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

Tianyu Zheng[1,2], Shang Gao[1], Bin Xiao[1], and Yubo Song[2]

[1] The Hong Kong Polytechnic University, Hongkong, China
[2] Southeast University, Nanjing, China

Abstract. In this paper, we propose *any-out-of-many proofs*, a logarithmic zero-knowledge scheme for proving knowledge of arbitrarily many secrets out of a public list. Unlike existing partial knowledge proofs that have to open the number of secrets [S&P'21,CRYPTO'21], our approach can make proofs for multiple secrets meanwhile hiding the exact number of them. As a result, it can achieve a higher-level of anonymity and efficiency. Furthermore, we enhance the efficiency of our scheme through a transformation that can adopt the improved inner-product argument in Bulletproofs [S&P'18], only $2 \cdot \lceil log_2(N) \rceil + 13$ elements need to be sent in a non-interactive proof.
We further use our proof scheme to implement both multiple ring signature schemes and RingCT protocols. For multiple ring signatures, we need to add a boundary constraint for the number k to avoid the proof of an empty secret set. Thus, an improved version called *bounded* any-out-of-many proof is presented, which preserves all nice features of the original protocol such as high anonymity and logarithmic size. For RingCT protocols, we show that both the original and bounded proofs can be used safely. The evaluation results indicate that our RingCT protocol is more efficient and secure than others. We also believe our techniques are applicable in other privacy-preserving occasions such as anonymous e-voting.

Keywords: zero-knowledge, partial knowledge proof, ring signature, confidential transaction

1 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 under guarantee in an honest-majority situation. To allow all transactions to be verified easily, some cryptocurrencies publicly record all transaction details in plaintext. For example, the balance and address of each account are accessible to anyone in the Bitcoin network [10]. Unfortunately, the public property hinders the original blockchain

scheme from private applications. As a result, this deficiency has impelled the development of private cryptocurrencies [22], [26], which provide confidentiality and anonymity while allowing any public verification.

Existing private cryptocurrencies use ring confidential transaction (RingCT) protocols [14], [19], [27], [28] to solve the privacy issue in two aspects: (1) confidentiality, which hides the amounts of money in a transaction; and (2) anonymity, which hides the identities of users in a transaction. For confidentiality, existing solutions adopt a balance proof to ensure the sums of inputs and outputs are equal, and a range proof to ensure each amount (account balance) lying in the valid range such as in S&P'18[8]. For the anonymity property, Groth et al. propose a logarithmic size one-out-of-many proof [16], which enables the prover to prove a statement about *one* message among a list of commitments, and only requiring a logarithmic number of commitments in transmission. This proof system can be used to construct a ring signature scheme, which provides anonymity by hiding the identity of the signer. However, due to several deficiencies of this system, the efficiency and security of the initial scheme are far from satisfactory. As the Pedersen vector commitment is not used in [16], the communication size of the scheme is very large. Even worse, one-out-of-many proofs can only open 1 out of N public commitments to zero, which contributes to incompatibilities when dealing with multiple inputs accounts in anonymous transactions. As the first deficiency has been solved in [5] through using Pedersen vector commitments, we will focus on the second one in the following content.

To explain the second deficiency, we suppose there are k spenders participating in the anonymous transaction, if the above method is applied, each of the spenders needs to generate a signature (constructed with one-out-of-many proofs) exclusively to ensure the validity of the transaction. Consequently, k distinct ring sets are required, and k corresponding signatures need to be transmitted in total. One solution to improve this inefficiency is leaking *multiple* secrets among one public list, as shown in S&P'20 [13] and CRYPTO'21 [1]. Although these two methods preserve the logarithmic-size property of one-out-of-many proofs, there are serial drawbacks. First, the many-out-of-many proof protocol only works for permutations with orbits of equal size [13], which is not a general k-out-of-N proof. Moreover, the verification cost of a partial knowledge proof [1] is prohibitively high, and the cyclic group operations are less efficient when using bilinear mapping. Finally, the anonymity of the protocol also needs to be considered. In previous work, accounts in the anonymous set are not independent. This incurs a problem that when an account is de-anonymized, some other accounts will also be excluded from the anonymous set. For example, in many-out-of-many proofs [13], the leakage of one secret will contribute to leaking the whole secret group since their positions in the anonymous set are relevant due to the public permutation. Thus, it is important to make accounts (secrets) distribute randomly (independently) in the anonymous set. In this work, our goal is to propose a more efficient proof scheme for leaking multiple secrets meanwhile with better anonymity for RingCT protocols.

1.1 Our Contributions

We conclude our contributions as follows:

Bulletproofs compression for general relations. Inspired by previous work [5], [8], we show how to apply the improved inner-product argument in Bulletproofs for general relations. We first show an approach to convert a quadratic relation into an inner-product form and then discuss how to transform multiple inner-product relations into one. With the help of the improved inner-product argument [8], logarithmic size proofs can be easily achieved.

Any-out-of-many proofs. We propose *any-out-of-many* proofs, a new technique to prove the knowledge of arbitrarily many of secrets among a public list. Different from the previous multiple secret proofs, our scheme does not reveal the exact number of secrets. We use our general model to transform these relations into an inner-product and reduce the proof size. By applying the Fiat-Shamir heuristic [4], we obtain logariyhmic non-interactive any-out-of-many proofs. Among all state-of-the-art approaches, our any-out-of-many proof has the smallest asymptotic proof size and the highest anonymity.

An efficient multiple ring signature scheme. As our original non-interactive any-out-of-many proofs allow to have *zero* secret ($k = 0$), directly applying our technique to (multiple) ring signatures is insecure. Hence we improve our proof scheme with a range proof to ensure the number k is positive. This new *bounded* any-out-of-many proofs can also be transformed and further reduced to logarithmic size with our general model.

RingCT protocol with a higher degree of anonymity. Based on our origin *any-out-of-many* proofs and *bounded any-out-of-many* proofs, we construct more secure and efficient RingCT protocols. The bounded RingCT Protocol can be directly applied to current anonymous cryptocurrency systems, while we prove the other one is also secure and applicable. Moreover, we can batch the any-out-of-many proof with range proofs in RingCT to further reduce the transcript size. Overall, our RingCT protocols are more efficient and secure comparing with existing approaches.

1.2 Related Work

As introduced by Cramer et al. in [11], a general method is proposed to prove the knowledge of at least k out of N solutions without revealing which k instances are involved with linear communication complexity. We denote these proof schemes as partial knowledge proofs uniformly, where $k \in [0, N]$. Based on this work, lots of novel Σ-protocols have been proposed, such as the group signature scheme presented in [9], [18], membership proofs in [12], [21] and also the ring signature schemes[24]. Ring signatures were first introduced by Rivest et al. in 2001, which enable a signer to sign a message as a user in an ad-hoc group without

revealing its identity. In the follow-up works such as [29], [15], some improved ring signature schemes are proposed based on non-interactive zero-knowledge proofs. With the popularity of blockchain technology, people find ring signature schemes can be used to provide anonymity in private cryptocurrencies. Thus, ring signatures once again arouse research interests and are widely applied to cryptocurrencies such as CryptoNote [25] and Ring Coin [2]. In Monero [27], such an application of ring signature is defined as a part of the process called RingCT. And later, Sun et al. formalize the syntax of RingCT protocol and also present formal security definitions of it. Since then, RingCT protocol has become an important and direct application of partial knowledge [11]. In [16], Groth introduces the one-out-of-many proof, an enhanced Σ-protocol with logarithmic communication complexity, with this scheme, the proof size of ring signatures can be improved from sublinear to logarithmic. Groth uses this scheme to instantiate both ring signature and zerocoin, and further applied in the following research such as Zether [7] and Monero [28].

Although one-out-of-many proofs [16] and their improved versions [5], [6] have been proved to be efficient and practical, there are still some limitations of this scheme. As mentioned above, one-out-of-many proofs can only open 1 out of N public commitments, which will contribute to incompatibilities when there are multiple input accounts in RingCT scenarios. Existing solutions for logarithmic-size partial knowledge proofs can be grossly categorized into two groups. One solution is to use a distinct ring for each true spender and batch multiple one-out-of-many proofs into one during the construction of an anonymous transaction. These methods have already been implemented in RingCT 2.0 and RingCT 3.0 [27],[28]. Unfortunately, the anonymous rings contribute to a great part of the transaction size when dealing with multiple inputs, which makes this approach inefficient and not compatible in other scenarios. Concisely, one-out-of-many proofs based approaches need to hide k spender accounts in a $k \times N$ matrix which each account is in (seperated) N-size anonymous set. As a result, a transaction needs to contain totally $k \cdot N$ accounts to hide n real users. To reduce the size of transactions, RingCT 2.0[27] batches public keys in the same row into one value. This method has reduced the account numbers in a transaction from $k \cdot n$ to n, while to achieve this, the index of the true spender's account in each group has to be the same. This limitation undoubtedly undermines the anonymity of the RingCT protocol, for that if one spender's identity in an anonymity group is leaked, then the identity of spenders in other anonymity groups will be leaked as well, simply because they are in the same row of the matrix. Thus, the anonymity of RingCT 2.0 is weak. Let σ denote the size of the anonymity space (described in the beginning of Section 4 in detail), and calculate it as the sum of possible distribution results for k spenders in N accounts. Then we say that anonymity space σ of RingCT 2.0 is N, which means that there are only N different ways to hide k spenders. Although in RingCT 3.0[28], Yuen et al. propose a method to break the distribution of real signers, as their method is still based on one-out-of-many proofs, which restricts that each row in a matrix can contain only

one account. We can calculate anonymity space of RingCT 3.0 as $\sigma = \binom{N}{k}$, and we can observe that there is still room for improvements

The other solution is designing a new efficient partial knowledge proof scheme, which can reveal multiple secrets at once, as proposed in [1], [13]. One major challenge is to preserve the logrithmic-size property of one-out-of-many proofs, as the optimization method of using Kronecker product [16] can not be directly applied in multi-secret situations. To solve this problem, Diamond proposes many-out-of-many proofs [13] to map multiple secrets to one based on permutations and a linear mapping. As a result, the task can be transformed to a one-out-of-many proof. Attema et al. introduce compressing proofs of k-Out-Of-N partial knowledge [1] to convert the index information to a polynomial function and prove the coefficients of the function to avoid binary proofs. Bulletproofs compression [8] can be further applied on these non-binary coefficients directly to reduce the proof size. In this paper, our any-out-of-many proofs also fall into this category for the partial knowledge proofs. Here we give a brief overview of some typical partial knowledge proofs as follows:

Table 1. Overview of some typical partial knowledge proofs

Proof type	Typical schemes	Number of secrets
Single secret proofs	[16], [5]	$k = 1$ and k is open
Multiple secret proofs	[13], [1]	$k \in [1, N]$ and k is open
Any secret proofs	Our method	$k \in [0, N]$ and k is hidden

1.3 Comparison with Other Approaches

The first wildly applied anonymous approach in RingCT protocols is one-out-of-many proofs [16]. One intriguing prosperity of this approach is it uses Kronecker's delta vector to achieve a logarithmic size proof. Specifically, instead of running a Σ-protocol on the index of the secret (an N-size binary vector) directly, the prover proves the knowledge of each bit of the secret index, which is only $2log(N)$ length. Some following-up schemes further improve the efficiency such as Bootle et al. use Perdesen vector commitment to reduce the proof size [5] to $3 \cdot \lceil log_2(N) \rceil + 7$, and Jivanyan and Mamikonyan propose a hierarchical approach to improve the prover complexity of the one-out-of-many proofs [17].

In comparison, in our protocol, we allow the prover to prove any number of secrets. When proving one secret, our approach also outperforms [16] and [5] with the proof size of $2 \cdot \lceil log_2(N) \rceil + 13$, especially for a large set. Hence, perhaps surprisingly, our simple protocols are comparable to dedicated solutions for the special case $k = 1$. (Compared with $1/N$)

To handle a multi-secret circumstance, Diamond et al. propose a generalization form of one-out-of-many proofs, namely many-out-of-many proofs[13]. Based on a public permutation of the indexes and a linear mapping, a prover

can map one index to many secrets, which can further prove the knowledge of this index for the knowledge of multiple secrets. Unfortunately, the protocol only works for permutations with orbits of equal size, which is not a general proof scheme for multi-secret. Furthermore, as the permutation is public, anyone can compute the index of all secrets when only one secret index is leaked.

The first general multi-secret proof is the Compressing Proofs of k-Out-Of-n Partial Knowledge, proposed by Attema et al. [1]. Instead of directly proving the knowledge of a Kronecker's delta vector (secret index vector), this proof scheme plants the 0 elements of the vector into a function and proves the knowledge of the coefficients of it to avoid binary proofs. Furthermore, the function is transformed into a homomorphism form which is Bulletproofs compression-friendly to achieve logarithmic size. Unfortunately, the proof size is $4 \cdot \lceil log_2(2 \cdot N - k + 1) \rceil - 1$, which is much larger comparing with [13]. Though the author argues that the size can be reduced to $2 \cdot \lceil log_2(2 \cdot N - k + 1) \rceil + 3$ with pairing-friendly elliptic curves. Both the efficiency and security level will be downgraded due to the bilinear pairing groups.

Different from many-out-of-many[13] the novel partial knowledge proofs[1], our proof scheme allows arbitrary number of secrets in a public set. Besides, our approach has the best performance under a large ring set size of N. We give a brief comparison with state-of-the-art approaches [13], [1], in Table 2, where we consider a situation to prove knowledge of k secrets in a ring set consists of N public elements.

