Lattice-based Zero-knowledge Proofs for Blockchain Confidential Transactions

Shang GAO[1], Tianyu ZHENG[1], Yu GUO[2], and Zhe PENG[1] Bin XIAO[1]

[1] The Hong Kong Polytechnic University, Hong Kong, China
[2] SECBIT Labs, Suzhou, China
{shanggao,jeffrey-zhe.peng,csbxiao}@polyu.edu.hk
tian-yu.zheng@connect.polyu.hk
yu.guo@secbit.io

Abstract. We propose new zero-knowledge proofs for efficient and postquantum ring confidential transaction (RingCT) protocols based on lattice assumptions in Blockchain systems. First, we introduce an inner-product based linear equation satisfiability approach for balance proofs with a wide range (e.g., 64-bit precision). Unlike existing balance proofs (MatRiCT and MatRiCT+) that require additional proofs for some "corrector values", our approach avoids the corrector values for better efficiency. Furthermore, we design a ring signature scheme to efficiently hide a user's identity in large anonymity sets. Different from existing approaches that adopt a one-out-of-many proof (MatRiCT and MatRiCT+), we show that a linear sum proof suffices in ring signatures, which could avoid the costly binary proof part. We further use the idea of "unbalanced" relations to build a logarithmic-size ring signature scheme. Finally, we show how to adopt these techniques in RingCT protocols and implement a prototype to compare the performance with existing approaches. The results show our solutions can reduce up to 50% and 20% proof size, 30% and 20% proving time, 20% and 20% verification time of MatRiCT and MatRiCT+, respectively. We also believe our techniques are of independent interest for other applications and are applicable in a generic setting.

Keywords: Lattice-based cryptography, zero-knowledge proof, balance proof, ring signature, RingCT, blockchain

1 Introduction

Cryptocurrencies adopt the blockchain technique where each participant maintains a ledger of all transactions to avoid any tampering attempts from minority attackers. In private/anonymous cryptocurrencies, the amount[3] stored in each

[3] In this paper, the "amount" refers to "account balance". We avoid using balance here as it conflicts with balance proofs.

account and the user's identity need to be hidden from the outside world. Meanwhile, it also requires public verification to ensure each transaction is valid. Existing solutions such as Monero [24b] and Zcash [SCG+14] adopt zero-knowledge proofs (ZKPs) to prove useful statements without leaking any private information. For instance, in Monero, a ring confidential transaction (RingCT) protocol is used with a *range proof* to show all amounts are non-negative and the difference between outputs and inputs is zero (balance property), and a *ring signature*-like approach to hide the identity of a spender with one-out-of-many proofs [Noe15] (recent research also demonstrates with more efficient partial knowledge proofs [ACF21, ZGSX23]). However, as the security of these implementations is mainly based on discrete logarithm assumptions, they are at risk of potential attacks from quantum computers.

This deficiency has impelled the development of "post-quantum" solutions. Without exception, blockchain communities also consider using quantum-secure techniques to fill the gap. Among all post-quantum approaches, lattice-based cryptography is one of the most promising candidates based on computational lattice problems. Unfortunately, the costs of lattice-based solutions increase significantly in comparison with those in discrete logarithm settings. Taking the range proof proposed by Esgin et al. [ESLL19] as an example, a single proof costs more than 90KB while the Bulletproofs protocol [BBB+18] costs less than 1KB. Even worse, as the amounts in a RingCT protocol need to be committed separately, the efficient aggregation approach in [ESLL19] cannot be adopted. MatRiCT [EZS+19] is the first practical lattice-based RingCT protocol to optimize the proof size in a blockchain environment and is currently applied in Hcash [24a]. By using a novel balance proof with hashed-message commitments (HMC) to show a transaction is valid, MatRiCT reduces the size of commitments and allows proofs on a wide range. Furthermore, it adopts techniques such as batched commitments and rejection sampling for secrets with a fixed Hamming weight in one-out-of-many proofs to improve the efficiency of the ring signature. MatRiCT+ [ESZ22] further improves the performance of MatRiCT by optimizing the underlying cyclotomic rings. However, both MatRiCT and MatRiCT+ require some "corrector values" in balance proofs. Proving corrector values that are correct imposes a high cost.

1.1 Our contribution

The main goal of this paper is to propose efficient, scalable, and practical ZKPs for existing post-quantum[4] anonymous cryptocurrencies such as Hcash [24a]. We focus on some key problems in lattice-based RingCT protocols and significantly reduce the proof size and proving/verification time with our new ZKP techniques. Besides, our approaches optimize the high-level ZKP relations, which are compilable with MatRiCT [EZS+19] and MatRiCT+ [ESZ22].

[4] The post-quantum security in this paper relies on the hardness of "post-quantum" lattice assumptions as with [ESLL19, EZS+19, ESZ22].

To achieve the high efficiency of our approach, we first extend the amortization technique in [ACF21] to deal with non-homomorphic functions, and propose a partially amortized proof for binary relations (Section 3). Furthermore, we design a new inner-product based linear equation satisfiability protocol (Section 4) which implies balance relations in RingCT. Additionally, we introduce a new unbalanced linear sum proof (Section 5) which proves a weaker but still secure relation to replace the one-out-of-many relation in ring signatures. Finally, to build a secure RingCT protocol, we build a linkable version of our ring signatures (signatures of the same signer can be linked by serial number in the public key) and check the count of serial numbers to avoid additional issues (Section 6).

1.2 Related work

In anonymous cryptocurrencies, RingCT protocols [Noe15, EZS+19, SALY17] adopt *range proofs* to show transaction amounts are valid and *ring signature*-like approaches to hide a spender's identity. We describe existing work in these two directions.

Range proofs. To guarantee the amount of each account in a confidential transaction is valid, range proofs [BBB+18] are used in RingCT protocols. By encapsulating the amounts in homomorphic commitments, the prover proves that 1) all the inputs and outputs are non-negative and 2) the sum of inputs equals outputs. The proofs can be succinct and efficient with a trusted setup [Gro16, GWC19], but will undermine the decentralized property of blockchain systems at the same time where no particular trusted authority should be involved. Though the trusted setup can be replaced by a secure multi-party computation, the process is costly and may not be reusable when the application (i.e., circuit) is updated [Gro16]. Currently, the smallest proof without a trusted setup is the Bulletproofs protocol [BBB+18], which leverages the vector compression idea in [BCC+16]. However, these approaches fail to address quantum attacks as they are proposed based on discrete logarithm assumptions.

One of the most promising post-quantum cryptography candidates is lattice-based cryptography. Esgin et al. propose new range proofs in lattice settings based on the unbounded-message commitment (UMC) scheme and further adopt a new Chinese remainder theorem (CRT) packing technique for efficient batch processing [ESLL19]. Unfortunately, the size of a UMC commitment is linear to the message size which is not suitable for large values such as amounts of different accounts. Besides, the batch processing in [ESLL19] is only efficient when the amounts of all accounts are committed together in a single commitment, while the amounts are usually committed separately in a RingCT protocol. The first *practical* lattice-based RingCT approach is MatRiCT [EZS+19] (applied in Hcash [24a]). Instead of using UMC to commit to an amount directly, MatRiCT commits to the bits of an amount with HMC [ESS+19] and further adopts a balance proof with some "corrector values" to show the sums of inputs and outputs are equal. MatRiCT+ [ESZ22] further reduces the proof size and running time of MatRiCT by optimizing the underlying cyclotomic rings. Here we focus

on MatRiCT since our improvements are based on the techniques proposed in MatRiCT, which are quite independent from the improvements in MatRiCT+. Though the efficiency has been improved compared with [ESLL19], a subtle issue prevents the use of MatRiCT and MatRiCT+ in general cases: the corrector values require additional range proofs when dealing with multiple input and output accounts.

Ring signatures. To hide the identity of a signer, ring signatures (one-out-of-many proofs) allow one to prove the knowledge of a secret key corresponding to an element in a set of public keys. The idea of the ring signature has been proposed by Rivest, Shamir, and Tauman [RST01]. In discrete logarithm settings, logarithmic-size ring signatures [BCC$^+$15, GK15, ZGSX23] have been used in different applications. Most of current anonymous cryptocurrencies are implemented based on discrete logarithm assumptions which is not post-quantum secure.

On the side of lattice settings, linear-size ring signatures have been proposed [TSS$^+$18, LAZ19], but these approaches are inefficient for large anonymous groups. Libert et al. [LLNW16] design a Merkle tree based accumulator and build a ZKP system for this accumulator. With these tools, logarithmic-size ring and group signatures are proposed. Furthermore, a linkable version of [LLNW16] (signatures created by the same signer can be linked) is introduced in [YAL$^+$17]. Though the signature size of [LLNW16, YAL$^+$17] is logarithmic, the zero-knowledge arguments applied in the accumulator require multiple protocol iterations (multi-shot proofs) to get a negligible soundness error. Esgin et al. [ESS$^+$19] introduce new tools for ZKPs to extend the discrete logarithm proof techniques in [GK15] to lattice settings. Logarithmic-size ring signatures can be easily achieved with these new techniques. A further improvement in [ESLL19] makes the underlying ZKPs achieve a negligible soundness error at a single protocol iteration and reduces the signature size accordingly. Following the blueprint of [ESLL19], MatRiCT [EZS$^+$19] batches commitments in binary proofs and improves the rejection sampling to build a more efficient ring signature scheme. Besides, MatRiCT uses two sets of compatible parameters for the ring signature to reduce the size. BLOOM [LN22] further reduces the proof size in large rings setting by utilizing ABDLOP commitments and Bimodal Gaussians.

2 Preliminaries

We describe some notations and background knowledge used in this paper.

2.1 Notations

We use $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ to denote the ring of integers modulo q represented by the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$. The rings are defined by $R = \mathbb{Z}[X]/(X^d + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^d + 1)$ where $d > 1$ is a power of 2. Bold-face lower-case letters such as $\boldsymbol{a}$ and bold-face capital letters such as $\boldsymbol{A}$ are used to denote column vectors and matrices respectively. Commitments are denoted by capital letters such as

C even though they may be vectors (except R for the ring and M, S, N for the size of different inputs). We use $(\boldsymbol{a}, \boldsymbol{b})$ to denote appending vector $\boldsymbol{a}$ to $\boldsymbol{b}$. For a vector $\boldsymbol{a} = (a_0, \cdots, a_{k-1})$, the norms are defined as $\|\boldsymbol{a}\| = \sqrt{\sum_{i=0}^{k-1} a_i^2}$, $\|\boldsymbol{a}\|_1 = \sum_{i=0}^{k-1} |a_i|$, and $\|\boldsymbol{a}\|_\infty = \max_i |a_i|$. The norms of a polynomial are defined in a similar way as a vector. Suppose $x \in \mathbb{Z}_q$, we denote $\boldsymbol{x}^k = (1, x, x^2, \cdots, x^{k-1})$. Furthermore, the inner-product of two k-dimensional vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=0}^{k-1} a_i b_i$ and the Hadamard product is denoted as $\boldsymbol{a} \circ \boldsymbol{b} = (a_0 \cdot b_0, \cdots, a_{k-1} \cdot b_{k-1})$. The Kronecker's delta is denoted as $\delta_{j,i}$ such that $\delta_{j,i} = 1$ when $j = i$ and otherwise $\delta_{j,i} = 0$. $\mathrm{HW}(x)$ denotes the Hamming weight of the coefficient vector of $x \in R$. Given a distribution/set $\mathbb{S}$, $a \leftarrow\!\!\$\ \mathbb{S}$ denotes sampling a from $\mathbb{S}$, or uniformly sampling from a set $\mathbb{S}$. Define $\mathbb{S}_{\mathcal{B}}$ as the subset of polynomials in $\mathcal{R}_q$ with infinity norm at most $\mathcal{B} \in \mathbb{Z}^*$.

The challenge space in a Σ-protocol is defined as follows:

$$\mathcal{C} = \{x \in R : \deg(x) = d - 1 \wedge \mathrm{HW}(x) = w \wedge \|x\|_\infty = p\}. \tag{1}$$

Clearly, we can observe $\|x\|_1 \leq pw$ and $|\mathcal{C}| = \binom{d}{w} \cdot (2p)^w$. We denote the set of challenge differences excluding zero as $\Delta\mathcal{C}$.

2.2 Lattice problems and HMC

M-SIS and M-LWE. We define the two well-known lattice problems [LS15], *module short integer solution* (M-SIS) and *module learning with errors* (M-LWE), which our schemes' security relies on.

Definition 1. *M-SIS*(n, m, q, γ). *Given* $\boldsymbol{A} \leftarrow R_q^{n \times m}$, *the goal of the problem is to find* $\boldsymbol{z} \in R_q^m$ *such that* $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \mod q$ *and* $0 < \|\boldsymbol{z}\| \leqslant \gamma$.

Definition 2. *M-LWE*$(n, m, q, \mathcal{B})$. *Let* $\boldsymbol{s} \leftarrow \mathbb{S}_{\mathcal{B}}^n$ *be a secret key. Define* $\mathsf{LWE}(q, \boldsymbol{s})$ *as the distribution obtained by sampling* $\boldsymbol{a} \leftarrow R_q^n$, $e \leftarrow \mathbb{S}_{\mathcal{B}}$ *and outputting* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e)$. *The goal of the problem is to distinguish between* m *given samples from either* $\mathsf{LWE}(q, \boldsymbol{s})$ *or* R_q^{n+1}.

Hashed-Message Commitment. Let $n, m, \mathcal{B}, q$ be positive integers with $m > n$. Suppose a prover commits to v-dimensional vectors over R_q for $v \geqslant 1$. The instantiation of the hashed-message commitment (HMC) scheme [EZS+19, BDL+18] is as follows:

- $\mathsf{CKeygen}(1^\lambda)$: Sample $\boldsymbol{G}_r \leftarrow R_q^{n \times m}$ and $\boldsymbol{G}_m \leftarrow R_q^{n \times v}$. Output $ck = \boldsymbol{G} = (\boldsymbol{G}_r, \boldsymbol{G}_m) \in R_q^{n \times (m+v)}$.
- $\mathsf{Commit}_{ck}(\boldsymbol{m})$: Sample $\boldsymbol{r} \leftarrow \{-\mathcal{B}, \cdots, \mathcal{B}\}^{md}$. Output $\boldsymbol{r}$ and $\mathrm{Com}_{ck}(\boldsymbol{m}; \boldsymbol{r}) = \boldsymbol{G} \cdot (\boldsymbol{r}, \boldsymbol{m}) = \boldsymbol{G}_r \cdot \boldsymbol{r} + \boldsymbol{G}_m \cdot \boldsymbol{m}$.
- $\mathsf{COpen}_{ck}(C, (y, \boldsymbol{m}', \boldsymbol{r}'))$: If $\|(\boldsymbol{m}', \boldsymbol{r}')\| \leqslant \gamma$ and $yC = \mathrm{Com}_{ck}(\boldsymbol{m}', \boldsymbol{r}')$ return 1, otherwise return 0.

The computationally hiding and computationally strong binding prosperities are defined as follows.

Definition 3. *For all PPT adversaries $\mathcal{A}$, computational hiding and computational strong γ-binding are defined respectively as*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{CKeygen}(1^\lambda); \\ (\boldsymbol{m}_0, \boldsymbol{m}_1) \leftarrow \mathcal{A}^{\mathsf{CKeygen}(ck)}; \\ b \leftarrow \{0,1\}; C \leftarrow \mathsf{Commit}_{ck}(\boldsymbol{m}_b) \end{array} : \mathcal{A}(C) = b\right] \approx \frac{1}{2}, \text{ and}$$

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{CKeygen}(1^\lambda); \\ (C, \boldsymbol{t}_0, \boldsymbol{t}_1) \leftarrow \mathcal{A}(ck) \end{array} : \begin{array}{l} (\boldsymbol{m}_0, \boldsymbol{r}_0) \neq (\boldsymbol{m}_1, \boldsymbol{r}_1) \wedge \\ \mathsf{COpen}_{ck}(C, \boldsymbol{t}_0) = 1 \wedge \\ \mathsf{COpen}_{ck}(C, \boldsymbol{t}_1) = 1 \end{array}\right] \approx 0,$$

where $\boldsymbol{t}_i = (y_i, \boldsymbol{m}_i, \boldsymbol{r}_i)$ for $i = \{0,1\}$ and the norm bound parameter in COpen is γ.

