $\langle \boldsymbol{b}_0, \boldsymbol{b}_1\rangle = \delta_b$ (e.g., after adopting the transformation of Equation (10) for $f_1(\boldsymbol{a}) = \boldsymbol{0}^n$ and $f_2(\boldsymbol{b}) = \boldsymbol{0}^m$). This can be simply converted into an inner-product form $\langle \boldsymbol{a}_0||(z\boldsymbol{b}_0), \boldsymbol{a}_1||(z\boldsymbol{b}_1)\rangle = \delta_a + z\delta_b$ with a challenge z. For one inner-product relation, the prover can use a similar process in the quadratic relation to encode $\tau(\boldsymbol{b}_0)$ and $\omega(\boldsymbol{b}_1)$ to $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$ respectively. Finally, we can use the Bulletproofs compression [15], [19] directly to reduce the proof size.

We give a formal definition of the above idea as follows. For multiple inner-product relation of different pairs $(\boldsymbol{b}_{0,0}, \boldsymbol{b}_{0,1}), ..., (\boldsymbol{b}_{k-1,0}, \boldsymbol{b}_{k-1,1})$, we can transform the following relation into one R_{ip}

$$R_{\text{mip-2}} = \left\{ \begin{array}{c} (\boldsymbol{g}_j, \boldsymbol{h}_j \in \mathbb{G}^{n_j}, B_j \in \mathbb{G}; \boldsymbol{b}_{j,0}, \boldsymbol{b}_{j,1} \in \mathbb{Z}_q^{n_j} \times \mathbb{Z}_q^{n_j}) : \\ B_j = \boldsymbol{g}_j^{\boldsymbol{b}_{j,0}} \boldsymbol{h}_j^{\boldsymbol{b}_{j,1}} \wedge \text{IP}_j, \forall j = 0, ..., k-1. \end{array} \right\},$$

where IP_j denotes $\langle \zeta_j(\boldsymbol{b}_{j,0}), \eta_j(\boldsymbol{b}_{j,1})\rangle = \delta_j$. $\zeta_j(\boldsymbol{b}_{j,0}) = \boldsymbol{\zeta}_j \circ \boldsymbol{b}_{j,0} + \boldsymbol{\mu}_j$ and $\eta_j(\boldsymbol{b}_{j,1}) = \boldsymbol{\eta}_j \circ \boldsymbol{b}_{j,1} + \boldsymbol{\nu}_j$ are linear relations for $j = 0, ..., k-1$. Note that we use n_j to denote different lengths. The transformation can be concluded with the following lemma:

Lemma 3: For $R_{\text{mip-2}}$ of multiple pairs $(\boldsymbol{b}_{j,0}, \boldsymbol{b}_{j,1}) \in \mathbb{Z}_q^{n_j} \times \mathbb{Z}_q^{n_j}, j = 0, ..., k-1$ with independent inner-product relations $\text{IP}_0, ..., \text{IP}_{k-1}$, it can be transformed into R_{ip} of a single pair $(\boldsymbol{\tau}, \boldsymbol{\omega}) \in \mathbb{Z}_q^{kn} \times \mathbb{Z}_q^{kn}$ satisfying an inner-product relation IP^*.

V. ANY-OUT-OF-MANY PROOFS

This section presents a new technique, *any-out-of-many* proof of partial knowledge without leaking the secret number k. We start by defining the *any-out-of-many* relation and reformulating it into a compression-friendly form. Then we present the basic proof scheme with elaborated descriptions of each step. Finally, we describe how to reduce the proof size of our scheme to logarithmic and discuss its potential applications.

A. *Basic Definitions*

We first define the relation of a partial knowledge proof with the hidden secret number k as follows:

$$R_{*/\text{N}} = \left\{ \begin{array}{c} (ck, \boldsymbol{P} \in \mathbb{G}^N; \boldsymbol{s} \in \mathbb{Z}_q^k) : \\ |s| \in [1, N], P_{i_j} = \text{Com}_{ck}(0; s_j) \ \forall j = 1, ..., k. \end{array} \right\},$$

where ck is the commitment key, $\boldsymbol{P}$ is a public set of commitments generated by ck. Besides, we use $\boldsymbol{s}$ to denote the set of secrets for arbitrary k commitments among $\boldsymbol{P}$, where $|\boldsymbol{s}| = k$. Specifically, each $s_j \in \boldsymbol{s}, j = 1, ..., k$ corresponds to a commitment $P_{i_j} = \mathrm{Com}_{ck}(0; s_j) = g^{s_j} \in \boldsymbol{P}, i_j \in [1, N]$, where i_j is defined in Section II-B that links the index of P_i to s_j. Therefore, the relation $R_{*/\mathrm{N}}$ requires that the prover does know the witness for partial $\boldsymbol{P}$ as secrets $\boldsymbol{s}$, and for each secret $s_j \in \boldsymbol{s}, j = 1, ..., k$, it is valid for P_{i_j}, i.e., it satisfies the relation $P_{i_j} = \mathrm{Com}_{ck}(0; s_j)$.

Following the intuition to prove $R_{*/\mathrm{N}}$ (Section II-B) in zero-knowledge, we describe more concretely. Equation (1) formulates the relation $R_{*/\mathrm{N}}$ by using an N-dimensional binary vector $\boldsymbol{b}$, the prover can send a k-dimensional encoding vector $\boldsymbol{f}_s \in \mathbb{Z}_p^k$ for $\boldsymbol{s}$ directly to the verifier for checking. However, instantiating a zkPoK is neither secure (leaking) nor efficient according to our analysis in Section II-B. Thus, we first amortize s_j's into one field element $f_s \in \mathbb{Z}_p$ as follows:

$$\begin{cases} \boldsymbol{f}_s[1] = ys_1 + r_1 \\ \boldsymbol{f}_s[2] = ys_2 + r_2 \\ \quad \cdots \\ \boldsymbol{f}_s[k] = ys_k + r_k \end{cases} \xrightarrow{\text{Amortize}} f_s = \textstyle\sum_{j=1}^{k} y^{i_j} s_j + r_s, \tag{13}$$

where $\boldsymbol{f}_s[j]$ denotes the j-th element in the vector. According to the amortization technique in Section III-C by Thomas et al. [17], since each s_j satisfies the same form of a homomorphic relation $P_{i_j} = g^{s_j}$, it is secure to amortize them into one encoding value with a challenge value y. The amortized relation is given in Equation (2). Note that Equation (2) has a similar form of the one-out-of-many proofs used in previous work [9]–[11] except that the binary vector $\boldsymbol{b}$ is not restricted to contain only one "1" bit.

The remaining task is to build a binary proof for $\boldsymbol{b}$ with a Bulletproofs-friendly form. It is known that $\boldsymbol{b} \in \{0,1\}^N$ is equivalent to the constraint $\boldsymbol{b} \circ (\mathbf{1}^N - \boldsymbol{b}) = \mathbf{0}$ [25], [26], which is a quadratic form conformed with our model in Section IV-A. Then according to Lemma 1, the binary constraint can be transformed into an inner-product form of $\langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle = \hat{t}$ with an extra challenge $x \in \mathbb{Z}_q$, where $\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \mathbf{1}^N - \boldsymbol{b}$.

Remarks. In some applications such as multiple ring signatures, we also need an additional constraint to ensure $|\boldsymbol{s}| = k$ is in a valid range (more detailed discussions in Section VIII-B). Since the binary form of $\boldsymbol{b}$ only ensures that $|\boldsymbol{s}|$ lies in the range $[0, N]$, the case $|\boldsymbol{s}| = 0$ may allow a malicious prover to prove the knowledge of the empty set, i.e., $\boldsymbol{s} = \emptyset, \boldsymbol{b} = \mathbf{0}^N$. Thankfully, we observe that attackers cannot leverage this case to compromise the security of RingCT since the input coins (account balances) are balanced with output coins by balance proofs. Additionally, with range proofs, the output coins must be zero in this empty secret case. Although a malicious prover can generate valid transactions without any real source account, our RingCT still ensures that no valid coin is transferred in a "zero-to-zero" transaction. In fact, "zero-to-zero" transactions are allowed in some existing anonymous cryptocurrencies such as Monero [2]. The only difference is that the user must own some input accounts with a 0 balance to generate "zero-to-zero" transactions in current approaches, while our approach allows these transactions even without owning any input accounts.

B. Proof Scheme

Before proposing the concrete scheme of our any-out-of-many proofs Π_*, we first define necessary notations as below.

Notations: commitment keys ck, gk are generated under security parameters λ_1, λ_2 and setup algorithm as $ck \leftarrow \mathrm{Setup}(1^{\lambda_1}), gk \leftarrow \mathrm{Setup}(1^{\lambda_2})$. $\boldsymbol{P} = (P_1, ..., P_N)$ represents a public commitment set consisting of $N \geq 1$ elements. Denote secret keys that the prover knows as a set $\boldsymbol{s} \in \mathbb{Z}_q^k$ with k (hidden) number of elements and each element s_j satisfies $P_{i_j} = \mathrm{Com}_{ck}(0; s_j)$. In case the public key encryption algorithm uses different security parameters from our zkPoK scheme, all used field and group elements are generated from a different key gk such as $\mathbb{Z}_q^N, \mathbb{Z}_q, \mathbb{G}$. We further explain Π_* as follows:

```
$\Pi_* : \langle \mathcal{P}(ck, \boldsymbol{g}, \boldsymbol{h}, u, v, \boldsymbol{P}; \boldsymbol{b}, \boldsymbol{s}), \mathcal{V}(ck, \boldsymbol{g}, \boldsymbol{h}, u, v), \boldsymbol{P} \rangle$
$\mathcal{P}$ :
  (1) $\alpha, \beta \leftarrow\!\$\ \mathbb{Z}_q$
  (2) $\boldsymbol{r}_0, \boldsymbol{r}_1 \leftarrow\!\$\ \mathbb{Z}_q^N$
  (3) $\boldsymbol{b}_0 := \boldsymbol{b}, \boldsymbol{b}_1 := \boldsymbol{b}_0 - \boldsymbol{1}^N$
  (4) $A := \boldsymbol{g}^{\boldsymbol{b}_0} \boldsymbol{h}^{\boldsymbol{b}_1} u^{\alpha}, B := \boldsymbol{g}^{\boldsymbol{r}_0} \boldsymbol{h}^{\boldsymbol{r}_1} u^{\beta}$
$\mathcal{P} \rightarrow \mathcal{V} : A, B$
$\mathcal{V}$ :
  (5) $y, z \leftarrow\!\$\ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V} : y, z$
$\mathcal{P}$ : Defines the following polynomials in $X$ and computes $t_1, t_2$
  (6) $\zeta(X) = (\boldsymbol{b}_0 + z \cdot \boldsymbol{1}^N + \boldsymbol{r}_0 \cdot X) \circ \boldsymbol{y}^{\boldsymbol{N}}$
  (7) $\eta(X) = z \cdot \boldsymbol{1}^N + \boldsymbol{b}_1 + \boldsymbol{r}_1 \cdot X$
  (8) $\hat{t}(X) = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle = \delta(y, z) + t_1 X + t_2 X^2$
$\mathcal{P}$ :
  (9) $\tau_1, \tau_2 \leftarrow\!\$\ \mathbb{Z}_q$
  (10) $E := \boldsymbol{P}^{(\boldsymbol{y}^N \circ \boldsymbol{r}_0)}\, \mathrm{Com}_{ck}(0; -r_s)$
  (11) $T_1 := v^{t_1} u^{\tau_1}, T_2 := v^{t_2} u^{\tau_2}$
$\mathcal{P} \rightarrow \mathcal{V} : E, T_1, T_2$
$\mathcal{V}$ :
  (12) $x \leftarrow\!\$\ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V} : x$
$\mathcal{P}$ :
  (13) $\boldsymbol{\zeta} := \zeta(\boldsymbol{b}_0), \boldsymbol{\eta} := \eta(\boldsymbol{b}_1)$
  (14) $\hat{t} := \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$
  (15) $\tau_x := \tau_1 x + \tau_2 x^2$
  (16) $\mu := \alpha + \beta x$
  (17) $f_s := \sum_{j=1}^{k} y^{i_j} s_j + r_s x$
$\mathcal{P} \rightarrow \mathcal{V} : \boldsymbol{\zeta}, \boldsymbol{\eta}, \hat{t}, \tau_x, \mu, f_s$
$\mathcal{V}$ : Checks if the following equations hold :
  (18) $v^{\hat{t}} u^{\tau_x} \stackrel{?}{=} v^{\delta(y,z)} T_1^x T_2^{x^2}$
  (19) $(\boldsymbol{g}')^{\boldsymbol{\zeta}} \boldsymbol{h}^{\boldsymbol{\eta}} u^{\mu} \stackrel{?}{=} A B^x \boldsymbol{g}^{-z \boldsymbol{1}^N} \boldsymbol{h}^{z \boldsymbol{1}^N}$
  (20) $\boldsymbol{P}^{\boldsymbol{\zeta}} \stackrel{?}{=} \mathrm{Com}_{ck}(0; f_s) E^x \boldsymbol{P}^{(-z \boldsymbol{y}^N)}$
  (21) $\hat{t} \stackrel{?}{=} \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$
```

- In steps 1 and 2, the prover randomly samples α, β from $\mathbb{Z}_q$ as the randomness for the Pedersen commitments and $\boldsymbol{r}_0, \boldsymbol{r}_1$ from $\mathbb{Z}_q^N$ as the masking value for the openings.
- In step 3, the prover generates binary vectors $\boldsymbol{b}_0, \boldsymbol{b}_1$ according to the transformation process in Section IV.