Table 2. Comparison of the proof size and anonymity between our proofs and other k-out-of-N approaches

	Proof size	Anonymity space
Many-out-of-many proofs [13]	$3 \lceil log_2(N) \rceil + 7$	N
Novel partial knowledge proofs [1]	$4 \lceil log_2(2N - k + 1) \rceil - 1$	$\binom{N}{k}$
Novel partial knowledge proofs (pairing)	$2 \lceil log_2(2N - k + 1) \rceil + 3$	$\binom{N}{k}$
Our any-out-of-many proofs	$2 \lceil log_2(N) \rceil + 13$	2^N

The result shows that the proof size of our approach has the smallest coefficient of 2. Though the proof size of the pairing method in [1] seems to be smaller, it is based on a bilinear pairing that needs a larger group in implementation. For example, when we set the order q of the group to 128, in order to achieve the same security level with the others, the pairing method in [1] has to use a larger group with the order of 256, which almost doubles the proof size. Even worse, the verification complexity of this partial knowledge proof is very high. The last column in the table gives the value of anonymity space, we define it as the space of possible combinations of the secrets in the ring set, which reflects the level of anonymity of the k-out-of-N proofs above. Obviously, the anonymity space of our approach is the highest among them, we also note that our anonymity level

does not depend on the number k, this excellent property gives our approach the ability to prove *arbitrary* number of secrets in a public set.

1.4 Organization of the Paper

The remaining paper is organized as follows. We review some fundamental definitions and building blocks that are used in this paper in Section 2. In Section 3, we show how to apply the inner-product argument in Bulletproofs to prove more general relations. Based on our general model, we further propose the any-out-of-many proof in Section 4 and discuss its applicabilities. Two applications of our any-out-of-many proofs, multiple ring signatures and RingCT protocols are presented in Section 5 and Section 6 respectively. Finally, in Section 7, we evaluate the performance and efficiency of our proposed approaches.

2 Preliminaries

In this section, we give a brief review of some fundamental definitions and building blocks which will be used in the following sections.

2.1 Notation.

Let λ be a security parameter. The setup algorithm generating a commitment key ck is written as, $ck \leftarrow Setup(1^\lambda)$, which indicates a cyclic group $\mathbb{G}$ is determined by ck. Let g, h, u, v be generators of $\mathbb{G}$, where all of them conform to the discrete-log assumption. $\mathbb{Z}_q$ denotes the ring of integers modulo q and $\mathbb{Z}_q^*$ denotes $\mathbb{Z}_q \backslash \{0\}$. 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.

We use bold-type lower-case letters to denote vectors, such as $\boldsymbol{a} \in \mathbb{F}^n$ is a vector with elements $a_1, ..., a_n \in \mathbb{F}$. Bold-type upper-case letters are used to denote matrices, such as $\boldsymbol{A} \in \mathbb{F}^{n \times m}$ is a matrix with n rows and m columns, where each elements $a_{i,j} \in \mathbb{F}$ $(i \in [1, n], j \in [1, m])$. Let $c \cdot \boldsymbol{a} = \sum_{i=1}^{n} c \cdot a_i$ denotes a scalar product of scalar c and vector $\boldsymbol{a}$; $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$; $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 \cdot b_1, ..., a_n \cdot b_n) \in \mathbb{F}^n$ 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 $\boldsymbol{k}^n = (1, k, k^2, ..., k^{n-1}) \in \mathbb{Z}_q^n$. The Hamming weight of a vector $\boldsymbol{b}$ is defined as $HW(\boldsymbol{b})$.

2.2 Commitment Scheme

A non-interactive commitment scheme (over a commitment key ck) can output a commitment $c \leftarrow \mathbb{G}$ with inputs secret message $m \leftarrow \mathbb{Z}_q$ and randomness $r \leftarrow \mathbb{Z}_q$ in the commitment stage. Then in the opening stage, m, r are revealed 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 hiding and binding properties as follows.

Definition 1. *(Hiding Property).* For all PPT (probabilistic polynomial time) adversaries $\mathcal{A}$,

$$\Pr\begin{bmatrix} ck \leftarrow \text{Setup}\left(1^\lambda\right); (m_0, m_1) \leftarrow \mathcal{A}(ck); \\ b \leftarrow \{0,1\}; c \leftarrow \text{Com}_{ck}\left(m_b\right) : \mathcal{A}(c) = b \end{bmatrix} \approx \frac{1}{2}. \tag{1}$$

Definition 2. *(Binding Property).* For all PPT adversaries $\mathcal{A}$,

$$\Pr\begin{bmatrix} ck \leftarrow \text{Setup}\left(1^\lambda\right); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck) : \\ m_0 \neq m_1 \wedge \text{Com}_{ck}\left(m_0, r_0\right) = \text{Com}_{ck}\left(m_1, r_1\right) \end{bmatrix} \approx 0. \tag{2}$$

The Pedersen commitment [20] and Pedersen vector commitment [8] are two natural examples of homomorphic commitment schemes, which are both perfectly hiding and computationally binding under the discrete-log assumption.

Definition 3. *(Pedersen Commitment).* With a cyclic group $\mathbb{G}$ determined by ck, we can use generator $g, h \leftarrow \mathbb{G}$, randomness $r \leftarrow \mathbb{Z}_q$ to commit to a message m as follows:

$$\text{Com}_{ck}(m, r) = g^m \cdot h^r \in \mathbb{G}. \tag{3}$$

Definition 4. *(Pedersen Vector Commitment).* The prover commits to an n-dimensional vector $m \in \mathbb{Z}_q^n$ to the verifier with $n+1$ different generators in $\mathbb{G}$

$$\text{Com}_{ck}(\boldsymbol{m}, r) = \boldsymbol{g}^{\boldsymbol{m}} \cdot h^r = (\prod_{i=1}^{n} g_i^{m_i}) \cdot h^r \in \mathbb{G}. \tag{4}$$

2.3 Bulletproofs Compression

Bulletproofs protocol is an interactive approach to compress n-size vectors to scalars with $log(n)$ times of iterations [8]. This scheme is mainly based on the inner-product argument[5], which aims to prove that the prover knows the openings of two Pedersen vector commitments that satisfy an inner-product relation, which can be described as follows:

Definition 5. *(Inner-Product Argument).* Given a commitment scheme with commitment key ck, with cyclic group $\mathbb{G}$ determined by ck and $g, h, u \leftarrow \mathbb{G}$, the prover outputs a commitment C to the secret vectors $\boldsymbol{a}, \boldsymbol{b}$ and their inner-product result $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$.

$$R_1 = \left\{ (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, C \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^n) : C = \boldsymbol{g}^{\boldsymbol{a}} \cdot \boldsymbol{h}^{\boldsymbol{b}} \cdot u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \right\}. \tag{5}$$

In order to convince the verifier of the commitment above, the prover needs to send vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ to the verifier, totally $2n$ elements. In order to reduce this size from linear to logarithmic, an elegant compression mechanism is given in [8]. Assume that n is an even number and $n' = n/2$, we define the slices of vectors:

$$\boldsymbol{a}_L = (a_0, ..., a_{n'-1}) \in \mathbb{F}^{n'}, \quad \boldsymbol{a}_R = (a_{n'}, ..., a_{n-1}) \in \mathbb{F}^{n'}, \tag{6}$$

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 value $x \in \mathbb{Z}_q$ as follows:

$$\begin{aligned} \boldsymbol{a'} &= x \cdot \boldsymbol{a}_L + \boldsymbol{a}_R, \quad \boldsymbol{b'} = \boldsymbol{b}_L + x \cdot \boldsymbol{b}_R \\ \boldsymbol{g'} &= \boldsymbol{g}_L \circ \boldsymbol{g}_R^x, \quad \boldsymbol{h'} = \boldsymbol{h}_L \circ \boldsymbol{h}_R^x, \end{aligned} \tag{7}$$

which indicates the following relation holds:

$$\begin{aligned} C' &= (\boldsymbol{g'})^{\boldsymbol{a'}} \cdot (\boldsymbol{h'})^{\boldsymbol{b'}} \cdot u^{\langle \boldsymbol{a'}, \boldsymbol{b'} \rangle} \\ &= \boldsymbol{g}_L^{\boldsymbol{a}_R} \cdot \boldsymbol{h}_L^{\boldsymbol{b}_R} \cdot u^{\langle \boldsymbol{a}_R, \boldsymbol{b}_L \rangle} \cdot (\boldsymbol{g}^{\boldsymbol{a}} \cdot \boldsymbol{h}^{\boldsymbol{b}} \cdot u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle})^x \cdot (\boldsymbol{g}_R^{\boldsymbol{a}_L} \cdot \boldsymbol{h}_R^{\boldsymbol{b}_L} \cdot u^{\langle \boldsymbol{a}_L, \boldsymbol{b}_R \rangle})^{x^2} \\ &= L \cdot C^x \cdot R^{x^2}, \end{aligned} \tag{8}$$

where $L = \boldsymbol{g}_L^{\boldsymbol{a}_R} \cdot \boldsymbol{h}_L^{\boldsymbol{b}_R} \cdot u^{\langle \boldsymbol{a}_R, \boldsymbol{b}_L \rangle}$ and $R = \boldsymbol{g}_R^{\boldsymbol{a}_L} \cdot \boldsymbol{h}_R^{\boldsymbol{b}_L} \cdot u^{\langle \boldsymbol{a}_L, \boldsymbol{b}_R \rangle}$. The new relation in Equation (8) indicates the original relation in Equation (5) still holds on the new elements $\boldsymbol{a'}, \boldsymbol{b'}, \boldsymbol{g'}, \boldsymbol{h'}, L, R$. Therefore, only half length of the origin vectors $\boldsymbol{a'}$ and $\boldsymbol{b'}$ and extra two group elements L and R need to be sent after engaging this method, and the origin vector of length n will be compressed to 1 after $\lceil log_2(n) \rceil$ rounds of iterations.

3 A General Compression Model for ZKP of Vectors

The Bulletproofs protocol converts range proof relations into an inner-product form, and then compresses the vectors through the inner-product argument [8], [5]. However, as Bulletproofs protocol only focuses on range proof relations, the application of this transformation is limited, which may not be feasible for other relations.

Motivated by previous work[8],[1], we present how to conduct this inner-product transformation for multiple quadratic relations, which is compatible with the inner-product argument to reduce the proof size. We start from a quadratic relation and further extend multiple quadratic relations.

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:

$$R_{quad} = \left\{ (\boldsymbol{g} \in \mathbb{G}^n, B \in \mathbb{G}; \boldsymbol{b} \in \mathbb{Z}_p^n) : B = \boldsymbol{g}^{\boldsymbol{b}}, f(\boldsymbol{b}) = \boldsymbol{0}^n \right\}, \tag{9}$$

where $f(\cdot)$ is a quadratic relation, $f(\boldsymbol{b}) = \boldsymbol{b} \circ (\boldsymbol{\alpha} \circ \boldsymbol{b} + \boldsymbol{\beta}) + \boldsymbol{\gamma}$, and $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ are public coefficients. A linear relation can be regarded as a special case of the quadratic relation where $\boldsymbol{\alpha} = \boldsymbol{0}^n$. When considering hiding, we can simply regard $\boldsymbol{b'} = (\boldsymbol{b}||r)$ and $\boldsymbol{g'} = (\boldsymbol{g}||h)$ in Pedersen vector commitments.

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 $\boldsymbol{b}_1$:

$$\boldsymbol{b}_0 \circ \boldsymbol{b}_1 = -\boldsymbol{\gamma} \quad \wedge \quad \boldsymbol{b}_1 = \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta}. \tag{10}$$

With a challenge $y \in \mathbb{Z}_q$, equation (10) can be further transformed into inner-product forms:

$$\langle \boldsymbol{y}^n, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \rangle = -\langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle \quad \wedge \quad \langle \boldsymbol{y}^n, \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta} - \boldsymbol{b}_1 \rangle = 0. \tag{11}$$

We can further aggregate these two equations with another challenge $z \in \mathbb{Z}_q$ and convert into one inner-product form:

$$\begin{aligned}
&\langle \boldsymbol{y}^n, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \rangle + z \cdot \langle \boldsymbol{y}^n, \boldsymbol{\alpha} \circ \boldsymbol{b}_0 + \boldsymbol{\beta} - \boldsymbol{b}_1 \rangle = -\langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle \\
\iff &\langle \boldsymbol{b}_0 \circ \boldsymbol{y}^n, \boldsymbol{b}_1 \rangle + \langle \boldsymbol{b}_0 \circ \boldsymbol{y}^n, z \cdot \boldsymbol{\alpha} \rangle + \langle \boldsymbol{y}^n, z \cdot \boldsymbol{\beta} \rangle - \langle \boldsymbol{y}^n, z \cdot \boldsymbol{b}_1 \rangle = -\langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle \\
\iff &\langle \boldsymbol{b}_0 - z \cdot \boldsymbol{1}^n, (z \cdot \boldsymbol{\alpha} + \boldsymbol{b}_1) \circ \boldsymbol{y}^n \rangle + \langle \boldsymbol{y}^n, z^2 \cdot \boldsymbol{\alpha} + z \cdot \boldsymbol{\beta} \rangle = -\langle \boldsymbol{y}^n, \boldsymbol{\gamma} \rangle \\
\iff &\langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle = \delta
\end{aligned} \tag{12}$$

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

In a Σ-protocol, instead of sending $\zeta(\boldsymbol{b}_0)$ and $\eta(\boldsymbol{b}_1)$ directly, a prover will response with $\boldsymbol{\zeta} = (\boldsymbol{b}_0 - z \cdot \boldsymbol{1}^n) + x \cdot \boldsymbol{s}_0$ and $\boldsymbol{\eta} = (z \cdot \boldsymbol{\alpha} + \boldsymbol{b}_1 + x \cdot \boldsymbol{s}_1) \circ \boldsymbol{y}^n$ with a challenge x and some masking values $\boldsymbol{s}_0$ and $\boldsymbol{s}_1$. Thus, Equation (12) becomes

$$\langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle = \langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle + x \cdot \big(\langle \boldsymbol{s}_0, \eta(\boldsymbol{b}_1) \rangle + \langle \zeta(\boldsymbol{b}_0), \boldsymbol{s}_1 \circ \boldsymbol{y}^n \rangle \big) + x^2 \cdot \langle \boldsymbol{s}_0, \boldsymbol{s}_1 \circ \boldsymbol{y}^n \rangle \tag{13}$$

As $\zeta(\cdot)$ is a linear function, we can compute $B_0 = \boldsymbol{g}^{\boldsymbol{\zeta}}$ based on B in Equation (9) 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 on $\langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$. We formally describe the above transforming process in Lemma 1.