Lemma 1. ***(Lemma 2.3 in [EZS+19])*** *For a (large) set of appropriately chosen parameters, if M-LWE$(m-n, m, q, \mathcal{B})$ problem is hard then the HMC is computationally hiding. If M-SIS$(n, m+v, q, 2\gamma)$ is hard, then the HMC is computationally strong γ-binding to the same relaxation factor y.*

Given a $\boldsymbol{r} \in R_q^m$, we write it into $\boldsymbol{r} = (\boldsymbol{r}', \boldsymbol{r}'')$ where $\boldsymbol{r}' \in R_q^n$ and $\boldsymbol{r}'' \in R_q^{m-n}$. Similarly, write $\boldsymbol{G}_r = (\boldsymbol{G}'_r, \boldsymbol{G}''_r)$ where $\boldsymbol{G}'_r \in R_q^{n \times n}$ and $\boldsymbol{G}''_r \in R_q^{n \times (m-n)}$. We have $\boldsymbol{G}_r \cdot \boldsymbol{r} = \boldsymbol{G}'_r \cdot \boldsymbol{r}' + \boldsymbol{G}''_r \cdot \boldsymbol{r}''$. When $\boldsymbol{G}'_r$ is non-singular (non-singular over all the fields of R_q splits into), $\boldsymbol{G}'_r \cdot \boldsymbol{r}'$ is indistinguishable from a random element $\boldsymbol{a} \in R_q^n$ (since $\boldsymbol{r}' = (\boldsymbol{G}'_r)^{-1} \cdot \boldsymbol{a}$). It indicates n M-LWE instances from M-LWE$(m-n, m, q, \mathcal{B})$. As the probability of $\boldsymbol{G}_r$ being singular over R_q^n is negligible in our settings (see [EZS+19, ESZ22]), the probability of $\boldsymbol{G}'_r$ being singular is also negligible (by swapping columns). Thus, the HMC is computational hiding.

When given an HMC collision $(\boldsymbol{m}_0, \boldsymbol{r}_0) \neq (\boldsymbol{m}_1, \boldsymbol{r}_1)$, we have $\boldsymbol{G} \cdot (\boldsymbol{m}_0, \boldsymbol{r}_0) = \boldsymbol{G} \cdot (\boldsymbol{m}_1, \boldsymbol{r}_1)$, which implies $\boldsymbol{G} \cdot (\boldsymbol{m}_1 - \boldsymbol{m}_0, \boldsymbol{r}_1 - \boldsymbol{r}_0) = 0$. This gives a solution to M-SIS$(n, m+v, q, 2\gamma)$ directly since $\|(\boldsymbol{m}_1 - \boldsymbol{m}_0, \boldsymbol{r}_1 - \boldsymbol{r}_0)\| \leq 2\gamma$. Thus, HMC is computational strong γ-binding. Based on the result in [MR09], the parameters should satisfy the following relation to ensure HMC is γ-binding:

$$\min\{q, 2^{2\sqrt{nd \log q \log \delta}}\} > 2\gamma, \tag{2}$$

where δ is a root Hermite factor to indicate the security level.

2.3 Lattice-based Σ-protocol

A Σ-protocol is a public coin interactive proof system to allow a prover to convince a verifier that a statement is true. It has three properties: completeness, knowledge soundness, and special honest-verifier zero-knowledge (SHVZK) [ESLL19, EZS+19]. Here we start from a simple example in discrete logarithm settings to show the knowledge of a secret $\boldsymbol{m}$ such that $C = \mathrm{Com}_{ck}(\boldsymbol{m})$ for a public C (randomness is omitted for simplicity). Specifically, the prover samples a masking vector $\boldsymbol{d}$ and sends $D = \mathrm{Com}_{ck}(\boldsymbol{d})$. The verifier challenges with a random challenge x and the prover responds with $\boldsymbol{f} = x\boldsymbol{m} + \boldsymbol{d}$. Finally, the verifier checks $\mathrm{Com}_{ck}(\boldsymbol{f}) = xC + D$.

Relaxed proof. Recall HMC. The opening algorithm COpen does not simply check $C \stackrel{?}{=} \mathrm{Com}_{ck}(\boldsymbol{m}')$ in common lattice-based schemes [DPLS18, ZGX25], but with a *relaxation factor* $y \in R_q$ as in [ESLL19, BLNS20]. This is due to the straightforward soundness proofs under lattice assumptions do not work (the challenge differences may not be *invertible* and the extracted witness may not be *short* in the soundness proof). Consider a simple case with two accepting transcripts, $(D, x_1, \boldsymbol{f}_1)$ and $(D, x_2, \boldsymbol{f}_2)$. Though the extractor can get $(x_2 - x_1)\mathrm{Com}(\boldsymbol{m}') = \mathrm{Com}(\boldsymbol{f}_2 - \boldsymbol{f}_1)$, it cannot simply output $\boldsymbol{m}' = (x_2 - x_1)^{-1}(\boldsymbol{f}_2 - \boldsymbol{f}_1)$ since: 1) $(x_2 - x_1)$ may not be invertible; and 2) even $(x_2 - x_1)$ is invertible, $(x_2 - x_1)^{-1}$ may not be short and the extracted opening will not be a valid one. One solution is to use a **relaxed proof** by relaxing the verification relation to overcome the complications [ESLL19, BLNS20]: instead of extracting the opening of $\mathrm{Com}(\boldsymbol{m}')$, the extractor is allowed to extract the opening of $y\mathrm{Com}(\boldsymbol{m})$ ($y = x_2 - x_1$ in this example). Since $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are short, $(\boldsymbol{f}_2 - \boldsymbol{f}_1)$ is still short. Accordingly, the proving and relaxed opening relations are different in relaxed proofs.

Rejection sampling. In lattice settings, the verifier also needs to check the norm bound of $\boldsymbol{f}$ to ensure the hardness of M-SIS problem in Section 2.2. Therefore, $\boldsymbol{d}$ must be sampled from a distribution with a larger range to hide $x\boldsymbol{m}$ term (denoted as $\mathbb{D}_\gamma$, where γ is the norm bound in M-SIS), while the bound of the distribution should be manageable for the hardness of M-SIS. Besides, the prover cannot output $\boldsymbol{f}$ directly since the distribution of $\boldsymbol{f}$ leaks the information of $\boldsymbol{m}$ when $\boldsymbol{d}$ is not uniformly sampled from the field. For instance, consider one-bit message $m \in \{0, 1\}$, the distribution of f is the same as the distribution of d when $m = 0$, and will shift by x when $m = 1$. Anyone can infer m by observing the distribution of $\boldsymbol{f}$. Existing solutions adopt an additional *rejection sampling* to reject responses that are out-of-bounds. Generally speaking, only $\boldsymbol{f}$'s that can be "touched" by all possible values of $\boldsymbol{m}$ and follow an expected distribution are acceptable. As we directly use the results of rejection sampling in this paper, we only briefly summarize rejection sampling [Lyu12] in Algorithm 1 (restricting the distribution of $(x, \boldsymbol{f})$ being independent of $\boldsymbol{m}$), where $T = \|x\boldsymbol{m}\|$ and $\phi = \gamma/T$. Returning 1 means $\boldsymbol{f}$ passes the rejection sampling.

Algorithm 1 Rejection Sampling [Lyu12]

$\mathrm{Rej}(\boldsymbol{f}, \boldsymbol{m}, \phi, T)$

1: $\gamma = \phi T$; $\mu(\phi) = \exp(\frac{12}{\phi} + \frac{1}{2\phi^2})$; $u \leftarrow [0, 1)$
2: **if** $u > \frac{1}{\mu(\phi)} \cdot \exp(\frac{-2\langle \boldsymbol{f}, \boldsymbol{m} \rangle + \|\boldsymbol{m}\|^2}{2\gamma^2})$ **then**
3: Return $\perp$
4: **end if**
5: Return 1

Lemma 2. ***(Theorem 4.6 in [Lyu12])*** *Let h be a probability distribution over $V \in \mathbb{Z}^s$ where $s \geq 1$ and the norm of all elements is less than T. Let $\boldsymbol{m} \in h$ and $\phi > 0$. Considering an algorithm that samples $\boldsymbol{d} \leftarrow\!\!\$\ \mathbb{D}^s_{\phi T}$ and outputs $Rej(\boldsymbol{f}, \boldsymbol{m}, \phi, T)$ for $\boldsymbol{f} := \boldsymbol{m} + \boldsymbol{d}$. The probability that the algorithm outputs 1 is within 2^{-100} of $1/\mu(\phi)$ where $\mu(\phi) = e^{12/\phi + 1/(2\phi^2)}$. When the output is 1, the statistical distance between the distribution of $\boldsymbol{f}$ and $\mathbb{D}^s_{\phi T}$ is at most 2^{-100}.*

We summarize the lattice-based Σ-protocol below. The major differences with that in discrete logarithm settings is highlighted in blue.

1. $\mathcal{P}$: Sample $\boldsymbol{d} \leftarrow\!\!\$\ \mathbb{D}^m_{\phi T}$ and set $D = \mathrm{Com}_{ck}(\boldsymbol{d})$.
2. $\mathcal{P} \to \mathcal{V}$: D.
3. $\mathcal{V} \to \mathcal{P}$: $x \leftarrow\!\!\$\ \mathcal{C}$.
4. $\mathcal{P}$: Set $\boldsymbol{f} := x\boldsymbol{m} + \boldsymbol{d}$ and run $\mathrm{Rej}(\boldsymbol{f}, x\boldsymbol{m}, \phi, T)$.
5. $\mathcal{P} \to \mathcal{V}$: $\boldsymbol{f}$.
6. $\mathcal{V}$: Check $\mathrm{Com}_{ck}(\boldsymbol{f}) \stackrel{?}{=} xC + D$ and $\|\boldsymbol{f}\| \stackrel{?}{\leq} 2\phi T\sqrt{md}$.

2.4 Vandermonde matrix and one-shot proof [ESLL19]

A $(k+1)$-dimensional Vandermonde matrix $\boldsymbol{V}$ is defined as follows for some $x_0, \cdots, x_k \in R$:

$$\boldsymbol{V} = \begin{pmatrix} 1 & x_0 & \cdots & x_0^k \\ 1 & x_1 & \cdots & x_1^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & \cdots & x_k^k \end{pmatrix}. \tag{3}$$

Let $\mathrm{adj}(\boldsymbol{V})$ denotes adjugate matrix of $\boldsymbol{V}$ and $\det(\boldsymbol{V})$ denotes the determinant of $\boldsymbol{V}$. We have $\det(\boldsymbol{V}) = \prod_{0 \leq i < j \leq k}(x_j - x_i)$. Let $(\Gamma_0, \cdots, \Gamma_k)$ be the last row of $\mathrm{adj}(\boldsymbol{V})$. Then

$$\Gamma_i = (-1)^{i+k} \prod_{\substack{0 \leq s < j \leq k \\ s,j \neq i}} (x_j - x_s). \tag{4}$$

Lemma 3. ***(Lemma 4 in [ESLL19])*** *Let $\kappa = \frac{k(k+1)}{2}$, we have $\|\det(\boldsymbol{V})\|_\infty \leq (2p)^\kappa w^{\kappa-1}$ when using the challenge space in Equation (1).*

The one-shot proof is a technique proposed in [ESLL19] to efficiently prove non-linear polynomial relations. Consider a k-degree polynomial relation with commitments $C_0 = \mathrm{Com}_{ck}(\boldsymbol{m}_0; \boldsymbol{r}_0), \cdots, C_k = \mathrm{Com}_{ck}(\boldsymbol{m}_k; \boldsymbol{r}_k)$. The prover encodes the message as $(\boldsymbol{f}, \boldsymbol{z}) \leftarrow (\sum_{i=0}^k x^i \boldsymbol{m}_i, \sum_{i=0}^k x^i \boldsymbol{r}_i)$ with a challenge x. The verifier checks the norms of $\boldsymbol{f}, \boldsymbol{z}$ and $\sum_{i=0}^k x^i C_i \stackrel{?}{=} \mathrm{Com}_{ck}(\boldsymbol{f}; \boldsymbol{z})$. This protocol has $(k+1)$-special soundness as we can extract a witness in one shot with the following approach.

Considering $(k+1)$ accepted transactions with distinct challenges x_i's and responses $(\boldsymbol{f}_i, \boldsymbol{z}_i)$'s where $i \in [0,k]$ (C_i's are the same). We have the following relation

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^k \\ 1 & x_1 & \cdots & x_1^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & \cdots & x_k^k \end{pmatrix} \cdot \begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_k \end{pmatrix} = \begin{pmatrix} \mathrm{Com}_{ck}(\boldsymbol{f}_0; \boldsymbol{z}_k) \\ \mathrm{Com}_{ck}(\boldsymbol{f}_1; \boldsymbol{z}_k) \\ \vdots \\ \mathrm{Com}_{ck}(\boldsymbol{f}_k; \boldsymbol{z}_k) \end{pmatrix}. \tag{5}$$

Let the Vandermonde matrix in Equation (5) be $\boldsymbol{V}$. Considering the property $\mathrm{adj}(\boldsymbol{V}) \cdot \boldsymbol{V} = \det(\boldsymbol{V}) \cdot \boldsymbol{I}_{k+1}$, we multiply both sides of Equation (5) by $\mathrm{adj}(\boldsymbol{V})$. Based on Equation (4), its last row becomes

$$\det(\boldsymbol{V}) \cdot C_k = \sum_{i=0}^{k} \Gamma_i \mathrm{Com}_{ck}(\boldsymbol{f}_i; \boldsymbol{z}_i) := \mathrm{Com}_{ck}(\widehat{\boldsymbol{m}}_k; \widehat{\boldsymbol{r}}_k), \tag{6}$$

where $(\widehat{\boldsymbol{m}}_k, \widehat{\boldsymbol{r}}_k) = (\sum_{i=0}^{k} \Gamma_i \boldsymbol{f}_i, \sum_{i=0}^{k} \Gamma_i \boldsymbol{z}_i)$ is a relaxed opening to yC_k with a relaxation factor $y = \det(\boldsymbol{V})$.

2.5 Amortized relation [ACF21, GQZ+24]

The amortization technique is used to open *multiple* linear forms for essentially the price of *one* [ACF21, GQZ+24]. In [ACF21], Attema et al. describe amortized exponentiations in discrete logarithm settings. For simplicity, we regard the randomness as one dimension of the secret. Consider the following relation

$$\mathcal{R}_{\mathrm{AmorExp}} = \left\{ \begin{array}{c} (ck, (B_i, P_i)_{i=1}^{S}), ((\boldsymbol{b}_i)_{i=1}^{S}) : \\ (\mathrm{Com}_{ck}(\boldsymbol{b}_i) = B_i, g(\boldsymbol{b}_i) = P_i)_{i=1}^{S} \end{array} \right\}, \tag{7}$$

where $g(\cdot)$ is a homomorphic function, it is equivalent to prove $\mathrm{Com}_{ck}(\sum_{i=1}^{S} \zeta^i \boldsymbol{b}_i) = \sum_{i=1}^{S} \zeta^i B_i$ and $g(\sum_{i=1}^{S} \zeta^i \boldsymbol{b}_i) = \sum_{i=1}^{S} \zeta^i P_i$ with a challenge ζ. The prover can further use one vector $\boldsymbol{a}$ to mask the response $\boldsymbol{f} = \boldsymbol{a} + \sum_{i=1}^{k} \zeta^i \boldsymbol{b}_i$ as with a standard Σ-protocol.

3 Partial Amortization

3.1 Batched verification of binary proofs

The binary proof is a fundamental building block in RingCT protocols (as well as other applications) to prove the knowledge of a binary vector. When proving multiple binary vectors, this can be done efficiently by committing all vectors in one commitment. Unfortunately, in RingCT protocols such as MatRiCT [EZS+19] and MatRiCT+ [ESZ22], this batching technique cannot be applied since each account is committed separately. For example, to mint coins for S accounts $(\boldsymbol{b}_i, \boldsymbol{r}_{b,i}, B_i)_{i=0}^{S-1}$ such that $B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i; \boldsymbol{r}_{b,i})$ (instead

of $\mathrm{Com}_{ck}(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}; \boldsymbol{r}_b)$), the prover needs to run a binary proof to prove the knowledge of $\boldsymbol{b}_i$'s and $\boldsymbol{b}_i$'s being binary vectors. Specifically, the prover needs to use S vectors $(\boldsymbol{t}_i, \boldsymbol{r}_{t,i})_{i=0}^{S-1}$ to mask $\boldsymbol{b}_i$'s and $\boldsymbol{r}_{b,i}$'s as $\boldsymbol{g}_i = x\boldsymbol{b}_i + \boldsymbol{t}_i$ and $\boldsymbol{z}_{b,i} = x\boldsymbol{r}_{b,i} + \boldsymbol{r}_{t,i}$ based on a challenge x. Accordingly, the verifier needs to check $\mathrm{Com}_{ck}(\boldsymbol{g}_i; \boldsymbol{z}_{b,i}) = xB_i + G_i$ holds for all i's, where $G_i = \mathrm{Com}_{ck}(\boldsymbol{t}_i; \boldsymbol{r}_{t,i})$ (a further check to ensure $\boldsymbol{b}_i \circ (\boldsymbol{1} - \boldsymbol{b}_i) = 0$ is also needed). Since the verification is conducted separately for each i, all $\boldsymbol{z}_{b,i}$'s must be included in the proof, which increases the proof size when dealing with multiple accounts.