- Step 4 computes the commitments A and B of the vectors $\boldsymbol{b}_0, \boldsymbol{r}_0$, $\boldsymbol{b}_1, \boldsymbol{r}_1$ respectively with random values α and β.
- After receiving the challenges, the prover computes the corresponding t_1, t_2 with the opening polynomials $\zeta(X), \eta(X)$ for the inner-product relation in steps 6,7,8.
- In steps 9 and 11, the prover samples random values τ_1, τ_2 from $\mathbb{Z}_q$ and computes commitments T_1, T_2 of t_1, t_2 for the verification in step 8.
- In step 10, the prover computes the commitment E for the verification of Equation (2).
- The prover calculates opening vectors and values in steps 13-17, including two opening vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ of the inner-product expression and their result $\hat{t}$. τ_x, μ are openings of random values, and f_s is the amortized opening of secret $\boldsymbol{s}$.
- Finally, according to the received transcript, the verifier checks that:
  - $\hat{t}$ is correctly computed with Equation (18);
  - the commitments of vectors $\boldsymbol{b}_0, \boldsymbol{b}_1$ are correct with Equation (19), where $(\boldsymbol{g}')^{\boldsymbol{y}^N} = \boldsymbol{g}$ for simplicity;
  - $\boldsymbol{P}^{(\boldsymbol{b} \circ \boldsymbol{y}^N)}$ is a commitment to zero with Equation (3);
  - the inner-product relation between vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ and $\hat{t}$ holds with Equation (4).

Theorem 1: Π_* is perfectly complete, and perfect special HVZK with a constant round complexity. Π_* has computational special soundness under the discrete logarithm assumption on $\mathbb{G}$.

The proof of the above theorem is given in Appendix B.

C. Optimizations

To further reduce the proof size in Π_*, we modify the protocol as follows. First, steps 19 and 20 can be aggregated into one as

$$(\boldsymbol{g}' \circ \boldsymbol{P}')^{\boldsymbol{\zeta}} \boldsymbol{h}^{\boldsymbol{\eta}} u^{\mu} \overset{?}{=} AB^x \boldsymbol{g}^{-z\boldsymbol{1}^N} \boldsymbol{h}^{z\boldsymbol{1}^N} \mathrm{Com}_{ck}(0; f_s) E^x \boldsymbol{P}^{(-z\boldsymbol{y}^N)},$$

where $\boldsymbol{P}' = (P_1^y, P_2^{y^2}, ..., P_N^{y^N}) \in \mathbb{G}^N$. The DL relation holds in the new equation because steps 19 and 20 are committed under different commitment keys ck, gk. It also follows the inner-product form of vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ under the group vectors $\boldsymbol{G} = \boldsymbol{g}' \circ \boldsymbol{P}', \boldsymbol{H} = \boldsymbol{h}$. The new inner-product relation is defined as follows:

$$R_{\text{ip*/n}} = \left\{ \begin{array}{c} (\boldsymbol{G}, \boldsymbol{H} \in \mathbb{G}^N, v, C \in \mathbb{G}; \boldsymbol{\zeta}, \boldsymbol{\eta} \in \mathbb{Z}_q^N) : \\ C = \boldsymbol{G}^{\boldsymbol{\zeta}} \boldsymbol{H}^{\boldsymbol{\eta}} v^{\langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle}. \end{array} \right\},$$

We claim that our new inner-product argument is almost the same as given the relation R_{ip} in Section III-D except that the group vector $\boldsymbol{G}$ is not uniformly sampled from $\mathbb{G}^N$. According to the conclusion (Corollary 1) given in the Omniring [11], we claim that this chosen vector $\boldsymbol{G}$ does not break the security of the inner-product argument.

Next, the prover executes the Bulletproofs compression process described in Section III-D. According to Equation (7), each round the prover compresses vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ into half length with received c from the verifier and sends the group elements L, R. After running the above process for $\lceil \log_2(N) \rceil$ rounds, the two N-dimensional vectors can be reduced to two field elements. And we claim that the following theorem for Π_{ip} holds.

Theorem 2: $\Pi_{\text{ip*/n}}$ is perfectly complete, and perfect special HVZK with $\lceil \log_2(N) \rceil$ round complexity. $\Pi_{\text{ip*/n}}$ has computational witness-extended emulation under DL assumption on $\mathbb{G}$.

Theorem 2 follows theorem 1 and the security of inner-product arguments concluded in Bulletproofs [19].

VI. RingCT Protocol

The RingCT protocol is a direct application of our *any-out-of-many* proofs. In this section, we will give a comprehensive discussion on the instantiation of this protocol based on the formalized definitions in Omniring [11].

A. Improved Tagging Schemes

We first introduce a technique to improve the efficiency of transaction validation for our RingCT protocol. For most existing cryptocurrency systems, transactions can only be validated by full nodes with the ledger's current state. Since ring signatures hide the identity of real spenders (signers), nodes in a RingCT-based system cannot distinguish which transaction output (TXO) is spent. Consequently, they must keep a record of *all* historical transaction outputs with ever-growing size. For example, Omnring [11] uses "tag" to mark each output uniquely, and other schemes apply similar techniques ("serial number" in Zcash [4] and RingCT 2.0 [9], and "key images" in Monero [2] and RingCT 3.0 [10]). Unfortunately, these techniques lead to a state size of gigabytes, raising common concerns about systems' usability and decentralized property as mentioned in [27].

To mitigate this problem, we propose a new transaction validation scheme with *constant* state size based on the trapdoor-less accumulator proposed by Boneh et al. in [22]. Prior to presenting the concrete scheme, we define some necessary notations first.

A hash-function with prime domains maps elements in $\mathbb{Z}$ to odd prime values. Denote $H_{\text{prime}} : \{0,1\}^* \rightarrow \text{Prime}(1^\lambda)$, where $\text{Primes}(1^\lambda)$ is the subset of odd prime generators in $\mathbb{G}$. Several feasible constructions of H_{prime} is presented in [28].

An algorithm $\text{Bezout}(x, y)$ outputs Bézout coefficients $a, b \in \mathbb{Z}$ for a pair of co-prime integers x, y satisfying the relation $ax + by = 1$ [29].

A trapdoorless universal accumulator is denoted with its current state $(A_t, \tilde{s})$ under the RSA assumption, where $A_t = g^{\tilde{s}} \bmod n \in \mathbb{G}$, and $\tilde{s}$ is the production of all accumulated odd prime generators in $\text{Primes}(1^\lambda)$. Typically, the accumulator has the following core operations:

- $\mathbf{AccGet}()$: Return current accumulator state $(A_t, \tilde{s})$.
- $\mathbf{AccAdd}(A_t, x)$: Add a new item $x \in \text{Primes}(1^\lambda)$ to A_t by computing $A_{t+1} = A_t^x \bmod n$.
- $\mathbf{AccMemWitGen}(A_t, \tilde{s}, x)$: Compute a membership witness for x, i.e., the accumulator without x as $g^{\tilde{s}/x}$. Return $w = g^{\tilde{s}/x}$.
- $\mathbf{AccNonMemWitGen}(A_t, \tilde{s}, x)$: Compute a non-membership witness for x based on the fact that $\text{gcd}(x, \tilde{s}) = 1$, i.e., $(a, b) \leftarrow \text{Bezout}(\tilde{s}, x)$. Return $w = (a, b)$.

Tagging schemes can be regarded as one-way permutations over group elements and therefore are used to mark the secret key uniquely. Lai et al. present a formalized description of tagging schemes in [11] as:

- $\mathbf{TagSetup}(1^\lambda)$: Sample the commitment key ck based on the security parameters 1^λ. Return ck.
- $\mathbf{TagKGen}(sk)$: Compute $pk = \text{Com}_{ck}(0; sk)$ based on the input sk. Return pk.
- $\mathbf{TagEval}(sk)$: Compute $\text{tag} = \text{Com}_{ck}(0; sk^{-1})$ based on the input sk. Return tag.

A non-interactive proof of exponentiation algorithm $\mathbf{NIPoE}(x, A, B)$ is proposed by Boneh [22], it generates a proof π to shown the relation that $A^x = B$ holds.

We improve the original tagging scheme with the accumulator [22] for our new RingCT protocol as follows:

By applying the improved tagging scheme ($\mathbf{TagSetup}^*$, $\mathbf{TagKGen}^*$,$\mathbf{TagPf}^*$,$\mathbf{TagVf}^*$), verifiers only need to check whether the proof output by $\mathbf{TagVf}^*$ is correct or not without comparing with all previous tag's.

We further introduce an amortization technique for multiple tags to reduce the proof size. Denote $\mathrm{tag}_{i_j} = \mathrm{Com}_{ck}(0; sk^{-1})$, we map k tags of real source accounts to $\boldsymbol{Q} \in \mathbb{G}^N$ with the same $\boldsymbol{b}$ in the ring signature. The prime tag $\mathrm{Q}_{i \neq i_j}$ of other ring accounts is generated with a randomly sampled tag. The general idea is amortizing $\boldsymbol{Q}$ in the same way as (2):

$$\boldsymbol{Q}^{\boldsymbol{y}^N \circ \boldsymbol{b}} = \prod_{j=1}^{k} Q_{i_j}{}^{y^{i_j} \cdot 1} = \tilde{Q}, \tag{14}$$

where $y \in \mathbb{Z}_q$ is the challenge and $\boldsymbol{b}$ is the binary vector defined in Section II-B. Accordingly, we can run the tagging scheme on the amortized value $\tilde{Q}$. It is not hard to see that accumulating $Q_{i_j}{}^{y^{i_j}}$ does not incur security problems in our tagging scheme since $\gcd(a^m, b^n) = 1$ holds if $\gcd(a, b) = 1$, where $a, b \in \mathrm{Primes}(1^\lambda), m, n \in \mathbb{Z}_q$.

```
AccSetup*(1^λ)
1 : $g \leftarrow\$ \mathbb{G}$
2 : $H_{\mathrm{prime}} : \mathbb{Z} \to \mathrm{Prime}(1^\lambda)$
3 : return $\mathrm{pp} = (g, H_{\mathrm{prime}})$

TagSetup*(pp, 1^λ)
1 : $ck \leftarrow \mathbf{TagSetup}(1^\lambda)$
2 : $(A_t, \tilde{s}) \leftarrow \mathbf{AccGet}()$
3 : return $\mathrm{pp}_{\mathrm{Tag}} = (ck, (A_t, \tilde{s}), H_{\mathrm{prime}}, g)$

TagKGen*(pp_Tag, sk)
1 : $\mathrm{tag} \leftarrow \mathbf{TagKGen}(sk)$
2 : $Q \leftarrow H_{\mathrm{prime}}(\mathrm{tag})$
3 : return $\mathrm{tag}, Q$

TagPf*(pp_Tag, p, (a, B))
1 : $A_{t+1} \leftarrow \mathbf{AccAdd}(A_t, Q)$
2 : $(a, b) \leftarrow \mathbf{AccNonMemWitGen}(A_t, \tilde{s}, Q)$
3 : $\pi_1 \leftarrow \mathbf{NIPoE}(Q, A_t, A_{t+1})$
4 : return $\pi_{\mathrm{Tag}} = (\pi_1, \pi_2)$

TagVf*(pp_Tag, π_Tag)
1 : $\pi_{\mathrm{Tag}}$ as $(\pi_1, \pi_2)$
2 : $b_0 \leftarrow \mathbf{NIPoE.Vf}(\pi_1)$
3 : $b_1 \leftarrow \mathbf{NIPoE.Vf}(\pi_2)$
4 : return $b_0 \cdot b_1$
```

Fig. 4: New tagging scheme.