Lemma 1. *Any quadratic relation of a vector $\boldsymbol{b}$ can be transformed into a compression-friendly form of inner-product relation, $\langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle = \delta$ by regarding $b_0 = b$ and $b_1 = 1 - b_0$.*

Proof Follow the process in equation (12) and we can get expected forms as $\zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1)$.

Multiple Inner-Product Relations. We further consider the following relation with k-many inner-product equations:

$$R_{IPs} = \left\{ \begin{array}{c} (\boldsymbol{g} \in \mathbb{G}^n, B_0, B_1 \in \mathbb{G}; \boldsymbol{b}_0, \boldsymbol{b}_1 \in \mathbb{Z}_p^n) : B_0 = \boldsymbol{g}^{\boldsymbol{b}_0}, B_1 = \boldsymbol{g}^{\boldsymbol{b}_1}, \\ \langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle = \delta_0, ..., \langle \zeta_{k-1}(\boldsymbol{b}_0), \eta_{k-1}(\boldsymbol{b}_1) \rangle = \delta_{k-1} \end{array} \right\}, \tag{14}$$

where $\zeta_i(\boldsymbol{b}_0) = \boldsymbol{\zeta}_i \circ \boldsymbol{b}_0 + \boldsymbol{\mu}_i$ and $\eta_i(\boldsymbol{b}_1) = \boldsymbol{\eta}_i \circ \boldsymbol{b}_1 + \boldsymbol{\nu}_i$ are linear relations. Note that B_0 and B_1 can also be committed in one commitment under different $\mathbb{G}$ elements, $B = \boldsymbol{g}^{\boldsymbol{b}_0} \boldsymbol{h}^{\boldsymbol{b}_1}$

We first consider two equations, $\langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle = \delta_0$ and $\langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1) \rangle = \delta_1$. Taking a challenge z, we can rewrite the two inner-products as:

$$\langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle + z \cdot \langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1) \rangle = \delta_0 + z \cdot \delta_1. \tag{15}$$

The left-hand side of Equation (15) can be rewritten as one inner-product relation:

$$\begin{aligned}
&\langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1)\rangle + z \cdot \langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1)\rangle \\
&= \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0\rangle + \langle \boldsymbol{\mu}_0, \eta_0(\boldsymbol{b}_1)\rangle + \langle \boldsymbol{b}_0, z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1\rangle + \langle z \cdot \boldsymbol{\mu}_1, \eta_1(\boldsymbol{b}_1)\rangle \\
&= \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1\rangle + \langle \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0, \boldsymbol{b}_1\rangle + \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle \\
&\quad + \langle z \cdot \boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1, \boldsymbol{b}_1\rangle + \langle z \cdot \boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle \\
&= \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1\rangle + \langle \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0 + z \cdot \boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1, \boldsymbol{b}_1\rangle \\
&\quad + \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle + \langle z \cdot \boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle. \qquad (16)
\end{aligned}$$

As $\eta_0(\cdot)$ and $\eta_1(\cdot)$ are linear functions, $\eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1$ is also linear. Let $\eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1 = \boldsymbol{\tau} \circ \boldsymbol{b}_1 + \boldsymbol{\omega}$, $\boldsymbol{\kappa}_0 = \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0 + z \cdot \boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1$ and $\kappa_1 = \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0\rangle + \langle z \cdot \boldsymbol{\mu}_1, \boldsymbol{\nu}_1\rangle$. We can rewrite Equation (16) as:

$$\begin{aligned}
&\langle \boldsymbol{b}_0, \boldsymbol{\tau} \circ \boldsymbol{b}_1 + \boldsymbol{\omega}\rangle + \langle \boldsymbol{\kappa}_0, \boldsymbol{b}_1\rangle + \kappa_1 \\
&= \langle \boldsymbol{\tau} \circ \boldsymbol{b}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle + \langle \boldsymbol{\kappa}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle - \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle + \kappa_1 \\
&= \langle \boldsymbol{\tau} \circ \boldsymbol{b}_0 + \boldsymbol{\kappa}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle - \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle + \kappa_1, \qquad (17)
\end{aligned}$$

where $\boldsymbol{\tau}^{-1}$ is to inverse each element of $\boldsymbol{\tau}$ (i.e., τ_i^{-1}). Thus, it is important to ensure $\tau_i \neq 0$ for the above transformation (which is true for most real-world applications). Taking Equation (16) and (17), we have one inner-product relation

$$\langle \tau(\boldsymbol{b}_0), \omega(\boldsymbol{b}_1)\rangle = \delta, \qquad (18)$$

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 $\delta = \delta_0 + z \cdot \delta_1 - \kappa_1 + \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega}\rangle$.

We can further iteratively conduct the above transformation to rewrite Equation (17) and $\langle \zeta_2(\boldsymbol{b}_0), \eta_2(\boldsymbol{b}_1)\rangle = \delta_2$ to a single inner-product form, and finally convert k-many inner-product relations into one.

When dealing multiple quadratic relations, $f_i(\boldsymbol{b}) = \boldsymbol{\alpha}_i \circ \boldsymbol{b} + \boldsymbol{\beta}_i = \mathbf{0}^n$, we need an additional step as the $\boldsymbol{b}_1$'s in Equation (10) can be different due to $\boldsymbol{\alpha}_i$'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.

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 (12) for $f_1(\boldsymbol{a}) = \mathbf{0}^n$ and $f_2(\boldsymbol{b}) = \mathbf{0}^m$). This can be simply converted into an inner-product form $\langle \boldsymbol{a}_0 || (z \cdot \boldsymbol{b}_0), \boldsymbol{a}_1 || (z \cdot \boldsymbol{b}_1)\rangle = \delta_a + z \cdot \delta_b$. 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 [5], [8] directly to reduce the proof size. We formally describe the transformation of multiple inner-product relations in Lemma 2.

Lemma 2. *For multiple inner-product relations, R_{IPs} defined in (14) and independent inner-product relations (our multiple-secret case), can be transformed into a compression-friendly form of one inner-product relation.*

Proof Follow the process described above mainly in equations 17, 18, we can get expected forms as $\zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1)$.

4 Any-out-of-Many Proofs

We present a new technique, *any-out-of-many* proof for partial knowledge argument. First, we give a general expression of multi-secret relations, and prove that logarithmic-size proofs can be easily achieved with our transformation in Section 3. Taking anonymity into consideration, we further introduce a new relation called any-out-of-many proof, which can provide better anonymity than former relations. Moreover, we discuss the applications and limitations of this novel relation and introduce a revised version, *bounded any-out-of-many* proofs in order to mitigate some of the limitations. We also show that both of these new protocols preserve the logarithmic-size property with our transformation in Section 3.

To begin with, we give a typical definition of multi-secret relations as follows:

Definition 6. *(multi-secret relation).* The following defines the multi-secret relation, which proves the knowledge of k ($k \in [1, N]$) openings $\boldsymbol{s} = (s_0, s_1, ..., s_{k-1})$ in the public set $\boldsymbol{P} = (P_0, ..., P_{N-1})$:

$$R_{k/N} = \left\{ \begin{matrix} (ck, \boldsymbol{P}, k; (\boldsymbol{b}, \boldsymbol{s})) : \boldsymbol{b} \in \{0,1\}^N \wedge \\ HW(\boldsymbol{b}) = k \wedge \boldsymbol{P^b} = \prod_{j=0}^{k-1} Com_{ck}(0; s_j) \end{matrix} \right\}. \tag{19}$$

We can regard the multi-exponentiation $\boldsymbol{P^b}$ as a commitment of the binary vector $\boldsymbol{b} \in \{0,1\}^N$. Note that the binary proof of $\boldsymbol{b}$ can be written as a quadratic relation $\boldsymbol{b} \circ (\boldsymbol{1}^N - \boldsymbol{b}) = \boldsymbol{0}^N$, which can be further transformed into an inner product relation. And $HW(\boldsymbol{b}) = k$ can be also regarded as an inner-product relation $\langle \boldsymbol{b}, \boldsymbol{1}^N \rangle = k$. Thus, the logarithmic-size multi-secret proof can be derived directly based on Lemma 1 in Section 3 and Bulletproofs compression mechanism. As this part is not the emphasis of our paper, details of this multi-secret proof are presented in Appendix A.

Next, we consider the anonymity of multi-secret relations above. We use the size of *possible distribution space* as indicator of measuring the anonymity of relations in Definition 6.2. This indicator is the number of all possible distributions of hiding some secrets in a ring set. For example, when dealing with a one-out-of-many situation, we assume the public set is perfectly undistinguishable, then for an outside observer, the secret can be the opening of any one of N public key with a equivalent possibility. Therefore we can calculate the size of anonymity space (possible distribution space) of this relation is N. And the size of many-out-of-many proof [13] is also N, this is mainly due to the openness of permutation method in this proof. We can aslo calculate the anonymity space of

the compressing partial knowledge proofs of k-Out-Of-N [1] as size of $\binom{N}{k}$, which is higher than many-out-of-many proofs [13]. Consequently, we can find that for multi-secret proofs listed above, their size of anonymity space can be further improved. Because the opening number k will definitely leak some information to observers and thus reduce the distribution space of secrets. Oppositely, if the exact number k of secrets is unknown to observers, the distribution of the secrets can be deemed as a randomly-distributed N-dimensional binary vector, and we can obtain the size of this relation as 2^N, which is higher than the size of former methods.

Intriguingly, we can achieve this goal by simply removing condition $HW(\boldsymbol{b}) = k$ from the relation (6.2). Because the binary proof of $\boldsymbol{b}$ has already prove the validity of $HW(\boldsymbol{b})$ (at most cases except $HW(b) = 0$, we will discuss this special case in Section 4.2.), which can be shown as $\boldsymbol{b} \in \{0,1\}^N \Longrightarrow HW(\boldsymbol{b}) \in [0, N]$. Thus, the new relation aims to prove that she does know some secrets to public keys, but won't reveal exactly how many secrets she knows. We typically name this relation as *any-out-of-many proof* and present its definition as follows:

Definition 7. *(Any-out-of-Many Relation).* The following defines the any-out-of-many relation, which proves the knowledge of arbitrarily many openings (written as vector $\boldsymbol{s}$ and $HW(\boldsymbol{s}) \in [0, N]$) in a public set $\boldsymbol{P} = (P_0, ..., P_{N-1})$:

$$R_{*/N} = \left\{ \begin{matrix} (ck, \boldsymbol{P}; (\boldsymbol{b}, \boldsymbol{s})) : \\ \boldsymbol{b} \in \{0,1\}^N \wedge \boldsymbol{P^b} = Com_{ck}(0; \sum_{s_i \in \boldsymbol{s}} s_i) \end{matrix} \right\} \tag{20}$$

Next, we present the construction of our logarithmic-size any-out-of-many proofs based on arguments in Section 3.