Our observation is that it is possible to batch the verification of $\mathrm{Com}_{ck}(\boldsymbol{g}_i; \boldsymbol{z}_{b,i}) = xB_i + G_i$ with the amortized technique in [ACF21, GQZ+24]. Unfortunately, since the binary constraint $\boldsymbol{b}_i \circ (\boldsymbol{1} - \boldsymbol{b}_i) = 0$ is *not* homomorphic, it cannot be regarded as the $g(\cdot)$ in Equation (7). This brings us to the idea of *partial amortization*: only using the batched verification for $\mathrm{Com}_{ck}(\boldsymbol{g}_i; \boldsymbol{z}_{b,i}) = xB_i + G_i$ (i.e., the knowledge of $\boldsymbol{b}_i$'s) and leaving the binary constraint part (i.e., $\boldsymbol{b}_i$'s are binary vectors) unchanged. Specifically, we replace $\mathrm{Com}_{ck}(\boldsymbol{b}_i; \boldsymbol{r}_{b,i}) = B_i$ with $\mathrm{Com}_{ck}(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{b}_i, \sum_{i=0}^{S-1} \zeta^i \boldsymbol{r}_{b,i}) = \sum_{i=0}^{S-1} \zeta^i B_i$ based on a challenge ζ. Since the remaining (non-homomorphic) binary constraint does not involve $\boldsymbol{r}_{b,i}$ (i.e., B_i), the prover can batch $\boldsymbol{r}_{b,i}$'s and only send one element $\boldsymbol{z}_b = \sum_{i=0}^{S-1} \zeta^i \boldsymbol{z}_{b,i}$.

Note that in the security (soundness) analysis, if we use the one-shot proof [ESLL19] directly to extract the relaxed openings of B_i's, we need to use the i-th row of a Vandermonde's adjugate matrix, which is much different from the form in Equation (4) when $i \neq S$. Additionally, the relaxation factor $\| \det(\boldsymbol{V}) \|_\infty$ increases significantly with S, which requires a larger parameter set to ensure the hardness of M-SIS and reduce the soundness error. Fortunately, as claimed in [EZS+19] and [ESZ22], the most common cases in cryptocurrencies are transactions with two outputs. Thus, we focus on the case of $S \leq 2$ in the soundness analysis of this paper. Besides, one may also reduce the growth with different challenges ζ_i's in a special challenge space to compute $\sum_{i=0}^{S-1} \zeta_i B_i$ and ensure the norm bound is relatively small as with [ESZ22].

3.2 Partially amortized binary proofs

Recall the amortized proof in Section 2.5. We generalize the commitment function $\mathrm{Com}_{ck}(\cdot)$ in Equation (7) to any homomorphic function $h(\cdot)$ and discuss the case when $g(\cdot)$ is *not* homomorphic. Suppose B_i and P_i's only share a part of common secrets, i.e., $h(\boldsymbol{b}_i, \boldsymbol{s}_i) = B_i$ and $g(\boldsymbol{b}_i, \boldsymbol{t}_i) = P_i$ for witness $(\boldsymbol{b}_i, \boldsymbol{s}_i, \boldsymbol{t}_i)_{i=0}^{S-1}$ (this is a common case in many applications when regarding $\boldsymbol{s}_i$'s and $\boldsymbol{t}_i$'s are different randomness, see our binary relation below as an example). Let $\boldsymbol{s} = \sum_{i=0}^{S-1} \zeta^i \boldsymbol{s}_i$ and $B = \sum_{i=0}^{S-1} \zeta^i B_i$. It is possible only to amortize the homomorphic part $h(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{b}_i, \boldsymbol{s}) = B$, and keep the $g(\cdot)$ part unchanged. Therefore, the prover only needs to send $(\boldsymbol{b}_i, \boldsymbol{t}_i)_{i=0}^{S-1}$ and $\boldsymbol{s}$ (instead of all $\boldsymbol{s}_i$'s) in the proof (we do not consider HVZK property here for simplicity).

To apply the above technique in lattice settings, besides checking $g(\boldsymbol{b}_i, \boldsymbol{t}_i) \stackrel{?}{=} P_i$ and $h(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{b}_i, \boldsymbol{s}) \stackrel{?}{=} B$, the verifier also needs additional norm checks to ensure $\|\boldsymbol{b}_i\|$'s and $\|\boldsymbol{t}_i\|$'s are smaller than the claimed bounds and $\|\boldsymbol{s}\|$ is short.

Protocol 1 Partially Amortized Binary Proof.

$\mathcal{P}_{bin}(ck, (B_i, \boldsymbol{b}_i, \boldsymbol{r}_{b,i})_{i=0}^{S-1}), \quad \mathcal{V}_{bin}(ck, (B_i)_{i=0}^{S-1})$

1: $\mathcal{V}_{bin} \Rightarrow \mathcal{P}_{bin}$: $\zeta \leftarrow \mathcal{C}$.
2: $\mathcal{P}_{bin}$: Set $T_1 = p\sqrt{2wk}$ and $T_2 = \mathcal{B}wp\sqrt{3md}$.
3: $\mathcal{P}_{bin}$: Sample masking values

$$\boldsymbol{r}_e \leftarrow \mathbb{S}_{\mathcal{B}}^m; \quad \boldsymbol{r}_f \leftarrow \mathbb{D}_{\phi_2 T_2}^m; \quad \boldsymbol{t}_i \leftarrow \mathbb{D}_{\phi_1 T_1}^k; \quad \boldsymbol{r}_{t,i} \leftarrow \mathbb{D}_{\phi_2 T_2}^m \ \ \forall i \in [0, S).$$

4: $\mathcal{P}_{bin}$: Compute the commitments

$$E = \mathrm{Com}_{ck}((\boldsymbol{t}_i \circ (\mathbf{1} - 2\boldsymbol{b}_i))_{i=0}^{S-1}; \boldsymbol{r}_e); \quad F = \mathrm{Com}_{ck}((-\boldsymbol{t}_i \circ \boldsymbol{t}_i)_{i=0}^{S-1}; \boldsymbol{r}_f);$$
$$G = \mathrm{Com}_{ck}(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{t}_i; \sum_{i=0}^{S-1} \zeta^i \boldsymbol{r}_{t,i}).$$

5: $\mathcal{P}_{bin} \Rightarrow \mathcal{V}_{bin}$: E, F, G.
6: $\mathcal{V}_{bin} \Rightarrow \mathcal{P}_{bin}$: $x \leftarrow \mathcal{C}$.
7: $\mathcal{P}_{bin}$: Compute $\boldsymbol{g}_i = x\boldsymbol{b}_i + \boldsymbol{t}_i$, $\forall i \in [0, S)$.
8: $\mathcal{P}_{bin}$: Run $\mathrm{Rej}((\boldsymbol{g}_i)_{i=0}^{S-1}, (x\boldsymbol{b}_i)_{i=0}^{S-1}, \phi_1, T_1)$.
9: $\mathcal{P}_{bin}$: Compute $\boldsymbol{z}_g = x\boldsymbol{r}_e + \boldsymbol{r}_f$, $\boldsymbol{z}_{b,i} = x\boldsymbol{r}_{b,i} + \boldsymbol{r}_{t,i}$, and $\boldsymbol{z}_b = \sum_{i=0}^{S-1} \zeta^i \boldsymbol{z}_{b,i}$.
10: $\mathcal{P}_{bin}$: Run $\mathrm{Rej}((\boldsymbol{z}_g, (\boldsymbol{z}_{b,i})_{i=0}^{S-1}), x(\boldsymbol{r}_e, (\boldsymbol{r}_{b,i})_{i=0}^{S-1}), \phi_2, T_2)$.
11: $\mathcal{P}_{bin} \Rightarrow \mathcal{V}_{bin}$: $(\boldsymbol{g}_i)_{i=0}^{S-1}, \boldsymbol{z}_g, \boldsymbol{z}_b$.
12: $\mathcal{V}_{bin}$: Conduct the following checks

$$\|g_{i,j}\| \overset{?}{\leq} 2\phi_1 T_1 \sqrt{d}, \ \forall i, j; \quad \|\boldsymbol{z}_g\| \overset{?}{\leq} 2\phi_2 T_2 \sqrt{md}; \quad \|\boldsymbol{z}_b\| \overset{?}{\leq} 2md\phi_2 T_2 (wp + 1);$$
$$xE + F \overset{?}{=} \mathrm{Com}_{ck}((\boldsymbol{g}_i \circ (x \cdot \mathbf{1} - \boldsymbol{g}_i))_{i=0}^{S-1}; \boldsymbol{z}_g); x \sum_{i=0}^{S-1} \zeta^i B_i + G \overset{?}{=} \mathrm{Com}_{ck}(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{g}_i; \boldsymbol{z}_b).$$

The correctness of the protocol follows directly. The soundness for $h(\cdot)$ part follows the soundness of the amortized exponentiations in [ACF21], which takes an $(S \times S)$-size Vandermonde matrix to extract $\boldsymbol{s}_i$ with $\sum_{i=0}^{S-1} \zeta_j^i \boldsymbol{s}_i$ on different challenges $(\zeta_j)_{j=0}^{S-1}$ (in lattice settings, it is a little bit different to extract a relaxed opening, see our binary proof for more details). Other parts $(\boldsymbol{b}_i, \boldsymbol{t}_i)_{i=0}^{S-1}$ can be derived directly from the prover's proof.

It is not hard to further add the zero-knowledge property to the above protocol. More specifically, we describe a partially amortized binary proof in Protocol 1 as a fundamental building block of our RingCT protocol, which shows B_i's are commitments to bits (for $S \leq 2$).

Definition 4. *The following defines the relations for multiple binary vectors, proving $\mathcal{R}_{bin}$ and relaxed opening $\mathcal{R}'_{bin}$:*

$$\mathcal{R}_{bin}(\mathcal{T}) = \left\{ \begin{array}{c} ((ck, (B_i)_{i=0}^{S-1}), (\boldsymbol{b}_i, \boldsymbol{r}_{b,i})_{i=0}^{S-1}) : \boldsymbol{b}_i \in \{0,1\}^k \\ \wedge \|\boldsymbol{r}_{b,i}\| \leq \mathcal{T} \wedge B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i; \boldsymbol{r}_{b,i}) \end{array} \right\},$$
$$\mathcal{R}'_{bin}(\widehat{\mathcal{T}}) = \left\{ \begin{array}{c} ((ck, (B_i)_{i=0}^{S-1}), (y, (\boldsymbol{b}_i, \widehat{\boldsymbol{r}}_{b,i})_{i=0}^{S-1}) : \boldsymbol{b}_i \in \{0,1\}^k \\ \wedge \|\widehat{\boldsymbol{r}}_{b,i}\| \leq \widehat{\mathcal{T}} \wedge yB_i = \mathrm{Com}_{ck}(y\boldsymbol{b}_i; \widehat{\boldsymbol{r}}_{b,i}) \end{array} \right\},$$

where $\mathcal{T}$ and $\widehat{\mathcal{T}}$ are norm bounds of $\boldsymbol{r}_{b,i}$ and $\widehat{\boldsymbol{r}}_{b,i}$ respectively and y is a relaxation factor.

To ensure the hardness of M-SIS in lattice settings, the prover needs to sample the masking values in special distributions (steps 2 and 3) and reject results that are out-of-bounds (steps 8 and 10). Accordingly, the verifier should check the bound based on the claimed bounds of the openings (step 12, first line).

A binary relation indicates two constraints, $\boldsymbol{b}_i \in \{0,1\}^k$ and $B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i; \boldsymbol{r}_{b,i})$. For the former one, it is equivalent to show $\boldsymbol{b}_i \circ (\boldsymbol{1} - \boldsymbol{b}_i) = \boldsymbol{0}$. Specifically, in a Σ-protocol, the prover encodes $\boldsymbol{b}_i$ as $\boldsymbol{g}_i = x\boldsymbol{b}_i + \boldsymbol{t}_i$ with a challenge x and masking vectors $\boldsymbol{t}_i$'s (step 7), which further allows the prover to check $xE + F = \mathrm{Com}_{ck}((\boldsymbol{g}_i \circ (x \cdot \boldsymbol{1} - \boldsymbol{g}_i))_{i=0}^{S-1}; \boldsymbol{z}_g)$, where $E = \mathrm{Com}_{ck}((\boldsymbol{t}_i \circ (\boldsymbol{1} - 2\boldsymbol{b}_i))_{i=0}^{S-1}; \boldsymbol{r}_e)$, $F = \mathrm{Com}_{ck}((-\boldsymbol{t}_i \circ \boldsymbol{t}_i)_{i=0}^{S-1}; \boldsymbol{r}_f)$, and $\boldsymbol{z}_g = x\boldsymbol{r}_e + \boldsymbol{r}_f$ (steps 4, 9, and 12). This part is same as a standard binary proof.

With our partial amortization, the latter constraint can be converted to $\mathrm{Com}_{ck}(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{b}_i; \sum_{i=0}^{S-1} \zeta^i \boldsymbol{r}_{b,i}) = \sum_{i=0}^{S-1} \zeta^i B_i$ under a challenge ζ. Since $\boldsymbol{g}_i = x\boldsymbol{b}_i + \boldsymbol{t}_i$, the prover needs to send the commitment G of the batched masking vectors and the batched randomness $\boldsymbol{z}_b$ to allow the verifier to check $x \sum_{i=0}^{S-1} \zeta^i B_i + G = \mathrm{Com}_{ck}(\sum_{i=0}^{S-1} \zeta^i \boldsymbol{g}_i; \boldsymbol{z}_b)$.

Theorem 1. *Suppose $S \leq 2$. Let $\kappa = S(S-1)/2$, $q/2 > 16(w\phi_1)^2 pkd$, $\gamma_{bin} = 4p^2\sqrt{d^3w^3\left(32dp^2k^3\phi_1^4 + 3\phi_2^2\mathcal{B}^2m^2\right)}$, and the HMC is hiding and γ_{bin}-binding. Protocol 1 has $(S,3)$-special soundness for relations $\mathcal{R}_{bin}(\mathcal{B}\sqrt{md})$ and $\mathcal{R}'_{bin}(8mdw^2p^2\phi_2\mathcal{B}\sqrt{3md})$ and SHVZK with a completeness error $1-1/(\mu(\phi_1)\mu(\phi_2))$ defined in Lemma 2.*

The proof for Theorem 1 is given in the full version [GZG+21].

4 Linear Equation for Balance Proofs

4.1 Corrector values in balance proofs

In existing RingCT protocols, to prove a transaction is valid, a spender (prover) needs to show 1) all the inputs and outputs are non-negative and 2) the difference between inputs and outputs is zero. The former relation can be checked in a range proof while the latter one is quite simple with a homomorphic commitment scheme. In lattice settings, some approaches use UMC to commit to an unbounded secret like amount [ESLL19, BDL+18]. However, as the size of

a UMC commitment grows linearly with the secret size, using a range proof directly is not practical in RingCT protocols.