B. Notation and Optimization

As mentioned in Section I, a transaction in a RingCT protocol needs to satisfy two properties: anonymity and confidentiality. For anonymity, we can build an efficient ring signature by leveraging the any-out-of-many proofs in Section V. For confidentiality, two specific constraints need to be ensured: (1) the balance between the inputs and outputs of the transaction and (2) the validity of the range of each output. Since these proof schemes are well-defined in previous work [9], [19], we do not give formal definitions due to space constraints. Here, we mainly focus on two optimization techniques. The first is rewriting the balance proof into amortization form and therefore combining it with our ring signature and tagging scheme. The second is aggregating the combined proofs with range proofs by concatenating the binary vector $\boldsymbol{b}$ and the vectors of value in range proofs. Both the above two techniques can reduce the proof size and verification complexity of our RingCT scheme.

We define the parameters used and introduce two optimization techniques in building the RingCT protocol. Define $\mathcal{R}, \mathcal{S}, \mathcal{T}$ as the set of ring accounts, real source accounts, and target accounts, respectively. Each account consists of the corresponding public key and coin. Other parameters used are defined in Table I.

1) Combining Balance Proofs: The balance proofs are used to prove the "balance" property of transaction amounts on both sides, i.e., the amount vector $\boldsymbol{a}_{\mathcal{S}}$ of the real source accounts and the amount

TABLE I: Notations of Parameters in RingCT

Notation	Description
$\boldsymbol{P} \in \mathbb{G}^{\lvert\mathcal{R}\rvert}$	Vector of ring account public keys.
$\boldsymbol{s} \in \mathbb{Z}_q^{\lvert\mathcal{S}\rvert}$	Vector of source account secret keys.
$\boldsymbol{b} \in \{0,1\}^{\lvert\mathcal{R}\rvert}$	Binary vector for any-out-of-many proofs.
$\boldsymbol{a}_{\mathcal{S}} \in \mathbb{Z}_q^{\lvert\mathcal{S}\rvert}$	Amount vector of source accounts.
$\boldsymbol{C}_{\mathcal{R}} \in \mathbb{Z}_q^{\lvert\mathcal{R}\rvert}$	Coin vector of ring accounts
$\boldsymbol{a}_{\mathcal{T}} \in \mathbb{Z}_q^{\lvert\mathcal{T}\rvert}$	Amount vector of target accounts.
$\boldsymbol{C}_{\mathcal{T}} \in \mathbb{Z}_q^{\lvert\mathcal{T}\rvert}$	Coin vector of target accounts.
$\mathbf{tag} \in \mathbb{G}^{\lvert\mathcal{R}\rvert}$	Tag vector of ring account.
$\boldsymbol{Q} \in \mathbb{G}^{\lvert\mathcal{R}\rvert}$	Prime tag vector generated from $\mathbf{tag}$.
$ck \leftarrow \mathrm{Setup}(1^\lambda)$	Commitment keys for tagging scheme.
$gk \leftarrow \mathrm{Setup}(1^\lambda)$	Commitment keys for Pedersen commitment scheme.

vector $\boldsymbol{a}_{\mathcal{T}}$ of the target accounts should satisfy the following relation:

$$\sum_{j=1}^{|\mathcal{S}|} a_{\mathcal{S},j} = \sum_{i=1}^{|\mathcal{T}|} a_{\mathcal{T},i}. \tag{15}$$

To provide confidentiality for $\boldsymbol{a}_{\mathcal{S}}$ and $\boldsymbol{a}_{\mathcal{T}}$, the prover needs to compute the commitments $C_{\mathcal{S},j} = \mathrm{Com}_{gk}(a_{\mathcal{S},j}; r_{\mathcal{S},j})$ for all $j = 1, ..., |\mathcal{S}|$ and $C_{\mathcal{T},j} = \mathrm{Com}_{gk}(a_{\mathcal{T},i}; r_{\mathcal{T},i})$ for all $i = 1, ..., |\mathcal{T}|$ as the coins of accounts.

Additionally, we need to use ring signatures (any-out-of-many proofs) to provide anonymity for the relation above. Specifically, we use the binary vector $\boldsymbol{b}$ in ring signature to indicate the indices of an $|\mathcal{S}|$-dimensional vector $\boldsymbol{C}_{\mathcal{S}}$ in a $|\mathcal{R}|$-dimensional vector $\boldsymbol{C}_{\mathcal{R}}$ (the unified ring) The amounts of ring coins not in $\boldsymbol{C}_{\mathcal{S}}$ can be generated as randoms.

Therefore, the modified relation of $\boldsymbol{C}_{\mathcal{R}}$ and $\boldsymbol{C}_{\mathcal{T}}$ is

$$\boldsymbol{C}_{\mathcal{R}}^{\boldsymbol{b}} = \prod_{i=1}^{|\mathcal{R}|} C_{\mathcal{R},i}^{b_i} = \prod_{j=1}^{|\mathcal{S}|} C_{\mathcal{S},i_j}^{b_i} = (\prod_{i=1}^{|\mathcal{T}|} C_{\mathcal{T},i}) \cdot R \tag{16}$$

where $R = \mathrm{Com}_{gk}(0; \sum_{j=1}^{|\mathcal{S}|} r_{\mathcal{S},j} - \sum_{i=1}^{|\mathcal{T}|} r_{\mathcal{T},i})$. It is not hard to see that (16) has a similar form with (2) of amortized secret key verification: both of them share the common vector $\boldsymbol{b}$. Thus, we can multiple each pair of P_i and $C_{\mathcal{R},i}$ into one base $P_i \cdot C_{\mathcal{R},i}$ for $i = 1, ..., k$ safely since P_j's and $C_{\mathcal{R},i}$'s are from different generators (ck and gk).

Similarly, this technique can also be applied to the relation of prime tag vector $\boldsymbol{Q}$ given in (14) and the relation of tag vector $\mathbf{tag}$ as follows:

$$\mathbf{tag}^{\boldsymbol{y}^N \circ \boldsymbol{b}} = \prod_{j=1}^{|\mathcal{S}|} \mathrm{tag}_{i_j}{}^{y^{i_j} \cdot 1} = \mathrm{Com}_{ck}(0; \sum_{j=1}^{|\mathcal{S}|} y_{i_j} s_j{}^{-1}). \tag{17}$$

Finally, we can combine the three proof schemes above to obtain a compact relation of $\boldsymbol{b}$ with commitments $\{(P_i^{u^2} \cdot \mathrm{tag}^u \cdot Q)^{y^i} \cdot C_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}$ as bases. This compact relation reduces three group elements of the proof size.

2) Aggregating Range Proofs: Besides balance proofs, the prover also needs to show that each amount in $\boldsymbol{a}_{\mathcal{T}}$ lies in a valid range of $[0, 2^{\beta} - 1]$ with range proofs. According to Bulletproofs [19], the prover first computes the binary representation of each value $a_{\mathcal{T},i}$ for all $i = 1, ..., |\mathcal{T}| - 1$ as $\boldsymbol{b}_{\mathcal{T},i} \in \{0,1\}^{\beta}$.

The prover transforms the relation of $\boldsymbol{b}_{\mathcal{T},i}$ into an inner-product form and computes the opening vector $\boldsymbol{\zeta}_{\mathcal{T},i}$ and $\boldsymbol{\eta}_{\mathcal{T},i}$ for $i = 1, ..., |\mathcal{T}|$ according to our generic compression model in Section IV. Similarly, we denote the opening vectors of the any-out-of-many proofs as $\boldsymbol{\zeta}_{\mathcal{R}}, \boldsymbol{\eta}_{\mathcal{R}}$. According to Lemma 3, it is feasible to aggregate the multiple inner-product relations above into one with vectors:

$$\begin{aligned} \boldsymbol{\tau} &= (\boldsymbol{\zeta}_{\mathcal{R}} || \boldsymbol{\zeta}_{\mathcal{T},1} || \cdots || \boldsymbol{\zeta}_{\mathcal{T},|\mathcal{T}|}) \in \mathbb{Z}_q^{|\mathcal{R}|+\beta|\mathcal{T}|}, \\ \boldsymbol{\omega} &= (\boldsymbol{\eta}_{\mathcal{R}} || \boldsymbol{\eta}_{\mathcal{T},1} || \cdots || \boldsymbol{\eta}_{\mathcal{T},|\mathcal{T}|}) \in \mathbb{Z}_q^{|\mathcal{R}|+\beta|\mathcal{T}|}. \end{aligned} \tag{18}$$

As a result, the above aggregation technique can reduce the proof size of RingCT from $O(\log(|\mathcal{R}|) + \log(\beta|\mathcal{T}|))$ to $O(\log(|\mathcal{R}| + \beta|\mathcal{T}|))$.

```
$\mathbf{Setup}(1^{\lambda}, 1^{\alpha}, 1^{\beta})$
1 :  $\mathrm{pp}_{\mathrm{HC}} \leftarrow \mathrm{HCSetup}(1^{\lambda})$
2 :  $\mathrm{pp}_{\mathrm{Tag}} \leftarrow \mathrm{TagSetup}(1^{\lambda})$
3 :  $\mathrm{pp} := (\beta, \mathrm{pp}_{\mathrm{HC}}, \mathrm{pp}_{\mathrm{Tag}})$
4 :  $\mathbf{return}\ \mathrm{pp}$
```

```
$\mathbf{SAKGen}(\mathrm{pp})$
1 :  $s \leftarrow\!\$\ \mathbb{Z}_q$
2 :  $P \leftarrow \mathrm{TagKGen}(s)$
3 :  $\mathrm{tag} \leftarrow \mathrm{TagEval}(s)$
4 :  $Q \leftarrow H_{\mathrm{Prime}}(\mathrm{tag})$
5 :  $\mathbf{return}\ (P, s, \mathrm{tag}, Q)$
```

```
$\mathbf{OTAccGen}(\mathrm{pp,P,a})$
1 :  $r \leftarrow\!\$\ \mathbb{Z}_q$
2 :  $C := \mathrm{Com}_{gk}(a; r)$
3 :  $\mathrm{act} := (P, C)$
4 :  $\mathrm{ask} := (a, r)$
5 :  $\mathbf{return}\ (\mathrm{act}, r)$
```

```
$\mathbf{Spend}(\mathrm{pp}, \{\mathrm{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}, \mathbf{tag}, \{\mathrm{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}, \mu)$
1 :  $\mathrm{tx} := \mathrm{tx}(\{\mathrm{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}, \mathbf{tag}, \{\mathrm{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}, \mu)$
2 :  $\mathrm{stmt} := ((\boldsymbol{P}, \boldsymbol{C}_{\mathcal{R}}), \boldsymbol{C}_{\mathcal{T}}, \mathbf{tag}, \tilde{Q})$
3 :  $\mathrm{wit} := (\boldsymbol{b}, \boldsymbol{s}, (\boldsymbol{a}_{\mathcal{S}}, \boldsymbol{r}_{\mathcal{S}},), (\boldsymbol{a}_{\mathcal{T}}, \boldsymbol{r}_{\mathcal{T}}))$
4 :  $\sigma \leftarrow \mathrm{NIPoK}(\mathrm{stmt}, \mathrm{wit})$
5 :  $\mathbf{return}\ (\mathrm{tx}, \sigma)$
```

```
$\mathbf{Vf}(\mathrm{pp,tx},\sigma)$
1 :  $\mathbf{parse}\ \mathrm{tx}\ \mathbf{as}\ (\{\mathrm{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}, \mathbf{tag}, \{\mathrm{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}, \mu)$
2 :  $\mathbf{compute}\ \mathrm{stmt} := ((\boldsymbol{P}, \boldsymbol{C}_{\mathcal{R}}), \boldsymbol{C}_{\mathcal{T}}, \mathbf{tag}, \tilde{Q})$
3 :  if $|\mathcal{T}| > 2^{\alpha}$ then $\mathbf{return}\ 0$
4 :  $b \leftarrow \mathrm{NIPoK.Vf}(\mathrm{stmt}, \sigma, \mathrm{tx})$
5 :  $\mathbf{return}\ b$
```

Fig. 5: RingCT construction.

