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 an arbitrary number of secrets out of a public list. Unlike existing k-out-of-N proofs [S&P'21, CRYPTO'21], our approach also hides the exact amount of secrets k, which can be used to achieve a higher anonymity level. 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, a *bounded* any-out-of-many proof, is presented, which preserves all nice features of the original protocol such as high anonymity and logarithmic size. As for the RingCT, both the original and bounded proofs can be used safely. The result of the performance evaluation indicates that our RingCT protocol is more efficient and secure than others. We also believe our techniques are applicable in other privacy-preserving occasions.

Keywords: zero-knowledge, k-out-of-N 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 maintained by every user in the network, hence the correctness and immutability of the whole system will be guaranteed in an honest-majority situation. To allow easy verification of all transactions, some cryptocurrencies adopt a public approach that records all transaction details in plaintext. For example, the balance and address of each account are accessible to anyone in the Bitcoin network [**?**]. Unfortunately, the public property hinders the original blockchain scheme from private applications. As a result, this deficiency has impelled the development of private cryptocurrencies [**?**], [**?**]. Providing confidentiality and

anonymity to a cryptocurrency system while allowing public verifications has been a hotspot in this field.

Existing private cryptocurrencies use ring confidential transaction (RingCT) protocols [**?**], [**?**], [**?**], [**?**] 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 [**?**]. For the anonymity property, Groth et al. propose a logarithmic size one-out-of-many proof [**?**], which enables the prover to prove a statement about *one* message among a list of commitments, and requiring only 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 protocol are far from satisfactory. For example, the Pedersen vector commitment is not used in [**?**], resulting in large communication cost. Even worse, one-out-of-many proofs can only open 1 out of N public commitments to zero, which contributes to incompatibilities in the case of multiple inputs accounts. As the first deficiency has been overcome in [**?**] through using Pedersen vector commitments, we will focus on the second one.

As one-out-of-many proofs only allow *one* valid user in a proof, while extra $N-1$ public accounts are needed as the anonymous set, this mechanism will lead to the inefficiency in the case of multiple inputs accounts, since one-out-of-many proofs should be conducted for every input account respectively. To solve this problem, one of the mainstream methods is leaking *multiple* secrets among one public list However, current proposals for multiple secrets proofs have either been prohibitively complex or undermined the anonymity. Neither meets our satisfaction. Although these two methods above can leak multiple secrets from one ring set and reduce the proof size to a logarithmic scale [**?**], [**?**], there are also some limitations of them. First, the many-out-of-many proof protocol only works for permutations with orbits of equal size [**?**], which is not a general k-out-of-N proof. Moreover, the verification cost of a partial knowledge proof [**?**] is very high, and the cyclic group operations are less efficient when using bilinear mapping. Besides, 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 [**?**], 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 permutation. Thus, it is important to make accounts (secrets) distribute randomly (independent) in the set. We will give a further discussion of these problems in Section 4.3. In this work, our goal is to propose a more efficient proof scheme for leaking multiple secrets meanwhile with better anonymity for RingCT.

1.1 Our Contributions

We conclude our contributions as follows:

Bulletproofs compression for general relations. Inspired by previous work [**?**], [**?**], we show how to adopt 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 [**?**], logarithmic size proofs can be easily achieved. In Section 3, we will describe the general model of transforming relations into inner-product forms, and futher compressing them.

Any-out-of-many proofs. We propose *any-out-of-many* proofs, a new technique to prove the knowledge of arbitrarily number of secrets among a public list. Different from the previous multiple secrets proofs, our scheme doesn't reveal the specific 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 [**?**], we obtain short 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 compressed by applying 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 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 compared with existing approaches.

1.2 Related Work

As introduced by Cramer et al. in [**?**], 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 define these proof schemes as k-out-of-N 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 [**?**], [**?**], membership proofs in [**?**], [**?**] and also the ring signature schemes. Ring signatures were first introduced by Rivest et al. in [**?**], 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 [**?**], [**?**], improved ring signature scheme are constructed based on non-interactive zero-knowledge proofs. With the popularity of blockchain technology, people find ring signature schemes can be applied to provide anonymity in private cryptocurrencies. Thus, ring signatures once again arouse research interests and are widely applied such as in CryptoNote [**?**] and Ring Coin [**?**]. In Monero [**?**], 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 present several formal security definitions. Since then, RingCT protocol has become an important and direct application of k-out-of-N proofs [**?**]. In [**?**], 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 it is also applied in following research e.g. Zether [**?**] and Monero [**?**].

Although one-out-of-many proofs [**?**] and the improved versions [**?**], [**?**] have been proved to be efficient and practical, there are still some unsatisfactory properties of this scheme. As mentioned above, one-out-of-many proofs can only open 1 out of N public commitments to zero, which will contribute to incompatibilities when there are multiple inputs accounts in a RingCT scenario. To guarantee anonymity for multiple users with logarithmic cost, existing mainstream solutions can be grossly divided into two groups. First solution is to batch mulitple one-out-of-many proofs into one during the construction of RingCT protocol. These methods have already been implemented in RingCT 2.0 and RingCT 3.0 [**?**,**?**] while excessive size of anonymous set and loss of generality makes them less desirable, we will give more discussions about these in Section 5. The other solution is designing a new logarithmic proof scheme which can reveal multiple secrets at once, as proposed in [**?**], [**?**]. One major challenge in realizing scheme is to reduce the proof size, as the optimization method of using Kronecker product [**?**] can not be directly applied in the multi-secret situation. To solve this, Diamond proposes many-out-of-many proofs [**?**] to map multiple secrets in one based on permutations and linear mapping, as a result the task can be transformed as an one-out-of-many question. Attema et al. introduce novel partial knowledge proofs [**?**] to convert the index information to a polynomial function and prove the coefficients of the function to avoid binary proofs. Bulletproofs compression [**?**] can be further applied on these non-binary coefficients directly to reduce the proof size. Our scheme of any-out-of-many proofs also follows the idea of second solution. Finally, We give a brief overview of some typical k-out-of-N proofs as follows:

1.3 Comparison with Other Approaches

The first wildly applied approach without a trusted setup is one-out-of-many proofs [**?**] as mentioned above. 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

Table 1. Overview of some typical k-out-of-N proofs

Proof type	Typical shcemes	Range of k
Single secret proofs	[**?**], [**?**]	1
Multiple secret proofs	[**?**], [**?**]	$[1, N]$
Any secret proofs	Our method	$[0, N]$

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 [**?**] 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 [**?**].

In comparison, in our protocol, we allow the prover to prove any number of secrets. When proving one secret, our approach also outperforms [**?**] and [**?**] 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, S&P'21[**?**] proposes a generalization form of one-out-of-many proofs, many-out-of-many proofs. 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 novel partial knowledge proof proposed in Crypto'21 [**?**]. 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 prove the knowledge of the coefficients of the function to avoid binary proofs. Furthermore, the function is transformed into a homomorphisms form which is Bulletproofs compression-friendly to achieve logarithmic size. Though this approach is the most general multi-secret proof, the proof size is the largest. Though the authors argue that the size of partial-knowledge proofs can be reduced to $2 \cdot \lceil log_2(2 \cdot N - k + 1) \rceil + 3$, pairing-friendly elliptic curves are needed in order to reduce the communication cost. As a result, both the efficiency and security level will be downgraded due to the pairing-friendly groups. Besides, the verification cost is also high as it requires checking a relation on each public commitment.

Different from many-out-of-many and novel partial knowledge proofs, our proof scheme allows arbitrarily number of secrets to be hidden in a public set. Besides, our approach has the best performance under a large ring set size of N. We give a brief review and comparison with other approaches [**?**], [**?**], which is presented in Table (2). In which we consider a situation of proving knowledge of

k secrets in a ring set consists of N public elements. 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 [**?**] seems to be smaller, it is based on a bilinear pairing that needs a larger group domain 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 [**?**] has to use a larger group with the order of 256, which almost doubles the proof size. Even worse, the verification complexity of the novel 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 *arbitrarily* number of secrets in a public set.