MatRiCT [EZS+19] and MatRiCT+ [ESZ22] commit to bits of each amount to reduce the cost of UMC. Thus, the former relation can be proved in a binary proof. For the latter one, it requires "corrector values" to ensure $\text{Bits}(x_1) + \text{Bits}(x_2)$ equals to $\text{Bits}(x_1 + x_2)$ after some corrections. For instance, suppose a prover wants to prove that the following relations hold for M inputs and S outputs:

$$a_i \geq 0,\ \forall i \in [0, M); \quad \wedge \quad b_i \geq 0,\ \forall i \in [0, S); \tag{8}$$

$$\sum_{i=0}^{M-1} a_i = \sum_{i=0}^{S-1} b_i; \tag{9}$$

where a_i's are amounts of input accounts and b_i's are amounts of output accounts. A balance proof first converts each amount into k bits, $\text{Bits}(a_i) = (a_{i,0}, \cdots, a_{i,k-1}) = \boldsymbol{a}_i$ and $\text{Bits}(b_i) = (b_{i,0}, \cdots, b_{i,k-1}) = \boldsymbol{b}_i$, and commits to each $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$. Then, the prover shows 1) $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are binary vectors for Equation (8) and 2) Equation (9) holds such that:

$$\begin{aligned}
&\sum_{i=0}^{M-1} a_i = \sum_{i=0}^{S-1} b_i \iff \sum_{i=0}^{M-1}\sum_{j=0}^{k-1} 2^j a_{i,j} = \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} 2^j b_{i,j} \\
\iff &\sum_{i=0}^{S-1} b_{i,j} - \sum_{i=0}^{M-1} a_{i,j} + \tau_j - 2\tau_{j+1} = 0, \quad \forall j \in [0, k),
\end{aligned}$$

where τ_j's are correct values to ensure $\sum_{i=0}^{S-1} b_{i,j} - \sum_{i=0}^{M-1} a_{i,j} + \tau_j - 2\tau_{j+1} = 0$ holds for all $j \in [0, k)$ and $\tau_0 = \tau_k = 0$.

The balance proof requires additional work to ensure τ_j's are properly generated. In general, the prover needs to ensure $\tau_j \in [-M + 1, S - 1]$ for $j \in (0, k)$ (Lemma 4.1 in [EZS+19]) with range proofs. It is acceptable to embed τ_j's in the binary proof of $b_{i,j}$'s when $M = 1$ and $S \leq 2$, as with the Algorithm 8 and 9 in [EZS+19]. However, in other cases, the cost of a standard range proof is not negligible. Taking the state-of-the-art range proof in [ESLL19] as an example, the additional 64-bit range proof costs more than 90KB, while other parts only cost about 100KB. MatRiCT+ [ESZ22] addresses this issue by converting each range proof into a binary proof and embedding them into the binary proof of output accounts. Nevertheless, it still requires additional commitments for τ_j's.

One observation is that the *corrector values* $(\tau_0, \cdots, \tau_k)$ are unnecessary for balance proofs. To prove Equation (9) holds, one can simply prove

$$\sum_{i=0}^{M-1}\sum_{j=0}^{k-1} 2^j a_{i,j} = \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} 2^j b_{i,j} \iff \sum_{j=0}^{k-1} 2^j \Big(\sum_{i=0}^{S-1} b_{i,j} - \sum_{i=0}^{M-1} a_{i,j} \Big) = 0. \tag{10}$$

Let $c_j = \sum_{i=0}^{S-1} b_{i,j} - \sum_{i=0}^{M-1} a_{i,j}$. We can rewrite Equation (10) as $\sum_{j=0}^{k-1} 2^j c_j = 0$. The fact behind this idea is that though $\text{Bits}(a_1) + \text{Bits}(a_2) \neq \text{Bits}(a_1 + a_2)$, we

have $\langle \mathrm{Bits}(a_1), \mathbf{2}^k \rangle + \langle \mathrm{Bits}(a_2), \mathbf{2}^k \rangle = \langle \mathrm{Bits}(a_1 + a_2), \mathbf{2}^k \rangle$. Accordingly, we can fully remove the range proofs and the commitments to τ_j's. Additionally, the prover can avoid sending the commitment of $\boldsymbol{c} = (c_j)_{j=0}^{k-1}$ as it can be computed locally by the verifier:

$$\mathrm{Com}_{ck}(\boldsymbol{c}; *) = \sum_{i=0}^{S-1} \mathrm{Com}_{ck}(\boldsymbol{b}_i; *) - \sum_{i=0}^{M-1} \mathrm{Com}_{ck}(\boldsymbol{a}_i; *). \tag{11}$$

More importantly, the range proofs of c_j's can be avoided when $\boldsymbol{a}_i$'s and $\boldsymbol{b}_i$'s are binary vectors since $c_j = \sum_{i=0}^{S-1} b_{i,j} - \sum_{i=0}^{M-1} a_{i,j}$ implies $c_j \in [-M, S]$.

When using the inner-product relation in lattice settings, a serious problem arises at the same time: after encoding c_j as $f_j = xc_j + d_j$ (d_j is a masking value and x is a challenge), $\sum_{j=0}^{k-1} 2^j f_j$ can be greater than q, i.e., $(\sum_{j=0}^{k-1} 2^j f_j \bmod q) \neq \sum_{j=0}^{k-1} 2^j f_j$. Accordingly, verifying $\sum_{j=0}^{k-1} 2^j f_j = x \sum_{j=0}^{k-1} 2^j c_j + \sum_{j=0}^{k-1} 2^j d_j$ in R_q may *not* imply $\sum_{j=0}^{k-1} 2^j c_j = 0$. A straightforward solution is to use a large q to avoid overflowing. However, such a solution will result in a large proof size, making it impractical for real-world applications. In this paper, we solve this problem with a new *cycle masking* approach to ensure both f_j's and $\sum_{j=0}^{k-1} 2^j f_j$ are short at the same time with proper d_j's (see Section 4.2 for more details).

4.2 Linear equation satisfiability

We generalize the balance relation to a *linear equation satisfiability*. Let S be a positive integer[5] and $\omega_0, \cdots, \omega_{S-1}$ be public integers. The linear function is defined as

$$F(X_0, \cdots, X_{S-1}) = \sum_{i=0}^{S-1} \omega_i X_i. \tag{12}$$

The linear equation satisfiability is to prove the knowledge of $(b_i)_{i=0}^{S-1}$ such that $F(b_0, \cdots, b_{S-1}) = 0$.

To support b_i's with a wide range in lattice settings, we commit to the bits of b_i's with $B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i; *)$, where $\boldsymbol{b}_i$ is the binary representation of b_i. Thus, $F(b_0, \cdots, b_{S-1})$ can be rewritten as:

$$F'(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}) = \sum_{i=0}^{S-1} \Big(\omega_i \cdot \langle \mathbf{2}^k, \boldsymbol{b}_i \rangle \Big). \tag{13}$$

Definition 5. *The following defines the linear equation relations, proving $\mathcal{R}_{LE}$ and relaxed opening $\mathcal{R}'_{LE}$:*

$$\mathcal{R}_{LE}(\mathcal{T}) = \left\{ \begin{array}{c} ((ck, (\omega_i, B_i)_{i=0}^{S-1}), (\boldsymbol{b}_i, \boldsymbol{r}_{b,i})_{i=0}^{S-1}) : \boldsymbol{b}_i \in \{0,1\}^k \wedge \|\boldsymbol{r}_{b,i}\| \leq \mathcal{T} \\ \wedge B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i; \boldsymbol{r}_{b,i}) \wedge F'(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}) = 0 \end{array} \right\},$$

$$\mathcal{R}'_{LE}(\widehat{\mathcal{T}}) = \left\{ \begin{array}{c} ((ck, (\omega_i, B_i)_{i=0}^{S-1}), (y, (\boldsymbol{b}_i, \widehat{\boldsymbol{r}}_{b,i})_{i=0}^{S-1}) : \boldsymbol{b}_i \in \{0,1\}^k \wedge \|\widehat{\boldsymbol{r}}_{b,i}\| \leq \widehat{\mathcal{T}} \\ \wedge yB_i = \mathrm{Com}_{ck}(y\boldsymbol{b}_i; \widehat{\boldsymbol{r}}_{b,i}) \wedge F'(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}) = 0 \end{array} \right\},$$

[5] In the partially amortized binary proof, we require $S \leq 2$. However, here S can exceed 2 if the binary proofs for $(S-2)$-many $\boldsymbol{b}_i$'s are not necessary.

where $\mathcal{T}$ and $\widehat{\mathcal{T}}$ are norm bounds of $\boldsymbol{r}_{b,i}$ and $\widehat{\boldsymbol{r}}_{b,i}$ respectively and y is a relaxation factor.

Inner-product based relation. The $\mathcal{R}_{LE}$ indicates two important relations: 1) B_i's are commitments to bits and 2) $F'(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}) = 0$. The former can be proved in our partially amortized binary proof. For the second, we can rewrite Equation (13) as

$$\begin{aligned} F'(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}) = 0 &\Longleftrightarrow \sum_{i=0}^{S-1} \left(\omega_i \sum_{j=0}^{k-1} 2^j b_{i,j}\right) = 0 \\ \Longleftrightarrow \sum_{j=0}^{k-1} \left(2^j \cdot \sum_{i=0}^{S-1} \omega_i b_{i,j}\right) &= \sum_{j=0}^{k-1} 2^j c_j = 0, \end{aligned} \tag{14}$$

where $b_{i,j}$ is the j-th element of $\boldsymbol{b}_i$ and $c_j = \sum_{i=0}^{S-1} \omega_i b_{i,j}$. Denote $\boldsymbol{c} = (c_0, \cdots, c_{k-1})$. The verifier can compute the commitment of $\boldsymbol{c}$ with ω_i's and B_i's: $C = \mathrm{Com}_{ck}(\boldsymbol{c}; *) = \sum_{i=0}^{S-1} \omega_i B_i$. Let $\boldsymbol{f} = x\boldsymbol{c} + \boldsymbol{d}$ with some masking values $\boldsymbol{d} = (d_0, \cdots, d_{k-1})$ and a challenge x, $D = \mathrm{Com}_{ck}(\boldsymbol{d}; *)$, $d_{sum} = \langle \boldsymbol{d}, \boldsymbol{2}^k \rangle$. We have

$$\begin{aligned} &\mathrm{Com}_{ck}(\boldsymbol{f}; *) = \mathrm{Com}_{ck}(x\boldsymbol{c} + \boldsymbol{d}; *) = xC + D, \\ &\langle \boldsymbol{f}, \boldsymbol{2}^k \rangle = \langle x\boldsymbol{c} + \boldsymbol{d}, \boldsymbol{2}^k \rangle = x\langle \boldsymbol{c}, \boldsymbol{2}^k \rangle + \langle \boldsymbol{d}, \boldsymbol{2}^k \rangle = d_{sum}, \end{aligned} \tag{15}$$

which ensure $F'(\boldsymbol{b}_0, \cdots, \boldsymbol{b}_{S-1}) = 0$ holds.

Cycle masking. It is important to note that if the second equation in Equation (15) is verified on R_q, it may not imply $\langle \boldsymbol{c}, \boldsymbol{2}^k \rangle = 0$ when q is small (the soundness error increases). A straightforward solution is to use a large q. Unfortunately, this makes q grow linearly with k and a larger q implies a larger proof size. One may consider computing and sending d_{sum} in R to avoid the overflow problem. However, the cost of d_{sum} is non-negligible. Here we describe a more elegant and efficient encoding scheme named *cycle masking* to find proper d_j's that ensures f_j's are short and $d_{sum} = 0$. Specifically, the prover samples $(d'_j)_{j=1}^{k-1}$ and sets $d'_0 = d'_k = 0$. By setting $d_j = d'_j - 2d'_{j+1}$, we have

$$d_{sum} = \langle \boldsymbol{d}, \boldsymbol{2}^k \rangle = \sum_{j=0}^{k-1} 2^j d'_j - \sum_{j=1}^{k} 2^j d'_j = d'_0 - 2^k d'_k = 0. \tag{16}$$

Therefore, the prover can avoid transmitting d_{sum} and f_0. Accordingly, the verifier computes $f_0 = -\sum_{j=1}^{k-1} 2^j f_j$ on R, verifies $f_0 \in R_q$, and only checks the first equation in (15).

Note that the new encoding scheme can be further generalized to ensure that both f_i's are short and the corresponding masking vector $\boldsymbol{d}$ satisfies $\langle \boldsymbol{d}, \boldsymbol{t} \rangle = 0$ for a public $\boldsymbol{t} = (t_0, t_1 t_0, t_2 t_1 t_0, \cdots, \prod_{i=0}^{k-1} t_i)$ (the i-th term is t_{i-1} times of the previous one). Specifically, the prover first sets $d'_0 = d'_k = 0$ and samples $(d'_j)_{j=1}^{k-1}$. Then, she computes $d_j = d'_j - t_{j+1} d'_{j+1}$. Accordingly, we have $\langle \boldsymbol{d}, \boldsymbol{t} \rangle =$

Protocol 2 Linear Equation Satisfiability.

$\mathcal{P}_{LE}(ck, (\omega_i, B_i)_{i=0}^{S-1}, (\boldsymbol{b}_i, \boldsymbol{r}_{b,i})_{i=0}^{S-1}), \quad \mathcal{V}_{LE}(ck, (\omega_i, B_i)_{i=0}^{S-1})$

1: $\mathcal{P}_{LE}$: Run $\mathcal{P}_{bin}$ to obtain $E, F, G \leftarrow \mathcal{P}_{bin}(ck, (B_i)_{i=0}^{S-1}, (\boldsymbol{b}_i, \boldsymbol{r}_{b,i})_{i=0}^{S-1})$.
2: $\mathcal{P}_{LE}$: Set $T_3 = \max(-\sum_{\omega_i<0} \omega_i, \sum_{\omega_i>0} \omega_i) p\sqrt{wk}$.
3: $\mathcal{P}_{LE}$: Compute $c_j = \sum_{i=0}^{S-1} \omega_i b_{i,j}, \forall j \in [0, k)$.
4: $\mathcal{P}_{LE}$: Sample $d'_0 = d'_k = 0, d'_j \leftarrow \mathbb{D}_{\phi_3 T_3} \ \forall j \in [1, k)$, and $\boldsymbol{r}_d \leftarrow \mathbb{D}^m_{\phi_2 T_2}$.
5: $\mathcal{P}_{LE}$: Compute masking values $d_j = d'_j - 2d'_{j+1}, \forall j \in [0, k)$.
6: $\mathcal{P}_{LE}$: Compute $D = \mathrm{Com}_{ck}((d_j)_{j=0}^{k-1}; \boldsymbol{r}_d)$.
7: $\mathcal{P}_{LE} \Rightarrow \mathcal{V}_{LE}$: D, E, F, G.
8: $\mathcal{V}_{LE} \Rightarrow \mathcal{P}_{LE}$: $x \leftarrow \mathcal{C}$.
9: $\mathcal{P}_{LE}$: Run $\mathcal{P}_{bin}$ to obtain $(\boldsymbol{g}_i)_{i=0}^{S-1}, \boldsymbol{z}_g, \boldsymbol{z}_b \leftarrow \mathcal{P}_{bin}(x)$.
10: $\mathcal{P}_{LE}$: Set $\boldsymbol{c}_1 = (c_i)_{i=1}^{k-1}$ and $\boldsymbol{d}_1 = (d_i)_{i=1}^{k-1}$.
11: $\mathcal{P}_{LE}$: Compute $\boldsymbol{f}_1 = x\boldsymbol{c}_1 + \boldsymbol{d}_1$.
12: $\mathcal{P}_{LE}$: Run $\mathrm{Rej}(\boldsymbol{f}_1, x\boldsymbol{c}_1, 3\phi_3, T_3)$.
13: $\mathcal{P}_{LE}$: Compute $\boldsymbol{r}_c = \sum_{i=0}^{S-1} \omega_i \boldsymbol{r}_{b,i}$ and $\boldsymbol{z} = x\boldsymbol{r}_c + \boldsymbol{r}_d$.
14: $\mathcal{P}_{LE}$: Run $\mathrm{Rej}(\boldsymbol{z}, x\boldsymbol{r}_c, \phi_2, T_2)$.
15: $\mathcal{P}_{LE} \Rightarrow \mathcal{V}_{LE}$: $\boldsymbol{f}_1, (\boldsymbol{g}_i)_{i=0}^{S-1}, \boldsymbol{z}_g, \boldsymbol{z}_b, \boldsymbol{z}$.
16: $\mathcal{V}_{LE}$: Compute $f_0 = -\sum_{j=1}^{k-1} 2^j \cdot f_j$ in R and $C = \sum_{j=1}^{k-1} \omega_i B_i$.
17: $\mathcal{V}_{LE}$: Conduct the following checks

$$f_0 \overset{?}{\in} R_q; \quad \|f_j\| \overset{?}{\leq} 6\phi_3 T_3 \sqrt{d}, \quad \forall j \in [0, k); \quad \|\boldsymbol{z}\| \overset{?}{\leq} 2\phi_2 T_2 \sqrt{md};$$
$$xC + D \overset{?}{=} \mathrm{Com}_{ck}((f_0, \cdots, f_{k-1}); \boldsymbol{z});$$
$$\mathcal{V}_{bin}(ck, (B_i)_{i=0}^{S-1}, E, F, G, (\boldsymbol{g}_i)_{i=0}^{S-1}, \boldsymbol{z}_b) \overset{?}{=} 1.$$

$\sum_{j=0}^{k-1} \left(d_j \prod_{i=0}^{j} t_i\right) = \sum_{j=0}^{k-1} \left((d'_j - t_{j+1} d'_{j+1}) \prod_{i=0}^{j} t_i\right) = d'_0 - d'_k \prod_{i=0}^{k-1} t_i = 0$. Besides linear equation satisfiability, another direct application is to ensure $\langle \boldsymbol{\delta}, \mathbf{1}^\beta \rangle = 1$ in one-out-of-many proofs ($\langle \boldsymbol{f}, \mathbf{1}^\beta \rangle = x$ in steps 3 and 12 in Protocol 3). Since all f_i's are small, we can use a smaller parameter set to reduce the proof size. But in our ring signatures (Protocol 3), the improvement will be less significant since we fully avoid the binary proof part. Thus, we describe our ring signatures with the standard encoding scheme for simplicity.