C. Protocol Instantiation

The RingCT protocol requires non-interactive zero-knowledge proofs to spend the coins with a valid transaction. In general, the relation of proofs should include the any-out-of-many proofs, range proofs, balance proofs, and tag proofs we mentioned before. By transformation and optimization, the relation can be written into a compact inner-product form and further be compressed to logarithmic size by the Bulletproofs technique [19]. Furthermore, according to the Fiat-Shamir heuristic [30], the prover can transform the interactive proof of knowledge into a non-interactive one (also known as the signature of knowledge) under the Random Oracle Model (ROM) [31].

$$R_{\text{NI-PoK}}(\text{pp}_{\text{Tag}},\text{pp}_{\text{HC}}) := \left\{ \begin{array}{c} \left(\mathbf{stmt} = ((\boldsymbol{P},\boldsymbol{C}_{\mathcal{R}}),\boldsymbol{C}_{\mathcal{T}},\mathbf{tag},\tilde{Q}); \mathbf{wit} = (\boldsymbol{b},\boldsymbol{s},(\boldsymbol{a}_{\mathcal{S}},\boldsymbol{r}_{\mathcal{S}},),(\boldsymbol{a}_{\mathcal{T}},\boldsymbol{r}_{\mathcal{T}}))\right) : \\ \forall j \in 1,...,|\mathcal{S}|, \begin{cases} P_{i_j} = \text{TagKGen}(0;s_j); \\ C_{\mathcal{R},i_j} = \text{Com}_{gk}(a_{\mathcal{S},j},r_{\mathcal{S},j}); \\ \text{tag}_{i_j} = \text{TagEval}(0;s_j^{-1}); \end{cases} \\ \forall i \in 1,...,|\mathcal{T}|, \begin{cases} C_{\mathcal{T},i} = \text{Com}_{gk}(a_{\mathcal{T},i},r_{\mathcal{T},i}); \\ a_{\mathcal{T},i} \in [0,2^{\beta}-1], \end{cases} \\ \tilde{Q} \text{ is well-formed and valid in the accumulator.} \end{array} \right\},$$

We denote our non-interactive zero-knowledge proof of knowledge as NI-PoK. And the relation $R_{\text{NI-PoK}}(\text{pp}_{\text{Tag}},\text{pp}_{\text{HC}})$ is given as follows, where pp_{Tag} is the public parameters for tagging scheme, pp_{HC} is the public parameters for Pedersen commitment scheme.

We propose a new RingCT protocol with $\Phi = (\mathbf{Setup},\mathbf{SAKGen},\mathbf{OTAccGen},\mathbf{Spend},\mathbf{Vf},\mathbf{CheckTag})$. Specifically, the protocol is designed as follows:

- $\mathbf{Setup}(1^{\lambda},1^{\alpha},1^{\beta})$: Take the security parameter 1^{λ} and integers $1^{\alpha},1^{\beta}$ as inputs, where 1^{α} indicates the upper bound of the number of outputs in the transaction (2^{α}), 1^{β} indicates the upper bound of the account balance (2^{β}). Call $\text{HCSetup}(1^{\lambda})$ to generate commitment key gk for Pedersen commitment scheme and hash function $H:\{0,1\}^* \to \{0,1\}^{\lambda}$ for NIPoK. Call $\text{TagSetup}(1^{\lambda})$ to generate commitment key ck for tagging scheme, current state $(A_t,\tilde{s})$, $H_{\text{Prime}}:\{0,1\}^* \to \text{Prime}(1^{\lambda})$ and g for the accumulator. Return $\text{pp} = (\beta,ck,H,gk,(A_t,\tilde{s}),H_{\text{Prime}},g)$.
- $\mathbf{SAKGen}(\text{pp})$: Sample $s \leftarrow\$ \mathbb{Z}$ as the secret key. Call $\text{TagKGen}(s)$ to compute P as the public key. Call $\text{TagEval}(s)$ to compute tag as the tag. Compute prime tag Q with $H_{\text{Prime}}(\text{tag})$. Return one-time key pair (P,s,tag,Q).
- $\mathbf{OTAccGen}(P,a)$: Mint a coin $C = \text{Com}_{gk}(a;r)$ with the amount a and a randomly generated value r. Generate account $\text{act} = (P,C)$. Generate account key $\text{ask} = (a,r)$. Return (act,r).
- $\mathbf{Spend}(\{\text{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|},\mathbf{tag},\{\text{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|},\mu)$: Compute prime tag $\tilde{Q}$ with $\boldsymbol{Q}$ of real source accounts. Generate transcript tx with $\{\text{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}$, $\mathbf{tag}$, $\{\text{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}$ and μ. Generate the signature σ with algorithm NIPoK algorithm, inputs stmt, wit, and tx. Return (tx,σ) as the transaction.

- $\mathbf{Vf}(\text{pp},(\text{tx},\sigma))$: Parse tx as $\{\text{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}$, $\mathbf{tag}$, $\{\text{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}$ and μ. Generate $\mathbf{stmt} = ((\boldsymbol{P},\boldsymbol{C}_{\mathcal{R}}),\boldsymbol{C}_{\mathcal{T}},\mathbf{tag},\tilde{Q})$. Check the signature σ with algorithm NIPoK.Vf, inputs stmt, and tx. Return true if all check passes, and false otherwise.

Note that we omit the steps generating master keys and receiving coins since they are almost identical to Omniring [11]. A sketch of the security properties for our RingCT protocol is also given as follows. The formal security definitions and their proofs can be referred to in appendix D.

Theorem 3: (Balance). Our protocol is unforgeable if the *DL* assumption holds in $\mathbb{G}$ in the ROM. Our protocol is equivalent w.r.t. insider corruption if the *DL* assumption holds in $\mathbb{G}$ in the ROM and the range proof is zero-knowledge. Our protocol is linkable w.r.t. insider corruption if the *DL* assumption holds in $\mathbb{G}$ in the ROM.

Theorem 4: (Anonymity). Our protocol is anonymous against the recipient, anonymous against ring insider, and anonymous under spender revelation if the q-decisional Diffie-Hellman inversion assumption holds in $\mathbb{G}$ in the ROM, where q is the number of Spend oracle query.

Theorem 5: (Non-slanderability). Our protocol is non-slanderable w.r.t. insider corruption if the *DL* assumption holds in $\mathbb{G}$ in the ROM.

VII. Evaluation

A. *Any-out-of-Many Proofs*

We first compare the proof size of our ZKP method with existing approaches [15], [17]. The theoretical performance of *any-out-of-many* proofs is given in Table II.

TABLE II: Comparison of the proof sizes between our schemes and other membership proof approaches [15], [17] for k secrets in a ring set of N public elements.

Scheme	# $\mathbb{G}$	# $\mathbb{Z}_q$	Pairing
$1/N$ [15]	$2\lceil\log_2 N\rceil + 7$	7	No
k/N [17]	$4\lceil\log_2(2N-k+1)\rceil - 5$	4	No
k/N with pairing [17]	$2\lceil\log_2(2N-k+1)\rceil - 1$ in $\mathbb{G}_T$	7	Yes
$*/N$ (this work)	$2\lceil\log_2 N\rceil + 3$	6	No

Note that the protocols above are all non-interactive versions, and some optimization techniques for reducing the constant terms are applied accordingly, Clearly, our proof scheme has the same coefficient of the logarithmic term as the one-out-of-many proof scheme, while the constant term is relatively small. For the k-out-of-N proof scheme [17], it seems that the bilinear pairing can help the scheme achieve better performance. However, we want to highlight that this scheme is impractical in real-world applications because the group elements in $\mathbb{G}_T$ is far larger than $\mathbb{G}$ under the same security level. For instance, the bls12-381 with pairing $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ can achieve 128-bit security with 381-bit field modulus q. Unfortunatelly, the pairing output in $\mathbb{G}_T$ is an element in the extended field $\mathbb{F}_{q^{12}}$, which is almost 18-times of a group element in secp256k1.

We also validate the theoretic results by implementing our *any-out-of-many* proofs scheme in Golang on both secp256k1 [32] and bls12-381 [33] (128-bit secure bilinear mapping curves for k-out-of-N proofs [17]) The proof size in terms of the ring size is shown in Figure 6. It is obvious that our scheme has a smaller asymptotic proof size compared with k-out-of-N proofs, and a smaller constant size than one-out-of-many proofs.

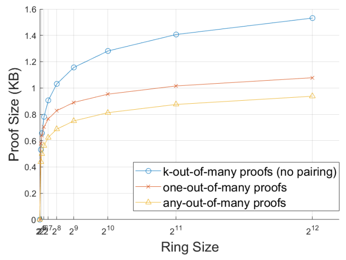

Fig. 6: Proof size comparison of our any-out-of-many proofs, partial knowledge proofs [17], and many-out-of-many proofs [18] with different size of public set.

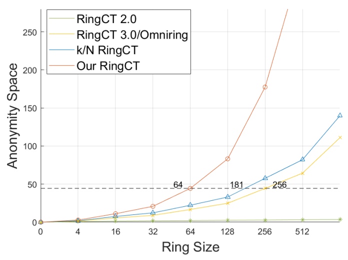

Fig. 7: Anonymity space comparison of our RingCT, Omniring [11], RingCT 3.0 [10] and RingCT 2.0 [9] with different ring size (number of output accounts $|\mathcal{T}| = |\mathcal{R}|/16$).

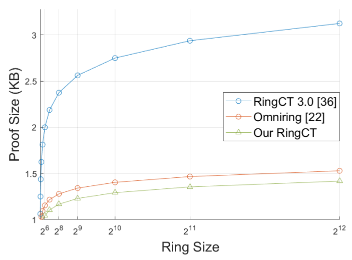

Fig. 8: Proof size comparison of our RingCT, Omniring [11], and RingCT 3.0 [10] with different ring size (number of output accounts $|\mathcal{T}| = |\mathcal{R}|/16$).

B. RingCT Protocol

We further compare the performance between our RingCT protocol, RingCT 3.0 [10], and Omniring [11]. First, we define a transaction with N input accounts, k real accounts among them, m output accounts, and 2^β upper bound of the amount. Suppose the range proof size takes $2\log_2(\beta|\mathcal{T}|)+9$ elements for each scheme (based on Bulletproofs [19]), the theoretical comparison of communication cost (including both the sizes of signature and range proofs) as well as anonymity space is given in Table III. As the RingCT 3.0 only aggregate k signatures, its proof size is $\mathcal{O}(\log(\beta|\mathcal{R}||\mathcal{T}|))$. Our RingCT scheme, in contrast, aggregates the signature with range proofs, which leads to a much smaller size of $\mathcal{O}(\log(|\mathcal{R}|+\beta|\mathcal{T}|))$.