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

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 $\mathcal{A}$,

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

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

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

The Pedersen commitment [**?**] and Pedersen vector commitment [**?**] are two natural examples of homomorphic commitment schemes which are 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:

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

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 $\mathbb{G}$

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

2.3 Bulletproofs Compression

Bulletproofs protocol is an interactive approach to compress n-size vectors to scalars with $log(n)$ times of iterations [**?**]. This scheme is mainly based on the inner product argument[**?**], which aims to prove that the prover knows the openings of two binding 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 $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$.

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

To verify the commitment, the prover needs to send vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ to the verifier, totally $2n$ elements as the proof size. In order to reduce this size from linear to logarithmic, an elegant compression mechanism is given in [**?**]. 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:

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

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 $R = \boldsymbol{g}_R^{\boldsymbol{a}_L} \cdot \boldsymbol{h}_R^{\boldsymbol{b}_L} \cdot u^{\langle \boldsymbol{a}_L, \boldsymbol{b}_R \rangle}$. As shown in the equations above, we can send shorter vectors $\boldsymbol{a'}$ and $\boldsymbol{b'}$ instead of $\boldsymbol{a}, \boldsymbol{b}$, with two extra elements L and R. Meanwhile, 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 and extra two group elements 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

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

Motivated by previous work, 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:

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

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 a quadratic relation where $\boldsymbol{\alpha} = \mathbf{0}^n$. When considering hiding, we can simply regard $\boldsymbol{b}' = (\boldsymbol{b}||r)$ and $\boldsymbol{g}' = (\boldsymbol{g}||h)$ in Pedersen vector commitment.

To ensure that $f(\boldsymbol{b}) = \mathbf{0}^n$ 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:

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

where $\zeta(\boldsymbol{b}_0) = \boldsymbol{b}_0 - z \cdot \mathbf{1}^n$, $\eta(\boldsymbol{b}_1) = (z \cdot \boldsymbol{\alpha} + \boldsymbol{b}_1) \circ \boldsymbol{y}^n$, 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 $\boldsymbol{l} = (\boldsymbol{b}_0 - z \cdot \mathbf{1}^n) + x \cdot \boldsymbol{s}_0$ and $\boldsymbol{r} = (z \cdot \boldsymbol{\alpha} + \boldsymbol{b}_1 + x \cdot \boldsymbol{s}_1) \circ \boldsymbol{y}^n$ with a challenge x and some masking values $\boldsymbol{s}_0$ and $\boldsymbol{s}_1$. Thus, Equation (12) becomes $\langle \boldsymbol{l}, \boldsymbol{r} \rangle = \langle \zeta(\boldsymbol{b}_0), \eta(\boldsymbol{b}_1) \rangle + x \cdot \left(\langle \boldsymbol{s}_0, \eta(\boldsymbol{b}_1) \rangle + \langle \zeta(\boldsymbol{b}_0), \boldsymbol{s}_1 \circ \boldsymbol{y}^n \rangle \right) + x^2 \cdot \langle \boldsymbol{s}_0, \boldsymbol{s}_1 \circ \boldsymbol{y}^n \rangle$. As $\zeta(\cdot)$ is a linear function, we can compute $B_0 = \boldsymbol{g}^{\boldsymbol{l}}$ based on B in Equation (9) and the commitment of $\boldsymbol{s}_0$, $S_0 = \boldsymbol{g}^{\boldsymbol{s}_0}$ (so does $B_1 = \boldsymbol{g}^{\boldsymbol{r}}$ 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 $\langle \boldsymbol{l}, \boldsymbol{r} \rangle$. According to the derivation process above, we can get the Lemma:

Lemma 1. *Any quadratic relation of a vector $\boldsymbol{b}$ which satisfies the form of $f(\boldsymbol{b}) = \boldsymbol{b} \circ (\boldsymbol{\alpha} \circ \boldsymbol{b} + \boldsymbol{\beta}) + \boldsymbol{\gamma}$ can be transformed into a compression-friendly form containing one inner-product.*

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

$$R = \left\{ \begin{matrix} (\boldsymbol{g} \in \mathbb{G}^n, B_0, B_1 \in \mathbb{G}; \boldsymbol{b}_0, \boldsymbol{b}_1 \in \mathbb{Z}_p^n) : B_0 = \boldsymbol{g}^{\boldsymbol{b}_0}, B_1 = \boldsymbol{g}^{\boldsymbol{b}_1}, \\ \langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle = \delta_0, ..., \langle \zeta_{k-1}(\boldsymbol{b}_0), \eta_{k-1}(\boldsymbol{b}_1) \rangle = \delta_{k-1} \end{matrix} \right\}, \tag{13}$$

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 commitment 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:

$$\langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle + z \cdot \langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1) \rangle = \delta_0 + z \cdot \delta_1. \tag{14}$$

The right-hand side of Equation (14) can be rewritten as one inner-product relation:

$$\begin{aligned} & \langle \zeta_0(\boldsymbol{b}_0), \eta_0(\boldsymbol{b}_1) \rangle + z \cdot \langle \zeta_1(\boldsymbol{b}_0), \eta_1(\boldsymbol{b}_1) \rangle \\ = & \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 \rangle + \langle \boldsymbol{\mu}_0, \eta_0(\boldsymbol{b}_1) \rangle + \langle \boldsymbol{b}_0, z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1 \rangle + \langle z \cdot \boldsymbol{\mu}_1, \eta_1(\boldsymbol{b}_1) \rangle \\ = & \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1 \rangle + \langle \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0, \boldsymbol{b}_1 \rangle + \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0 \rangle \\ & + \langle z \cdot \boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1, \boldsymbol{b}_1 \rangle + \langle z \cdot \boldsymbol{\mu}_1, \boldsymbol{\nu}_1 \rangle \\ = & \langle \boldsymbol{b}_0, \eta_0(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_0 + z \cdot \eta_1(\boldsymbol{b}_1) \circ \boldsymbol{\zeta}_1 \rangle + \langle \boldsymbol{\mu}_0 \circ \boldsymbol{\eta}_0 + z \cdot \boldsymbol{\mu}_1 \circ \boldsymbol{\eta}_1, \boldsymbol{b}_1 \rangle \\ & + \langle \boldsymbol{\mu}_0, \boldsymbol{\nu}_0 \rangle + \langle z \cdot \boldsymbol{\mu}_1, \boldsymbol{\nu}_1 \rangle. \end{aligned} \tag{15}$$

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 (15) as:

$$\begin{aligned} & \langle \boldsymbol{b}_0, \boldsymbol{\tau} \circ \boldsymbol{b}_1 + \boldsymbol{\omega} \rangle + \langle \boldsymbol{\kappa}_0, \boldsymbol{b}_1 \rangle + \kappa_1 \\ = & \langle \boldsymbol{\tau} \circ \boldsymbol{b}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega} \rangle + \langle \boldsymbol{\kappa}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega} \rangle - \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega} \rangle + \kappa_1 \\ = & \langle \boldsymbol{\tau} \circ \boldsymbol{b}_0 + \boldsymbol{\kappa}_0, \boldsymbol{b}_1 + \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega} \rangle - \langle \boldsymbol{\kappa}_0, \boldsymbol{\tau}^{-1} \circ \boldsymbol{\omega} \rangle + \kappa_1, \end{aligned} \tag{16}$$

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 (15) and (16), we have one inner-product relation

$$\langle \tau(\boldsymbol{b}_0), \omega(\boldsymbol{b}_1) \rangle = \delta, \tag{17}$$

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 (16) 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 $\boldsymbol{l}$ and $\boldsymbol{r}$ respectively. Finally, we can use the Bulletproofs compression [**?**], [**?**] directly to reduce the proof size.