Formal protocol. We formally describe our linear equation satisfiability protocol in Protocol 2. Similar to lattice-based Σ-protocols, the prover needs to sample the masking values from special distributions (steps 2 and 4) and reject results that are out-of-bounds (steps 12 and 14). Accordingly, the verifier checks the norms of the openings based on the claimed bounds (step 17, first line).

The prover and verifier run the first 5 steps of partially amortized binary proof (Protocol 1) to generate the commitment E, F, G. c_j's in Equation (14) are derived in step 3 and their masking values, d_j's, are generated in steps 4 and 5.

After receiving the challenge x, the prover generates the responses of the binary proof based on steps 7 to 11 in Protocol 1. As $\langle \boldsymbol{f}, \mathbf{2}^k \rangle = 0$ holds, the prover can avoid sending f_0 in step 11. In step 13, the randomness for $\boldsymbol{c}$ (i.e., $\boldsymbol{r}_c$) is derived based on $\boldsymbol{r}_{b,i}$'s since $c_j = \sum_{i=0}^{S-1} \omega_i b_{i,j}$.

Finally, in step 16, the verifier computes f_0 to ensure $\langle \boldsymbol{f}, \mathbf{2}^k \rangle = 0$ holds. Here she also needs to run on R instead of R_q to avoid the overflow problem and returns false if f_0 is not in R_q. The commitment of $\boldsymbol{c}$ is derived based on B_i's. The check on C and D in Step 17 ensures f_i's are properly generated from c_i's and the last verification $\mathcal{V}_{bin}$ (step 12 in Protocol 1) ensure $\boldsymbol{b}_i$'s are binary vectors.

The following theorem ensures the security of Protocol 2. For simplicity, we ignore the security requirement of the binary proof part as the balance constraint can use a different parameter set.

Theorem 2. *Let $\gamma_{LE} = 4\phi_3 md\mathcal{B}wp(\|\boldsymbol{\omega}\|_1^2+S+1)^{1/2}$ and the HMC is hiding and γ_{LE}-binding. Protocol 5 has $(S,3)$-special soundness for relations $\mathcal{R}_{LE}(\mathcal{B}\sqrt{md})$ and $\mathcal{R}'_{LE}(\gamma_{LE})$ and SHVZK with a completeness error $1-1/(\mu(\phi_1)\mu(\phi_2)\mu(\phi_3))$ defined in Lemma 2.*

Proof. Here we only focus on the SHVZK of $\boldsymbol{c}$ to argue the new encoding scheme leaks no information. Other parts are similar to standard proofs for lattice-based Σ-protocols. The full proof is formally given in the full version [GZG+21].

SHVZK of $\boldsymbol{c}$. The simulator samples $f'_j \leftarrow D^d_{\phi_3 T_3}$ for all $j \in [1,k)$ and sets $f'_0 = f'_j = 0$. Then, it computes $f_j = f'_j - 2f'_{j+1}$ to build $\boldsymbol{f}$ and sets $\boldsymbol{f}_1 = (f_1, \cdots, f_{k-1})$. Clearly, $\langle \boldsymbol{f}, \mathbf{2}^k \rangle = 0$ holds and $\boldsymbol{f}_1$ is close to the real distribution. □

5 Linear Sum for Ring Signatures

5.1 Ring signatures without binary proofs

Binary proofs in ring signatures. In most of existing ring signatures [ESLL19, EZS+19,ESZ22], a one-out-of-many proof is used to show a prover (signer) knows an opening of a public key P_l in a public key set $(P_0, \cdots, P_{N-1})$. The idea for this proof is regarding a public key as a commitment to zero. Thus, by constructing a secret binary sequence $\boldsymbol{\delta} = (\delta_{l,0}, \cdots, \delta_{l,N-1})$ with Hamming weight 1, a prover proves 1) $\boldsymbol{\delta}$ is well-formed and 2) $\sum_{i=0}^{N-1} \delta_{l,i} P_i = P_l$ is a commitment to 0. A straightforward solution for the former relation is to use a binary proof to show $\boldsymbol{\delta}$ is a binary vector and $\sum_{i=0}^{N-1} f_i = \sum_{i=0}^{N-1} (x\delta_i + a_i) = x$ for a challenge x and some masking values a_i's where $\sum_{i=0}^{N-1} a_i = 0$. Unfortunately, the proof size of this approach is $O(N)$ due to the size of $\boldsymbol{\delta}$.

The efficient logarithmic-size ring signatures "compress" $\boldsymbol{\delta}$ to several shorter delta vectors and allow the verifier to "reconstruct" $\boldsymbol{\delta}$ with these vectors [ESLL19, EZS+19,ESZ22]. Suppose $N = \beta^k$. Write $l = (l_0, \cdots, l_{k-1})$ and $i = (i_0, \cdots, i_{k-1})$ as the representations in base β such that $\delta_{l,i} = \prod_{j=0}^{k-1} \delta_{l_j, i_j}$. Instead of proving

that an N-size vector $\boldsymbol{\delta}$ is well-formed, the prover only needs to prove k-many β-size vectors, $(\delta_{l_j,0}, \cdots, \delta_{l_j,\beta-1})_{j=0}^{k-1}$, are well-formed, which reduces the proof size to $O(k\beta)$.

One observation is that the binary proof requires a larger parameter set than other parts of the proof to ensure security. This is due to 1) the hardness of the M-SIS problem and 2) $\delta_{l,i}(1-\delta_{l,i}) = 0$ may not hold in R_q for a smaller q (the relaxation factor for this constraint is large) [EZS+19, ESZ22]. Though the binary proof is simple, its larger parameters indicate a larger proof size. Motivated by this, we analyze ring signatures and find proving $\boldsymbol{\delta}$ being a *binary* sequence is redundant. For example, a signer can prove knowing the opening to $2P_l$ instead of P_l without sacrificing security. Generally speaking, it is sufficient to relax the one-out-of-many proof by proving the knowledge of an opening to $\sum_{i=0}^{N-1} b_i P_i$ in ring signatures, where b_i's are short and not all b_i's are 0 (relaxing $\delta_{l,i}$ to b_i). While reducing binary proof is nice in itself, we would like to highlight that it is particularly important for ring signatures. As "*the binary proof requires a much bigger modulus than (other parts of) the one-out-of-many proof*" [EZS+19], avoiding the binary proof fully releases ring signatures from the burden of large parameters. Therefore, instead of running a full one-out-of-many proof, ring signatures can use a much more efficient *linear sum* proof with a small modulus.

New ring signatures. Let $\boldsymbol{r}$ be a private key and P_l be the corresponding public key in a public key set $\boldsymbol{P} = (P_0, \cdots, P_{N-1})$ for some $N \geq 1$ and $0 \leq l < N$. The goal of ring signatures is to show the knowledge of a secret key(s) corresponding to a public key(s) in $\boldsymbol{P}$. Based on the idea in Section 5.1, we show that proving the knowledge of an opening of a *short non-zero linear sum* relation of the public keys suffices for ring signatures, i.e., knowing some bounded b_i's and an opening to $\sum_{i=0}^{N-1} b_i P_i$ where at least one b_i is not zero.

Theorem 3. *In ring signatures, if the commitment scheme is computational hiding and γ-binding, then the ring signature scheme described above is unforgeable with respect to insider corruption*[6] *in the random oracle model.*

Proof. Assume there exists a PPT adversary $\mathcal{F}$ that can efficiently forge a ring signature with non-negligible probability, we have an adversary $\mathcal{A}$ which can break the binding property of the commitment scheme, and solve the M-SIS problem accordingly.

$\mathcal{A}$ samples $\boldsymbol{r} \leftarrow \{-\mathcal{B}, \cdots, \mathcal{B}\}$ and computes an invalid public key $pk_l = \mathrm{Com}_{ck}(1, 0, \cdots, 0; \boldsymbol{r})$. Due to the hiding probability of the commitment scheme, $\mathcal{F}$ cannot distinguish pk_l with other public keys. Then, $\mathcal{A}$ runs $\mathcal{F}$ for $(k+1)$ times to get $(k+1)$ forgeries with distinct challenges and same $A, B, (E_j)_{j=0}^{k-1}$ based on the forking lemma (pk_l is not corrupted). Furthermore, $\mathcal{A}$ runs the extractor of Protocol 3 with the $(k+1)$ forgeries to get valid $b'_i = y^k b_i$ for $i \in [0, N)$ and a valid opening $(\mathbf{0}; \boldsymbol{s})$ of $y \sum_{i=0}^{N-1} b_i \cdot pk_i$ for some public keys. With

[6] The insider corruption allows the attacker to obtain private keys to some public keys with corruption queries. Accordingly, the signature forgery should not include these "corrupted" public keys in its ring.

non-negligible probability, we have $b_l \neq 0$ since $\mathcal{F}$ can only make polynomially many registration queries to $\mathcal{A}$. Then, $\mathcal{A}$ uses all private keys but $\boldsymbol{r}_l$ to compute $\boldsymbol{s}' = y^{k-1}\boldsymbol{s} - \sum_{i \neq l} b'_i \boldsymbol{r}_i$. Since $b_l \neq 0$, we have $b'_l \neq 0$ (with non-negligible probability), and thus we find a binding collision for the HMC, $((b'_l, 0, \cdots, 0); b'_l \boldsymbol{r})$ and $(\boldsymbol{0}; \boldsymbol{s}')$. Detailed proof is given in the full version [GZG+21]. □

5.2 Unbalanced linear sum relation

It is important to note that the linear sum proof may be difficult to adopt the "compressing" technique in [ESLL19,EZS+19,ESZ22] to achieve logarithmic-size ring signatures as there may not exist $(b_{j,0}, \cdots, b_{j,\beta-1})$ such that $b_i = \prod_{j=0}^{k-1} b_{j,i_j}$ for all $i \in [0, N)$ and finding such a solution can be very inefficient. This brings us to the idea of adopting "unbalanced" relations as in relaxed proofs: using a stricter relation in proving, but checking the original relation in verifying. For instance, as a linear sum relation is sound for ring signatures and a one-out-of-many relation is stricter than the linear sum relation, a prover can use $b_i = \delta_{l,i}$ in the one-out-of-many relation to generate a proof. The verifier checks the linear sum relation instead of the one-out-of-many relation. The unbalanced linear sum protocol is similar to the one-out-of-many protocol without the binary proof part, which avoids the constraint of $b_i \in \{0, 1\}$. Additionally, to ensure at least one b_i is not zero, the verifier checks whether $\|\boldsymbol{b}\| > 0$. The unbalanced linear sum relations are defined as follows:

Definition 6. *The following defines the unbalanced relations for our unbalanced linear sum proof, proving $\mathcal{R}_{LS}$ and relaxed opening $\mathcal{R}'_{LS}$:*

$$\mathcal{R}_{LS}(\mathcal{T}) = \left\{ \begin{array}{c} ((ck, \boldsymbol{P}), (l, \boldsymbol{r})) : \\ l \in [0, N) \wedge \|\boldsymbol{r}\| \leq \mathcal{T} \wedge P_l = \mathrm{Com}_{ck}(\boldsymbol{0}; \boldsymbol{r}) \end{array} \right\},$$

$$\mathcal{R}'_{LS}(\widehat{\mathcal{T}}_b, \widehat{\mathcal{T}}_r) = \left\{ \begin{array}{c} ((ck, \boldsymbol{P}), (y, \boldsymbol{b}, \widehat{\boldsymbol{r}})) : \\ \|\boldsymbol{b}\| > 0 \wedge \|b_i\| \leq \widehat{\mathcal{T}}_b \wedge \|\widehat{\boldsymbol{r}}\| \leq \widehat{\mathcal{T}}_r \wedge y \sum_{i=0}^{N-1} b_i P_i = \mathrm{Com}_{ck}(\boldsymbol{0}; \widehat{\boldsymbol{r}}) \end{array} \right\},$$

where y is a relaxation factor and $\mathcal{T}$, $\widehat{\mathcal{T}}_b$, and $\widehat{\mathcal{T}}_r$ are norm bounds of $\boldsymbol{r}$, b_i, and $\widehat{\boldsymbol{r}}$ respectively.

Though our "unbalanced" relation is derived from relaxed relations, the motivations behind are different. In our approach, we start from the *verifier's* side and show verifying a linear sum proof suffices in ring signatures. To improve the efficiency, we restrict the prover's relation and require the prover to run under a one-out-of-many relation. The key idea is to find a *strict* and *efficient* relation for provers. On the other hand, existing relaxed proofs start from the *prover's* side and find straightforward soundness proofs do not work. They need to relax the relation on the verifier's side to overcome the complications. The key idea is to find a *relaxed* but *sound* relation for verifiers. Thus we use the term "unbalanced relations" to distinguish with relaxed relations.

Logarithmic-size proof. To achieve a logarithmic-size unbalanced linear sum proof, a prover can directly apply the "compressing" technique in [ESLL19,

EZS+19]. Here we briefly describe this technique to fill some gaps before showing the details of the formal protocol. Specifically, the prover finds and commits to k-many sequences $(\delta_{l_j,0}, \cdots, \delta_{l_j,\beta-1})_{j=0}^{k-1}$ such that $\delta_{l,i} = \prod_{j=0}^{k-1} \delta_{l_j,i_j}$ for all $i \in [0, N)$. After receiving a challenge x, the prover's response contains $f_{j,i} = x\delta_{l_j,i} + a_{j,i}$ with some masking values $a_{j,i}$'s. Let $\boldsymbol{\delta'} = (\delta_{l_0,0}, \cdots, \delta_{l_{k-1},\beta-1})$, $\boldsymbol{a} = (a_{0,0}, \cdots, a_{k-1,\beta-1})$, and $\boldsymbol{f} = (f_{0,0}, \cdots, f_{k-1,\beta-1})$. The prover needs to show 1) $(\delta_{l_j,0}, \cdots, \delta_{l_j,\beta-1})$'s are short non-zero vectors and are properly committed and 2) $\delta_{l,i}$'s can be constructed with $\delta_{l,i} = \prod_{j=0}^{k-1} \delta_{l_j,i_j}$ such that $\sum_{i=0}^{N-1} \delta_{l,i} P_i$ being a commitment to zero.

For the first constraint, the prover shows the following equations hold:

$$\begin{aligned} &\mathrm{Com}_{ck}(\boldsymbol{f}; *) = \mathrm{Com}_{ck}(x\boldsymbol{\delta'} + \boldsymbol{a}; *) = xB + A; \\ &\sum_{i=0}^{\beta-1} f_{j,i} = x \sum_{i=0}^{\beta-1} \delta_{l_j,i} + \sum_{i=0}^{\beta-1} a_{j,i} = x + \sum_{i=0}^{\beta-1} a_{j,i}, \quad \forall j \in [0, k). \end{aligned} \tag{17}$$

The second equation ensures at least one element in $(\delta_{l_j,i})_{i=0}^{\beta-1}$ is not 0 for all j's as $\sum_{i=0}^{\beta-1} f_{j,i} = x + \sum_{i=0}^{\beta-1} a_{j,i}$ implies $\sum_{i=0}^{\beta-1} \delta_{l_j,i} = 1$. Moreover, proving $\boldsymbol{\delta'}$ being "short" is done in the norm check of HMC openings. Besides, the second equation is not a necessary condition for the linear sum relation. However, based on the unbalanced relations in Section 5.1, the prover can efficiently show the second equation holds with a one-out-of-many relation.