TABLE III: Comparison of the proof size and anonymity space between our proofs and other RingCTs

Scheme	**Proof size** (# $\mathbb{G}$ and $\mathbb{Z}_q$)	**Anonymity**	**Pairing**	**Trusted Setup**
RingCT 2.0 [9]	$2\lceil\log_2(\beta|\mathcal{R}||\mathcal{S}||\mathcal{T}|)\rceil$	$(|\mathcal{R}|/|\mathcal{S}|)^{|\mathcal{S}|}$	Yes	Yes
RingCT 3.0 [10]	$2\lceil\log_2(\beta|\mathcal{R}||\mathcal{S}||\mathcal{T}|)\rceil$	$(|\mathcal{R}|/|\mathcal{S}|)^{|\mathcal{S}|}$	No	No
Omniring [11]	$2\lceil\log_2((3+2|\mathcal{R}|+4|\mathcal{S}|+\beta|\mathcal{T}|)\rceil$	$\binom{|\mathcal{R}|}{|\mathcal{S}|}$	No	No
Our RingCT	$2\lceil\log_2(|\mathcal{R}|+\beta|\mathcal{T}|)\rceil$	$2^{|\mathcal{R}|}$	No	No

We further compare the *anonymity space* of RingCT protocols under different ring sizes as shown in Figure 7. Clearly, our approach can achieve the highest anonymity level with the smallest ring size. Taking an empirical anonymity space of 44.3 for real-world applications (represented by the dotted line), it only requires a 64-sized ring in our scheme, while k-out-of-N requires a ring of size 181 and RingCT 3.0 requires a ring of size 256, which are nearly 1.8 and 3 times more.

To analyze the real-world performance of our RingCT protocol, we build a prototype in Golang based on the elliptic curve Secp256k1 library with 128-bit security [34]; the codes can be referred to the link (https://github.com/egoistzty/Any-out-of-Many-Proofs). All experiments are performed on an Intel i5-4210U system throttled to 1.70 GHz, using a single thread and less than 200 MB of memory.

We first analyze the proof size of our RingCT protocol. The ring size scales from 1 to 4096 in our experiment. Figure 8 demonstrates the size of a single transformation in different RingCT protocols in terms of the ring size, with different numbers of output accounts, i.e., $m = N/16$. Clearly, our RingCT has a smaller asymptotic proof size compared with RingCT 3.0, mainly due to the aggregation technique used in Section VI-B. Besides, our approach is also smaller than Omniring [11] due to a smaller constant term. When the ring size is bigger than 500, the proof size of our method is almost half of RingCT 3.0, which fits with the theoretical result well.

The average running time of our RingCT protocol under different rings is shown in Figure 9 and Figure 10. Since Bulletproofs compression requires linear time for proving and verifying, the running time of our approach scales linearly with the ring size. To generate a transaction with a 16-size ring (larger than the existing ring size of 11 in Menoro), our approach only costs 249 ms, and the verification is almost 5 times faster than proving. Thus, the time cost of our RingCT protocol is acceptable in real-world applications. Compared to Omniring, our RingCT protocol runs faster both on proving and verification. For partial knowledge proofs, our any-out-of-many proofs also outperform the k-out-of-N proofs on the proving time since we avoid pairing in our design.

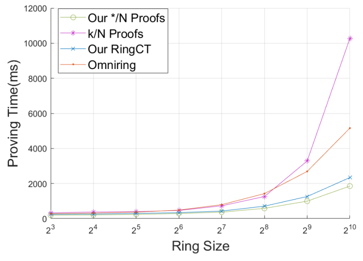

Fig. 9: Proving time comparison of our any-out-of-many proofs, partial knowledge proofs [17], our RingCT protocol and Omniring [11] with different size of the public set (ring size).

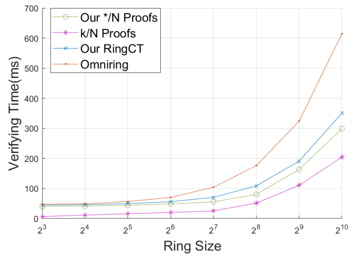

Fig. 10: Verifying time comparison of our any-out-of-many proofs, partial knowledge proofs [17], our RingCT protocol and Omniring [11] with different size of public set (ring size).

VIII. Discussions

A. *Validity of the coefficients in Generic Compression Model*

We first discuss how to ensure τ_i is invertible in Equation (12). Based on the deduction in Section IV, we can find τ_i is determined by $\eta_{0,i}$, $\zeta_{0,i}$, $\eta_{1,i}$, $\zeta_{1,i}$, and the challenge z, which can be written as follows:

$$\tau_i = \eta_{0,i}\zeta_{0,i} + z\eta_{1,i}\zeta_{1,i}. \tag{19}$$

As the coefficients of $\boldsymbol{b}_0$ and $\boldsymbol{b}_1$ terms in $\eta_{j,i}$ and $\zeta_{j,i}$ are determined by the challenge y (or is a constant) for $j \in \{0, 1\}$, the probability of τ_i being non-invertible is negligible under a super-poly challenge space. In an interactive proof, the verifier can ensure τ_i is invertible by selecting proper y and z. Moreover, acquiring a non-invertible value τ_i will not contribute to failure when running the non-interactive protocol. Since the challenge values y, z are determined by the Hash function based on the Fiat-Shamir heuristic, the prover only needs to $rollback$ to the previous step and regenerates the blinding vectors. The hash values will be totally different.

B. *Further Applications of Any-out-of-many Proofs*

1) Multiple Ring Signature: Different from the RingCT protocol, a multiple ring signature [35] aims to provide anonymity for multiple independent signers. That is, the secret key of each signer is unknown to others. It is feasible to build this scheme by replacing the core proof scheme in the Schnorr multi-signatures model proposed by Maxwell [36] with our any-out-of-many proofs. Besides, an additional range proof for $k \in [1, N)$ should be applied to prevent the $\boldsymbol{s} = \emptyset$ case mentioned in Section V-A since generating a signature without real signer is illegal in this scenario. This can be done by proving $\langle \boldsymbol{b}, \mathbf{1}^N \rangle \in [1, N)$ with the range proof in [19].

2) Coin Mixing: Coin mixing is another practical approach to enhance the anonymity of cryptocurrencies by combing multiple transactions into one to obscure the relationship between spenders and recipients. Current coin mixing schemes need a trustless third party (Mixer) to perform the mixing process, while the security and anonymity of such centralized service are questionable. Our any-out-of-many proofs can also be applied in this application by using a ring size precisely of the number of source accounts, i.e., proving N-out-of-N relation. Additionally, proof for $k = N$ should be included. Since N is public, it can be done efficiently with the Bulletproofs compression for $\langle \boldsymbol{b}, \mathbf{1}^N \rangle = N$. To compute the opening of amortized secret keys of multiple spenders, we believe a secure Multiple Party Computation [37] is also needed.

C. Open Problems

Our generic inner-product transformation focuses on quadratic relations. This transformation can be applied in more applications if extended to polynomial relations. Though we have some progress in downgrading an n-degree relation, the transformation requires $O(n)$ elements to be committed, which cannot reduce the proof size even with Bulletproofs compression. Therefore, ensuring only a minimum number of elements are sent in the transformation is critical.

For the tagging scheme, we highlight that each tag of the real account or decoy account must be unique. This may cause the exhaustion of prime numbers used in the accumulator after the network deals with a large number of transactions. An alternative approach to avoid this problem is to use the updatable public key scheme proposed in Quisquis [38], while it introduces extra work of updating the key. Therefore, we believe a more efficient way is to design a partial knowledge proof for real account tags without leaking k.

Acknowledgement

We want to express our gratitude to Mr. Rui Song from the Hong Kong Polytechnic University for proofreading assistance. This research has received partial support from HK RGC GRF under Grants PolyU 15216721 and PolyU 15207522.

Appendix

A. Formal Definitions

1) Commitment Scheme: We give definitions of the hiding and binding properties of a secure commitment scheme.

Definition 1: (Hiding Property). A commitment scheme is hiding if a commitment does not reveal the message. For all PPT (probabilistic polynomial time) adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{l} ck \leftarrow \operatorname{Setup}\left(1^{\lambda}\right); (m_0, m_1) \leftarrow \mathcal{A}(ck); b \leftarrow \{0,1\} \\ r \leftarrow \mathbb{Z}_q; c = \operatorname{Com}_{ck}(m_b, r) : \mathcal{A}(c) = b \end{array}\right] \approx \frac{1}{2},$$

where $\mathcal{A}$ outputs messages $m_0, m_1 \in \mathbb{M}$. If the probability is exactly $1/2$ we say the commitment scheme is perfect hiding.

Definition 2: (Binding Property). A commitment scheme is binding if a commitment can only be opened to one message. For all PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c} ck \leftarrow \operatorname{Setup}\left(1^{\lambda}\right); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck): \\ m_0 \neq m_1 \wedge \operatorname{Com}_{ck}(m_0, r_0) = \operatorname{Com}_{ck}(m_1, r_1) \end{array}\right] \approx 0,$$

where $\mathcal{A}$ outputs messages $m_0, m_1 \in \mathbb{M}$ and $r_0, r_1 \in \mathbb{Z}_q$. If the probability is exactly 0 we say the commitment scheme is perfect binding.

2) Σ-Protocol: Σ-protocol is a type of 3-move interactive proof system between two parties, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. With the setup algorithm mentioned above, the prover can convince the verifier that a statement is true. Specifically, we define R as a polynomial-time decidable ternary relation, with a commitment key ck generated by the algorithm $\operatorname{Setup}(1^{\lambda})$, a statement c and its witness r, where $(ck, c, r) \in \mathcal{R}$.

First, the prover generates an initial message m according to the given parameter $(ck, c, r) \in \mathcal{R}$. After receiving the message m, the verifier chooses a challenge value $x \leftarrow \mathbb{Z}_q^*$ and sends it to the prover. Then prover computes the response message z according to the challenge x. Finally, the verifier will check the proof with (ck, c, m, x, z) and returns 1 if accepted. We call the triple $(\text{Setup}, \mathcal{P}, \mathcal{V})$ a Σ-Protocol if it satisfies the following three properties.

Definition 3 (Perfect Completeness): $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is perfectly complete if for all probabilistic polynomial time adversaries $\mathcal{A}$

$$\Pr\left[\ \mathcal{V}(ck, c, m, x, z) = 1 \ \middle|\ \begin{array}{l} ck \leftarrow \text{Setup}(1^\lambda); (c, r) \leftarrow \mathcal{A}(ck); \\ m \leftarrow \mathcal{P}(ck, c, r); x \xleftarrow{} \mathbb{Z}_q^*; z \leftarrow \mathcal{P}(x) \end{array}\ \right] = 1.$$

Definition 4 (n-Special Soundness): $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is n-special sound if there exists an efficient PPT extractor $\mathcal{E}$ that can extract the witness r given n accepting transcripts with the same m.

$$\Pr\left[\ (ck, c, r) \in R \ \middle|\ \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); \\ (c, m, x_1, z_1, ..., x_n, z_n) \leftarrow \mathcal{A}(ck); \\ \mathcal{V}(ck, c, m, x_i, z_i) = 1, \forall i \in [1, n]; \\ r \leftarrow \mathcal{E}(ck, c, m, x_1, z_1, ..., x_n, z_n) \end{array}\ \right] = 1 - \mu(\lambda),$$

where the function $\mu(\lambda)$ is negligible, and we say that the protocol is n-special sound.

Definition 5 (Honest Verifier Zero-Knowledge): $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is special honest verifier zero-knowledge if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive probabilistic polynomial time adversaries $\mathcal{A}$

$$\left| \begin{array}{l} \Pr\left[\ \mathcal{A}(m, z) = 1 \ \middle|\ \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); (c, r, x) \leftarrow \mathcal{A}(ck); \\ a \leftarrow \mathcal{P}(ck, c, r); z \leftarrow \mathcal{P}(x); \end{array}\ \right] - \\ \Pr\left[\ \mathcal{A}(m, z) = 1 \ \middle|\ \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); (c, r, x) \leftarrow \mathcal{A}(ck); \\ (m, z) \leftarrow \mathcal{S}(ck, c, r); \end{array}\ \right] \end{array} \right| \leqslant \mu(\lambda).$$

B. Details for Section IV

1) Section IV-A: In (10), we can rewrite the two equations with a challenge $z \in \mathbb{Z}_q$ into an inner-product form:

$$\begin{array}{rl} & \langle \boldsymbol{y}^n, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \rangle + z \langle \boldsymbol{y}^n, \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta} - \boldsymbol{b}_1 \rangle \\ = & \langle \boldsymbol{b}_0 \circ \boldsymbol{y}^n, \boldsymbol{b}_1 \rangle + \langle \boldsymbol{b}_0 \circ \boldsymbol{y}^n, z\boldsymbol{\alpha} \rangle + \langle \boldsymbol{y}^n, z\boldsymbol{\beta} \rangle - \langle \boldsymbol{y}^n, z\boldsymbol{b}_1 \rangle \\ = & \langle \boldsymbol{b}_0 - z\mathbf{1}^n, (z\boldsymbol{\alpha} + \boldsymbol{b}_1) \circ \boldsymbol{y}^n \rangle + \left\langle \boldsymbol{y}^n, z^2\boldsymbol{\alpha} + z\boldsymbol{\beta} \right\rangle \\ = & -\langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle \\ \iff & \langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle = \delta, \end{array}$$

where $\zeta(\boldsymbol{b}_0) = \boldsymbol{b}_0 - z\mathbf{1}^n$, $\eta(\boldsymbol{b}_1) = (z\boldsymbol{\alpha} + \boldsymbol{b}_1) \circ \boldsymbol{y}^n$, and $\delta = -\langle \boldsymbol{y}^n, z^2\boldsymbol{\alpha} + z\boldsymbol{\beta} \rangle - \langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle$.