Thus, we get another lemma as follows:

Lemma 2. *For any number of quadratic relations, each has its own vector $\boldsymbol{b}_i$, all of them can be transformed into a compression-friendly form containing one inner-product.*

4 Any-out-of-Many Proofs

In this section, we present our any-out-of-many proofs. Based on the prior discussions, it's clear that our first challenge is to design an elegant logarithmic zero-knowledge proof without sacrificing the efficiency and generality. Motivated by the our transformation of general relations in Section 3, we can convert the k-out-of-N relation into an inner-product form based on our transformation model for general relations in Section 3. The relation of a k-out-of-N proof is given as follows:

Definition 6. *(k-out-of-N relation).* The following defines the k-out-of-N relation, which proves the knowledge of k openings $\boldsymbol{s}' = (s_{l_0}, ..., s_{l_{k-1}})$ in a public set $\boldsymbol{P} = (P_0, ..., P_{N-1})$:

$$R_{k/N} = \left\{ \begin{matrix} (ck, \boldsymbol{P}; (\boldsymbol{b}, \boldsymbol{s}')) : \\ \boldsymbol{b} \in \{0,1\}^N \wedge HW(\boldsymbol{b}) = k \wedge \boldsymbol{P}^{\boldsymbol{b}} = \prod_{j=0}^{k-1} Com_{ck}(0; s'_j) \end{matrix} \right\}. \tag{18}$$

Since we can regard the multi-exponentiation $\boldsymbol{P}^{\boldsymbol{b}}$ as a commitment of vector $\boldsymbol{b}$, $\boldsymbol{b} \in \{0,1\}^N$ as a quadratic relation $\boldsymbol{b} \circ (\mathbf{1}^N - \boldsymbol{b}) = \mathbf{0}^N$, and $HW(\boldsymbol{b}) = k$ as an inner-product relation $\langle \boldsymbol{b}, \mathbf{1}^N \rangle = k$, it is feasible to transform the k-out-of-N relation into compression-friendly forms according to Lemma 1 presented in Section 3.

The second challenge is anonymity. As we have shown that the distribution of secrets is critical to the anonymity and security, designing an approach of which secrets are independently distributed in the anonymous set is important. This property is similar to the concept of "anonymity against insider attacks"

defined by Yuen et al. in RingCT 3.0 [**?**], which is used to highlight the irrelevance between the different spenders among the anonymous set (we give a more detailed discussion in Section 5.2). Here we can simply apply this property as an indicator of measuring the anonymity of k-out-of-N proofs. Thus, we say the anonymity of the partial knowledge proof [**?**] is higher than many-out-of-many proofs [**?**] as the indexes of all secrets can be predicted when the permutation method and any one of indexes is leaked. Furthermore, we find that openning the number of secrets k, will also leak some information. This may cause some serious problems especially when $k \approx N$, because the outsider can randomly choose an account from the ring set, and has a large possibility to believe it is a true spender. Due to this reason, a large ring set and relatively small size of secrets should be set for real-world applications, which may cause an inefficiency. So we can observe that if there exists any method which does not open the exact value of k, the efficiency and anonymity of the scheme would be enhanced significantly. Thankfully, this goal can be achieved by removing the relation of $HW(\boldsymbol{b}) = k$ directly. This is simply because the range of $HW(\boldsymbol{b})$ can be inferred from the binary proof ($\boldsymbol{b} \in \{0,1\}^N \implies HW(\boldsymbol{b}) \in [0, N]$). So the value of $HW(\boldsymbol{b})$ will always be valid (The occassion of $HW(\boldsymbol{b}) = 0$ will be discussed in Section 5). We name this novel scheme as *any-out-of-many proof* and present its relation 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 $\boldsymbol{s'} = (s_{l_0}, ..., s_{l_{k-1}})$ (including $k = 0$ case) in a public set $\boldsymbol{P} = (P_0, ..., P_{N-1})$:

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

Next, we present the construction of our logarithmic-size any-out-of-many proofs.

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

We give some definitions first to explain our protocol clearly. Let $\boldsymbol{P} = (P_0, ..., P_{N-1})$ represents a public list consists of N elements ($N \geq 1$), and the element P_i in the public list can be deemed as a commitment to 0, $P_i = Com_{ck}(0; s_i)$, with the given security parameter λ_1 and $ck \leftarrow \text{Setup}(1^{\lambda_1})$. As the commitment scheme in the public key generation and the ZKP process may be different, here we use another security parameter to distinguish these two proceeds. The commitment scheme used in the ZKP process is described as $gk \leftarrow \text{Setup}(1^{\lambda_2})$. and $\mathbb{Z}_q^N, \mathbb{Z}_q, \mathbb{G}$ we used in the protocol are determined by gk. The Prover needs to demonstrate that she knows k secret keys to the corresponding elements in the public list. For convenience, we define the subset of secret keys that the Prover knows as a vector $\boldsymbol{s'} = (s_{l_0}, ..., s_{l_{k-1}})$. And the commitment to the vector $\boldsymbol{r}$ is $\boldsymbol{P'} = (P_{l_0}, ..., P_{l_{k-1}})$, where the vector $\boldsymbol{l} = (l_0, ..., l_{k-1})$ represents the index of the commitments in public list, $l_i \in [0, N)$, We can use a binary vector $\boldsymbol{b}$ to describe the relation

between $\boldsymbol{P'}$ and $\boldsymbol{P}$, where $\|\boldsymbol{L'}\|_1 = \prod_{j=0}^{k-1} P_{l_j} = \prod_{i=0}^{N-1} P_i^{b_i} = \boldsymbol{P^b}$, and vector $\boldsymbol{b}$ satisfies:

$$\boldsymbol{b} = (b_0, ..., b_{N-1}),\ b_i = \begin{cases} 0\ (i \in \boldsymbol{l}) \\ 1\ (i \notin \boldsymbol{l}) \end{cases} \tag{20}$$

According to our general transformation model, we can transform relations given in Definition 4.2 into an inner-product form. The multi-exponentiation $\boldsymbol{P^b}$ can be regarded as a commitment. And we can prove a quadratic relation $\boldsymbol{b} \circ (\boldsymbol{1}^N - \boldsymbol{b})$ instead of proving $\boldsymbol{b} \in \{0,1\}^N$. According to the transformation steps for *quadratic relaion* given in Section 3. The relation can be rewritten as:

$$\begin{aligned} & f(\boldsymbol{b_0}, \boldsymbol{b_1}) = \boldsymbol{0}^n \\ \Longleftrightarrow\ & \left\langle \boldsymbol{y}^N, \boldsymbol{b}_0 \circ \boldsymbol{b}_1 \right\rangle + z \cdot \left\langle \boldsymbol{y}^N, \boldsymbol{b}_0 - \boldsymbol{1}^N - \boldsymbol{b}_1 \right\rangle = 0 \\ \Longleftrightarrow\ & \left\langle \boldsymbol{b}_0 - z \cdot \boldsymbol{1}^N, (z \cdot \boldsymbol{1}^N + \boldsymbol{b}_1) \circ \boldsymbol{y}^N \right\rangle + \left\langle \boldsymbol{y}^N, z^2 \cdot \boldsymbol{1}^N - z \cdot \boldsymbol{1}^N \right\rangle = 0 \\ \Longleftrightarrow\ & \langle \boldsymbol{\zeta_0}, \boldsymbol{\eta_0} \rangle + \delta(y, z) = 0 \end{aligned} \tag{21}$$

where $\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \boldsymbol{1^N}$, $y, z \in \mathbb{Z}_q$ are challenge values.

Based on the inner product form relation in equation (21), we can construct *any-out-of-many* proofs as shown in Protocol 1. We use ck for generating public keys $\boldsymbol{P}$ with g, and gk for the zero knowledge proofs of Relation 4.2. Group elements $\boldsymbol{h}, u, v$ are determined by gk. The $\boldsymbol{h'}$ in Protocol 1 is $\boldsymbol{h'} = (h_1, h_2^{(y^{-1})}, h_3^{(y^{-2})}, ..., h_N^{(y^{-N+1})})$, which is for concise expression. Different from Bulletproofs, in this protocol we commit to blinding values b_0, b_1 separately with two challenge values $c, d \in \mathbb{Z}_q$, in order to check that $\boldsymbol{P^b} = \prod_{j=0}^{k-1} Com_{ck}(0; s'_j)$ holds. Thus, the openings $\zeta(c), \eta(c)$ of the two vectors $\boldsymbol{\zeta_0}, \boldsymbol{\eta_0}$ in the inner product above are written as,

$$\zeta(c) = \boldsymbol{b}_0 - z \cdot \boldsymbol{1}^N + \boldsymbol{s}_0 \cdot c \tag{22}$$

$$\eta(c) = (z \cdot \boldsymbol{1}^N + d \cdot \boldsymbol{b}_1 + \boldsymbol{s}_1 \cdot c) \circ \boldsymbol{y^N} \tag{23}$$

$$\delta(y, z) = \left\langle \boldsymbol{y}^N, z^2 \cdot \boldsymbol{1}^N - z \cdot \boldsymbol{1}^N \right\rangle \tag{24}$$

$$\begin{aligned} t(c) &= \langle \zeta(c), \eta(c) \rangle \\ &= \delta(y, z) + t_1 \cdot c + t_2 \cdot c^2 \end{aligned} \tag{25}$$

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.*

we give the proof of the above theorem in Appendix A.