For the second constraint, the verifier computes the product $p_i(x) = \prod_{j=0}^{k-1} f_{j,i_j}$:

$$\begin{aligned} p_i(x) &= \prod_{j=0}^{k-1} f_{j,i_j} = \prod_{j=0}^{k-1} (x\delta_{l_j,i_j} + a_{j,i_j}) \\ &= x^k \cdot \prod_{j=0}^{k-1} \delta_{l_j,i_j} + \sum_{j=0}^{k-1} p_{i,j} \cdot x^j = \delta_{l,i} x^k + \sum_{j=0}^{k-1} p_{i,j} x^j, \end{aligned} \tag{18}$$

where $p_{i,j}$'s are functions of δ_{l_j,i_j}'s (i.e., l) and $a_{j,i}$'s. Equation (18) holds for all $i \in [0, N)$. As $p_{i,j}$'s are independent of x, the prover can pre-compute $p_{i,j}$'s and send $E_j = \sum_{i=0}^{N-1} p_{i,j} P_i$ to allow the verifier to cancel out the coefficients of the terms $1, x, \cdots, x^{k-1}$ before receiving x (the randomness is omitted here for simplicity). For x^k part, it can be set to the prover's public key P_l with $\sum_{i=0}^{N-1} \delta_{l,i} P_i$.

Formal protocol. We formally describe our unbalanced linear sum proof protocol in Protocol 3. Similar to lattice-based Σ-protocols, the prove samples all masking values from special distributions (steps 2 and 5) and rejects results that are out-of-bounds (steps 11 and 13). Accordingly, the verifier should check the norms of the openings based on the claimed bounds (step 16).

Steps 3 and 4 generate the masking values $a_{j,i}$'s for $\delta_{l_j,i}$'s and ensure $\sum_{i=0}^{\beta-1} a_{j,i} = 0$ (which further ensure $\sum_{i=0}^{\beta-1} (x\delta_{l_j,i} + a_{j,i}) = x$). $p_{i,j}$'s in step 3 are derived from Equation (18).

Protocol 3 Unbalanced Linear Sum Proof.

$\mathcal{PP}_{LS}(ck, \boldsymbol{P}, (l, \boldsymbol{r}))$, $\quad \mathcal{V}_{LS}(ck, \boldsymbol{P})$

1: $\mathcal{P}_{LS}$: Set $T_1 = p\sqrt{kw}$ and $T_2 = (wp)^k \mathcal{B}\sqrt{2md}$.
2: $\mathcal{P}_{LS}$: Sample

$$\boldsymbol{r}_a \leftarrow \mathbb{D}^m_{\phi_2 T_2}; \quad \boldsymbol{r}_b \leftarrow \mathbb{S}^{md}_{\mathcal{B}}; \quad a_{j,1}, \cdots, a_{j,\beta-1} \leftarrow \mathbb{D}_{\phi_1 T_1}, \forall j \in [0, k).$$

3: $\mathcal{P}_{LS}$: Set $a_{j,0} = -\sum_{i=1}^{\beta-1} a_{j,i}, \forall j$ and $\boldsymbol{a} = (a_{j,i})^{k-1,\beta-1}_{j=0,i=0}$.
4: $\mathcal{P}_{LS}$: Compute $\boldsymbol{\delta} = (\delta_{l_j,i})^{k-1,\beta-1}_{j=0,i=0}$ and $p_{i,j}$'s based on l.
5: $\mathcal{P}_{LS}$: Sample $\boldsymbol{\rho}_0 \leftarrow \mathbb{D}^m_{\phi_2 T_2}$ and $\boldsymbol{\rho}_j \leftarrow \mathbb{S}^m_{\mathcal{B}}, \forall j \in [0, k)$
6: $\mathcal{P}_{LS}$: Compute the commitments

$$E_j = \sum_{i=0}^{N-1} p_{i,j} P_j + \mathrm{Com}_{ck}(\mathbf{0}; \boldsymbol{\rho}_j), \quad \forall j \in [0, k),$$
$$B = \mathrm{Com}_{ck}(\boldsymbol{\delta}; \boldsymbol{r}_b), \qquad A = \mathrm{Com}_{ck}(\boldsymbol{a}; \boldsymbol{r}_a).$$

7: $\mathcal{P}_{LS} \Rightarrow \mathcal{V}_{LS}$: $A, B, (E_j)^{k-1}_{j=0}$.
8: $\mathcal{V}_{LS} \Rightarrow \mathcal{P}_{LS}$: $x \leftarrow \mathcal{C}$.
9: $\mathcal{P}_{LS}$: Set $\boldsymbol{\delta}_1 = (\delta_{l_j,i})^{k-1,\beta-1}_{j=0,i=1}$ and $\boldsymbol{a}_1 = (a_{j,i})^{k-1,\beta-1}_{j=0,i=1}$.
10: $\mathcal{P}_{LS}$: Compute $\boldsymbol{f}_1 = x\boldsymbol{\delta}_1 + \boldsymbol{a}_1$.
11: $\mathcal{P}_{LS}$: Run $\mathrm{Rej}(\boldsymbol{f}_1, x\boldsymbol{\delta}_1, \phi_1, T_1)$.
12: $\mathcal{P}_{LS}$: Compute $\boldsymbol{z}_b = x\boldsymbol{r}_b + \boldsymbol{r}_a$ and $\boldsymbol{z}_r = x^k \boldsymbol{r} - \sum_{j=0}^{k-1} x^j \boldsymbol{\rho}_j$.
13: $\mathcal{P}_{LS}$: Run $\mathrm{Rej}((\boldsymbol{z}_b, \boldsymbol{z}_r), (x\boldsymbol{r}_b, x^k\boldsymbol{r} - \sum_{j=1}^{k-1} x^j \boldsymbol{\rho}_j), \phi_2, T_2)$.
14: $\mathcal{P}_{LS} \Rightarrow \mathcal{V}_{LS}$: $\boldsymbol{f}_1, \boldsymbol{z}_b, \boldsymbol{z}_r$.
15: $\mathcal{V}_{LS}$: Compute $f_{j,0} = x - \sum_{i=1}^{\beta-1} f_{j,i}, \forall j \in [0, k)$.
16: $\mathcal{V}_{LS}$: Conduct the following checks

$$\|f_{j,i}\| \overset{?}{\leq} 2\phi_1 T_1 \sqrt{d}, \forall j \in [0, k), \forall i \in [1, \beta); \quad \|f_{j,0}\| \overset{?}{\leq} 2\phi_1 T_1 \sqrt{\beta d}, \forall j \in [0, k);$$
$$\|\boldsymbol{z}_b\|, \|\boldsymbol{z}_r\| \overset{?}{\leq} 2\phi_2 T_2 \sqrt{md}; \qquad xB + A \overset{?}{=} \mathrm{Com}_{ck}((f_{0,0}, \cdots, f_{k-1,\beta-1}); \boldsymbol{z}_b);$$
$$\sum_{i=0}^{N-1} (\prod_{j=0}^{k-1} f_{j,i_j}) P_j - \sum_{j=0}^{k-1} E_j x^j \overset{?}{=} \mathrm{Com}_{ck}(\mathbf{0}; \boldsymbol{z}_r).$$

Upon receiving the challenge x, the prover generates the responses $\boldsymbol{f}_1$, $\boldsymbol{z}_b$, and $\boldsymbol{z}_r$. For $\boldsymbol{f}$, the prover can avoid sending $f_{j,0}$'s as $\sum_{i=0}^{\beta-1} f_{j,i} = x$ holds. In step 12, $\boldsymbol{z}_r$ is the response to randomness in P_l and E_j's based on Equation (18).

Finally, the verifier computes $f_{j,0} = x - \sum_{i=1}^{\beta-1} f_{j,i}$ for all $j \in [0, k)$ as $\sum_{i=0}^{\beta-1} f_{j,i} = x$, which ensures $\sum_{i=0}^{\beta-1} \delta_{l_j,i} = 1$ (and further ensures at least one element in $(\delta_{l_j,0}, \cdots, \delta_{l_j,\beta-1})$ is not 0 for all j's). The last two checks ensure that $\sum_{i=0}^{N-1} \delta_{l,i} P_i$ is a commitment to 0.

Theorem 4. *Let* $\gamma_{LS} = (4\phi_1\sqrt{k\beta})^k d^{k-\frac{1}{2}}$ *and* $\gamma'_{LS} = (k+1)2^{\kappa'+2}\sqrt{2}\phi_2\mathcal{B}md^2w^\kappa p^{\kappa+1}$ *for* $\kappa = k(k+1)/2$ *and* $\kappa' = k(k-1)/2$*, the HMC is hiding and* γ_{LS}*-binding. Protocol 6 has* $(k+1)$*-special soundness for relations* $\mathcal{R}_{LS}(\mathcal{B}\sqrt{md})$ *and* $\mathcal{R}'_{LS}(\gamma_{LS}, \gamma'_{LS})$ *and SHVZK with a completeness error* $1 - 1/(\mu(\phi_1)\mu(\phi_2))$ *defined in Lemma 2.*

The proof is given in the full version [GZG+21].

6 RingCT Protocol

6.1 Overview

Since both Protocol 2 and Protocol 3 are public coin, we can use Fiat-Shamir heuristic to transform them into non-interactive protocols and build a RingCT protocol. We first show how to combine the two proofs and then address some additional issues such as the unbalancing problem and double-spending.

Combining two proofs. In a RingCT protocol, a spender needs to prove a transaction is valid and hides the identity of input accounts simultaneously. This can be achieved by adding decoy accounts into inputs and proving the balance and linear sum (ring signature) relations in one proof. Note that the binary proofs for inputs can be reduced as they have been verified as output accounts in previous transactions. Let $\mathtt{CNK}_{in} = (\boldsymbol{r}_{a,i})_{i=0}^{M-1}$ and $\mathtt{CNK}_{out} = (\boldsymbol{r}_{b,i})_{i=0}^{S-1}$ be the sets of input and output coin keys respectively (i.e. randomness), $\mathtt{CN}_{in} = (A_i)_{i=0}^{M-1}$ and $\mathtt{CN}_{out} = (B_i)_{i=0}^{M-1}$ be the sets of input and output coins (commitments to $\boldsymbol{a}_i$'s and $\boldsymbol{b}_i$'s, i.e., $A_i = \mathrm{Com}_{ck}(\boldsymbol{a}_i; \boldsymbol{r}_{a,i})$ and $B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i; \boldsymbol{r}_{b,i})$). Consider NM inputs $(\mathtt{CN}_{in}^{(j)})_{j=0}^{N-1} = (A_i^{(j)})_{i=0,j=0}^{M-1,N-1}$, which have M spender's accounts (the spender owns the amount values and coin keys at index $j = l$, $(\boldsymbol{a}_i^{(l)}, \boldsymbol{r}_{a,i}^{(l)})_{i=0}^{M-1}$), and $(N-1)M$ decoy accounts, $\mathtt{CN}_{in}^{(j)}$ where $j \neq l$. To transfer funds to S output accounts, $(B_i = \mathrm{Com}_{ck}(\boldsymbol{b}_i, \boldsymbol{r}_{b,i}))_{i=0}^{S-1}$, the spender needs to send an additional commitment C and compute public keys $(P_j)_{j=0}^{N-1}$ as follows:

$$\begin{aligned} C &= \mathrm{Com}_{ck}(\boldsymbol{c}; \boldsymbol{r}'_c), \\ P_j &= \sum_{i=0}^{S-1} B_i - \sum_{i=0}^{M-1} A_i^{(j)} - C, \quad \forall j \in [0, N). \end{aligned} \tag{19}$$

Ideally, we regard P_l as a commitment to zero with the private key (randomness) $\boldsymbol{r} = \sum_{i=0}^{S-1} \boldsymbol{r}_{b,i} - \sum_{i=0}^{M-1} \boldsymbol{r}_{a,i}^{(l)} - \boldsymbol{r}'_c$. The spender can further show P_l is a commitment to zero as in our ring signature scheme, which proves the amount balance and hides the identity at the same time. Unfortunately, as linear sum proof only ensures $\sum_{i=0}^{N-1} \widetilde{b}_i P_i$ is a commitment to zero instead of each P_i (we use $\widetilde{b}_i$ to distinguish with the output amount b_i), the above approach will incur an *unbalancing problem*. For instance, if the spender owns *two* input accounts at indices s and t with $a^{(s)} = 2$ and $a^{(t)} = 1$, she can mint $b = 4$ coins $(b \neq a^{(s)} + a^{(t)})$ by setting $M = S = 1$, $\widetilde{b}_s = 3$, and $\widetilde{b}_t = -2$. As P_s is a commitment to $b - a^{(s)}$

(i.e., 2) and P_t is a commitment to $b - a^{(t)}$ (i.e., 3), $\widetilde{b}_s P_s + \widetilde{b}_t P_t$ is a commitment to 0 (here we use the amounts directly instead of their bits for simplicity). This is due to the security proof of our ring signatures relies on P_i's being correctly generated (i.e., commitments to 0), which may not be true in RingCT as P_i's are derived from different accounts. Prior to show our solution, we describe the linkable version of our ring signature to avoid double spending.

Avoid double-spending. To avoid double-spending, we extend our ring signature (Protocol 3) to provide linkability by checking the serial number of each input account to ensure it is not included in previous transactions. This could be done by following the blueprint of MatRiCT [EZS+19] and MatRiCT+ [ESZ22]. Consider a new commitment key $\boldsymbol{H}$. A serial number SN is a public commitment to zero under $\boldsymbol{H}$ with the signing key $\boldsymbol{r}$ as the randomness, i.e., $\mathsf{SN} = \boldsymbol{H} \cdot \boldsymbol{r}$. At step 6 of Protocol 3, the prover needs to additionally compute $F_j = \boldsymbol{H} \cdot \boldsymbol{\rho}_j$ for all $j \in [0, k)$. In the verification, the verifier can 1) link the proof with previous ones based on SN and 2) check SN is correct with $x^k \cdot \mathsf{SN} - \sum_{j=0}^{k-1} x^j F_j = \boldsymbol{H} \cdot \boldsymbol{z}_b$.

In a RingCT protocol, each account has an additional account key pair, $(\mathsf{pk}, \mathsf{sk})$ such that $\mathsf{pk} = \mathrm{Com}_{rk}(\boldsymbol{0}, \mathsf{sk})$ under a set of different public parameters rk. For each input account, $i \in [0, M)$, the spender places it at index l of an N-size ring $\boldsymbol{PK}_i$ and runs the above linkable version of Protocol 3, $\mathcal{P}_{LS}(rk, \boldsymbol{PK}_i, \mathsf{SN}_i, (l, \mathsf{sk}_i))$. Besides, as described in MatRiCT+ [ESZ22], the linkable ring signatures for M input accounts can be aggregated into one with $\sum_{i=0}^{M-1} \alpha_i \mathsf{pk}_i + P_i$ as the public keys and $\sum_{i=0}^{M-1} \alpha_i \mathsf{sk}_i + \boldsymbol{r}$ as the signing key, where α_i's are challenges. To avoid double-spending, the verifier simply checks the serial numbers are distinct and not included in previous transactions.

Avoiding unbalancing problem. To address the unbalancing problem described above, we show a simple and efficient approach to ensure the spender can only use *one* P_i to run the linear sum proof. Recall the linkable version of our unbalanced linear sum proof. The serial number ensures a spender cannot 1) avoid sending any serial number of her real account and 2) include the serial numbers of other's accounts. Thus, the serial number set of a valid transaction *must* be the serial numbers of all real input accounts. Accordingly, the number of serial numbers must be $\mathrm{HW}(\boldsymbol{b}) \cdot S$, which reveals how many accounts are used as real inputs in our unbalanced linear sum proof. Therefore, to ensure *one* P_i out of an N-size list, the verifier checks *the number of serial numbers being* S.

The security of this approach can be derived directly from the prosperities of serial numbers. If the count of serial numbers is different from the count of real input accounts in a transaction (and causes the unbalancing problem), it either excludes the serial numbers of the real accounts or includes the serial numbers of others' accounts. Since the serial number and secret key share the same secret $\widetilde{\boldsymbol{b}}$ to indicate the indices, it cannot pass the verification in either case.