4.1 Any-out-of-Many Proofs with Logarithmic Size

With given security parameters λ_1 and $ck \leftarrow \text{Setup}(1^{\lambda_1})$, generate $\boldsymbol{P} = (P_0, ..., P_{N-1})$ representing a public list consists of N elements ($N \geq 1$). The element $P_i = Com_{ck}(0; s_i)$ in the public list can be deemed as a commitment to 0 with the secret key s_i as the randomness. As the commitment scheme used in public key generation and the following ZKP (zero-knowledge proof) process may be different, we use another security parameter $gk \leftarrow \text{Setup}(1^{\lambda_2})$ to distinguish these two parts. The commitment scheme in ZKP process is conducted with parameters $\mathbb{Z}_q^N, \mathbb{Z}_q, \mathbb{G}$ determined by gk. The Prover needs to show that she knows some of secret keys to the corresponding elements in the public list. For simplicity, we define secret keys that the Prover knows as a set $\boldsymbol{s}$ with arbitrary (hidden) number of elements. And the commitment to the vector $\boldsymbol{s}$ is $\boldsymbol{P}_s$, which is a subset of the public list set $\boldsymbol{P}$. We can use a binary vector $\boldsymbol{b}$ to describe the relation between $\boldsymbol{P}_s$ and $\boldsymbol{P}$ as follows

$$\boldsymbol{b} = (b_0, ..., b_{N-1}),\ b_i = \begin{cases} 0\ (s_i \in \boldsymbol{s}) \\ 1\ (s_i \notin \boldsymbol{s}) \end{cases} \tag{21}$$

For a set $\boldsymbol{P}$ includes N public elements and its subset $\boldsymbol{P}_s$, they satisfy

$$\boldsymbol{P}_s = \{P_i \mid b_i = 1\} \tag{22}$$

Next, we show that how to transform the relation above in Definition 7 into an inner-product form. As mentioned above, the multi-exponentiation $\boldsymbol{P^b}$ is regarded as a commitment of $\boldsymbol{b}$ (obviously an extra blinding value is needed). Instead of proving $\boldsymbol{b} \in \{0,1\}^N$, we prove a quadratic relation of $f(\boldsymbol{b}) = \boldsymbol{b} \circ (\boldsymbol{1}^N - \boldsymbol{b})$. According to Lemma 1, the quadratic relation $f(\boldsymbol{b}) = \boldsymbol{0}^N$ can be rewritten as follows with two challenges $y, z \in \mathbb{Z}_q$.

$$\begin{aligned} & \left\langle \boldsymbol{y}^N, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \right\rangle + z \cdot \left\langle \boldsymbol{y}^N, \boldsymbol{b}_0 - \boldsymbol{1}^N - \boldsymbol{b}_1 \right\rangle = 0 \\ \iff & \left\langle \boldsymbol{b}_0 - z \cdot \boldsymbol{1}^N, (z \cdot \boldsymbol{1}^N + \boldsymbol{b}_1) \circ \boldsymbol{y}^N \right\rangle = \left\langle \boldsymbol{y}^N, z \cdot \boldsymbol{1}^N - z^2 \cdot \boldsymbol{1}^N \right\rangle \\ \iff & \langle \boldsymbol{\zeta}_0, \boldsymbol{\eta}_0 \rangle = \delta(y, z) \end{aligned} \tag{23}$$

where $\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \boldsymbol{1^N}$.

Different from Bulletproofs, in this protocol we commit to secret binary vectors $\boldsymbol{b}_0, \boldsymbol{b}_1$ separately with two challenge values $c, d \in \mathbb{Z}_q$ (d is generated and transmitted along with y, z), in order to exclusively guarantee that $\boldsymbol{P^b} = Com_{ck}(0; \sum_{s_i \in \boldsymbol{s}} s_i)$ holds. Thus, the openings $\boldsymbol{\zeta}, \boldsymbol{\eta}$ of the two vectors $\boldsymbol{\zeta_0}, \boldsymbol{\eta_0}$ in the inner product above are written as,

$$\boldsymbol{\zeta} = \boldsymbol{b}_0 - z \cdot \boldsymbol{1}^N + \boldsymbol{s}_0 \cdot c \tag{24}$$

$$\boldsymbol{\eta} = (z \cdot d \cdot \boldsymbol{1}^N + d \cdot \boldsymbol{b}_1 + \boldsymbol{s}_1 \cdot c) \circ \boldsymbol{y^N} \tag{25}$$

$$\delta(y, z, d) = \left\langle d \cdot \boldsymbol{y}^N s, z^2 \cdot \boldsymbol{1}^N - z \cdot \boldsymbol{1}^N \right\rangle \tag{26}$$

$$\begin{aligned} \hat{t} &= \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle \\ &= \delta(y, z, d) + t_1 \cdot c + t_2 \cdot c^2 \end{aligned} \tag{27}$$

Note that expression $\delta(y, z, d)$ is slightly different from the origin $\delta(y, z)$ because of the introduction of challenge d, but it can still be calculated by verifier without any other information except the proof transcript.

Now we consider providing the zero-knowledge property of the scheme and further construct an interactive protocol. As mentioned above, we use ck for generating public keys $\boldsymbol{P}$ with g, and gk for the zero knowledge proofs of Relation 7. Group elements $\boldsymbol{h}, u, v$ are determined by gk, which are used to generate Pederson commitments of secret vectors. Random values $\alpha, \beta, \gamma, r_s$ and vectors $\boldsymbol{s}_0, \boldsymbol{s}_1$ are the corresponding blinding parameters in generating commitments A, B, C, D.

Several tricks are used in Protocol 1 for security or simplicity. First, we use $\boldsymbol{H} = (h_1, h_2^{(y^{-1})}, h_3^{(y^{-2})}, ..., h_N^{(y^{-N+1})})$ for concise expression. Besides, we can use $Y_i = P_i \cdot g_i^{hash(\boldsymbol{P})}$ to replace the public keys for hiding the DL relations between different public elements as RingCT 3.0 proposed [28], where $hash(\boldsymbol{P})$ denotes the hash value of the public key vector $\boldsymbol{P}$.

As shown in Protocol 1, the prover generates commitments A, B of binary vectors $\boldsymbol{b}_0, \boldsymbol{b}_1$ respectively, with random values α, β. Then she generates a commitment C of two blinding vectors $\boldsymbol{s}_0, \boldsymbol{s}_1$ with another random value γ. D is the commitment of the blinding value of the sum of secrets $\sum_{s_i \in \boldsymbol{s}} s_i$). After transmission of these commitments above, the verifier will generate three challenges

y, z, d. Based on y, z, d, prover computes commitments of coefficients t_1, t_2 with random values tau_1, tau_2. And the verifier check the inner product value $\hat{t}$ with a random challenge x. Next, the prover calculate opening vectors and values, including the two opening vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ of the inner product, and their result $\hat{t}$, τ_c, μ, f_α are the openings of random values, f_s is the opening of secret $\sum_{s_i \in \boldsymbol{s}} s_i$). Finally, according to the transcripts above, the verifier will check equation (1) in Protocol 1 that the commitment of the result $\hat{t}$ is correct; check equation (2) that commitments of vectors $\boldsymbol{\zeta}_0, \boldsymbol{\eta}_0$ are correct; check equation (3) that $\boldsymbol{P}^{\boldsymbol{b_0}}$ is a commitment to zero; check equation (4) that the inner product relation between vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ and $\hat{t}$ holds.

Theorem 1. *The Protocol 1 for proving knowledge of multiple secrets out of N commitments opening to 0 is perfectly complete. It is perfect* $(n+1)$*-special sound if the commitment scheme is perfectly binding. It is perfect special honest verifier zero-knowledge if the commitment scheme is perfectly hiding.*

The proof of the above theorem will be given in Appendix A.

We give a linear-sized *any-out-of-many* proof with compression-friendly forms in Protocol 1.

Based on the improved inner-product argument in Bulletproofs [8], we can further achieve a logarithmic-sized *any-out-of-many* proof by compressing the vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$. We give the compression form of vectors in one recursion using definitions in Section 2.3 as follows

$$\begin{aligned} \boldsymbol{\zeta'} &= c \cdot \boldsymbol{\zeta}_L + \boldsymbol{\zeta}_R, \quad \boldsymbol{\eta'} = \boldsymbol{\eta}_L + c \cdot \boldsymbol{\eta}_R \\ \boldsymbol{Y'} &= \boldsymbol{Y}_L + \boldsymbol{Y}_R^c, \quad \boldsymbol{H'} = \boldsymbol{H}_L^c + \boldsymbol{H}_R \end{aligned} \tag{28}$$

With $L = \boldsymbol{Y}_L^{\boldsymbol{\zeta}_R} \cdot \boldsymbol{H}_L^{\boldsymbol{\eta}_R} \cdot v^{\langle \boldsymbol{\zeta}_R, \boldsymbol{\eta}_L \rangle}$ and $R = \boldsymbol{Y}_R^{\boldsymbol{\zeta}_L} \cdot \boldsymbol{H}_R^{\boldsymbol{\eta}_L} \cdot v^{\langle \boldsymbol{\zeta}_L, \boldsymbol{\eta}_R \rangle}$, we can iteratively do the compression according to equation (8) in Section 2.3 and achieve the logarithmic proof size. Specifically, we only need to send $2 \cdot \lceil log_2(N) \rceil + 6$ group elements and 7 field elements in total. The detailed description of this part will be given in Appendix B.

The protocol above can also be converted into a non-interactive one based on a standard Fiat-Shamir transform [4], of which challenges are generated by hashes of the transcript of the interaction up to that point.

Protocol 1 Any-out-of-Many Proof

$\mathcal{P}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}; k, \boldsymbol{b}, \sum_{s_i \in \boldsymbol{s}} s_i)$		$\mathcal{V}(g, \boldsymbol{h}, u, v, \boldsymbol{Y})$
$\alpha, \beta, \gamma, r_s \leftarrow \mathbb{Z}_q$		
$\boldsymbol{s}_0, \boldsymbol{s}_1 \leftarrow \mathbb{Z}_q^N$		
$\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \mathbf{1}^N$		
$A = \boldsymbol{Y}^{\boldsymbol{b}_0} \cdot u^\alpha$		
$B = \boldsymbol{h}^{\boldsymbol{b}_1} \cdot u^\beta$		
$C = \boldsymbol{Y}^{\boldsymbol{s}_0} \cdot \boldsymbol{h}^{\boldsymbol{s}_1} \cdot u^\gamma$		
$D = g^{r_s} \cdot u^{r_\alpha}$		
	$\xrightarrow{A,B,C,D}$	
		$y, z, d \leftarrow \mathbb{Z}_q$
	$\xleftarrow{y,z,d}$	
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_q$		
$T_1 = v^{t_1} \cdot u^{\tau_1}$		
$T_2 = v^{t_2} \cdot u^{\tau_2}$		
	$\xrightarrow{T_1,T_2}$	
		$c \leftarrow \mathbb{Z}_q$
	$\xleftarrow{c}$	
$\boldsymbol{\zeta} = \zeta(c)$		
$\boldsymbol{\eta} = \eta(c)$		
$\hat{t} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$		
$\tau_c = \tau_2 \cdot c^2 + \tau_1 \cdot c$		
$\mu = \alpha + \beta \cdot d + \gamma \cdot c$		
$f_s = \sum_{s_i \in \boldsymbol{s}} s_i) \cdot c + r_s$		
$f_\alpha = \alpha \cdot c + r_\alpha$		
	$\xrightarrow{\boldsymbol{\zeta},\boldsymbol{\eta},\hat{t},\tau_c,\mu,f_s,f_\alpha}$	
		(1) $v^{\hat{t}} \cdot u^{\tau_c} \stackrel{?}{=} v^{\delta(y,z,d)} \cdot T_1^c \cdot T_2^{c^2}$
		(2) $\boldsymbol{Y}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{H})^{\boldsymbol{\eta}} \cdot u^\mu \stackrel{?}{=} A \cdot B^d \cdot C^c \cdot \boldsymbol{Y}^{-z \cdot \mathbf{1}^N} \cdot \boldsymbol{h}^{z \cdot \mathbf{1}^N}$
		(3) $g^{f_s} \cdot u^{f_\alpha} \stackrel{?}{=} A^c \cdot D$
		(4) $\hat{t} \stackrel{?}{=} \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$

4.2 Bounded Any-out-of-Many Proofs

In our *any-out-of-many* proofs, the prover can prove that she has no secret by running the protocol with $\boldsymbol{s} = \emptyset$ or $\boldsymbol{b} = \mathbf{0}^N$ (in definition 7). This is not a defect of our approach because the *any-out-of-many* relation does allow proving arbitrary number of secrets, obviously an empty secret set shall also be included. However, in some applications, proving an empty secret set is unacceptable. For instance, in (multiple) ring signatures, since the signature is used to authenticate the effectiveness of the message, it has to be generated based on at least one secret key. If we allow the secret set $\boldsymbol{s} = \emptyset$, anyone can sign a message on behalf of a group of users, without any secret key (We present a more detailed discussion of this problem in Section 5.1), and it will undoubtedly breach the security. As a result, we need to restrict the range of element number in set $\boldsymbol{s}$ when applying *any-out-of-many* proofs to some applications. Concretely, to ensure the element number of $\boldsymbol{s}$ lies in a valid range, the prover can equivalently commit to the range of hamming weight of binary vector $\boldsymbol{b}$. We present a general description of this relation and name it as *bounded any-out-of-many* proof.