In the protocol, several tricks are used to apply the compression method to the relations smoothly and securely. First, we pad the secret vector $\boldsymbol{s'}$ with zeros to extend it to a vector $\boldsymbol{s}$ of length N, with the equation $\boldsymbol{P^b} = \sum_{i=0}^{N-1} Com_{ck}(0; \boldsymbol{s}_i)$ still holds. The purpose of this trick is to keep the blinding vectors of $\boldsymbol{b}$ and $\boldsymbol{s'}$ in the same length, so that they can be compressed simultaneously. What's more, according to RingCT 3.0[**?**], we use $Y_i = P_i \cdot g_i^{hash(\boldsymbol{P})}$ to replace the public keys

Protocol 1 Any-out-of-Many Proof

$\mathcal{P}(g, \boldsymbol{h}, u, v, \boldsymbol{Y}, (\boldsymbol{b}; \boldsymbol{s}))$		$\mathcal{V}(g, \boldsymbol{h}, u, v, \boldsymbol{Y})$
$\alpha, \beta, \gamma \leftarrow \mathbb{Z}_q$		
$\boldsymbol{r}_s, \boldsymbol{s}_0, \boldsymbol{s}_1 \leftarrow \mathbb{Z}_q^N$		
$\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \mathbf{1}^N$		
$A = \boldsymbol{Y}^{\boldsymbol{b}_0} \cdot u^\alpha$		
$B = \boldsymbol{h}^{\boldsymbol{b}_1} \cdot u^\beta$		
$C = \boldsymbol{Y}^{\boldsymbol{s}_0} \cdot \boldsymbol{h}^{\boldsymbol{s}_1} \cdot u^\gamma$		
$D = (g \cdot \mathbf{1}^N)^{\boldsymbol{r}_s} \cdot u^{r_\alpha}$		
	$\xrightarrow{A,B,C,D}$	
		$y, z \leftarrow \mathbb{Z}_q$
	$\xleftarrow{y,z}$	
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_q$		
$T_1 = v^{t_1} \cdot u_1^\tau$		
$T_2 = v^{t_2} \cdot u_2^\tau$		
	$\xrightarrow{T_1,T_2}$	
		$c, d \leftarrow \mathbb{Z}_q$
	$\xleftarrow{c,d}$	
$\boldsymbol{\zeta} = \zeta(c)$		
$\boldsymbol{\eta} = \eta(c)$		
$\hat{t} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$		
$\tau_c = \tau_2 \cdot c^2 + \tau_1 \cdot c$		
$\mu = \alpha + \beta \cdot d + \gamma \cdot c$		
$\boldsymbol{f}_s = \boldsymbol{s} + \boldsymbol{r}_s \cdot c$		
$f_\alpha = \alpha + r_\alpha \cdot c$		
	$\xrightarrow{\boldsymbol{\zeta},\boldsymbol{\eta},\hat{t},\tau_c,\mu,\boldsymbol{f}_s,f_\alpha}$	
		(1) $v^{\hat{t}} \cdot u^{\tau_c} \stackrel{?}{=} v^{\delta(y,z)} \cdot T_1^c \cdot T_2^{c^2}$
		(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) $(g \cdot \mathbf{1}^N)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \stackrel{?}{=} A^x \cdot D$
		(4) $\hat{t} \stackrel{?}{=} \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$

for hiding the DL relations between different public elements, where $hash(\boldsymbol{P})$ denotes the hash value of the public key vector $\boldsymbol{P}$.

Notably, the verifier needs to check equation (1) in Protocol 1 that the commitment of the result $\hat{t}$ is correct, check equation (2) that the commitments of vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ are correct, check equation (3) that $\boldsymbol{P}^{\boldsymbol{b_o}}$ is a commitment to zero. and check equation (4) that the inner product relation of vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ holds.

Now we can compress the vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ with the improved inner-product argument in Bulletproofs[**?**]. Note that vector $\boldsymbol{f}_s$ can also be compressed, since the equation (3) is independent of the inner product relation, we can directly multiply them together into one E as follows:

$$E = v^{\hat{t}} \cdot u^{\tau_c} \cdot \boldsymbol{Y}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu} \cdot (g \cdot \mathbf{1}^N)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \tag{26}$$

Then we can give the compression form of the vectors in one recursion using definitions in Section 2.4.

$$\boldsymbol{\zeta'} = c \cdot \boldsymbol{\zeta}_0 + \boldsymbol{\zeta}_1 \tag{27}$$

$$\boldsymbol{\eta'} = \boldsymbol{\eta}_0 + c \cdot \boldsymbol{\eta}_1 \tag{28}$$

$$\boldsymbol{f}'_s = \boldsymbol{f}_{s,0} + c \cdot \boldsymbol{f}_{s,1} \tag{29}$$

And the computing result indicates that a batch method can be applied. Thus, for each recursion of compression, we can reduce the length of vectors above by half with only two extra group elements to be transferred. The detailed process of this part is given in Appendix A. Finally, we can reduce the proof size from linear to logarithmic totally with $2 \cdot \lceil log_2(N) \rceil + 6$ group elements and 7 field elements.

What's more, the protocol above can also be converted based on into a non-interactive one based on a standard Fiat-Shamir transform [**?**], of which challenges are generated by hashes of the transcript of the interaction up to that point. The Fiat-Shamir transformation can also be applied in other interactive protocols in this paper.

4.2 Extension and Discussion

Based on the definition of any-out-of-many proofs in Definition , we can observe that our proofs will hold for any binary vector $\boldsymbol{b}$, even when $\boldsymbol{b} = \mathbf{0}^n$. This is not a defect in our approach as the goal of our any-out-of-many proof is to prove arbitrary number of elements in a set, undoubtedly an empty set also belongs to it. However, in some applications, proving an empty 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 by at least one valid user. If the number k is allowed to be zero, then anyone can generate a valid signature on behalf of a group of users, no matter whether he/she holds the secret key (we give more details of multiple ring signature application in Section 5.1). To solve this problem, the range of k should be restricted in some occassions, specifically, a range proof needs to be added to the relations of origin any-out-of-many proofs

to ensure k is non-zero. We set a new vector $\boldsymbol{a} \in \mathbb{Z}_q^{log(N)}$ which is the binary representation of $k-1$, then we make a range proof of $\boldsymbol{a}$ to ensure $\langle \boldsymbol{a}, \mathbf{2}^n \rangle$ lies in range $[0, N-1]$, which indicates $k \in [1, N]$. Note that vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be aggregated into one vector $(\boldsymbol{a} \parallel \boldsymbol{b})$ based on Lemma 2. And the result vector satisfies the following relations, we call it a *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 the knowledge of at least one opening $\boldsymbol{s'} = (s'_0, ..., s'_{k-1})$ $(k \geq 1)$ in a public set $\boldsymbol{P} = (P_0, ..., P_{N-1})$:

$$R_{+/N} = \left\{ \begin{array}{c} (ck, gk, A, \boldsymbol{P}); (\boldsymbol{a}, \boldsymbol{b}, r, \boldsymbol{s})) : \\ \boldsymbol{P^b} = \prod_{i=0}^{N-1} Com_{ck}(0; s_i) \wedge A = Com_{gk}(\boldsymbol{a}; r) \\ \wedge (\boldsymbol{a} \parallel \boldsymbol{b}) \in \{0,1\}^{log(N)+N} \wedge \langle (\boldsymbol{a} \parallel \boldsymbol{b}), (-\mathbf{2}^{log(N)} \parallel \mathbf{1}^N) \rangle = 1 \end{array} \right\} \quad (30)$$

Although we increase the length of vectors, the compression mechanism will eliminate its impact and finally the proof size of the logarithmic *bounded any-out-of-many* proof are $2 \cdot \lceil log_2(N + log_2(N)) \rceil + 7$ group elements and 7 field elements. We present more details of the bounded any-out-of-many proof in Appendix B.