6.2 RingCT functions

We present the full set of algorithms in our RingCT protocol. Consider the case with M input amounts $(a_i)_{i=0}^{M-1}$ and S output amounts $(b_i)_{i=0}^{S-1}$. The balance

proof part in RingCT protocols is a special case of linear equation satisfiability, where $N = S + M$, $(\omega_0, \cdots, \omega_{S-1}) = (1, \cdots, 1)$, and $(\omega_S, \cdots, \omega_{S+M-1}) = (-1, \cdots, -1)$. Accordingly, Equation (12) can be expressed as

$$F(a_0, \cdots, a_{M-1}, b_0, \cdots, b_{S-1}) = \sum_{i=0}^{S-1} b_i - \sum_{i=0}^{M-1} a_i. \tag{20}$$

Let $\mathbb{B}$ be the set of public keys and coin keys of all registered accounts, $\mathbb{S}$ be the set of serial numbers for spent output accounts, $\mathtt{CNK}_{in}$ and $\mathtt{CNK}_{out}$ be the sets of real input and output coin keys respectively (i.e. randomness), $\mathtt{CN}_{in}$ and $\mathtt{CN}_{out}$ be the sets of real input and output coins (commitments to amount with coin keys as randomness), $\mathtt{SK}_{in}$ and $\mathtt{SK}_{out}$ be the sets of real input and output secret keys respectively, $\mathtt{PK}_{in}$ and $\mathtt{PK}_{out}$ be the sets of real input and output public keys respectively, $\mathtt{SN}_{in}$ be the set of serial numbers of real inputs, $\mathtt{CN}_d$ and $\mathtt{PK}_d$ be the sets of decoy coins and decoy public keys.

In Protocol 2, denote the initial commitments as $\mathsf{CMT}_{LE} = (D, E, F, G)$, the prover's response as $\mathsf{RSP}_{LE} = (\boldsymbol{f}_1, (\boldsymbol{g}_i)_{i=0}^{S-1}, \boldsymbol{z}, \boldsymbol{z}_g, \boldsymbol{z}_b)$, and $\mathsf{CMT}^*_{LE} = E$. In Protocol 3, denote the initial commitment as $\mathsf{CMT}_{LS} = (A, B, (E_j)_{j=0}^{k-1})$, the prover's response as $\mathsf{RSP}_{LS} = (\boldsymbol{f}_1, \boldsymbol{z}_b, \boldsymbol{z}_r)$, and $\mathsf{CMT}^*_{LS} = (B, (E_j)_{j=1}^{k-1})$.

• **Setup**(1^λ): Run $\boldsymbol{G} \leftarrow \mathsf{CKeygen}$ and set $ck = \boldsymbol{G}$. Run $\boldsymbol{G}' \leftarrow \mathsf{CKeygen}$ and set $rk = \boldsymbol{G}'$. Run $\boldsymbol{H} \leftarrow \mathsf{CKeygen}$ and set $sk = \boldsymbol{H}$. Choose a hash function $H : \{0,1\}^* \to \mathcal{C}$. Return $pp = (ck, rk, sk, H)$.

• **Mint**(pp, v): Sample $\boldsymbol{r} \leftarrow \{-\mathcal{B}, \cdots, \mathcal{B}\}^{md}$ and compute $\mathrm{Bits}(v) = (v_0, \cdots, v_{k-1})$, $B = \mathrm{Com}_{ck}(\boldsymbol{v}; \boldsymbol{r})$. Return $(\mathsf{cn}, \mathsf{cnk}) = (B, \boldsymbol{r})$.

• **KeyGen**(pp): Sample $\boldsymbol{r}' \leftarrow \{-\mathcal{B}, \cdots, \mathcal{B}\}^{md}$ and compute $P = \mathrm{Com}_{rk}(\boldsymbol{0}; \boldsymbol{r}')$. Return $(\mathsf{pk}, \mathsf{sk}) = (P, \boldsymbol{r}')$.

• **SerialGen**(pp, sk): Compute $\mathsf{SN} = \mathrm{Com}_{sk}(\boldsymbol{0}; \boldsymbol{r}')$ where $\boldsymbol{r}' = \mathsf{sk}$. Return SN.

• **Spend**$(pp, (a_i)_{i=0}^{M-1}, (b_i)_{i=0}^{S-1}, \mathtt{CN}_{in}, \mathtt{CNK}_{in}, \mathtt{PK}_{in}, \mathtt{SK}_{in}, \mathtt{SN}_{in}, \mathtt{CN}_d, \mathtt{PK}_d)$: Choose $l \leftarrow [0, N-1]$. Parse $\mathtt{CN}_{in} = (A_i^{(l)})_{i=0}^{M-1}$, $\mathtt{CNK}_{in} = (\boldsymbol{r}_{a,i}^{(l)})_{i=0}^{M-1}$, $\mathtt{PK}_{in} = (P_{a,i}^{(l)})_{i=0}^{M-1}$, and $\mathtt{SK}_{in} = (\boldsymbol{r}_{a,i}^{\prime(l)})_{i=0}^{M-1}$. Set $\mathrm{Bits}(a_i) = \boldsymbol{a}_i$ for $i \in [0, M)$ and $\mathrm{Bits}(b_i) = \boldsymbol{b}_i$ for $i \in [0, S)$. Call **Mint**$(pp, b_i) = (\mathsf{cn}_i, \mathsf{cnk}_i) = (B_i, \boldsymbol{r}_{b,i})$, **KeyGen**$(pp) = (\mathsf{pk}_i, \mathsf{sk}_i) = (P_{b,i}, \boldsymbol{r}'_{b,i})$, and **SerialGen**$(pp, \mathsf{sk}_i) = \mathsf{SN}_i$ for $i \in [0, S)$ for output accounts. Set $\mathtt{CN}_{out} = (\mathsf{cn}_i)_{i=0}^{S-1}$, $\mathtt{CNK}_{out} = (\mathsf{cnk}_i)_{i=0}^{S-1}$, $\mathtt{PK}_{out} = (\mathsf{pk}_i)_{i=0}^{S-1}$, $\mathtt{SK}_{out} = (\mathsf{sk}_i)_{i=0}^{S-1}$, and $\mathtt{SN}_{out} = (\mathsf{SN}_i)_{i=0}^{S-1}$. Set $\mathtt{CN}_d = (A_i^{(j)})_{i=0,j=0}^{M-1,N-1}$ and $\mathtt{PK}_d = (P_{a,i}^{(j)})_{i=0,j=0}^{M-1,N-1}$ for $j \neq l$. Proceed as follows:

1. Compute $\zeta = H(pp, \mathtt{CN}_{in}, \mathtt{PK}_{in}, \mathtt{SN}_{in}, \mathtt{CN}_d, \mathtt{PK}_d)$ as the first challenge in the partially amortized binary proof (Protocol 1).
2. Run $\mathcal{P}_{LE}(ck, ((1, B_i)_{i=0}^{S-1}, (-1, A_i)_{i=0}^{M-1}), ((\boldsymbol{b}_i, \boldsymbol{r}_{b,i})_{i=0}^{S-1}, (\boldsymbol{a}_i, \boldsymbol{r}_{a,i})_{i=0}^{S-1}))$ to generate CMT_{LE} based on the first 7 steps of Protocol 2.
3. Compute $C = \mathrm{Com}_{ck}(\boldsymbol{c}; \boldsymbol{r}'_c)$ and $P_j = \sum_{i=0}^{S-1} B_i - \sum_{i=0}^{M-1} A_i^{(j)} - C$ for $j \in [0, N)$ in Equation (19).
4. Set $\boldsymbol{P} = (P_0, \cdots, P_{N-1})$. Run $\mathcal{P}_{LS}(ck, \boldsymbol{P}, (l, \sum_{i=0}^{S-1} \boldsymbol{r}_{b,i} - \sum_{i=0}^{M-1} \boldsymbol{r}_{a,i}^{(l)} - \boldsymbol{r}'_c))$ steps 1 to 7 in Protocol 3 to generate CMT_{LS}.

5. For each $i \in [0, M-1]$, set $\boldsymbol{P}_i = (P_{a,i}^{(0)}, \cdots, P_{a,i}^{(N-1)})$. Run $\mathcal{P}_{LS}(rk, \boldsymbol{P}_i, (l, \boldsymbol{r}'_{a,i}))$ steps 1 to 7 in Protocol 3 to generate $\mathsf{CMT}_{LS}^{(i)}$ for all i's.
6. Additionally compute $F_{i,j} = \boldsymbol{H} \cdot \boldsymbol{\rho}_{i,j}$ for all $i \in [0, M-1]$ and $j \in [0, k)$ at step 6 of Protocol 3 (as described in Section 6.1). Set $\boldsymbol{F} = (F_{i,j})_{i=0,j=0}^{M-1,k-1}$ and $\boldsymbol{F}^* = (F_{i,j})_{i=0,j=1}^{M-1,k-1}$.
7. Compute $x = H(pp, \mathtt{CN}_{in}, \mathtt{PK}_{in}, \mathtt{SN}_{in}, \mathtt{CN}_d, \mathtt{PK}_d, \mathtt{CN}_{out}, \mathtt{PK}_{out}, \mathtt{SN}_{out}, \mathsf{CMT}_{LE}, (\mathsf{CMT}_{LS}^{(i)})_{i=0}^{M-1}, \mathsf{CMT}_{LS}, \boldsymbol{F})$.
8. Compute RSP_{LE} by running the remaining steps of $\mathcal{P}_{LE}$ (Protocol 2).
9. Compute $\mathsf{RSP}_{LS}^{(i)}$'s and RSP_{LS} by running the remaining steps of $\mathcal{P}_{LS}$ for all i's (Protocol 3).
10. Set $\pi = (\mathsf{CMT}_{LE}^*, (\mathsf{CMT}_{LS}^{*(i)})_{i=0}^{M-1}, \mathsf{CMT}_{LS}^*, \boldsymbol{F}^*, x, \mathsf{RSP}_{LE}, (\mathsf{RSP}_{LS}^{(i)})_{i=0}^{M-1}, \mathsf{RSP}_{LS})$. Return $(\mathtt{CK}_{out}, \mathtt{CNK}_{out}, \mathtt{PK}_{out}, \mathtt{SK}_{out}, \mathtt{SN}_{out}, \pi)$.

• **CheckAct**$(\mathbb{B}, \mathtt{PK}, \mathtt{CNK})$: For all $\mathsf{pk}_i \in \mathtt{PK}$ and $\mathsf{cnk}_i \in \mathtt{CNK}$, if all $(\mathsf{pk}_i, \mathsf{cnk}_i)$'s appear in $\mathbb{B}$, output 1. Otherwise, output 0.

• **IsSpent**$(\mathbb{S}, \mathtt{SN})$: For all $\mathsf{SN}_i \in \mathtt{SN}$, if any SN_i appears more than once in $\mathtt{SN}$ or appears in $\mathbb{S}$, output 1. Otherwise, output 0.

• **Verify**$(pp, \mathbb{S}, \mathtt{CN}_{in}, \mathtt{PK}_{in}, \mathtt{SN}_{in}, \mathtt{CN}_d, \mathtt{PK}_d, \mathtt{CN}_{out}, \mathtt{PK}_{out}, \mathtt{SN}_{out}, \pi)$: Compute $\zeta = H(pp, \mathtt{CN}_{in}, \mathtt{PK}_{in}, \mathtt{SN}_{in}, \mathtt{CN}_d, \mathtt{PK}_d)$ as the first challenge in the partially amortized binary proof (Protocol 1). Run **CheckAct**$(\mathbb{B}, \mathtt{PK}_{in}, \mathtt{CNK}_{in})$. If the output is 0, return 0. Run **IsSpent**$(\mathbb{S}, \mathtt{SN}_{in})$. If the output is 1, return 0. Parse $\pi = (\mathsf{CMT}_{LE}^*, (\mathsf{CMT}_{LS}^{*(i)})_{i=0}^{M-1}, \mathsf{CMT}_{LS}^*, \boldsymbol{F}^*, x, \mathsf{RSP}_{LE}, (\mathsf{RSP}_{LS}^{(i)})_{i=0}^{M-1}, \mathsf{RSP}_{LS})$. Proceed as follows:

1. Return 0 if $|\mathtt{CN}_{in}| \neq M$ or $|\mathtt{PK}_{in}| \neq M$ or $|\mathtt{SN}_{in}| \neq M$.
2. Rebuild CMT_{LE} based on the last two verifications in step 12 of Protocol 1 and the forth verification in step 17 of Protocol 2, $(\mathsf{CMT}_{LS}^{(i)})_{i=0}^{M-1}$ and CMT_{LS} based on the forth verification in step 16 of Protocol 3. Rebuild $\boldsymbol{F}$ by setting $F_{i,0} = x^k \cdot \mathsf{SN}_i - \sum_{j=1}^{k-1} x^j F_{i,j} - \boldsymbol{H} \cdot \boldsymbol{z}_{b,i}$ for all i's.
3. Return 0 if $x \neq H(pp, \mathtt{CN}_{in}, \mathtt{PK}_{in}, \mathtt{SN}_{in}, \mathtt{CN}_d, \mathtt{PK}_d, \mathtt{CN}_{out}, \mathtt{PK}_{out}, \mathtt{SN}_{out}, \mathsf{CMT}_{LE}, (\mathsf{CMT}_{LS}^{(i)})_{i=0}^{M-1}, \mathsf{CMT}_{LS}, \boldsymbol{F})$, otherwise return 1.

$\mathbb{B}$ and $\mathbb{S}$ will be updated once transactions are recorded on the blockchain.

7 Evaluation

Implementation. To evaluate the performance of the proposed proofs, we give a reference implementation[7] of our approaches in Golang. The underling polynomial ring operations are implemented with LaGo [LJC24]. To compare with MatRiCT and MatRiCT+, we evaluate the performance of our approaches (as well as MatRiCT and MatRiCT+) under settings: $\phi_1 = \phi_2 = 15$, $\mathcal{B} = 1$, $(d, w, p) = (64, 56, 8)$, $q = 2^{49} - 2^{18} + 2^9 + 1$, $(n, m) = (29, 60)$, $\widehat{q} = 2^{31} - 2^{18} + 2^3 + 1$, and $(\widehat{n}, \widehat{m}) = (18, 38)$ for MatRiCT; and $(d, w) = (256, 56)$, $q = 167770241$

[7] `https://github.com/GoldSaintEagle/RingCT_Implementation`

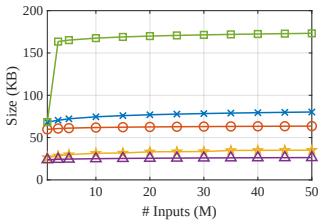

Fig. 1: Balance proof size.

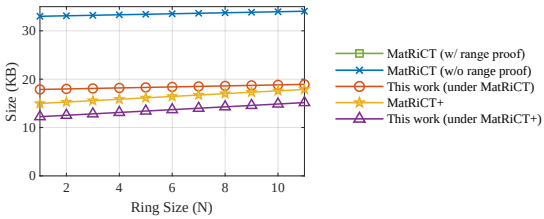

Fig. 2: Ring signature size.

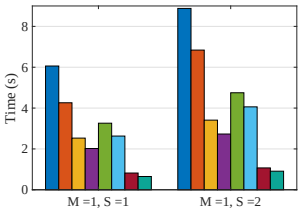

Fig. 3: Time cost of balance proofs.

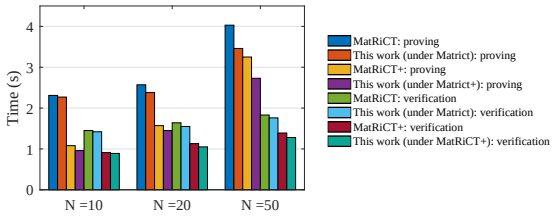

Fig. 4: Time cost of ring signatures.

($\approx 2^{27}$), $(n, \kappa) = (4, 4)$, $\widehat{q} = 2^{34} - 2^{26} - 2^7 + 1$, and $(\widehat{n}, \widehat{\kappa}) = (5, 5)$ for MatRiCT+. Note these settings suffice the 128-bit security of partially amortized binary proof when the number of outputs are small (≤ 2) in our experiments. All experiments are performed on a personal laptop equipped with Intel i7-8750H 2.20GHz CPU and 8GB memory.

Proof size: balance proof. We first evaluate the performance of our balance proof. Referring to [EZS+19, ESZ22], we consider the scenario that requires 64-bit precision for amounts (i.e., $k = 64$) and fix the number of output account ($S = 1$). The balance proof size growth with the number of input accounts is depicted in Figure 1. As we do not consider anonymity in our balance proofs, the proof size does not increase much with M except when $M = 1$. This is an expected result as M only contributes to the size of proof elements which is logarithmic to the proof size. Furthermore, there is a clear burst in MatRiCT when $M = 3$ if we use full range proofs to show the validate of corrector values. This problem can be avoided by embedding the range relation in the binary proofs of output accounts as with MatRiCT+ (MatRiCT w/o range proof in Figure 1). Nevertheless, our approach can save 15% size of MatRiCT (with range proof) when $M = 1$ and more than 50% in other cases, and nearly 20% size of MatRiCT+.