Definition 8. *(Bounded Any-out-of-Many Relation).* The following defines the relation of *bounded any-out-of-many* proof, which proves that she knows at least one secret $\boldsymbol{s}$ of the public element in $\boldsymbol{P}$:

$$R_{+/N} = \left\{ \begin{matrix} (ck, \boldsymbol{P}; (\boldsymbol{b}, \boldsymbol{s})) : \boldsymbol{b} \in \{0,1\}^N \wedge \\ HW(\boldsymbol{b}) \in [1, N] \wedge \boldsymbol{P}^{\boldsymbol{b}} = Com_{ck}(0; \sum_{s_i \in \boldsymbol{s}} s_i) \end{matrix} \right\} \tag{29}$$

To prove that $HW(\boldsymbol{b}) \in [1, N]$, we set a new vector $\boldsymbol{a} \in \mathbb{Z}_q^{logN}$ which is the binary form of $HW(\boldsymbol{b}) - 1$, and further use a range proof to ensure $\langle \boldsymbol{a}, \mathbf{2}^n \rangle$ lies in $[0, N-1]$, indicating that element number of set $\boldsymbol{s}$ lies in $[1, N]$. Thus, we can transform the relation above into more specific equations as follows:

$$(\boldsymbol{a} \parallel \boldsymbol{b}) \circ (\mathbf{1}^{logN+N} - (\boldsymbol{a} \parallel \boldsymbol{b})) = \mathbf{0}^{logN+N} \tag{30}$$

$$\big\langle (\boldsymbol{a} \parallel \boldsymbol{b}), (-\mathbf{2}^{logN} \parallel \mathbf{1}^N) \big\rangle = 1 \tag{31}$$

$$\boldsymbol{P}^{\boldsymbol{b}} = Com_{ck}(0; \sum_{s_i \in \boldsymbol{s}} s_i) \tag{32}$$

According to Lemma 2, it is feasible to rewrite these equations into compression-friendly forms. Similar to the former process in Section 4.1, by using vectors $\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{b}_0, \boldsymbol{b}_1$, as well as blinding vectors $\boldsymbol{s}_0, \boldsymbol{s}_1 \in \mathbb{Z}_q^{logN+N}$, challenges y, z, c, d, e, we can get the coefficients of inner-product relation $\hat{(t)} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$ as follows

$$\begin{aligned} \boldsymbol{\zeta} &= (\boldsymbol{a}_0 \parallel e \cdot \boldsymbol{b}_0) - z \cdot \mathbf{1}^{logN+N} + \boldsymbol{s}_0 \cdot c \\ \boldsymbol{\eta} &= (z \cdot \mathbf{1}^{logN+N} + d \cdot (\boldsymbol{a}_1 \parallel \boldsymbol{b}_1) + \boldsymbol{s}_1 \cdot c) \circ \boldsymbol{y}^{logN+N} \\ &\quad -z^2 \cdot (\mathbf{2}^{logN} \parallel \mathbf{0}^N) + z^3 \cdot (\mathbf{0}^{logN} \parallel \mathbf{1}^N) \\ \delta(y, z, d, e) &= z \cdot d \cdot \big\langle \mathbf{1}^{logN}, \boldsymbol{y}^{logN} \big\rangle + z \cdot d \cdot e \cdot \big\langle \mathbf{1}^N, \boldsymbol{y}^N \big\rangle \\ &\quad -z^2 \cdot \big\langle \mathbf{1}^{logN+N}, \boldsymbol{y}^{logN+N} \big\rangle - z^3 \cdot \big\langle \mathbf{1}^{logN}, \mathbf{2}^{logN} \big\rangle + z^4 \cdot \big\langle \mathbf{1}^N, \mathbf{1}^N \big\rangle \\ \hat{t} &= \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle = z^3 \cdot e \cdot v - z^2 \cdot (v - 1) + \delta(y, z, d, e) + t_1 \cdot c + t_2 \cdot c^2 \end{aligned}$$

Protocol 2 Bounded Any-out-of-Many Proof

$\mathcal{P}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}; k, \boldsymbol{a} \parallel \boldsymbol{b}, \sum_{s_i \in \boldsymbol{s}} s_i)$		$\mathcal{V}(g, \boldsymbol{h}, u, v, \boldsymbol{Y})$
$\alpha_1, \alpha_2, \beta, \gamma \leftarrow \mathbb{Z}_q$ $\boldsymbol{r}_s, \boldsymbol{s}_0, \boldsymbol{s}_1 \leftarrow \mathbb{Z}_q^{logN+N}$ $\boldsymbol{a}_0 = \boldsymbol{a}, \boldsymbol{a}_1 = \boldsymbol{a}_0 - \boldsymbol{1}^{logN}$ $\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \boldsymbol{1}^{N}$ $A_1 = \boldsymbol{Y}_{[0:logN-1]}^{\boldsymbol{a}_0} \cdot u^{\alpha_1}$ $A_2 = \boldsymbol{Y}_{[logN:N-1]}^{\boldsymbol{b}_0} \cdot u^{\alpha_2}$ $B = \boldsymbol{h}^{(\boldsymbol{a}_1 \| \boldsymbol{b}_1)} \cdot u^{\beta}$ $C = \boldsymbol{Y}^{\boldsymbol{s}_0} \cdot \boldsymbol{h}^{\boldsymbol{s}_1} \cdot u^{\gamma}$ $D = (g \cdot \boldsymbol{1}^{N+logN})^{\boldsymbol{r}_s} \cdot u^{r_\alpha}$		
	$\xrightarrow{A_1, A_2, B, C, D}$	
		$y, z \leftarrow \mathbb{Z}_q$
	$\xleftarrow{y, z}$	
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_q$ $T_1 = v^{t_1} \cdot u_1^{\tau}$ $T_2 = v^{t_2} \cdot u_2^{\tau}$		
	$\xrightarrow{T_1, T_2}$	
		$c, d, e \leftarrow \mathbb{Z}_q$
	$\xleftarrow{c, d, e}$	
$\boldsymbol{\zeta} = \zeta(c)$ $\boldsymbol{\eta} = \eta(c)$ $\hat{t} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$ $\tau_c = \tau_2 \cdot c^2 + \tau_1 \cdot c$ $\mu = \alpha_1 + \alpha_2 \cdot e + \beta \cdot d + \gamma \cdot c$ $f_s = \sum_{s_i \in \boldsymbol{s}} s_i + \boldsymbol{r}_s \cdot c$ $f_\alpha = \alpha_1 + r_\alpha \cdot c$		
	$\xrightarrow{\boldsymbol{\zeta}, \boldsymbol{\eta}, \hat{t}, \tau_c, \mu, f_s, f_\alpha}$	
		$v^{\hat{t}} \cdot u^{\tau_c} \stackrel{?}{=} v^{\delta(y,z)} \cdot T_1^c \cdot T_2^{c^2}$
$\boldsymbol{Y}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{H})^{\boldsymbol{\eta}} \cdot u^{\mu} \stackrel{?}{=}$		$A_1 \cdot A_2^e \cdot B^d \cdot C^c \cdot \boldsymbol{Y}^{-z \cdot \boldsymbol{1}^{logN+N}} \cdot$
$\boldsymbol{H}^{z \cdot \boldsymbol{1}^{logN+N}} \cdot -z^2 \cdot$		$(\boldsymbol{2}^{logN} \parallel \boldsymbol{0}^N) + z^3 \cdot (\boldsymbol{0}^{logN} \parallel \boldsymbol{2}^N)$
		$(g \cdot \boldsymbol{1}^n)^{f_s} \cdot u^{f_\alpha} \stackrel{?}{=} A_1^c \cdot D$
		$\hat{t} \stackrel{?}{=} \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$

where $\boldsymbol{s}_0, \boldsymbol{s}_1$ are used as blinding vectors, d, e are two extra challenges ensuring that A_1 can be verified independently apart from A_2, B. We present the protocol of *bounded any-out-of-many* proofs. $\boldsymbol{h}, \boldsymbol{Y}$ are all vectors of $logN + N$ dimensions, and $\boldsymbol{Y}_{[logN:N-1]}$ is the vector of N public keys, $\boldsymbol{Y}_{[0:logN-1]}$ can be a random generator vector.

We can ensure the security of this protocol.

Theorem 2. *The Protocol 2 for knowledge of bounded number k of secrets out of N commitments opening to 0 is perfectly complete. It is perfect* $(n+1)$*-special sound if the commitment scheme is perfectly binding. It is perfect special honest verifier zero-knowledge if the commitment scheme is perfectly hiding.*

We can also compress vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ using the same way in Bulletproofs[8]. Although we increase the length of vectors, the Bullletproofs compression mechanism will eliminate its impact. The size of the logarithmic *bounded any-out-of-many* proof contains $2 \cdot \lceil log_2(N + log_2(N)) \rceil + 7$ group elements and 7 field elements. As the steps described above are very similar to *any-out-of-many* proof, so we won't give more details here for simplicity.

4.3 RingCT application

The original *any-out-of-many* proof can be applied directly without sacrificing security in RingCT. This is because in the verification stage, the signature is checked along with balance proofs using the binary vector $\boldsymbol{b}$, the simplified process can be expressed as follows:

$$\sum_{i=0}^{N-1} a_{in,i} \cdot b_i = \sum_{j=0}^{M-1} a_{out,j} \tag{33}$$

where $a_{in,i}$, $a_{out,j}$ denote non-negative transaction amounts (balances) of input and output accounts respectively. We can find that if the prover has an empty secret set, which means the binary vector $\boldsymbol{b}$ is $\mathbf{0}$, then the sum of totally M output amounts is equals to zero. Notably, since every amount is non-negative (guaranteed by range proofs), to keep equation (33) hold, the only possible value of the output amount is zero. Thus, although this transaction message can be signed with an empty secret set, the signer can only generate some output accounts with 0 balance, which is meaningless and unprofitable. Besides, it is neither a wise choice for an adversary to launch Denial-of-Service attacks by using these "zero-to-zero" transactions, this is mainly because the transaction fee is also required, so the cost of launching a Dos attack will be prohibitively high. In fact, the current anonymous cryptocurrencies, such as Monero [22], already allow "zero-to-zero" transactions. The only difference is that users must own some input accounts with 0 balance to generate "zero-to-zero" transactions in existing approaches, while our approach allows these transactions even without owning any input accounts.

5 Applications

5.1 Application 1: Multiple Ring Signature

As mentioned above, our bounded any-out-of-many proofs in Protocol 2 can be used to construct an efficient multiple ring signature scheme, which consists of a quadruple of PPT algorithms Π = (**MSetup**,**MKeygen**,**MSign**,**MVerify**), and the multiple ring signature scheme is designed as follows:

- **MSetup**(1^λ): Generate commitment keys ck,gk and a hash function H : $0,1^* \rightarrow \mathcal{C}$. Return $pp = (ck, gk, H)$.
- **MKeygen**(pp):Sample arbitrary number of $s_i \leftarrow \mathbb{Z}_q$ to construct secret set $\boldsymbol{sk}$), and compute corresponding $pk_i = Com_{ck}(0; s_i)$. Generate N-dimensional vector $\boldsymbol{pk}$ with all pk_i and some extra public keys which are not actually used. Return $(\boldsymbol{pk}, \boldsymbol{sk})$.
- **MSign**$(M, \boldsymbol{pk}, pp, \boldsymbol{sk})$: Sign on the message M on behalf of a group represented by public keys $\boldsymbol{pk} = (pk_0, pk_1, ..., pk_{N-1})$ Proceeds as follows:
  1. Run the first stage in Protocol 2 as $\mathcal{P}(ck, gk, \boldsymbol{pk}, (\boldsymbol{b}; \boldsymbol{sk}'))$ and output the initial commitment $CMT_1 = (A_1, A_2, B, C, D)$.
  2. Compute $y = H(ck, M, \boldsymbol{pk}, CMT_1)$ and $z = H(gk, M, \boldsymbol{pk}, CMT_1)$.
  3. With the generated y and z, compute $CMT_2 = (T_1, T_2)$ according to Protocol 2.
  4. Compute $c = H(ck, M, \boldsymbol{pk}, CMT_2)$,$d = H(gk, M, \boldsymbol{pk}, CMT_2)$.
  5. Compute the $RSP = (\tau_c, \mu, \hat{t}, f_\alpha, \sum_{s_i \in \boldsymbol{s}} s_i, \zeta_{log_2(logN+N)}, \eta_{log_2(logN+N)}, P, L_1, R_1..., L_{log_2(logN+N)}, R_{log_2(logN+N)})$.
  6. Return the signature $\pi = (B, S, T_2, x, y, z, RSP)$.
- **MVerify**$(M, \boldsymbol{pk}, \pi, pp)$: Verify the signature with $\boldsymbol{pk} = (pk_0, pk_1, ..., pk_{2N-1})$ and signature $\pi = (B, C, D, T_2, y, z, c, d, e, RSP)$. Proceeds as follows:
  1. Compute Commitment A_1, A_2 and T_1 based on the equations in verification part of Protocol 2 and set $CMT_1 = (A_1, A_2, B, C, D)$, $CMT_2 = (T_1, T_2)$.
  2. Return 0 if there is at least one challenge in (y, z, c, d, e) conflicts with the hash value.
  3. Return the output of $\mathcal{V}(ck, \boldsymbol{pk}, gk)$ with $(CMT_1, CMT_2, y, z, c, d, e, RSP)$.

The correctness and anonymity of our multiple ring signature can be derived directly from the completeness and Special-HVZK of Protocol 1. The unforgeability of the multiple ring signature is formally described as follows:

Theorem 3. *The scheme Π is a ring signature scheme with perfect correctness. It has perfect anonymity if the commitment scheme is perfectly hiding. It is unforgeable in the random oracle model if the commitment scheme is perfectly hiding and computationally binding.*

This Theorem can be proved easily according to the security assumption of *bounded any-out-of-many* proofs in Theorem 2. Note that the signature $\pi = (B, C, D, T_2, y, z, c, d, e, \tau_c, \mu, \hat{t}, f_\alpha, \sum_{s_i \in \boldsymbol{s}} s_i,$
$\zeta_{log_2(logN+N)}, \eta_{log_2(logN+N)}, P, L_1, R_1..., L_{log_2(logN+N)}, R_{log_2(logN+N)})$ includes $2 \cdot \lceil logN + N \rceil + 5$ elements in $\mathbb{G}$ and 12 elements in $\mathbb{Z}_q$.

5.2 Application 2: RingCT

Another application of our *any-out-of-many* proofs is RingCT protocols. As mentioned in Section 1.2, previous one-out-of-many proofs based approaches such as RingCT 2.0 and RingCT 3.0 [27], [28] can not achieve an expected anonymity level, mainly due to the intrinsic property of the ZKP scheme. With *any-out-of-many* proofs, we can set the number of true spenders in one anonymous ring set to an arbitrary unpublished value (in $[0, N-1]$, which will significantly enhance the anonymity of the protocol. This is mainly because one spender's identity is no more relevant to another, and the leakage of one will not lead to further leakage of the other. So the anonymity level of the RingCT based on *any-out-of-many* proofs is $\sigma = 1/2^N$ for any number of real spenders hidden in N accounts, which is a significant improvement.