For RingCT protocols, the original any-out-of-many proof can be applied to it securely. This is because in the verification stage, the signature is checked along with the balance proof using the same vector $\boldsymbol{b}$, the simplified process can be expressed as follows:

$$\prod_{i=0}^{N-1} (P_i \cdot A_{in,i})^{b_i} = \prod_{j=0}^{k-1} Com_{ck}(0; s_j) \cdot \prod_{l=0}^{m-1} A_{out,l} \quad (31)$$

where P_i denotes the public key of an input account, and $A_{in,i}$, $A_{out,i}$ denote the non-negative transaction amounts (balances) of an input account and output account respectively. Notably, if the number k is zero, which also means $\boldsymbol{b} = \mathbf{0}^N$, we can find that there are no accounts that truly participate in this transaction. Although this transaction message can be signed by none of the group users, the output amounts must be zero. Therefore, users can only generate accounts with 0 balance and cannot illegally mint coins by themselves. Also, as the transaction fee is still needed when this "zero-to-zero" transaction is issued, it is not attractive for an adversary to launch Denial-of-Service attacks by utilizing this characteristic. In fact, the existing RingCT protocol also allows "zero-to-zero" transactions. The only difference is that users must own some input account with 0 balance to generate "zero-to-zero" transactions in existing approaches, while our approach allows users to generate "zero-to-zero" transactions even without owning any input account.

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. This scheme con-

sists of a quadruple of PPT algorithms $\Pi = (\mathbf{MSetup}, \mathbf{MKeygen}, \mathbf{MSign}, \mathbf{MVerify})$, and the multiple ring signature scheme is designed as follows:

- $\mathbf{MSetup}(1^\lambda)$: Generate commitment keys ck,gk and a hash function $H : 0,1^* \to \mathcal{C}$. Return $pp = (ck, gk, H)$.
- $\mathbf{MKeygen}(pp)$:Sample $(s_{l_0}, ..., s_{l_{k-1}}) \leftarrow \mathbb{Z}_q^k$ as $\boldsymbol{sk'}$, and compute $\boldsymbol{pk'_{l_i}} = Com_{ck}(0; s_{l_i})$. Return $(\boldsymbol{pk'}, \boldsymbol{sk'})$.
- $\mathbf{MSign}(M, \boldsymbol{pk}, pp, \boldsymbol{sk})$: Sign on the message M on behalf of a group represented by public keys $\boldsymbol{pk} = (pk_0, pk_1, ..., pk_{2N-1})$ and $\boldsymbol{pk'} \subset \boldsymbol{pk}$, as well as the corresponding secret binary vecstor $\boldsymbol{b}$. Proceeds as follows:
  1. Run the first stage in Protocol 2 as $\mathcal{P}(ck, gk, \boldsymbol{pk}, (\boldsymbol{b}; \boldsymbol{sk'}))$ and ouput 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 $c = H(ck, M, \boldsymbol{pk}, CMT_2)$,$d = H(gk, M, \boldsymbol{pk}, CMT_2)$, $d = H(ck, gk, M, \boldsymbol{pk}, CMT_2)$.
  5. Compute the $RSP = (\tau_c, \mu, \hat{t}, f_\alpha, f_{s,log_2(2N)}, \zeta_{log_2(2N)}, \eta_{log_2(2N)}, P, L_1, R_1..., L_{log_2(2N)}, R_{log_2(2N)})$.
  6. Return the signature $\pi = (B, S, T_2, x, y, z, RSP)$.
- $\mathbf{MVerify}(M, \boldsymbol{pk}, \pi, pp)$: 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 SHVZK of Protocol 1. The unforgeability of the multiple ring signature is formally described as follows:

Theorem 2. *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.*

Note that the signature $\pi = (B, C, D, T_2, y, z, c, d, e, \tau_c, \mu, \hat{t}, f_\alpha, f_{s,log_2(2N)}, \zeta_{log_2(2N)}, \eta_{log_2(2N)}, P, L_1, R_1..., L_{log_2(2N)}, R_{log_2(2N)})$ includes $2 \cdot \lceil log_2(2N) \rceil + 5$ elements in $\mathbb{G}$ and 12 elements in $\mathbb{Z}_q$.

5.2 Application 2: RingCT

In this part, we present a new RingCT protocol based on our *any-out-of-many* proofs. In previous RingCT protocols based on one-out-of-many proofs, for example,[**?**], a $k \times n$ matrix is used to represent k anonymity groups of input accounts(public keys), and each group consists of n accounts, in which we assume

the s-th account is the true spender. As a result, a transaction needs to contain $N = k \cdot n$ public keys to hide the true spender by one-out-of-many proofs. To reduce the size of the transaction, RingCT 2.0 [**?**] uses an extra group element u to accumulate the 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 undoubtfully 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 also leak, simply because they are in the same row of the matrix.

So the level of anonymity of RingCT 2.0 is low. Specifically, we let σ denote the size of the anonymous space, and compute it as the sum of possible permutation results for k spenders in N accounts. Then we say that anonymous space σ for RingCT 2.0 is n, which means that there are only n different ways to hide k spenders. Although in RingCT 3.0[**?**], Yuen et al. propose a method to break the distribution of real signers, as their method is based on one-out-of-many proofs, which restricts that each row in a matrix can contain only one account, there is still room for improvements.

With *any-out-of-many* proofs, we can increase the number of true spenders in one anonymous ring to an arbitrary unknown k, which will significantly enhance the level of anonymity of the protocol, 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 k spenders hidden in N accounts, which is a significant improvement.

Next, we will give a description of the novel RingCT Protocol constructed with our *any-out-of-many* proofs, we give the definitions of parameters first. Without loss of generality, we denote a group of N input accounts by $\boldsymbol{A} = \{(pk_{in,i}, cn_{in,i})\}, (i \in [0, N-1])$ where $pk_{in,i}$ is the $i-th$ account address and $cn_{in,i}$ is the coin with respect to this account. $\boldsymbol{A_s} = \{(pk_{s,j}, cn_{s,j})\}, (j \in [0, k-1])$ is the subset of $\boldsymbol{A}$, including k true spenders' accounts in total, the definition of index l is given in Section 4.1. Instead of arranging the accounts into a matrix, we directly arrange all N accounts into a vector as follows,

$$\left(pk_{s,0}, pk_{s,1}, ..., pk_{s,N-1}\right) \tag{32}$$

and the accounts of true spenders' $pk_{s,\boldsymbol{l}} = (pk_{s,l_0}, ..., pk_{s,l_{k-1}})$ are independently distributed in this vector.

Protocol Description According to the discussion above, we can construct our new RingCT Protocol with a quintuple Φ=(**Setup**,**KeyGen**,**Mint**,**Spend**,**Verify**), and 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). on input pp, the algorithm samples $(sk_{s,l_0}, ..., sk_{s,l_{k-1}}) \leftarrow \mathbb{Z}_q^k$ as $\boldsymbol{sk_s}$, and compute $pk_{s,j} = Com_{ck}(0; sk_{s,j})$. Return $(\boldsymbol{pk_s}, \boldsymbol{sk_s})$.