Proof size: ring signature. We further evaluate the performance of our ring signature, and compare with MatRiCT [EZS+19] and MatRiCT+ [ESZ22]. As N is relatively small in existing anonymous cryptocurrencies (e.g., $N = 11$ in Monero), we fix $k = 1$ and $\beta = N$. The signature size growth with the ring size N is depicted in Figure 2. Since we set $\beta = N$, all approaches scale linearly with the

ring size. Compared with MatRiCT, our approach reduces about 50% proof size. This is contributed by removing the binary proofs, which allows us to efficiently run under the smaller parameter set. The improvement is less significant in MatRiCT+ settings. This is mainly due to the cyclotomic rings optimization decreases the gap between different parameters, i.e., $(q, n, \kappa) \approx (2^{34}, 4, 4)$ and $(\widehat{q}, \widehat{n}, \widehat{\kappa}) \approx (2^{27}, 5, 5)$. Nevertheless, our approach can still save about 15% proof size of MatRiCT+.

Time consumption. Finally, we compare the proving and verification time of our approaches with MatRiCT [EZS+19] and MatRiCT+ [ESZ22]. The results are depicted in Figure 3. Our inner-product based approach reduces nearly 30% proving time of the MatRiCT and 20% of MatRiCT+ as we do not involve c_i's in binary proofs. Besides, since the commitment of c_i's is derived from A_i's and B_i's, our approach also reduce the time of committing to corrector values. Furthermore, our approach reduces about 20% verification time of both MatRiCT and MatRiCT+. The main reason is verifying the inner-product relation (step 11 of Protocol 2) is much more efficient than the balance relation with corrector values. Therefore, the efficiency of binary verification is also improved without corrector values.

The performances of ring signatures are depicted in Figure 4. Our unbalanced linear sum approach can reduce nearly 15% proving time of MatRiCT and MatRiCT+ when $N = 50$. This is mainly contributed by avoiding the binary proof parts in our approach. When N is small, the improvement is less significant as the binary proof cost is only a small portion of the whole cost. The improvement in verification is less significant due to the same reason. Nevertheless, our approaches outperform MatRiCT and MatRiCT+ in all settings.

8 Discussion

Security of parameters. Based on Lemma 1, we can find that the security of M-SIS increases with n whereas the security of M-LWE increases with $m - n$. Thus, we balance the security aspects to choose parameters such that $m \approx 2n$. Furthermore, MatRiCT [EZS+19] and MatRiCT+ [ESZ22] aim for the root Hermite factor of $\delta \approx 1.0045$ in Equation (2) for both M-LWE and M-SIS, which indicates 128-bit post-quantum security [APS15]. Here we show that our parameters ensure a same security level.

Prior to present the results, we show a technique used in [EZS+19] to reduce γ_{bin} by scarifying some completeness (increasing the completeness error).

In the binary proof of MatRiCT, after generating $\boldsymbol{g}_i$'s, the prover and verifier further check $\|(\boldsymbol{g}_i \circ (x \cdot \mathbf{1} - \boldsymbol{g}_i))_{i=0}^{S-1}\|$ be a factor $4d$ smaller than the theoretical bound (after steps 8 and 12 in Protocol 1, Equation (33) in the full version [GZG+21]). [EZS+19] shows these changes can significantly reduce γ_{bin} while ensuring the completeness error less than 1%. Therefore, the norm bound of $\|(\widehat{\boldsymbol{s}}; \widehat{\boldsymbol{r}}_f)\|$ becomes

$$\|(\widehat{\boldsymbol{s}}; \widehat{\boldsymbol{r}}_f)\| = 4p^2 \sqrt{d^3 w^3 \big(p^2 k^3 S^3 \phi_1^4 + \phi_2^2 \mathcal{B}^2 m^2 (S+1)\big)}. \tag{21}$$

Accordingly, γ_{bin} becomes

$$\begin{aligned}\gamma_{\text{bin}} &= \max\{B_{Bin}, B_{Amor}\} \\ &:= \max\Big\{4p^2\sqrt{d^3w^3\big(p^2k^3S^3\phi_1^4+\phi_2^2\mathcal{B}^2m^2(S+1)\big)}, \\ &\qquad 2^{\kappa+1}S\mathcal{B}p^\kappa w^{\kappa-1}md^2\phi_2\sqrt{md(S+1)}\sum_{i=0}^{S-1}(wp)^i\Big\}.\end{aligned}$$

Observe that B_{Bin} is the bound in MatRiCT (Lemma 5.5 in [EZS+19]) introduced by the binary constraint (our solution has a smaller bound in this term since we use the new encoding scheme), and B_{Amor} is introduced by our partial amortization technique. Meanwhile, $B_{Amor} < B_{Bin}$ when $S \leq 2$, which indicates our parameters satisfy Equation (2). Specifically, when $S = 2$, $2\gamma_{bin} \approx 2.86 \times 10^{14}$, which is smaller than $\min\{q, 2^{2\sqrt{nd\log q\log\delta}}\} \approx 4.10 \times 10^{14}$ in Equation (2).

Additionally, $2(2wp)^{\kappa+1}\phi_1\sqrt{wkS} \approx 2.04\times10^9$ and $8(w\phi_1)^2pkSd \approx 3.70\times10^{11}$ when $S = 2$, while $q/2 \approx 2.81\times10^{14}$, which is greater than both terms. Thus, we can safely use the our parameters for binary proofs when $S \leq 2$, which satisfies most cases for confidential transactions.

For the linear equation satisfiability, we set $\|\boldsymbol{\omega}\|_1^2 = S+M$ in γ_{LE}. When $S = 2$ and $M = 50$, we have $2\gamma_{LE} \approx 1.34\times10^9$, which is smaller than $\min\{\widehat{q}, 2^{2\sqrt{\widehat{n}d\log\widehat{q}\log\delta}}\} \approx 1.44 \times 10^9$ in Equation (2). Thus, the settings satisfy the security of our linear equation satisfiability.

Lastly, for the unbalanced linear sum proof, we have $2\gamma_{LS} \approx 6788$ when $N = 50$, which is much smaller than 1.44×10^9 (derived from the left hand size of Equation (2)). It also indicates we can use a much smaller parameter set for ring signature applications.

Compatible with other techniques. As we improve the underlying ZKPs of RingCT protocols, our approaches preserve all distinguishing features of MatRiCT, such as being compatible with extractable commitment techniques, which allows one to design an auditable RingCT protocol by placing a "mini trapdoor" in HMC (enumerating all possible values to recover the committed message).

Besides MatRiCT, other techniques in MatRiCT+ [ESZ22] to optimize the underlying cyclotomic rings can also be applied in our approaches. Specifically, a new CRT-packing technique is proposed in power-of-2 cyclotomic rings to reduce the modulus with binary CRT slots (and reduce the commitment size accordingly). Furthermore, MatRiCT+ optimizes challenges in cyclotomic rings to reduce their Hamming weights [ESZ22]. As both techniques are *"general and of independent interest for lattice-based proof systems"* [ESZ22], our approaches can regard them as optimized settings to further improve efficiency.

Discrete logarithm settings and other applications. Since our techniques do not rely on lattice settings, the results are believed to be of independent interest for RingCT protocols in a generic setting and other applications.

Partially amortized binary proof can be directly applied in discrete logarithm settings to batch $\boldsymbol{z}_{b,i}$'s regardless of S since the norm checks are no longer needed.

Partial amortization extends the amortization technique in [ACF21] which can benefit other applications with non-homomorphic constraints (e.g., reduce the cost of responses to randomness in many Σ-protocols).

Linear equation satisfiability is compatible with bit-based commitments with Equation (13) (commit to the bits of the secret instead of its value). It also has a wider application, such as in the R1CS check to build SNARKs. To prove the knowledge of $\boldsymbol{z}$ and $\boldsymbol{z}_A$ such that $A \cdot \boldsymbol{z} = \boldsymbol{z}_A$, one can run our linear equation satisfiability by setting $z_{A,i} - \sum_j A_{i,j} z_j = 0$.

The *new encoding scheme* allows a prover to ensure both $\langle \boldsymbol{d}, \boldsymbol{t} \rangle = 0$ and the encoded result are short, which can be used with smaller parameters to reduce the proof size in lattice settings as mentioned in Section 4.2.

Unbalanced linear sum proof can also be applied in discrete logarithm settings directly to improve their performance of ring signatures by removing the binary proof part. Note that under the discrete logarithm assumption, b_i's do not necessarily have to be short. Besides RingCT protocols, other ring-signature-based applications can also be benefited. For instance, in an *anonymous e-voting*, voters can use our linkable ring signature to sign their votes. The tallying center verifies the signatures to ensure validity and avoid multiple voting without knowing the identities of the voters.

First, an $O(\sqrt{N})$-size commitment scheme is proposed in [BBC+18] by encoding N-many secrets into S where $v = l = O(\sqrt{N})$. Unfortunately, when adopting this approach, the $\langle \boldsymbol{f}, \mathbf{2}^k \rangle$ in Equation (15) cannot be calculated directly as Z will "batch" some f_i's when computing $S \cdot C$. For instance, consider the first element in Z, $z_{0,0} = \sum_{i=0}^{l-1} s_{0,i} \cdot c_{i,0} + y_{0,0}$. As $f_0 = s_{0,0} \cdot c_{0,0} + y_{0,0}$, we have $2^0 \cdot z_{0,0} = 2^0 \cdot f_0 + 2^0 \cdot (\sum_{i=1}^{l-1} s_{0,i} \cdot c_{i,0}) = 2^0 \cdot f_0 + 2^0 \cdot e_0$. Therefore, it is important to allow the verifier to cancel out $\sum_{i=0}^{k-1} 2^i \cdot e_i$ without leaking any information when computing $\langle \boldsymbol{f}, \mathbf{2}^k \rangle$. The same issue occurs in ring signatures when computing $\sum_{i=0}^{\beta-1} f_{j,i}$ in Equation (17) and $\prod_{j=0}^{k-1} f_{l_j,i_j}$ in Equation (18). The latter one is a much thornier problem when using $z_{i,j}$'s to compute $\prod_{j=0}^{k-1} f_{j,i_j}$. Second, Bootle et al. show the proof size of a Σ-protocol can be reduced to $O(N^{\frac{1}{d+1}})$ with d-levelled commitments or $O(\log^2(N))$ with Bulletproofs folding [BLNS20]. Though the result is promising, we find it is hard to be applied in our approaches due to the same reasons above. Besides, the sizes of the extracted solutions (denoted by "slack" in [BLNS20]) also increase.

Open problems. Though our partial amortization technique works in most RingCT cases, directly applying it with a larger S requires larger parameters, which hinders from a wider application. It is very useful to improve the amortization technique for a larger S. Besides, our ring signature approach avoids the binary proof part based on the fact that a one-out-of-many relation is not a *necessary* condition for ring signatures. An interesting question is finding the *sufficient and necessary* condition for ring signatures, which may further avoid unnecessary parts of our linear sum proof for a better efficiency. Finally, the linear sum relation yields a "many-out-of-many" relation [Dia20]. Unlike [Dia20] which generates *many* public key index from a *single* secret l with permutations,

the linear sum relation maps b_i's to P_i's directly. Thus, logarithmic-size linear sum proofs seem promising solutions.

Acknowledgment

We gratefully acknowledge Dr. Xingye LU from the University of Hong Kong for the helpful technical discussions about lattice-based cryptography, Dr. Zuoxia Yu from the University of Wollongong and Dr. Aoning HU from the Southeast University for the discussion about ring signatures, as well as Dr. Muhammed Esgin from Monash University for the discussion of MatRiCT and pointing out some misleading parts. This research is partially supported by HK RGC GRF PolyU 15216721/Q86A, 15207522/Q93W, 15209822, and NSFC Youth 62302418/ZGJV.

References

ACF21. Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing Proofs of k-out-of-n Partial Knowledge. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2021.

APS15. Martin R Albrecht, Rachel Player, and Sam Scott. On the Concrete Hardness of Learning With Errors. In *Journal of Mathematical Cryptology*, 2015.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2018.

BBC+18. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël Del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear Lattice-based Zero-knowledge Arguments for Arithmetic Circuits. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2018.

BCC+15. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short Accountable Ring Signatures Based on DDH. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)*. Springer, 2015.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2016.

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More Efficient Commitments from Structured Lattice Assumptions. In *Proc. of the International Conference on Security and Cryptography for Networks (SCN)*. Springer, 2018.

BLNS20. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A Non-PCP Approach to Succinct Quantum-safe Zero-knowledge. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2020.

Dia20. Benjamin E Diamond. "Many-out-of-Many" Proofs with Applications to Anonymous Zether. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2020.

DPLS18. Rafaël Del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-Based Group Signatures and Zero-Knowledge Proofs of Automorphism Stability. In *Proc. of the ACM Conference on Computer & Communications Security (CCS)*. ACM, 2018.

ESLL19. Muhammed F Esgin, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2019.

ESS+19. Muhammed F Esgin, Ron Steinfeld, Amin Sakzad, Joseph K Liu, and Dongxi Liu. Short Lattice-Based One-Out-of-Many Proofs and Applications to Ring Signatures. In *Proc. of the International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2019.

ESZ22. Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. MatRiCT+: More Efficient Post-Quantum Private Blockchain Payments. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*, 2022.

EZS+19. Muhammed F Esgin, Raymond K Zhao, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol. In *Proc. of the ACM Conference on Computer & Communications Security (CCS)*. ACM, 2019.

GK15. Jens Groth and Markulf Kohlweiss. 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.

GQZ+24. Shang Gao, Chen Qian, Tianyu Zheng, Yu Guo, and Bin Xiao. Σ-Check: Compressed Σ-protocol Theory from Sum-check. *IACR Cryptology ePrint Archive, Paper 2024/1654*, 2024.

Gro16. Jens Groth. On the Size of Pairing-based Non-Interactive Arguments. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2016.

GWC19. Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive Arguments of Knowledge. *IACR Cryptology ePrint Archive, Paper 2019/953*, 2019.

GZG+21. Shang Gao, Tianyu Zheng, Yu Guo, Zhe Peng, and Bin Xiao. Lattice-based Zero-knowledge Proofs for Blockchain Confidential Transactions. *IACR Cryptology ePrint Archive, Paper 2021/1674*, 2021.

LAZ19. Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A Practical Lattice-Based (Linkable) Ring Signature. In *Proc. of the International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2019.

LJC24. Shaohua Li, Philipp Jovanovic, and ChristianMct. LAGO. `https://github.com/dedis/lago`, 2024.

LLNW16. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures without Trapdoors. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2016.

LN22. Vadim Lyubashevsky and Ngoc Khanh Nguyen. BLOOM: Bimodal Lattice One-out-of-Many Proofs and Applications. In *Proc. of the Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 95–125. Springer, 2022.

LS15. Adeline Langlois and Damien Stehlé. Worst-Case to Average-Case Reductions for Module Lattices. In *Designs, Codes and Cryptography*. Springer, 2015.

Lyu12. Vadim Lyubashevsky. Lattice Signatures Without Trapdoors. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2012.

MR09. Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

Noe15. Shen Noether. Ring Signature Confidential Transactions for Monero. *IACR Cryptology ePrint Archive, Paper 2015/1098*, 2015.

24a. Hash Team. Hcash. `https://h.cash/`, 2024.

24b. Monero Project. Monero. `https://www.getmonero.org/`, 2024.

RST01. Ronald L Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In *Proc. of the Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 2001.

SALY17. Shi-Feng Sun, Man Ho Au, Joseph K Liu, and Tsz Hon Yuen. 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.

SCG+14. Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2014.

TSS+18. Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-Quantum One-Time Linkable Ring Signature and Application to Ring Confidential Transactions in Blockchain (Lattice RingCT v1. 0). In *Proc. of the Australasian Conference on Information Security and Privacy (ACISP)*. Springer, 2018.

YAL+17. Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Lattice-Based Techniques for Accountable Anonymity: Composition of Abstract Stern's Protocols and Weak PRF with Efficient Protocols from LWR. *IACR Cryptology ePrint Archive, Paper 2017/781*, 2017.

ZGSX23. Tianyu Zheng, Shang Gao, Yubo Song, and Bin Xiao. Leaking Arbitrarily Many Secrets: Any-out-of-Many Proofs and Applications to RingCT Protocols. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2023.

ZGX25. Lizhen Zhang, Shang Gao, and Bin Xiao. Lattice-based Σ-Protocols for Polynomial Relations with Standard Soundness. *IACR Cryptology ePrint Archive, Paper 2025/313*, 2025.