Based on the idea of *any-out-of-many* proofs, we propose a new RingCT protocol which allows arbitrarily many real spenders in one anonymous ring, which significantly enhances the level of anonymity of the protocol. Since identities of spenders are irrelevant to each other, de-anonymizing one account will not affect others, and we can calculate the anonymity space of our new RingCT as $\sigma = 2^N$ which is far higher than spaces of RingCT 2.0 and RingCT 3.0 as presented above.

Prior to describe our RingCT Protocol based on *any-out-of-many* proofs, we first give some definitions of parameters. We denote an anonymous set representing N accounts as $\boldsymbol{A}_i n$, with N tuples $(pk_{in,i}, cn_{in,i}), (i \in [0, N-1])$, where $pk_{in,i}$ is the i-th account address and $cn_{in,i}$ is the coin with respect to this account. $\boldsymbol{A}_i n$ includes arbitrary number of real spender accounts, we denote them as $\boldsymbol{A}_s$. Instead of arranging the accounts into a matrix, we directly arrange all N accounts into a vector as follows,

$$\boldsymbol{A}_i n = \{(pk_{in,0}, cn_{in,0}), (pk_{in,1}, cn_{in,1}), ..., (pk_{in,N-1}, cn_{in,N-1})\} \tag{34}$$

$\boldsymbol{A}_i n$ can be regarded as a N-dimensional vector with tuple elements $(pk_{in,i}, cn_{in,i})$, and all of real accounts in $\boldsymbol{A}_s$ are distributed randomly in $\boldsymbol{A}_i n$.

Protocol Description According to the discussion above, we propose a new RingCT Protocol with a quintuple Φ=(**Setup**,**KeyGen**,**Mint**,**Spend**,**Verify**). Specifically, the protocol is designed as follows:
Setup(1^λ). Generate commitment keys ck,gk and a hash function $H : \{0,1\}^* \to \mathcal{C}$. Return $pp = (ck, gk, H)$.
KeyGen(pp). For all real spenders, compose secret key vector $\boldsymbol{sk}_s$ with $sk_{s,j} \leftarrow \mathbb{Z}_q$ based on input pp, the number of $sk_{s,j}$ in $\boldsymbol{sk}_s$ is hidden; compose public key vector $\boldsymbol{pk}_s$ with $pk_{s,j} = Com_{ck}(0; sk_{s,j}), sk_{s,j} \in \boldsymbol{sk}_s$. Return $\boldsymbol{pk}_s$.
Mint$(\boldsymbol{pk}_s, \boldsymbol{sk}_s, \boldsymbol{a}_s)$. For each input account address $pk_{s,j}$ of all real spenders and its corresponding balance amount $a_{s,j}$, choose a blinding value $r_{s,j} \in \boldsymbol{r}_s$ as a coin key, $(r_{s,j} \leftarrow \mathbb{Z}_q)$ and compute a commitment $cn_{s,j} = Com_{ck}(a_{s,j}, r_{s,j})$ to mint a coin for each account. Compose public account balance vector $\boldsymbol{cn_s}$ with $cn_{s,j}$ of all accounts. Compose secret account balance vector $\boldsymbol{ck_s}$ with tuple $ck_{s,j} =$

$(a_{s,j}, r_{s,j}), a_{s,j} \in \boldsymbol{a}_s$ of all accounts. Return real spenders' account vector $\boldsymbol{A_s}$ composed with all $(pk_{s,j}, cn_{s,j}), pk_{s,j} \in \boldsymbol{pk}_s, cn_{s,j} \in \boldsymbol{cn_s}$, and the corresponding secret key vector $\boldsymbol{A_{sk}}$ composed with all $(sk_{s,j}, ck_{s,j}), sk_{s,j} \in \boldsymbol{sk}_s, ck_{s,j} \in \boldsymbol{ck_s}$.
Spend$(M, \boldsymbol{A}_s, \boldsymbol{A}_{sk}, \boldsymbol{pk}_{out})$. Construct an anonymous set $\boldsymbol{A}$ of N tuples, including all tuples in $\boldsymbol{A}_s$ and other $N - HW(\boldsymbol{A}_s)$ tuples, and corresponding vectors $\boldsymbol{pk}_{in}$ and $\boldsymbol{cn}_{in}$. The relation between $\boldsymbol{A}_i n$ and $\boldsymbol{A}_s$ can be represented by a binary vector $\boldsymbol{b}$ which satisfies $\boldsymbol{A}_i n^{\boldsymbol{b}} = \boldsymbol{A}_s$. Generate transaction tx as well as several proofs by following steps:

1. Output accounts generation: M is the transaction message. Through calling the **KeyGen**(pp), generate $\boldsymbol{pk}_{out}$ which denotes public keys of m output accounts, $\boldsymbol{pk}_{out} = pk_{out,0}, ..., pk_{out,m-1}$. Through calling the **Mint**$(\boldsymbol{pk}_{out}, \boldsymbol{sk}_{out}, \boldsymbol{a}_{out})$, generate $\boldsymbol{A}_o ut$ composed with all $(pk_{out,i}, cn_{out,i}), pk_{out,i} \in \boldsymbol{pk}_{out}, cn_{out,i} \in \boldsymbol{cn_{out}}$. where $\boldsymbol{a}_{out}$ satisfies $\sum_{a_{s,j} \in \boldsymbol{a}_s} a_{s,j} = \sum_{i=0}^{m-1} a_{out,i}, r_{out,i} \leftarrow \mathbb{Z}_q, cn_{out,i} = Com_{ck}(a_{out,i}, r_{out,i})$.
2. Range proof generation: According to the method in the [8], generate a range proof ϕ_{range} to ensure that every balance amounts of $\boldsymbol{a}_{out}$ is within the valid range (amounts in $\boldsymbol{a}_{in}$ have been already checked as the output amounts in the former transaction).
3. Balance proof generation: Generate a proof $\phi_{balance}$ to show the sum of input accounts equals to the sum of output accounts, $\sum_{a_{s,j} \in \boldsymbol{a}_s} a_{s,j} = \sum_{i=0}^{m-1} a_{out,i}$, which can be verified on the public commitments $(\boldsymbol{cn}_{in})^{\boldsymbol{b}} = \prod_{i=0}^{m-1} \boldsymbol{cn}_{out} \cdot Com_{ck}(0; \prod_{r_{s,j} \in \boldsymbol{r}_s} r_{s,j} - \prod_{i=0}^{m-1} r_{out,i})$.
4. Signature generation: Generate a proof ϕ_{Sig} to prove that the transaction is signed by a group of accounts, which can be written as $(\boldsymbol{pk}_{in})^{\boldsymbol{b}} = \sum_{sk_{s,j} \in \boldsymbol{sk}_s} Com_{ck}(0; sk_{s,j})$.
5. Serial number generation: Compute $s_j = H(pk_{s,j})^{sk_{s,j}}$ as the serial number of account $pk_{s,j}$, which can be used to prevent double-spending. Return the vector of serial numbers $\boldsymbol{S}$ composed with all s_j.

As the above range proof, balance proof and signature can all be converted into inner-product forms, we further aggregate them into one compression-friendly form based on the multiple range proof proposed in [8]. For simplicity, we assume the amounts of all balances in $\boldsymbol{a}_{out}$ lie in the range $[0, 2^l - 1]$ $(2^l > max\{a_{out,i}\})$, and we use vectors $\boldsymbol{b_{in,j}}$ and $\boldsymbol{b_{out,i}}$ to represent the binary forms of $a_{in,j}, a_{out,i}$, finally we get a vector $\boldsymbol{b_0} \in \mathbb{Z}_q^{N+m \cdot l}$ as follows:

$$\boldsymbol{b_0} = (\boldsymbol{b} \parallel \boldsymbol{b_{out,0}} \parallel ... \parallel \boldsymbol{b_{out,m-1}}) \tag{35}$$

The algorithm only needs to generate one proof ϕ that prove the above 3 relation in a zero-knowledge way, as well as the tx (containing M, $\boldsymbol{A}_{in}$ and $\boldsymbol{A}_{out}$) and serial number $\boldsymbol{S}$.
MVerify$(tx, \phi, \boldsymbol{S}, pp)$. With a transaction $tx = (M, \boldsymbol{A}_{in}, \boldsymbol{A}_{out})$ which can be sliced into message M, input and output account set $\boldsymbol{A}_{in}, \boldsymbol{A}_{out}$, as well as proof ϕ for tx, serial number vector $\boldsymbol{S}$ and the security parameter pp, the verifier checks whether a transaction is valid with a two-step verification:

1. Serial Number Check: The verifier checks whether the serial numbers in $\boldsymbol{S}$ are properly generated. If any of them is generated improperly, the verifier will reject the transaction. The verifier checks whether the serial numbers in $\boldsymbol{S}$ exists in previous transactions. If any of them exists, the verifier will reject the transaction since it indicates that this account has been already spent.
2. Balance Check: Based on the given tx and pp, the verifier checks whether the proof ϕ is properly generated, whether sums of input and output accounts are equal, and whether balances of output accounts are non-negative. If any of the above checks failed, the verifier will reject the transaction.

We can further calculate the size of above RingCT protocol, which is 16 field elements and $2 \cdot \lceil log_2(N + m \cdot l) \rceil + 8$ group elements.

Theorem 4. *Our RingCT scheme is unforgeable if the Discrete-Logarithmic assumption holds in $\mathbb{G}$ in the random oracle model. Our RingCT scheme is anonymous against recipients if the q-DDHI (q-Decision Diffie-Hellman Inversion) assumption holds in $\mathbb{G}$ in the ROM, where q is the number of Spend oracle query. Our RingCT scheme is anonymous against ring insiders if the q-DDHI assumption holds in $\mathbb{G}$ in the ROM and RP is a secure zero-knowledge range proof. Our RingCT scheme is non-slanderable w.r.t. insider corruption if the DL assumption holds in $\mathbb{G}$ in the random oracle model.*

Detailed discussion and relevant definitions are given in Appendix C.

6 Evaluation

6.1 Any-out-of-Many Proofs

The theoretical evaluation of *any-out-of-many* proofs is already given in Table 2. And here we implement our *any-out-of-many* proofs scheme in Golang and integrate it into an elliptic curve library of $secp128r1$ with 128-bit parameters [23], in order to evaluate the performance. All experiments were performed on an Intel i5-4210U system throttled to 1.70 GHz and using a single thread. Less than 200 MB of memory was used in all experiments.

We first compare the proof size of our proofs with existing approaches [13], [1], where the same $secp128r1$ library is integrated into many-out-of-many proofs [13]. Meanwhile, a bilinear mapping curve $bn256$ [3] with 256-bit is used when implementing the novel partial knowledge proofs, in order to achieve a same security level as former approaches. The proof size in terms of the anonymous group size (in logarithmic form) is depicted in figure 1.

When the anonymous set is relatively small, the proof size of our approach is not optimal. However, our proof size quickly becomes the smallest among all approaches when $log(N) > 6$. This is mainly because our *any-out-of-many* scheme has the smallest scalar of the logarithmic part while a relatively larger number of field elements.

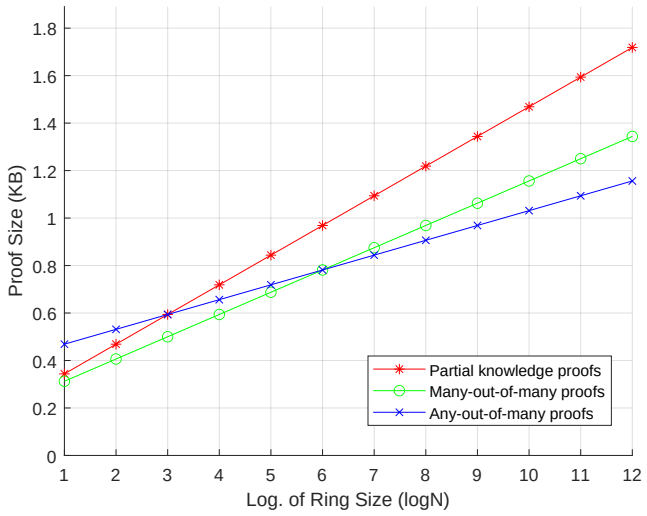

Fig. 1. Proof size comparison of any-out-of-many proofs, partial knowledge proofs and many-out-of-many proofs with different size of public set

Moreover, as we use Bulletproofs compression mechanism in our proofs, both of the prover and verifier complexities are $\mathcal{O}(N)$ for our approach, which is computed in Section 4.1. Thus, we say that our approach is more efficient than many-out-of-many proofs, which has $\mathcal{O}(N \cdot log(N)^2)$ prover complexity.

6.2 RingCT Protocol

We further compare the performance between our RingCT protocol and RingCT 3.0 [28]. First, we define a transaction with N input accounts, k real spenders among them, m output accounts, and the valid range $[0, 2^l)$ for each output balance amount.

The theoretical evaluations of communication cost and anonymity are given in Table 6.2. As the RingCT 3.0 only aggregate the k signatures, its proof size is $\mathcal{O}(log(N \cdot m))$. Relatively, our RingCT scheme aggregates the signature with range proofs, which leads to a smaller size of $\mathcal{O}(log(N + m))$.