Mint($\boldsymbol{pk_s}, \boldsymbol{a_s}$). For each input account address $pk_{s,j}$ and its corresponding balance amount $a_{s,j}$, the algorithm chooses a blinding value $r_{s,j} \leftarrow \mathbb{Z}_q$ and computes commitment $c_{s,j} = Com_{ck}(a_{s,j}, r_{s,j})$ to mint a coin for each account. Let $\boldsymbol{cn_s} = (c_{s,0}, .., c_{s,k-1})$ denote the public vector of account balance, and $\boldsymbol{ck_s} = ((r_{s,0}, a_{s,0}), .., (r_{s,k-1}, a_{s,k-1}))$ denote the secret key vector to the public balance vector. Return spender accounts vector $\boldsymbol{A_s} = \{(pk_{s,j}, cn_{s,j})\}, (j \in [0, k-1])$, and the corresponding secret keys vector $\boldsymbol{A_{sk}} = \{(sk_{s,j}, ck_{s,j})\}, (j \in [0, k-1])$.
Spend($M, \boldsymbol{A_s}, \boldsymbol{A_{sk}}, \boldsymbol{pk_{out}}$). With input accounts $\boldsymbol{A_s}$, the algorithm constructs an arbitrary account group $\boldsymbol{A}$ which contains $\boldsymbol{A_s}$ and other $N-k$ accounts, $\boldsymbol{A} = \{(pk_{in,i}, cn_{in,i})\}, (i \in [0, N-1])$, the relation between $\boldsymbol{A}$ and $\boldsymbol{A_s}$ can be represented by a binary vector $\boldsymbol{b}$ which satisfies $\boldsymbol{A^b} = \boldsymbol{A_s}$. M is the transaction message and $\boldsymbol{pk_{out}}$ denotes the m output accounts, $\boldsymbol{pk_{out}} = pk_{out,0}, ..., pk_{out,m-1}$ This algorithm will output a transaction tx as well as several proofs by following steps:

1. Output coins generation: Set balance vector $\boldsymbol{a_{out}}$ for all output accounts in $\boldsymbol{pk_{out}}$, such that the input and output balances satisfy $\sum_{j=1}^{k} a_{s,j} = \sum_{l=1}^{m} a_{out,l}$, then choose a blinding value $r_{out,l} \leftarrow \mathbb{Z}_q$ and computes commitment $c_{out,l} = Com_{ck}(a_{out,l}, r_{out,l})$ to mint a coin for each account, $\boldsymbol{cn_{out}} = (c_{out,0}, .., c_{out,m-1})$ and $\boldsymbol{ck_{out}} = ((r_{out,0}, a_{out,0}), .., (r_{out,m-1}, a_{out}^{m-1}))$ Thus we have $\boldsymbol{R} = (\boldsymbol{pk_{out}}, \boldsymbol{cn_{out}})$.
2. Range proof generation: According to the method in the [**?**], the algorithm generates a range proof ϕ_{range} to ensure that every balance amounts of $\boldsymbol{a_{in}}$ and $\boldsymbol{a_{out}}$ is within the valid range.
3. Balance proof generation: Generate a proof $\phi_{balance}$ to prove that the sum of input accounts equals to the sum of output accounts, $\sum_{j=1}^{k} a_{s,j} = \sum_{l=1}^{m} a_{out,l}$, which can be also expressed as $(\boldsymbol{cn_{in}})^{\boldsymbol{b}} = \prod_{l=0}^{m-1} \boldsymbol{cn_{out}} \cdot Com_{ck}(0; \prod_{j=0}^{k-1} r_{s,j} - \prod_{l=0}^{m-1} r_{out,l})$.
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 $(\boldsymbol{pk_{in}})^{\boldsymbol{b}} = \sum_{j=0}^{k-1} Com_{ck}(0; sk_{s,j})$.
5. Serial number generation: Compute $s_j = H(pk_{s,j})^{sk_{s,j}} (j \in [0, k-1])$ as the serial number of account $pk_{s,j}$, which can be used to prevent double-spending. Return the vector of serial numbers $\boldsymbol{S} = (s_0, ..., s_{k-1})$.

We can observe that the 3 proofs above are all based on the compression methods in improved inner-product argument, thus we can aggregate them into one proof. First, step (3),(4) both prove the relation of the binary vector $\boldsymbol{b}$, and we can simply combine the two public vectors $\boldsymbol{cn_{in}}$ and $\boldsymbol{pk_{in}}$ into one vector $\boldsymbol{P} = \boldsymbol{cn_{in}} \circ \boldsymbol{pk_{in}}$, which is the same as the public list defined in our k-out-of-many proofs. Next, inspired by the multiple range proof proposed in [**?**], we find it is feasible to combine our proof of binary vector $\boldsymbol{b}$ with the binary vector of balances in step (2). For simplicity, we assume the amounts of all balances in $\boldsymbol{a_{in}}$ and $\boldsymbol{a_{out}}$ lie in the range $[0, 2^o - 1]$, and we use vectors $\boldsymbol{b_{in,i}}$ and $\boldsymbol{b_{out,l}}$ to represent the binary forms of $a_{in,i}, a_{out,l}$, finally we get a vector $\boldsymbol{b_L} \in \mathbb{Z}_q^{o \cdot m + (o+1) \cdot N}$ as

follows:

$$\boldsymbol{b_L} = (\boldsymbol{b} \parallel \boldsymbol{b_{in,0}} \parallel ... \parallel \boldsymbol{b_{in,N-1}} \parallel \boldsymbol{b_{out,0}} \parallel ... \parallel \boldsymbol{b_{out,m-1}}) \tag{33}$$

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, $\boldsymbol{A}$ and $\boldsymbol{R}$) and serial number $\boldsymbol{S}$.

MVerify(tx, ϕ, S, pp). With a transaction tx containing M, $\boldsymbol{A}$ and $\boldsymbol{R}$, the proof ϕ for tx, serial number vector S and the security parameter pp, the verifier can check whether a set of accounts of spenders from the account group A was spent for a transaction tx towards output address group R in a correct way.

1. The verifier first checks whether the serial numbers in S exists in the past transactions, return -1 if exists, since it indicates that this account has been already spent.
2. The verifier checks the proof ϕ based on the given tx,S and pp to ensure that the commitments to the relations in steps (2)(3)(4) and the serial number in S are correct.

Finally, we can compute the size of the aggregated proof, including signature and range proof, is $2 \cdot \lceil log_2(N \cdot (o+1) + m \cdot o) \rceil + 8$ elements in $\mathbb{G}$ and 16 elements in $\mathbb{Z}_q$.

Theorem 3. *Our RingCT scheme is unforgeable if the DL assumption holds in 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 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 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 G in the random oracle model.*

Detailed discussion and relevant definitions are given in Appendix C.

6 Evaluation

6.1 Efficiency of Any-out-of-Many Proofs

As shown in Table (2), our *any-out-of-many* proofs can achieve a logarithmic proof size as well as the highest level of anonymity. In this section, simulations have been done to quantitatively evaluate the performance of the efficiency. In our simulations, we implement our proofs scheme using the elliptic curve secp128r1, of which the security parameter $\lambda = 128$. Meanwhile, to achieve the same security level, a bilinear mapping curve with the order of 256 needs to be used when implementing the partial knowledge proofs. Here we choose the curve bn256, and Fig.1 presents the growth of proof size with the ring size increasing of three different ZKP schemes. We can observe that when the public set number is small, the proof size of our approach is not optimal, however, as the size

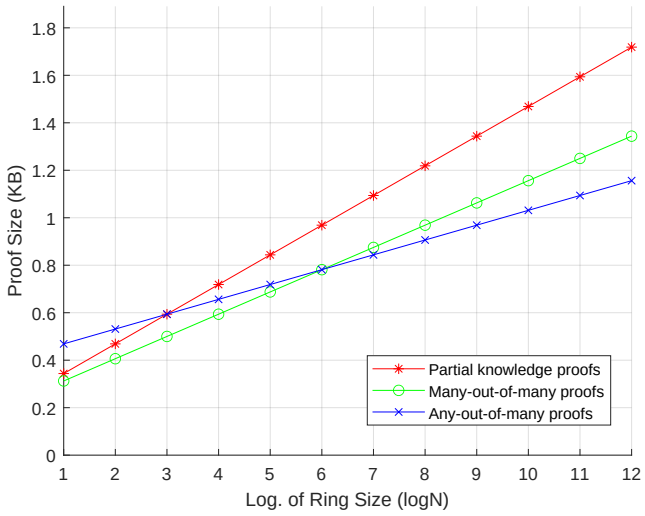

Fig. 1. Proof size comparison of any-out-of-many proofs, partial knowledge proofs and many-out-of-many proofs with different size of public set

of the public set increases, the proof size of our method becomes smaller than others. 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 Efficiency of RingCT

We mainly evaluate the performance between our protocol and RingCT 3.0 [**?**] in table 2. In the table above, we define a transaction, with N input accounts, k true spenders among them, m output accounts, and the valid range $[0, 2^o)$ for each input and output amount. The proof size in Table 2 consists of the size of signatures and range proofs.+ 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 lower size of $\mathcal{O}(log(N + m))$. We can spot this difference intuitively in the following figure, where the proof size of our RingCT protocol is far less than RingCT3.0.