In a Σ-protocol, instead of sending $\zeta(\boldsymbol{b}_0)$ and $\eta(\boldsymbol{b}_1)$ directly, a prover will response with $\boldsymbol{\zeta} = (\boldsymbol{b}_0 - z\mathbf{1}^n) + x\boldsymbol{s}_0$ and $\boldsymbol{\eta} = (z\boldsymbol{\alpha} + \boldsymbol{b}_1 + x\boldsymbol{s}_1) \circ \boldsymbol{y}^n$ with a challenge $x \in \mathbb{Z}_q$ and some masking values $\boldsymbol{s}_0, \boldsymbol{s}_1 \in \mathbb{Z}_q^n$. Thus, Equation (20) becomes

$$\langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle = \langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle + xt_1 + x^2 t_2,$$

where $t_1 = \left(\langle \boldsymbol{s}_0, \eta(\boldsymbol{b}_1) \rangle + \langle \zeta(\boldsymbol{b}_0), \boldsymbol{s}_1 \circ \boldsymbol{y}^n \rangle\right)$, $t_2 = \langle \boldsymbol{s}_0, \boldsymbol{s}_1 \circ \boldsymbol{y}^n \rangle$. As $\zeta(\cdot)$ is a linear function, we can compute $B_0 = \boldsymbol{g}^{\boldsymbol{\zeta}}$ based on B in relation R_{quad} and the commitment of $\boldsymbol{s}_0$, $S_0 = \boldsymbol{g}^{\boldsymbol{s}_0}$ (so does $B_1 = \boldsymbol{g}^{\boldsymbol{\eta}}$ based on $\boldsymbol{g}^{\boldsymbol{b}_1}$ and $\boldsymbol{g}^{\boldsymbol{s}_1}$ for the linear function $\eta(\cdot)$) and further apply the Bulletproofs compression directly with inputs $\boldsymbol{\zeta}, \boldsymbol{\eta}, \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$.

2) Section IV-B: The left-hand side of Equation (11) can be rewritten as one inner-product relation:

$$\begin{aligned}
& \langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1)\rangle + z\langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1)\rangle \\
= & \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0\rangle + \langle \boldsymbol{\mu}_0, \eta_0(\boldsymbol{b}_1)\rangle \\
& + \langle \boldsymbol{b}_0, z\eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1\rangle + \langle z\boldsymbol{\mu}_1, \eta_1(\boldsymbol{b}_1)\rangle \\
= & \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z\eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1\rangle + \langle \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0, \boldsymbol{b}_1\rangle \\
& + \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle + \langle z\boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1, \boldsymbol{b}_1\rangle + \langle z\boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle \\
= & \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z\eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1\rangle \\
& + \langle \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0 + z\boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1, \boldsymbol{b}_1\rangle + \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle + \langle z\boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle.
\end{aligned}$$

As η_0 and η_1 are linear functions, $\eta_0(\boldsymbol{b}_1)\circ\boldsymbol{\zeta}_0 + z\eta_1(\boldsymbol{b}_1)\circ\boldsymbol{\zeta}_1$ is also linear. Let $\eta_0(\boldsymbol{b}_1)\circ\boldsymbol{\zeta}_0 + z\eta_1(\boldsymbol{b}_1)\circ\boldsymbol{\zeta}_1 = \boldsymbol{\tau} \circ \boldsymbol{b}_1 + \boldsymbol{\omega}$, $\boldsymbol{\kappa}_0 = \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0 + z\boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1$ and $\kappa_1 = \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle + \langle z\boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle$. We can rewrite Equation (20) as:

$$\begin{aligned}
& \langle \boldsymbol{b}_0, \boldsymbol{\tau} \circ \boldsymbol{b}_1 + \boldsymbol{\omega}\rangle + \langle \boldsymbol{\kappa}_0, \boldsymbol{b}_1\rangle + \kappa_1 \\
= & \langle \boldsymbol{\tau} \circ \boldsymbol{b}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle + \langle \boldsymbol{\kappa}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle \\
& - \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle + \kappa_1 \\
= & \langle \boldsymbol{\tau} \circ \boldsymbol{b}_0 + \boldsymbol{\kappa}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle - \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle + \kappa_1,
\end{aligned}$$

where $\boldsymbol{\tau}^{-1}$ is to inverse each element of $\boldsymbol{\tau}$ (i.e., τ_i^{-1}). Thus, the prover needs to ensure $\tau_i \neq 0$ for the above transformation (which is true with all but negligible probability for a challenge space larger enough, more discussions in Section VIII-A). Taking Equation (20) and (20), we have one inner-product relation

$$\langle \tau(\boldsymbol{b}_0), \omega(\boldsymbol{b}_1)\rangle = \delta,$$

where $\tau(\boldsymbol{b}_0) = \boldsymbol{\tau} \circ \boldsymbol{b}_0 + \boldsymbol{\kappa}_0$, $\omega(\boldsymbol{b}_1) = \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}$, and $\delta = \delta_0 + z\delta_1 - \kappa_1 + \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle$.

C. Proof of Theorem 1

In this section, the security proofs of *any-out-of-many* proofs in Theorem 1 is presented.

1) Completeness.: With correct transcripts from the honest prover, the verifier can verify the following equations hold:

Equation (1) in Protocol 4:

$$\begin{aligned}
v^{\hat{t}} u^{\tau_x} &= v^{\delta(y,z) + t_1 x + t_2 x^2} \\
&= v^{\delta(y,z)} v^{t_1} x v^{t_2 x^2} \\
&= v^{\delta(y,z)} T_1^x T_2^{x^2}.
\end{aligned}$$

Equation (2) in Protocol 4:

$$\begin{aligned}
& \boldsymbol{g}'^{\boldsymbol{\zeta}} \boldsymbol{h}^{\boldsymbol{\eta}} u^{\mu} \\
= & \boldsymbol{g}^{(\boldsymbol{b}_0 - z\cdot\boldsymbol{1}^N + x\boldsymbol{s}_0)} \boldsymbol{h}^{(z\cdot\boldsymbol{1}^N + \boldsymbol{b}_1 + x\boldsymbol{r}_1)} u^{\alpha + \beta x} \\
= & (\boldsymbol{g}^{\boldsymbol{b}_0} \boldsymbol{h}^{\boldsymbol{b}_1} u^{\alpha})(\boldsymbol{g}^{\boldsymbol{r}_0} \boldsymbol{h}^{\boldsymbol{r}_1} u^{\beta})^x \boldsymbol{g}^{-z\cdot\boldsymbol{1}^N} \boldsymbol{h}^{z\cdot\boldsymbol{1}^N} \\
= & AB^x (\boldsymbol{g}^{-1}\boldsymbol{h})^{z\cdot\boldsymbol{1}^N}.
\end{aligned}$$

Equation (3) in Protocol 4:

$$\begin{aligned}
\boldsymbol{P}^{\boldsymbol{\zeta}} &= \boldsymbol{P}^{(\boldsymbol{b}_0 - z\cdot\mathbf{1}^N + x\boldsymbol{s_0})\circ \boldsymbol{y}^N} \\
&= \boldsymbol{P}^{\boldsymbol{b}_0\circ\boldsymbol{y}^N}\cdot[\mathrm{Com}_{ck}(0;r_s)]^x \cdot \boldsymbol{P}^{(-z\cdot\mathbf{1}^N\circ\boldsymbol{y}^N)} \\
&\quad \cdot[\boldsymbol{P}^{\boldsymbol{r}_0\circ\boldsymbol{y}^N}\cdot\mathrm{Com}_{ck}(0;-r_s)]^x \\
&= \mathrm{Com}_{ck}(0;f_s)\cdot E^x\cdot\boldsymbol{P}^{(-z\cdot\mathbf{1}^N\circ\boldsymbol{y}^N)}.
\end{aligned}$$

Equation (4) in Protocol 4:

$$\begin{aligned}
\hat{t} &= \delta(y,z) + t_1 x + t_2 x^2 \\
&= \left\langle (\boldsymbol{b}_0 - z\cdot\mathbf{1}^N + x\boldsymbol{r}_0)\circ\boldsymbol{y}^{\boldsymbol{N}}, z\cdot\mathbf{1}^N + \boldsymbol{b}_1 + x\boldsymbol{r}_1\right\rangle \\
&= \langle\boldsymbol{\zeta},\boldsymbol{\eta}\rangle\,.
\end{aligned}$$

2) Special-HVZK: With randomly chosen elements $(B, T_2, \tau_c,\ \mu, f_s, \boldsymbol{\zeta}, \boldsymbol{\eta})$, and the challenge values (x,y,z) from their corresponding domains, the simulator can compute:

$$\begin{aligned}
\hat{t} &= \langle\boldsymbol{\zeta},\boldsymbol{\eta}\rangle\,, \\
T_1 &= [v^{\hat{t}}u^{\tau_x}v^{-\delta(y,z)}\cdot T_2^{-x^2}]^{1/x}, \\
A &= (\boldsymbol{g'}^{\boldsymbol{\zeta}}(\boldsymbol{h})^{\boldsymbol{\eta}}u^{\mu})\cdot B^{-x}(\boldsymbol{g}\boldsymbol{h}^{-1})^{z\cdot\mathbf{1}^N}, \\
E &= [\boldsymbol{P}^{\boldsymbol{\zeta}}\cdot\mathrm{Com}_{ck}^{-1}(0;f_s)\cdot\boldsymbol{P}^{(z\cdot\mathbf{1}^N\circ\boldsymbol{y}^N)}]^{1/x}.
\end{aligned}$$

We can observe that the transcript output by the simulator has an identical distribution with the true prover. Thus, an honest verifier can not distinguish the transcripts generated above from those generated in the legal conversation if the Pederson commitment is perfect hiding. And Protocol 4 is said to be perfect special-HVZK.