Table 3. Comparison of the proof size and anonymity between our proofs and other k-out-of-N approaches

	$\mathbb{G}$ elements	$\mathbb{Z}_q$ elements	Anonymity space
RingCT 3.0 [28]	$2 \cdot \lceil log_2(N^2 + N \cdot m \cdot l) \rceil + 11$	12	$(N/k)^k$
Our RingCT	$2 \cdot \lceil log_2(N + m \cdot l) \rceil + 8$	16	2^N

We can spot this difference intuitively in the following figure, where the proof size of our RingCT protocol is far less than RingCT3.0.

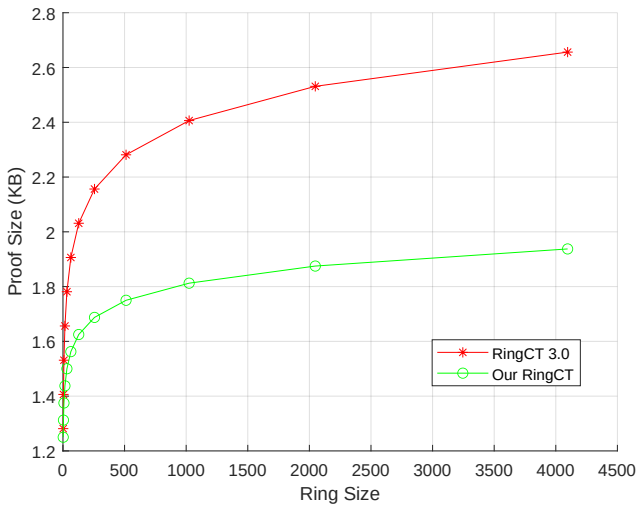

Fig. 2. Proof size comparison of our RingCT and RingCT 3.0 with different ring size

And the figures above fit with the theoretical results well.

References

1. Attema, T., Cramer, R., Fehr, S.: Compressing Proofs of k-out-of-n Partial Knowledge. In: Proc. of the Annual International Cryptology Conference (CRYPTO). Springer (2021)
2. Back, A.: Bitcoins with homomorphic value (validatable but encrypted). `https://bitcointalk.org/index.php?topic=305791`, 2013. [Online; accessed 1-May-2015]
3. Barreto, P.S., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: International Workshop on Selected Areas in Cryptography. pp. 319–331. Springer (2005)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73 (1993)
5. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short Accountable Ring Signatures Based on DDH. In: Proc. of the European Symposium on Research in Computer Security (ESORICS). Springer (2015)
6. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 327–357. Springer (2016)
7. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. In: International Conference on Financial Cryptography and Data Security. pp. 423–443. Springer (2020)
8. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short Proofs for Confidential Transactions and More. In: Proc. of the IEEE Symposium on Security and Privacy (S&P). IEEE (2018)
9. Camenisch, J., Michels, M.: A group signature scheme with improved efficiency. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 160–174. Springer (1998)
10. Conti, M., Kumar, E.S., Lal, C., Ruj, S.: A survey on security and privacy issues of bitcoin. IEEE Communications Surveys & Tutorials **20**(4), 3416–3452 (2018)
11. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Annual International Cryptology Conference. pp. 174–187. Springer (1994)
12. Demuynck, L., De Decker, B.: How to prove list membership in logarithmic time. CW Reports, KU Leuven, Department of Computer Science, vol. CW470 (2006)
13. Diamond, B.E.: "Many-out-of-Many" Proofs with Applications to Anonymous Zether. In: Proc. of the IEEE Symposium on Security and Privacy (S&P). IEEE (2020)
14. Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol. In: Proc. of the ACM Conference on Computer & Communications Security (CCS). ACM (2019)
15. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
16. Groth, J., Kohlweiss, M.: One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin. In: Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). Springer (2015)
17. Jivanyan, A., Mamikonyan, T.: Hierarchical One-out-of-Many Proofs with Applications to Blockchain Privacy and Ring Signatures. In: Prof. of the Asia Joint Conference on Information Security (AsiaJCIS). IEEE (2020)

18. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Australasian Conference on Information Security and Privacy. pp. 325–335. Springer (2004)
19. Noether, S.: Ring Signature Confidential Transactions for Monero. In: IACR Cryptology ePrint Archive (2015)
20. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)
21. Peng, K., Bao, F.: Achieving high efficiency in membership proof without compromising or weakening any security property. In: 2010 10th IEEE International Conference on Computer and Information Technology. pp. 1044–1049. IEEE (2010)
22. Project, M.: Monero. `https://www.getmonero.org/` (2020)
23. Research, C.: Sec 2: Recommended elliptic curve domain parameters. `https://www.secg.org/SEC2-Ver-1.0.pdf`
24. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 552–565. Springer (2001)
25. van Saberhagen, N.: Cryptonote v2.0. `https://cryptonote.org/whitepaper.pdf`, 2013
26. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized Anonymous Payments from Bitcoin. In: Proc. of the IEEE Symposium on Security and Privacy (S&P). IEEE (2014)
27. Sun, S.F., Au, M.H., Liu, J.K., Yuen, T.H.: RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero. In: Proc. of the European Symposium on Research in Computer Security (ESORICS). Springer (2017)
28. Yuen, T.H., Sun, S.f., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: RingCT 3.0 for Blockchain Confidential Transaction: Shorter Size and Stronger Security. In: Cryptology ePrint Archive (2019)
29. Zhang, F., Kim, K.: Id-based blind signature and ring signature from pairings. In: International conference on the theory and application of cryptology and information security. pp. 533–547. Springer (2002)

Appendix A

A.1 k-out-of-N Proofs

Here we give the technical details and security proofs of *k-out-of-N* proofs. According to relation in definition and parameters defined in Section 4, we can write the inner-product form as follows:

$$\begin{aligned}
&\left\langle \boldsymbol{y}^N, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \right\rangle + z \cdot \left\langle \boldsymbol{y}^N, \boldsymbol{b}_0 - \mathbf{1}^N - \boldsymbol{b}_1 \right\rangle + z^2 \cdot \left\langle \mathbf{1}^N, \boldsymbol{b}_0 \right\rangle = z^2 \cdot k && (36)\\
\Longleftrightarrow &\left\langle \boldsymbol{b}_0 - z \cdot \mathbf{1}^N, (z \cdot \mathbf{1}^N + \boldsymbol{b}_1) \circ \boldsymbol{y}^N + z^2 \cdot \mathbf{1}^N \right\rangle && (37)\\
&= z^2 \cdot k + \left\langle \boldsymbol{y}^N, z \cdot \mathbf{1}^N - z^2 \cdot \mathbf{1}^N \right\rangle - z^3 \cdot N\\
\Longleftrightarrow &\langle \boldsymbol{\zeta}_0, \boldsymbol{\eta}_0 \rangle = z^2 \cdot k - z^3 \cdot N + \delta(y, z)
\end{aligned}$$

With two blinding vectors $\boldsymbol{s}_0, \boldsymbol{s}_1$, we can get opening vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$.

$$\begin{aligned}
\boldsymbol{\zeta} &= \boldsymbol{b}_0 - z \cdot \mathbf{1}^N + \boldsymbol{s}_0 \cdot c && (38)\\
\boldsymbol{\eta} &= (z \cdot d \cdot \mathbf{1}^N + d \cdot \boldsymbol{b}_1 + \boldsymbol{s}_1 \cdot c) \circ \boldsymbol{y}^{\boldsymbol{N}} && (39)\\
\delta(y, z, d) &= \left\langle d \cdot \boldsymbol{y}^N s, z^2 \cdot \mathbf{1}^N - z \cdot \mathbf{1}^N \right\rangle && (40)\\
\hat{t} &= \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle\\
&= z^2 \cdot k - z^3 \cdot N + \delta(y, z, d) + t_1 \cdot c + t_2 \cdot c^2 && (41)
\end{aligned}$$

We give the linear-sized k-out-of-N proof in the following protocol, and the general compression process will be described in next part.

Protocol 3 k-out-of-N Proof

$\mathcal{P}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}; \boldsymbol{b}, \sum_{i=0}^{k-1} s_i)$		$\mathcal{V}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}; k)$
$\alpha, \beta, \gamma, r_s \leftarrow \mathbb{Z}_q$ $\boldsymbol{s}_0, \boldsymbol{s}_1 \leftarrow \mathbb{Z}_q^N$ $\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \mathbf{1}^N$		
	$\xrightarrow{A,B,C,D}$	
		$y, z, d \leftarrow \mathbb{Z}_q$
	$\xleftarrow{y,z,d}$	
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_q$ $T_1 = v^{t_1} \cdot u^{\tau_1}$ $T_2 = v^{t_2} \cdot u^{\tau_2}$		
	$\xrightarrow{T_1,T_2}$	
		$c \leftarrow \mathbb{Z}_q$
	$\xleftarrow{c}$	
$\boldsymbol{\zeta} = \zeta(c)$ $\boldsymbol{\eta} = \eta(c)$ $\hat{t} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$ $\tau_c = \tau_2 \cdot c^2 + \tau_1 \cdot c$ $\mu = \alpha + \beta \cdot d + \gamma \cdot c$ $f_s = \sum_{s_i \in \boldsymbol{s}} s_i) \cdot c + r_s$ $f_\alpha = \alpha \cdot c + r_\alpha$		
	$\xrightarrow{\boldsymbol{\zeta},\boldsymbol{\eta},\hat{t},\tau_c,\mu,f_s,f_\alpha}$	
		$v^{\hat{t}} \cdot u^{\tau_c} \stackrel{?}{=} v^{z^2 \cdot k - z^3 \cdot N + \delta(y,z,d)} \cdot T_1^c \cdot T_2^{c^2}$ $\boldsymbol{Y}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{H})^{\boldsymbol{\eta}} \cdot u^{\mu} \stackrel{?}{=}$ $A \cdot B^d \cdot C^c \cdot \boldsymbol{Y}^{-z \cdot \mathbf{1}^N} \cdot \boldsymbol{h}^{z \cdot \mathbf{1}^N}$ $g^{f_s} \cdot u^{f_\alpha} \stackrel{?}{=} A^c \cdot D$ $\hat{t} \stackrel{?}{=} \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$

A.2 Compression Process

We introduce the detailed compression process with improved inner-product argument in Bulletproofs [8]. As the process of three protocols are broadly similar, here we choose the any-out-of-many proof as exmaple. As described in Section 4.1, the any-out-of-many proof can be verified by checking the commitment C that

$$C = v^{\hat{t}} \cdot u^{\tau_c} \cdot \boldsymbol{P}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu} \cdot (g \cdot \mathbf{1}^n)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \tag{42}$$

and we compressed the compute the compressed vectors and the corresponding group elments with a challenge x as:

$$\begin{aligned}
\boldsymbol{\zeta'} &= x \cdot \boldsymbol{\zeta}_L + \boldsymbol{\zeta}_R \\
\boldsymbol{\eta'} &= \boldsymbol{\eta}_L + x \cdot \boldsymbol{\eta}_R \\
\boldsymbol{P'} &= \boldsymbol{P}_L \circ \boldsymbol{P}_R^x \\
\boldsymbol{h''} &= \boldsymbol{h'}_L^x \circ \boldsymbol{h'}_R
\end{aligned}$$

Further we can compute

$$\begin{aligned}
(\boldsymbol{P'})^{\boldsymbol{\zeta'}} &= {\boldsymbol{P}_L}^{\boldsymbol{\zeta}_R} \cdot \boldsymbol{P}^{\boldsymbol{\zeta} \cdot x} \cdot {\boldsymbol{P}_R}^{\boldsymbol{\zeta}_L \cdot x^2} \\
&= L_1 \cdot \boldsymbol{P}^{\boldsymbol{\zeta} \cdot x} \cdot {R_1}^{x^2}
\end{aligned}$$

$$\begin{aligned}
(\boldsymbol{h''})^{\boldsymbol{\eta'}} &= {\boldsymbol{h'}_R}^{\boldsymbol{\eta}_L} \cdot \boldsymbol{h'}^{\boldsymbol{\eta} \cdot x} \cdot {\boldsymbol{h'}_L}^{\boldsymbol{\eta}_R \cdot x^2} \\
&= L_2 \cdot \boldsymbol{h'}^{\boldsymbol{\eta} \cdot x} \cdot {R_2}^{x^2}
\end{aligned}$$

$$\begin{aligned}
\langle \boldsymbol{\zeta'}, \boldsymbol{\eta'} \rangle &= \langle \boldsymbol{\zeta}_R, \boldsymbol{\eta}_L \rangle + \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle + x^2 \cdot \langle \boldsymbol{\zeta}_L, \boldsymbol{\eta}_R \rangle \\
&= L_3 + \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle + x^2 \cdot R_3
\end{aligned}$$

Therefore, we can compute the new C' as:

$$\begin{aligned}
C' &= v^{\hat{t}'} \cdot u^{\tau_c} \cdot \boldsymbol{P'}^{\boldsymbol{\zeta'}} \cdot (\boldsymbol{h''})^{\boldsymbol{\eta'}} \cdot u^{\mu} \cdot (g \cdot \mathbf{1}^{n'} \cdot g^x \cdot \mathbf{1}^{n'})^{\boldsymbol{f}'_s} \cdot u^{f_\alpha} \\
&= (L_1 \cdot L_2 \cdot L_3) \cdot C \cdot (R_1 \cdot R_2 \cdot R_3)^{x^2} \\
&= L \cdot C \cdot R^{x^2}
\end{aligned}$$