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 [**?**]	$2 \cdot \lceil log_2(N^2 \cdot o + N \cdot m \cdot o) \rceil + 11$	12	$(N/k)^k$
Our RingCT	$2 \cdot \lceil log_2(N \cdot (o+1) + m \cdot o) \rceil + 8$	16	2^N

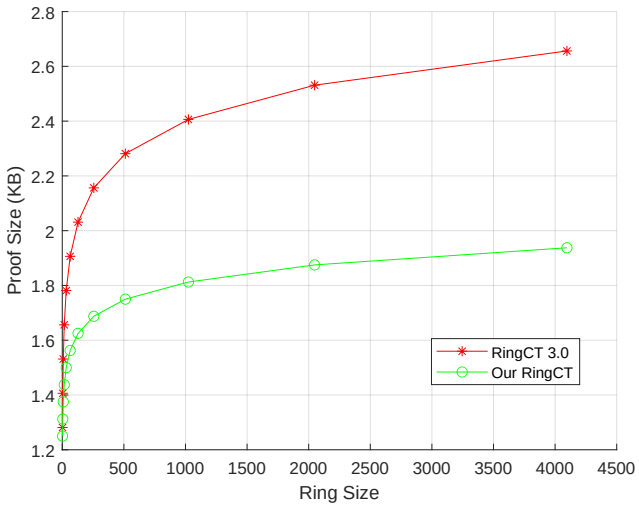

Fig. 2. Proof size comparison of our RingCT and RingCT 3.0 with different ring size

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. 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)
4. Bellare, M., Rogaway, P.: Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. ACM (1995)
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. 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)
24. van Saberhagen, N.: Cryptonote v2.0. `https://cryptonote.org/whitepaper.pdf`, 2013
25. 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)
26. 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)
27. 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)
28. 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

Here we give the technical details and security proofs of our any-out-of-many proofs.

Compression Process for Protocol 1

As described in Section 4.1, the any-out-of-many proof can be verified by checking the commitment C that

$$C = v^{\hat{t}} \cdot u^{\tau_c} \cdot \boldsymbol{P}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu} \cdot (g \cdot \mathbf{1}^n)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \tag{34}$$

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

$$\begin{aligned}
\boldsymbol{\zeta'} &= x \cdot \boldsymbol{\zeta}_L + \boldsymbol{\zeta}_R \\
\boldsymbol{\eta'} &= \boldsymbol{\eta}_L + x \cdot \boldsymbol{\eta}_R \\
\boldsymbol{f}'_s &= x \cdot \boldsymbol{f}_{s,L} + \boldsymbol{f}_{s,R} \\
\boldsymbol{P'} &= \boldsymbol{P}_L \circ \boldsymbol{P}_R^x \\
\boldsymbol{h''} &= \boldsymbol{h'}_L^x \circ \boldsymbol{h'}_R
\end{aligned}$$

Further we can compute

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

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

$$\begin{aligned}
(g \cdot \mathbf{1}^{n'} \cdot g^x \cdot \mathbf{1}^{n'})^{\boldsymbol{f}'_s} &= (g \cdot \mathbf{1}^{n'})^{\boldsymbol{f}_{s,R}} \cdot (g \cdot \mathbf{1}^n)^{\boldsymbol{f}_s} (g \cdot \mathbf{1}^{n'})^{x^2 \cdot \boldsymbol{f}_{s,L}} \\
&= L_3 \cdot (g \cdot \mathbf{1}^n)^{\boldsymbol{f}_s} \cdot {R_3}^{x^2}
\end{aligned}$$

$$\begin{aligned}
\langle \boldsymbol{\zeta'}, \boldsymbol{\eta'} \rangle &= \langle \boldsymbol{\zeta}_R, \boldsymbol{\eta}_L \rangle + \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle + x^2 \cdot \langle \boldsymbol{\zeta}_L, \boldsymbol{\eta}_R \rangle \\
&= L_4 + \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle + x^2 \cdot R_4
\end{aligned}$$

Therefore, we can compute the new C' as:

$$\begin{aligned}
C' &= v^{\hat{t}'} \cdot u^{\tau_c} \cdot \boldsymbol{P'}^{\boldsymbol{\zeta'}} \cdot (\boldsymbol{h''})^{\boldsymbol{\eta'}} \cdot u^{\mu} \cdot (g \cdot \mathbf{1}^{n'} \cdot g^x \cdot \mathbf{1}^{n'})^{\boldsymbol{f}'_s} \cdot u^{f_\alpha} \\
&= (L_1 \cdot L_2 \cdot L_3 \cdot L_4) \cdot C \cdot (R_1 \cdot R_2 \cdot R_3 \cdot R_4)^{x^2} \\
&= L \cdot C \cdot R^{x^2}
\end{aligned}$$

As a result, the prover compresses vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}, \boldsymbol{f}_s$ 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:

Compression Process:

$$
\begin{aligned}
&if\ n = 1: \\
&\mathcal{P} \quad \rightarrow \mathcal{V}:\ \tau_x, \mu, \hat{t}, f, l, r \\
&\mathcal{V} \quad checks\ if: \\
&\qquad C = v^{\hat{t}} \cdot u^{\tau_c} \cdot P^{\zeta} \cdot (h')^{\eta} \cdot u^{\mu} \cdot (g \cdot 1^n)^{f_s} \cdot u^{f_\alpha} \\
&\qquad \hat{t} = \langle \zeta, \eta \rangle \\
&else\ \ (n > 1): \\
&\mathcal{P} \quad Computes: \\
&\qquad n' = n/2 \\
&\qquad L = \boldsymbol{P}_L{}^{\boldsymbol{\zeta}_R} \cdot \boldsymbol{h}'_R{}^{\boldsymbol{\eta}_L} \cdot (g \cdot \mathbf{1}^{n'})^{\boldsymbol{f}_{s,R}} \cdot v^{\langle \boldsymbol{\zeta}_R, \boldsymbol{\eta}_L \rangle} \\
&\qquad R = \boldsymbol{P}_R{}^{\boldsymbol{\zeta}_L} \cdot \boldsymbol{h}'_L{}^{\boldsymbol{\eta}_R} \cdot (g \cdot \mathbf{1}^{n'})^{\boldsymbol{f}_{s,L}} \cdot v^{\langle \boldsymbol{\zeta}_L, \boldsymbol{\eta}_R \rangle} \\
&\mathcal{P} \quad \rightarrow \mathcal{V}:\ L, R \\
&\mathcal{V} \quad Computes: \\
&\qquad x \leftarrow \mathbb{Z}_q \\
&\mathcal{V} \quad \rightarrow \mathcal{P}:\ x \\
&\mathcal{P} \quad and\ \mathcal{V}\ Computes: \\
&\qquad \boldsymbol{P'} = \boldsymbol{P}_L \circ \boldsymbol{P}_R^x \\
&\qquad \boldsymbol{h''} = \boldsymbol{h'}_L^x \circ \boldsymbol{h'}_R \\
&\qquad C' = L \cdot C^x \cdot R^{x^2} \\
&\mathcal{P} \quad Computes: \\
&\qquad \boldsymbol{\zeta'} = x \cdot \boldsymbol{\zeta}_L + \boldsymbol{\zeta}_R \\
&\qquad \boldsymbol{\eta'} = \boldsymbol{\eta}_L + x \cdot \boldsymbol{\eta}_R \\
&\qquad \boldsymbol{f}'_s = x \cdot \boldsymbol{f}_{s,L} + \boldsymbol{f}_{s,R}
\end{aligned}
$$