3) Special Soundness: Suppose the adversary can return the transcripts of the same witness as the extractor rewinds with different challenges. We use the subscript i to denote the elements in the return transcript of $i{-}th$ rewind. To extract A, the adversary returns two transcripts for rewinding with 2 different challenges x_0, x_1. According to the two verification equations written as $(\boldsymbol{g}')^{\boldsymbol{\zeta}_i}\boldsymbol{h}^{\boldsymbol{\eta}_i}u^{\mu_i} = AB^{x_i}(\boldsymbol{g}^{-1}\circ\boldsymbol{h})^{z\cdot\mathbf{1}^N}$, where $i\in\{0,1\}$, we can compute A as

$$A = (\boldsymbol{g}')^{(\boldsymbol{b}_0' - z\cdot\mathbf{1}^N)\circ\boldsymbol{y}^N}\cdot\boldsymbol{h}^{(z\cdot\mathbf{1}^N+\boldsymbol{b}_1')}u^{\alpha'}(\boldsymbol{g}\circ\boldsymbol{h}^{-1})^{z\cdot\mathbf{1}^N} = \boldsymbol{g}^{\boldsymbol{b}_0'}\boldsymbol{h}^{\boldsymbol{b}_1'}u^{\alpha'}.$$

Similarly, we can also extract B by using the same transcripts.

$$B \;=\; (\boldsymbol{g}')^{(\boldsymbol{s}_0'\circ\boldsymbol{y}^N)}\boldsymbol{h}^{\boldsymbol{s}_1'}u^{\beta'} = \boldsymbol{g}^{\boldsymbol{s}_0'}\boldsymbol{h}^{\boldsymbol{s}_1'}u^{\beta'}.$$

Putting back the extracted values into the verification Equation (2) in Protocol 1, we can get:

$$\begin{aligned}
(\boldsymbol{g}')^{\boldsymbol{\zeta}}\boldsymbol{h}^{\boldsymbol{\eta}}u^{\mu} &= A\cdot B^x\cdot(\boldsymbol{g}^{-1}\circ\boldsymbol{h})^{z\cdot\mathbf{1}^N} \\
&= \boldsymbol{g}^{\boldsymbol{b}_0'}\boldsymbol{h}^{\boldsymbol{b}_1'}u^{\alpha'}[\boldsymbol{g}^{\boldsymbol{s}_0'}\boldsymbol{h}^{\boldsymbol{s}_1'}u^{\beta'}]^x\cdot(\boldsymbol{g}^{-1}\circ\boldsymbol{h})^{z\cdot\mathbf{1}^N}.
\end{aligned}$$

Since the above holds for all challenge c, and assuming the DL assumption holds between $\boldsymbol{g}$, $\boldsymbol{h}$ and u, we can extract the vectors $\boldsymbol{\zeta},\boldsymbol{\eta}$ as

$$\begin{aligned}
\boldsymbol{\zeta} &= (\boldsymbol{b}_0' - z\cdot\mathbf{1}^N + x\boldsymbol{s}_0')\circ\boldsymbol{y}^N, \\
\boldsymbol{\eta} &= \boldsymbol{b}_1' + z\cdot\mathbf{1}^N + x\boldsymbol{s}_1'.
\end{aligned}$$

Elements T_1, T_2 can also be extracted by using other 3 rewinding transcripts with challenge x_2, x_3, x_4. The rewinding transcripts are $v^{\hat{t_i}} u^{\tau_{x_i}} = v^{\delta(y,z)} T_1^{x_i} T_2^{{x_i}^2}$ according to Equation (1) in Protocol 4, where $i \in \{2, 3, 4\}$. Then we can compute T_1, T_2

$$T_1 = v^{t'_1} u^{\tau'_1},\ T_2 = v^{t'_2} u^{\tau'_2}.$$

Also, putting back the extracted values into the verification Equation (1), we have

$$v^{\hat{t}} u^{\tau_x} = v^{\delta(y,z)} T_1^x T_2^{x^2} = v^{\delta(y,z)} (v^{t'_1} u^{\tau'_0})(v^{t'_2} u^{\tau'_1})^x.$$

As the equation above holds for all challenge c and the DL assumption between u, v is not known to the adversary, we have:

$$\hat{t} \quad = \quad \delta(y,z) + t'_1 + t'_2 x.$$

Note that we can also compute $\hat{t}$ as follows with the vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ above.

$$\begin{aligned}
\hat{t} &= \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle \\
&= \left\langle (\boldsymbol{b}'_0 - z \cdot \mathbf{1}^N + x\boldsymbol{r}'_0) \circ \boldsymbol{y^n}, \boldsymbol{b}'_1 + z \cdot \mathbf{1}^N + x\boldsymbol{r}'_1 \right\rangle \\
&= \left\langle \boldsymbol{y}^N, \boldsymbol{b}'_0 \circ \boldsymbol{b}'_1 \right\rangle + z \left\langle \boldsymbol{y}^N, \boldsymbol{b}'_0 - \mathbf{1}^N - \boldsymbol{b}'_1 \right\rangle \\
&\quad + \delta(y,z) + t''_1 + t''_2 x.
\end{aligned}$$

Since the expression above holds for all challenges x, y, z, we can get the relations as follows by comparing the two expressions of $\hat{t}$. $\boldsymbol{b}'_0 \circ \boldsymbol{b}'_1 = \mathbf{0}^N, \boldsymbol{b}'_0 - \mathbf{1}^N - \boldsymbol{b}'_1 = \mathbf{0}^N$, and further ensure that $\boldsymbol{b}'_0$ is a binary vector.

On the other hand, using the same set of 2 rewinding transcripts, we can also calculate the following expression based on Equation (3) in Protocol 1.

$$\begin{aligned}
\boldsymbol{P}^{(\boldsymbol{b}'_0 - \boldsymbol{z} \cdot \mathbf{1}^N) \circ \boldsymbol{y}^N} &= \text{Com}_{ck}(0; \sum_{i \in \boldsymbol{S}} y^i s'_i) \cdot \boldsymbol{P}^{(-z \cdot \mathbf{1}^N \circ \boldsymbol{y}^N)}, \\
\boldsymbol{P}^{(\boldsymbol{b}'_0 \circ \boldsymbol{y}^N)} &= \text{Com}_{ck}(0; \sum_{i \in \boldsymbol{S}} y^i s'_i).
\end{aligned}$$

Note that the expression above can actually be regarded as a product of N different exponentiation,

$$P_0{}^{b'_{0,0}} \cdot (P_1{}^y)^{b'_{0,1}} \cdot \ldots \cdot (P_{N-1}{}^{y^{N-1}})^{b'_{0,N-1}} = \text{Com}_{ck}(0; \sum_{i \in \boldsymbol{S}} y^i s'_i),$$

and since $\boldsymbol{b}'_0$ is a binary vector, all of the values $b'_{0,0}$ are either 0 or 1. As a result, with totally $|\boldsymbol{S}| + 1$ challenges $y_0, y_1, ..., y_{|\boldsymbol{S}|}$, we can calculate the following equations:

$$P_i = \text{Com}_{ck}(0; s'_i) \quad \forall i \in \boldsymbol{S}.$$

Thus, we can reveal the set $\boldsymbol{S}$ of the witnesses, which also represents the secret vector $\boldsymbol{b}$ of the challenger. What's more, each witness s'_i can be extracted from Equation (3) in Protocol 4 with $|\boldsymbol{S}| + 1$ rewinding transcripts. Finally, we can extract $\boldsymbol{x}'_{sk} = \boldsymbol{sk}'$ if the Pederson commitment is perfect binding. Thus, Protocol 4 is special sound.

D. Security Proofs for RingCT

In RingCT 3.0 [10], Yuen et al. propose a stronger security model for their new RingCT protocol, taking the recipient and insider attack into consideration. This is mainly because, in the previous version (e.g., RingCT 2.0 [9]), the security definition of anonymity only emphasizes on the indistinguishability of the transcripts. However, the author points out that the anonymity of the RingCT protocol is more complicated than the anonymity of linkable ring signatures. This is because the recipient in a RinCT protocol will be given the knowledge of the input and output amount, and the level of anonymity of RingCT protocol may be reduced consequently. For a transaction with multiple input accounts, multiple linkable ring signatures are generated. Yet, they are correlated when validating the balance of input and output amount. This extra relationship may reduce the level of anonymity. Moreover, the previous model of anonymity neither considers the threat of insider attacks. As a result, the authors in [9] develop a stronger security model by defining two new models for anonymity, anonymity against the recipient (who knows all the output account secret keys and their amounts) and anonymity against ring insider (who knows some input account secret keys and their amounts). We will first give a comprehensive description of this security model for RingCT and prove that our RingCT protocol also satisfies these stronger security requirements.

Perfect Correctness. The perfect correctness property requires that a user can spend any group of her accounts w.r.t. an arbitrary set of groups of input accounts, each group containing the same number of accounts as the group she intends to spend.

Balance. The balance property requires that any malicious user cannot

1) spend any account of an honest user,
2) spend her own accounts with the sum of input amount being different from that of output amount, and
3) double-spend any of her accounts.

Therefore, the balance property can be modeled by three security models: unforgeability, equivalence, and linkability.

Anonymity against Recipients. The anonymity against recipients property requires that without the knowledge of any input account secret key and input amount (which are within a valid Range: from 0 to a maximum value), the spenders' accounts are successfully hidden among all the honestly generated accounts, even when the output accounts and the output amounts are known.

Anonymity against Insider Attacks. The anonymity against ring insiders requires that without the knowledge of the output account secret key and output amount (which are within a valid Range), the spenders' accounts are successfully hidden among all uncorrupted accounts.

Non-slanderability. The non-slanderability property requires that a malicious user cannot prevent any honest user from spending. It is infeasible for any malicious user to produce a valid spending that shares at least one serial number with an honestly generated spending.

The Perfect Correctness property follows in a straightforward manner. So we only consider the Balance, Anonymity, and Non-slanderability properties in our security proofs. Before describing the proof in detail, we need to give a formal definition of our security model first. Here are several oracles that will be used in the security proofs:

- $\mathbf{PKGen}(i)$: on the $i-th$ query picks randomness r_i, runs $(P_i, sk_i) \leftarrow$ **SAKGen**(pp) and returns P_i.
- $\mathbf{ActGen}(P, a)$: on input a public key P and an amount a, runs $(C, r) \leftarrow$ **OTAccGen**(P, a), and outputs account (act, r) for address P.

- $\mathbf{Spend}(\{\text{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}, \mathbf{tag}, \{\text{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}, \mu)$: returns $\sigma \leftarrow \mathbf{Spend}(\{\text{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}, \mathbf{tag}, \{\text{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}, \mu)$, provided (P_i, s_i) has been generated by $\mathbf{PKGen}$ and $P_i \in \boldsymbol{P}$.
- $\mathbf{Corrupt}(i)$: returns s_i provided P_i has been generated by $\mathbf{PKGen}$.

We denote **Orc** as a tuple of oracles including **PKGen**, **ActGen**, **Spend**, **Corrupt**. And we say that a tuple of *PPT* algorithms of Φ = (**Setup**, **SAKGen**, **OTAccGen**, **Spend**, **Vf**, **CheckTag**) is a RingCT Protocol satisfying the properties as follows:

Definition 6 (Unforgeability): A RingCT protocol is called unforgeable if it is infeasible to forge a valid transaction without controlling one of the members in the ring. Formally, it is unforgeable when for all PPT adversaries $\mathcal{A}$:

$$\Pr\left\{\begin{array}{c} pp \leftarrow \mathbf{Setup}(1^\lambda); \\ (\text{tx}, \sigma, \mathbf{I}) \leftarrow \mathcal{A}^{\mathbf{Orc}}(\text{pp}) \end{array} : \mathbf{MVerify}_{\text{pp}}(\text{tx}, \sigma, \mathbf{tag}, \boldsymbol{Q}) = 1\right\} \approx 0,$$

where $\text{tx} = (\{\text{acc}_{\mathcal{R},i}\}_{i=1}^{|\mathcal{R}|}, \mathbf{tag}, \{\text{acc}_{\mathcal{T},i}\}_{i=1}^{|\mathcal{T}|}, \mu)$.