As a result, the prover compresses vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ to half of their length, and only with two extra group element L, R, the verifier can check the commitments with the new compressed vectors. We give descriptions of the compression steps below:

Protocol 4 Compression Process

$\mathcal{P}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}; \boldsymbol{b}, \sum_{i=0}^{k-1} s_i)$		$\mathcal{V}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}; k)$
$if\ n = 1:$		
	$\xleftarrow{\tau_x, \mu, \hat{t}, f, l, r}$	
		$\mathcal{V} checks\ if:$
		$C = v^{\hat{t}} \cdot u^{\tau_c} \cdot P^{\zeta} \cdot (h')^{\eta} \cdot u^{\mu} \cdot (g \cdot 1^n)^{f_s} \cdot u^{f_\alpha}$
		$\hat{t} = \langle \zeta, \eta \rangle$
$else if n > 1:$		
$\mathcal{P}\ Computes:$		
$n' = n/2$		
$L = \boldsymbol{P}_L{}^{\boldsymbol{\zeta}_R} \cdot \boldsymbol{h}'_R{}^{\boldsymbol{\eta}_L} \cdot v^{\langle \boldsymbol{\zeta}_R, \boldsymbol{\eta}_L \rangle}$		
$R = \boldsymbol{P}_R{}^{\boldsymbol{\zeta}_L} \cdot \boldsymbol{h}'_L{}^{\boldsymbol{\eta}_R} \cdot v^{\langle \boldsymbol{\zeta}_L, \boldsymbol{\eta}_R \rangle}$		
	$\xleftarrow{L, R}$	
		$x \leftarrow \mathbb{Z}_q$
	$\xrightarrow{x}$	
		$\mathcal{V}\ Computes:$
		$\boldsymbol{P}' = \boldsymbol{P}_L \circ \boldsymbol{P}_R^x$
		$\boldsymbol{H}' = \boldsymbol{H}_L^x \circ \boldsymbol{H}_R$
		$C' = L \cdot C^x \cdot R^{x^2}$
$\boldsymbol{\zeta'} = x \cdot \boldsymbol{\zeta}_L + \boldsymbol{\zeta}_R$		
$\boldsymbol{\eta'} = \boldsymbol{\eta}_L + x \cdot \boldsymbol{\eta}_R$		
	$\xleftarrow{\boldsymbol{\zeta'}, \boldsymbol{\eta'}}$	

Appendix B

B.1 Proof of Theorem 2

Before giving the security proofs, we review the security property of the Zero-knowledge proofs first.

Σ-Protocol Σ-protocol is a type of 3-move interactive proof systems between two parties, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. With the setup algorithm mentioned above, prover can convince 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 Setup(1^λ), a statement c and the corresponding witness r, where $(ck, c, r) \in \mathcal{R}$

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

Definition 9. (Perfect Completeness)
$(\text{Setup}, \mathcal{P}, \mathcal{V})$ is perfectly complete if for all probabilistic polynomial time adversaries $\mathcal{A}$

$$\Pr\left[\mathcal{V}(ck, c, m, x, z) = 1 \middle| \begin{array}{l} ck \leftarrow \text{Setup}(1^\lambda); (c, r) \leftarrow \mathcal{A}(ck); \\ m \leftarrow \mathcal{P}(ck, c, r); x \leftarrow \mathbb{Z}_q^*; z \leftarrow \mathcal{P}(x) \end{array}\right] = 1 \tag{43}$$

Definition 10. (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.

$$\Pr\left[(ck, c, r) \in R \middle| \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); \\ (c, m, x_1, z_1, ..., x_n, z_n) \leftarrow \mathcal{A}(ck); \\ \mathcal{V}(ck, c, m, x_i, z_i) = 1, \forall i \in [1, n]; \\ r \leftarrow \mathcal{E}(ck, c, m, x_1, z_1, ..., x_n, z_n) \end{array}\right] = 1 - \mu(\lambda) \tag{44}$$

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

Definition 11. (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 $\mathcal{A}$

$$\left|\begin{array}{l} \Pr\left[\mathcal{A}(m, z) = 1 \middle| \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); (c, r, x) \leftarrow \mathcal{A}(ck); \\ a \leftarrow \mathcal{P}(ck, c, r); z \leftarrow \mathcal{P}(x); \end{array}\right] - \\ \Pr\left[\mathcal{A}(m, z) = 1 \middle| \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); (c, r, x) \leftarrow \mathcal{A}(ck); \\ (m, z) \leftarrow \mathcal{S}(ck, c, r); \end{array}\right] \end{array}\right| \leqslant \mu(\lambda) \tag{45}$$

Completeness. With correct transcripts from prover, the verifier can verify the following equations:

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

$$\begin{aligned} \boldsymbol{P}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu} &= \boldsymbol{P}^{\boldsymbol{b}_0 - z\cdot \boldsymbol{R}^n + \boldsymbol{s_0}\cdot c} \cdot (\boldsymbol{h'})^{(z\cdot \boldsymbol{R}^n + d\cdot \boldsymbol{b}_1 + \boldsymbol{s_R}\cdot c)\circ \boldsymbol{y^n}} \cdot u^{\alpha+\beta\cdot d+\gamma\cdot c} \\ &= (\boldsymbol{P}^{\boldsymbol{b}_0} \cdot u^{\alpha}) \cdot (\boldsymbol{h}^{\boldsymbol{b}_R} \cdot u^{\beta})^d \cdot (\boldsymbol{P}^{\boldsymbol{s}_0} \cdot \boldsymbol{h}^{\boldsymbol{s}_R} \cdot u^{\gamma})^c \cdot \boldsymbol{P}^{-z\cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z\cdot \boldsymbol{R}^n} \\ &= A \cdot B^d \cdot C^c \cdot \boldsymbol{P}^{-z\cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z\cdot \boldsymbol{R}^n} \end{aligned}$$

$$\begin{aligned} (g \cdot \boldsymbol{R}^n)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} &= (g \cdot \boldsymbol{R}^n)^{\boldsymbol{s}+\boldsymbol{r}_s\cdot c} \cdot u^{\alpha+r_\alpha\cdot c} \\ &= A^x \cdot D \end{aligned}$$

$$\hat{t} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$$

Special-HVZK. With randomly chosen elements $(A, C, T_2, \tau_c, \mu, f_\alpha, \boldsymbol{f_s}, \boldsymbol{\zeta}, \boldsymbol{\eta})$, and the challenge values (x, y, c, d) from their corresponding domains, the simulator can compute:

$$\begin{aligned} \hat{t} &= \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle \\ T_1 &= v^{-\delta(y,z)/c} \cdot T_2^{-c} \\ B &= (\boldsymbol{P}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu})^{1/d} \cdot (A \cdot C^c \cdot \boldsymbol{P}^{-z\cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z\cdot \boldsymbol{R}^n})^{-1/d} \\ D &= (g \cdot \boldsymbol{R}^n)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \cdot A^{-x} \end{aligned}$$

Thus, a honest verifier can not distinguish the transcripts generated above from the transcripts in true convesations, if the Pederson commitment is perfectly hiding. The Protocol 1 is said to be perfect special honest verifier zero-knowledge.

$n+1$-Special Soundness. Suppose the adversary can return the transcript 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. The adversary returns two transcripts for rewinding with 2 different challenge c_1, c_2, which can be verified as:

$$\begin{cases} A \cdot B^d \cdot C^{c_R} \cdot \boldsymbol{P}^{-z\cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z\cdot \boldsymbol{R}^n} = \boldsymbol{P}^{\boldsymbol{\zeta}_R} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}_R} \cdot u_1^{\mu} \\ A \cdot B^d \cdot C^{c_2} \cdot \boldsymbol{P}^{-z\cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z\cdot \boldsymbol{R}^n} = \boldsymbol{P}^{\boldsymbol{\zeta}_2} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}_2} \cdot u_2^{\mu} \end{cases}$$

and the extractor can extract $A \cdot B^d$ as:

$$A \cdot B^d = u^{\alpha'} \cdot \boldsymbol{P}^{\boldsymbol{b}'_0} \cdot \boldsymbol{h}^{\boldsymbol{b}'_1}$$

and extract C as:

$$C = \boldsymbol{P}^{\boldsymbol{s}'_0} \cdot \boldsymbol{h}^{\boldsymbol{s}'_1} \cdot u^{\gamma'}$$

With $A \cdot B^d$ and C known, the extractor can substitute the following equation as:

$$\begin{aligned} \boldsymbol{P}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h}')^{\boldsymbol{\eta}} \cdot u^{\mu} &= A \cdot B^d \cdot C^c \cdot \boldsymbol{P}^{-z \cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z \cdot \boldsymbol{R}^n} \\ &= (u^{\alpha'} \cdot \boldsymbol{P}^{\boldsymbol{b}'_0} \cdot \boldsymbol{h}^{\boldsymbol{b}'_1}) \cdot (\boldsymbol{P}^{\boldsymbol{s}'_0} \cdot \boldsymbol{h}^{\boldsymbol{s}'_1} \cdot u^{\gamma'})^c \cdot \boldsymbol{P}^{-z \cdot \boldsymbol{R}^n} \cdot \boldsymbol{h}^{z \cdot \boldsymbol{R}^n} \end{aligned}$$

thus the vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ can be extracted as:

$$\begin{aligned} \boldsymbol{\zeta} &= \boldsymbol{b}'_0 - z \cdot \boldsymbol{R}^n + \boldsymbol{s}'_0 \cdot c \\ \boldsymbol{\eta} &= (z \cdot \boldsymbol{R}^n + d \cdot \boldsymbol{b}'_1 + \boldsymbol{s}'_1 \cdot c) \circ \boldsymbol{y}^{\boldsymbol{n}} \end{aligned}$$

Elements T_1, T_2 can also be extracted with other 3 challenges c_3, c_4, c_5 and the rewinding transcripts:

$$\begin{cases} v^{\hat{t_3}} \cdot u^{\tau_{c,3}} = v^{\delta(y,z)} \cdot T_1^{c_3} \cdot T_2^{{c_3}^2} \\ v^{\hat{t_4}} \cdot u^{\tau_{c,4}} = v^{\delta(y,z)} \cdot T_1^{c_4} \cdot T_2^{{c_4}^2} \\ v^{\hat{t_5}} \cdot u^{\tau_{c,5}} = v^{\delta(y,z)} \cdot T_1^{c_5} \cdot T_2^{{c_5}^2} \end{cases}$$

we can compute T_1, T_2:

$$\begin{aligned} T_1 &= v^{t'_1} \cdot u^{\tau'_1} \\ T_2 &= v^{t'_2} \cdot u^{\tau'_2} \\ v^{\hat{t}} \cdot u^{\tau_c} &= v^{\delta(y,z)} \cdot T_1^c \cdot T_2^{c^2} \\ &= v^{\delta(y,z)} \cdot (v^{t'_1} \cdot u^{\tau'_1})^c \cdot (v^{t'_2} \cdot u^{\tau'_2})^{c^2} \\ \hat{t} &= \delta(y,z) + t'_1 \cdot c + t'_2 \cdot c^2 \end{aligned}$$

Also we can compute $\hat{t}$ in another equation as:

$$\begin{aligned} \hat{t} &= \langle \zeta(c), \eta(c) \rangle \\ &= \langle \boldsymbol{b}_0 - z \cdot \boldsymbol{R}^n + \boldsymbol{s}_0 \cdot c, (z \cdot \boldsymbol{R}^n + d \cdot \boldsymbol{b}_1 + \boldsymbol{s}_1 \cdot c) \circ \boldsymbol{y}^{\boldsymbol{n}} \rangle \\ &= \langle \boldsymbol{y}^n, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \rangle + z \cdot \langle \boldsymbol{y}^n, \boldsymbol{b}_0 - \boldsymbol{R}^n - \boldsymbol{b}_1 \rangle + t''_1 \cdot c + t''_2 \cdot c^2 \end{aligned}$$

compared with the two expressions of $\hat{t}$, which will always hold for all challenges x, y, c, d, we can get the relations of: $\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \boldsymbol{1^n}$

What's more, by knowing $A \cdot B^d = u^{\alpha'} \cdot \boldsymbol{P}^{\boldsymbol{b}'_0} \cdot \boldsymbol{h}^{\boldsymbol{b}'_1}$ holds for all challenge d, we can deduce that $A = u^{\alpha'} \cdot \boldsymbol{P}^{\boldsymbol{b}'_0} = (g \cdot \boldsymbol{R}^n)^{\boldsymbol{x'_{sk}}} \cdot u^{\alpha'}$.

Similarly, by knowing rewinding transcripts for different challenge c_6, c_7 as

$$\begin{cases} (g \cdot \boldsymbol{R}^n)^{\boldsymbol{f}_s, 6} \cdot u^{f_{\alpha,6}} = A^x \cdot D \\ (g \cdot \boldsymbol{R}^n)^{\boldsymbol{f}_s, 7} \cdot u^{f_{\alpha,7}} = A^x \cdot D \end{cases}$$

we can get another equation of $A = (g \cdot \boldsymbol{R}^n)^{\boldsymbol{sk'}} \cdot u^{\alpha'}$, Finally, we can extract $\boldsymbol{x'_{sk}} = \boldsymbol{sk'}$ if the Pederson commitment is perfectly binding. Thus, the Protocol 1 is $(n+1)$-special sound.

Appendix C

In this section we give the strong security model defined in RingCT 3.0, and the security proofs will be given in the full-version article.

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.

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

Perfect Correctness. 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.

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 a honestly generated spending.