Proof of Theorem 4

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 $\mathcal{A}$

$$\Pr\left[\mathcal{V}(ck, c, m, x, z) = 1 \middle| \begin{array}{l} ck \leftarrow \text{Setup}(1^\lambda); (c, r) \leftarrow \mathcal{A}(ck); \\ m \leftarrow \mathcal{P}(ck, c, r); x \leftarrow \mathbb{Z}_q^*; z \leftarrow \mathcal{P}(x) \end{array}\right] = 1 \tag{35}$$

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.

$$\Pr\left[(ck, c, r) \in R \middle| \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); \\ (c, m, x_1, z_1, ..., x_n, z_n) \leftarrow \mathcal{A}(ck); \\ \mathcal{V}(ck, c, m, x_i, z_i) = 1, \forall i \in [1, n]; \\ r \leftarrow \mathcal{E}(ck, c, m, x_1, z_1, ..., x_n, z_n) \end{array}\right] = 1 - \mu(\lambda) \tag{36}$$

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 $\mathcal{A}$

$$\left|\begin{array}{l} \Pr\left[\mathcal{A}(m, z) = 1 \middle| \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); (c, r, x) \leftarrow \mathcal{A}(ck); \\ a \leftarrow \mathcal{P}(ck, c, r); z \leftarrow \mathcal{P}(x); \end{array}\right] - \\ \Pr\left[\mathcal{A}(m, z) = 1 \middle| \begin{array}{l} ck \leftarrow \text{Setup}\left(1^\lambda\right); (c, r, x) \leftarrow \mathcal{A}(ck); \\ (m, z) \leftarrow \mathcal{S}(ck, c, r); \end{array}\right] \end{array}\right| \leqslant \mu(\lambda) \tag{37}$$

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 $A = u^{\alpha'} \cdot \boldsymbol{P}^{\boldsymbol{b}'_0} = (g \cdot \boldsymbol{R}^n)^{\boldsymbol{x}'_{\boldsymbol{sk}}} \cdot u^{\alpha'}$.

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 $\boldsymbol{x}'_{\boldsymbol{sk}} = \boldsymbol{sk'}$ if the Pederson commitment is perfectly binding. Thus, the Protocol 1 is $(n+1)$-special sound.

Appendix B

In this appendix, a full-version Bounded any-out-of-many proofs is presented. Based on the relations of *bounded any-out-of-many* proofs in definition 8, we can get the inner-product form with polynomials $\zeta(c), \eta(c)$ as

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

where y, z, c, d, e are challenges.

Now we can compress the vectors $\boldsymbol{\zeta}, \boldsymbol{\eta}$ in the same way as Bulletproofs. Note that vector $\boldsymbol{f}_s$ can also be compressed, since the equation (3) is independent of the inner product relation, we can multiply their commitments together into one as follows:

$$C = v^{\hat{t}} \cdot u^{\tau_c} \cdot \boldsymbol{Y}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu} \cdot (g \cdot \mathbf{1}^N)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \tag{38}$$

And we give the compression form of the vectors in one recursion using to definitions in Section 2.4.

$$\boldsymbol{\zeta'} = c \cdot \boldsymbol{\zeta}_0 + \boldsymbol{\zeta}_1 \tag{39}$$

$$\boldsymbol{\eta'} = \boldsymbol{\eta}_0 + c \cdot \boldsymbol{\eta}_1 \tag{40}$$

$$\boldsymbol{f}'_s = \boldsymbol{f}_{s,0} + c \cdot \boldsymbol{f}_{s,1} \tag{41}$$

And the computing result indicates that a batch method can be applied. Thus, for each recursion of compression, we can reduce the length of vectors above by half with only two extra group elements to be transferred.

Theorem 4. *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.*

The proofs of this theorem is similar to the any-out-of-many proofs, thus we do not give a further description here. We present the protocol based on these relations as follows:

Protocol 2 Bounded Any-out-of-Many Proof

$\mathcal{P}(g, \boldsymbol{h}, u, v, \boldsymbol{P}, ((\boldsymbol{a} \parallel \boldsymbol{b}); \boldsymbol{s}))$		$\mathcal{V}(g, \boldsymbol{h}, u, v, \boldsymbol{P})$
$\alpha_1, \alpha_2, \beta, \gamma \leftarrow \mathbb{Z}_q$		
$\boldsymbol{r}_s, \boldsymbol{s}_0, \boldsymbol{s}_1 \leftarrow \mathbb{Z}_q^{N+log(N)}$		
$\boldsymbol{a}_0 = \boldsymbol{a}, \boldsymbol{a}_1 = \boldsymbol{a}_0 - \mathbf{1}^{log(N)}$		
$\boldsymbol{b}_0 = \boldsymbol{b}, \boldsymbol{b}_1 = \boldsymbol{b}_0 - \mathbf{1}^{N}$		
$A_1 = \boldsymbol{P}_L^{\boldsymbol{a}_0} \cdot u^{\alpha_1}$		
$A_2 = \boldsymbol{P}_R^{\boldsymbol{b}_0} \cdot u^{\alpha_2}$		
$B = \boldsymbol{h}^{(\boldsymbol{a}_1 \| \boldsymbol{b}_1)} \cdot u^{\beta}$		
$C = \boldsymbol{P}^{\boldsymbol{s}_0} \cdot \boldsymbol{h}^{\boldsymbol{s}_1} \cdot u^{\gamma}$		
$D = (g \cdot \mathbf{1}^{N+log(N)})^{\boldsymbol{r}_s} \cdot u^{r_\alpha}$		
	$\xrightarrow{A_1, A_2, B, C, D}$	
		$y, z \leftarrow \mathbb{Z}_q$
	$\xleftarrow{y, z}$	
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_q$		
$T_1 = v^{t_1} \cdot u_1^{\tau}$		
$T_2 = v^{t_2} \cdot u_2^{\tau}$		
	$\xrightarrow{T_1, T_2}$	
		$c, d, e \leftarrow \mathbb{Z}_q$
	$\xleftarrow{c, d, e}$	
$\boldsymbol{\zeta} = \zeta(c)$		
$\boldsymbol{\eta} = \eta(c)$		
$\hat{t} = \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$		
$\tau_c = \tau_2 \cdot c^2 + \tau_1 \cdot c$		
$\mu = \alpha_1 + \alpha_2 \cdot e + \beta \cdot d + \gamma \cdot c$		
$\boldsymbol{f}_s = \boldsymbol{s} + \boldsymbol{r}_s \cdot c$		
$f_\alpha = \alpha_1 + r_\alpha \cdot c$		
	$\xrightarrow{\boldsymbol{\zeta}, \boldsymbol{\eta}, \hat{t}, \tau_c, \mu, \boldsymbol{f}_s, f_\alpha}$	
		$v^{\hat{t}} \cdot u^{\tau_c} \stackrel{?}{=} v^{\delta(y,z)} \cdot T_1^c \cdot T_2^{c^2}$
		$\boldsymbol{P}^{\boldsymbol{\zeta}} \cdot (\boldsymbol{h'})^{\boldsymbol{\eta}} \cdot u^{\mu} \stackrel{?}{=} A_1 \cdot A_2^e \cdot B^d \cdot C^c \cdot$ $\boldsymbol{P}^{-z \cdot \mathbf{1}^{2N}} \cdot \boldsymbol{h'}^{z \cdot \mathbf{1}^{2N} - z^2 \cdot (\mathbf{2}^N \| \mathbf{0}^N) + z^3 \cdot (\mathbf{0}^N \| \mathbf{2}^N)}$
		$(g \cdot \mathbf{1}^n)^{\boldsymbol{f}_s} \cdot u^{f_\alpha} \stackrel{?}{=} A_1^c \cdot D$
		$\hat{t} \stackrel{?}{=} \langle \boldsymbol{\zeta}, \boldsymbol{\eta} \rangle$

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.