Definition 7 (Equivalence): A RingCT protocol is called equivalent w.r.t insider corruption if it is infeasible to forge a valid transaction where the sum of the input amount is different from the sum of the output amount. Formally, it is equivalent w.r.t insider corruption when for all PPT adversaries $\mathcal{A}$:

$$\Pr\left\{\begin{array}{c} pp \leftarrow \mathbf{Setup}(1^\lambda); \\ (\text{tx}, \sigma, \mathbf{I}) \leftarrow \mathcal{A}^{\mathbf{Orc}}(\text{pp}) \end{array} : \begin{array}{c} \mathbf{Vf}_{\text{pp}}(\text{tx}, \sigma, \mathbf{tag}, \boldsymbol{Q}) = 1; \\ \mathbf{ActGen}(P, s, C, r) \neq 0; \\ \sum_j a_{\mathcal{S},j} \cdot b_j = \sum_i a_{\mathcal{R},i} \end{array}\right\} \approx 0.$$

Definition 8 (Linkability): A RingCT protocol is called linkable if it is infeasible to forge two valid transactions with distinct serial number vectors $\mathbf{Q}_1$, $\mathbf{Q}_2$ where at most $|\mathbf{Q}_1| + |\mathbf{Q}_2| - 1$ input accounts are corrupted or not generated by the challenger. Formally, it is linkable when for all PPT adversaries $\mathcal{A}$:

$$\Pr\left\{\begin{array}{c} pp \leftarrow \mathbf{Setup}(1^\lambda); \\ \{\text{tx}_i, \sigma_i, \boldsymbol{Q}_i, \}_{i=1,2} \\ \leftarrow \mathcal{A}^{\mathbf{Orc}}(\text{pp}) \end{array} : \begin{array}{c} \mathbf{Vf}_{\text{pp}}(\text{tx}, \phi, \mathbf{tag}, \boldsymbol{Q}) = 1; \\ \mathbf{Q}_1 \cap \mathbf{Q}_2 = \emptyset; |\boldsymbol{A}_{corrupt}| \\ \leq |\mathbf{Q}_1| + |\mathbf{Q}_2| - 1 \end{array}\right\} \approx 0.$$

Definition 9 (Non-slanderability): A RingCT protocol is called non-slanderable if it is infeasible to forge a valid spending that shares at least one serial number with a previously generated honest spending. Formally, it is non-slanderable when for all PPT adversaries $\mathcal{A}$:

$$\Pr\left\{\begin{array}{c} \text{pp} \leftarrow \mathbf{Setup}(1^\lambda); \\ (\mathbf{Q}', \text{tx}, \sigma, \mathbf{Q}) \\ \leftarrow \mathcal{A}^{\mathbf{Orc}}(\text{pp}) \end{array} : \begin{array}{c} \mathbf{Vf}_{pp}(\text{tx}, \phi, \mathbf{Q}) = 1; \\ \mathbf{ActGen}(P, s, C, r) \neq 0; \\ \sum_j a_{\mathcal{S},j} \cdot b_j = \sum_i a_{\mathcal{R},i} \end{array}\right\} \approx 0.$$

1) Proof of Theorem 3: Our scheme is unforgeable if the *DL* assumption holds in $\mathbb{G}$ in the random oracle model. Our scheme is equivalent w.r.t. insider corruption if the *DL* assumption holds in $\mathbb{G}$ in the random oracle model and RP is a secure zero-knowledge range proof. Our scheme is linkable w.r.t. insider corruption if the *DL* assumption holds in $\mathbb{G}$ in the random oracle model.

2) Proof of Theorem 4: Our scheme is anonymous against the recipient if the q-DDHI assumption holds in $\mathbb{G}$ under the ROM, where q is the number of Spend oracle queries. Our scheme is anonymous against ring insiders if the q-DDHI assumption holds in $\mathbb{G}$ under the ROM and the range proof is secure.

Proof. Assume that an adversary tends to crack the anonymity of our RingCT protocol. Given a transcript included $\boldsymbol{A}_{out}$, proof ϕ and image vector $\boldsymbol{I}$, the adversary can only refer to the identity information of real spenders from the ϕ and $\boldsymbol{I}$. Therefore, a simulator can be applied to simulate the proof and key images in the reduction process. Based on the q-DDHI assumption and Random Oracle model, the simulator can reduce the de-anonymization attack from the adversary to a break instance to the hard problem.

Specifically, suppose the simulator is given the DDHI problem instance $(g, g^a, ..., g^{a^q}, T)$ and wants to decide if $T = g^{1/a}$. To achieve this goal, the simulator interacts with an adversary who claimed that he has the ability to de-anonymizing the RingCT protocol. The simulator first samples some distinct randomnesses $\epsilon, h^*, h_1, ..., h_q \in \mathbb{Z}_q$ and let $u = g^{\epsilon \cdot \prod_{i=1}^{q}(a-h^*+h_i)}$ as part of the system parameters. Next we describe how the simulator runs the oracles $\mathbf{PKGen}$, $\mathbf{ActGen}$, $\mathbf{Spend}, \mathbf{Corrupt}$ above.

To respond to the $\mathbf{PKGen}$ query, the simulator samples the long-time key pair $(ltpk, ltsk)$ and computes the one-time public key P honestly in most cases. Except for one time, the simulator generates a faked public key as $ltpk^* = (g^{a-h^*, g^{x2}})$. Then the simulator runs the $\mathbf{ActGen}$ oracle correctly excluding the $ltpk^*$.

To respond to the $\mathbf{Corrupt}$ query, the simulator returns the secret key of every act queried, except for act^*, it declares failure and exits.

To respond to the $\mathbf{Spend}$ query with inputs $M, \boldsymbol{acc}_{\mathcal{R}}, \boldsymbol{acc}_{\mathcal{S}}, \boldsymbol{acc}_{\mathcal{T}}$, if the public key $P^* = g^{a-h^*} g^{h_j}, Sj \in [1, q]$ belongs to the set $\boldsymbol{acc}_{\mathcal{S}}$, then the simulator can generate the key image tag* based on the parameters $g, g^a, ..., g^{a^q}, T$, otherwise it declares failure and exits. For the remaining part of this algorithm, the simulator simply follows the correct steps and generates the transcripts accordingly.

To begin with the reduction process, the simulator first issues $\mathbf{PKGen}$ queries in polynomial times. After he acquired enough public keys for cracking, he sends a request for a RingCT transcript on the parameters $M, \boldsymbol{P}_{\mathcal{R}}, \boldsymbol{P}_{\mathcal{T}}$. The simulator first checks if $P^* = (g^{a-h^*} g^{h_j}) \in \boldsymbol{P}_{\mathcal{R}}$. For other well-formed public keys, the simulator retrieves their secret keys it records and randomly generates a valid set of values $\boldsymbol{acc}_{\mathcal{S}}, \boldsymbol{acc}_{\mathcal{T}}$ to simulate the range proofs and balance proofs. For the ring signature, the simulator will forge the key images of P_* as tag$^* = g^{\epsilon \cdot \prod_{i=1, i \neq j}^{q}(a-h^*+h_i)/a} = g^{\epsilon \cdot \sum_{i=0}^{q-1} p(a)} T^{\epsilon} \cdot k^{(-1)}$, where $p(a) = \sum_{i=0}^{q-1} k^(i) a^i$ is a (q-1)-degree polynomial of a. Obviously, the simulator can compute the expression of tag* by knowing the parameters $g, g^a, ..., g^{a^q}, T$ only. The simulated transcript of $\boldsymbol{acc}_{\mathcal{R}}, \boldsymbol{acc}_{\mathcal{T}}, \sigma, \mathbf{tag}$ will be given to the adversary.

Finally, the adversary shall output the public key P_{adv} of a real spender if they successfully de-anonymize the RingCT protocol. And with probability of $1/(N - k^*)$, $P^* = P_{adv}$, where N is the ring size and k^* is the number corrupted accounts in $\boldsymbol{acc}_{\mathcal{S}}$. Then the simulator can output $T = g^{1/a}$ as the solution to the DDHI problem. As a result, the proof above reduces the de-anonymization attack issued by the adversary to a break of the q-DDHI hard assumption with non-negligible probability.

3) Proof of Theorem 5: Our scheme is non-slanderable w.r.t. insider corruption if the *DL* assumption holds in $\mathbb{G}$ under the ROM. The proof is the same as RingCT 3.0.

References

[1] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, “A survey on security and privacy issues of bitcoin,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[2] M. Project, “Monero,” https://www.getmonero.org/, 2020.

[3] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 459–474.

[4] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, “Zcash protocol specification,” *GitHub: San Francisco, CA, USA*, p. 1, 2016.

[5] A. Pertsev, R. Semenov, and R. Storm, “Tornado cash privacy solution version 1.4,” 2019.

[6] J. Yang, S. Gao, G. Li, R. Song, and B. Xiao, “Reducing gas consumption of tornado cash and other smart contracts in ethereum,” in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 921–926.

[7] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu, “Matrict: efficient, scalable and post-quantum blockchain confidential transactions protocol,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 567–584.

[8] S. Noether, “Ring Signature Confidential Transactions for Monero,” in *IACR Cryptology ePrint Archive*, 2015.

[9] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, “Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero,” in *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22*. Springer, 2017, pp. 456–474.

[10] T. H. Yuen, S.-f. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, “Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security,” in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 2020, pp. 464–483.

[11] R. W. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, “Omniring: Scaling private payments without trusted setup,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 31–48.

[12] J. Groth and M. Kohlweiss, “One-out-of-many proofs: Or how to leak a secret and spend a coin,” in *Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*. Springer, 2015, pp. 253–280.

[13] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan *et al.*, “An empirical analysis of traceability in the monero blockchain,” *arXiv preprint arXiv:1704.04299*, 2017.

[14] A. Kumar, C. Fischer, S. Tople, and P. Saxena, “A traceability analysis of monero’s blockchain,” in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 153–173.

[15] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit, “Short accountable ring signatures based on ddh,” in *Computer Security–ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*. Springer, 2016, pp. 243–265.

[16] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, “Zether: Towards privacy in a smart contract world,” in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 423–443.

[17] T. Attema, R. Cramer, and S. Fehr, “Compressing proofs of k-out-of-n partial knowledge,” in *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV*. Springer, 2021, pp. 65–91.

[18] B. E. Diamond, “Many-out-of-many proofs and applications to anonymous zether,” in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1800–1817.

[19] 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.

[20] J. Groth, “On the size of pairing-based non-interactive arguments,” in *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*. Springer, 2016, pp. 305–326.

[21] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, “Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2111–2128.

[22] D. Boneh, B. Bünz, and B. Fisch, “Batching techniques for accumulators with applications to iops and stateless blockchains,” in *Annual International Cryptology Conference*. Springer, 2019, pp. 561–586.

[23] T. P. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in *Annual international cryptology conference*. Springer, 1991, pp. 129–140.

[24] T. Attema and R. Cramer, “Compressed sigma-protocol theory and practical application to plug & play secure algorithmics,” in *Annual International Cryptology Conference*. Springer, 2020, pp. 513–543.

[25] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu, “Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications,” in *Annual International Cryptology Conference*. Springer, 2019, pp. 115–146.

[26] S. Gao, T. Zheng, Y. Guo, and B. Xiao, “Efficient and post-quantum zero-knowledge proofs for blockchain confidential transaction protocols,” *Cryptology ePrint Archive*, 2021.

[27] A. Chepurnoy, C. Papamanthou, S. Srinivasan, and Y. Zhang, “Edrax: A cryptocurrency with stateless transaction validation,” *Cryptology ePrint Archive*, 2018.

[28] P.-A. Fouque and M. Tibouchi, “Close to uniform prime number generation with fewer random bits,” in *International Colloquium on Automata, Languages, and Programming*. Springer, 2014, pp. 991–1002.

[29] E. W. Weisstein, “Bézout’s theorem,” *https://mathworld. wolfram. com/*, 1999.

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

[31] M. Chase and A. Lysyanskaya, “On signatures of knowledge,” in *Annual International Cryptology Conference*. Springer, 2006, pp. 78–96.

[32] D. R. Brown, “Sec 2: Recommended elliptic curve domain parameters,” *Standars for Efficient Cryptography*, 2010.

[33] D. Moody, R. Peralta, R. Perlner, A. Regenscheid, A. Roginsky, and L. Chen, “Report on pairing-based cryptography,” *Journal of research of the National Institute of Standards and Technology*, vol. 120, p. 11, 2015.

[34] Decred, “Secp256k1 golang package,” 2020. [Online]. Available: https://pkg.go.dev/github.com/decred/dcrd/dcrec/secp256k1/v3

[35] T. Okamoto, R. Tso, M. Yamaguchi, and E. Okamoto, “A k-out-of-n ring signature with flexible participation for signers,” *Cryptology ePrint Archive*, 2018.

[36] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, “Simple schnorr multi-signatures with applications to bitcoin,” *Designs, Codes and Cryptography*, vol. 87, no. 9, pp. 2139–2164, 2019.

[37] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, “Secure multi-party computation: theory, practice and applications,” *Information Sciences*, vol. 476, pp. 357–372, 2019.

[38] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, “Quisquis: A new design for anonymous cryptocurrencies,” in *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25*. Springer, 2019, pp. 649–678.