MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol

Muhammed F. Esgin
Monash University & Data61, CSIRO
Australia
Muhammed.Esgin@monash.edu

Raymond K. Zhao
Monash University
Australia
Raymond.Zhao@monash.edu

Ron Steinfeld
Monash University
Australia
Ron.Steinfeld@monash.edu

Joseph K. Liu
Monash University
Australia
Joseph.Liu@monash.edu

Dongxi Liu
Data61, CSIRO
Australia
Dongxi.Liu@data61.csiro.au

ABSTRACT

We introduce MatRiCT, an efficient RingCT protocol for blockchain confidential transactions, whose security is based on "post-quantum" (module) lattice assumptions. The proof length of the protocol is around two orders of magnitude shorter than the existing post-quantum proposal, and scales efficiently to large anonymity sets, unlike the existing proposal. Further, we provide the first full implementation of a post-quantum RingCT, demonstrating the practicality of our scheme. In particular, a typical transaction can be generated in a fraction of a second and verified in about 23 ms on a standard PC. Moreover, we show how our scheme can be extended to provide auditability, where a user can select a particular authority from a set of authorities to reveal her identity. The user also has the ability to select no auditing and all these auditing options may co-exist in the same environment.

The key ingredients, introduced in this work, of MatRiCT are 1) the shortest to date scalable ring signature from standard lattice assumptions with no Gaussian sampling required, 2) a novel balance zero-knowledge proof and 3) a novel extractable commitment scheme from (module) lattices. We believe these ingredients to be of independent interest for other privacy-preserving applications such as secure e-voting. Despite allowing 64-bit precision for transaction amounts, our new balance proof, and thus our protocol, does not require a range proof on a wide range (such as 32- or 64-bit ranges), which has been a major obstacle against efficient lattice-based solutions.

Further, we provide new formal definitions for RingCT-like protocols, where the real-world blockchain setting is captured more closely. The definitions are applicable in a generic setting, and thus are believed to contribute to the development of future confidential transaction protocols in general (not only in the lattice setting).

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3354200

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Lattice, Zero-Knowledge, RingCT, Post-Quantum, Ring Signature, Group Signature

ACM Reference Format:
Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. 2019. MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3319535.3354200

1 INTRODUCTION

Zero-knowledge proofs (ZKP) are a fundamental tool used in numerous privacy-preserving applications, and they have recently become a crucial part of privacy-aware blockchain-based applications such as private/anonymous cryptocurrencies, e.g., Monero and Zcash. When coupled with *commitments*, they allow users to prove useful statements without leaking private information. For example, Monero uses the RingCT protocol [23] to realize confidential transactions. However, the currently deployed solutions in these systems do not provide post-quantum security. As stated in Zcash's FAQ page [29], the developers "plan to monitor developments in postquantum-secure components, and if/when they are mature and practical, update the Zcash protocol to use them." Therefore, there is an evident need to design quantum-secure alternatives of currently deployed privacy-preserving protocols. Our main goal in this paper is to introduce an *efficient* and *post-quantum*[1] RingCT protocol based on computational lattice problems (in particular, M-SIS and M-LWE), accompanied by a full implementation.

Overview of RingCT protocol. In a blockchain environment, the two main entities for the purposes of our work are *spenders*, who create a transaction and its proof of validity, and *verifiers*, who check the validity of a transaction and its proof. There are also *recipients* of the transactions, but they do not play a central role in RingCT system.

[1] Post-quantum security referred in this work does not necessarily involve a security proof in the quantum random oracle model (even if random oracle model is used). This is in the same spirit as in, e.g., [6–8, 14].

Table 1: Proof length comparison (in KB) of "post-quantum" RingCT proposals, supporting multiple inputs/outputs.

Anonymity level	1/10		1/100	
#inputs → #outputs	1 → 2	2 → 2	1 → 2	2 → 2
LRCT v2.0 [30]	>8000	>10000	>50000	>70000
MatRiCT: Our Work	**93**	**110**	**103**	**120**
LRCT v2.0 [30]	PK Size: 100 KB		Modulus: $\approx 2^{196}$	
MatRiCT: Our Work	PK Size: 4 KB		Modulus: $< 2^{53}$	

A RingCT protocol allows users to create *confidential transactions* on blockchain so that the spender's identity as well as the transaction amount is hidden from the outside world. Additionally, the protocol must guarantee that the transaction is a *valid* spending. This is mainly captured by a fundamental property, *balance*, which requires that the total amount being spent by the spender is exactly equal to the total amount received by the recipients where no double-spending or negative amount spending occurs.

Previous RingCT protocols, e.g. [23, 28, 30, 32], make use of three core ingredients: 1) a homomorphic commitment scheme, allowing one to hide some secret with the ability to later reveal it while ensuring that the secret committed in the first place and the opened one are the same, 2) a linkable ring signature [17, 25] (or one-out-of-many proof), allowing one to prove knowledge of a secret key corresponding to an (undisclosed) element in a set of public keys, and 3) a range proof, showing that a secret committed value falls within a certain range. In RingCT protocol, the transaction amount is hidden via the use of the commitment scheme[2], and the spender's identity is hidden via the use of the ring signature. The main purpose of the range proof is for guaranteeing the validity of a transaction by proving that the real transaction amount hidden in a commitment is in a valid (positive) range.

Monero cryptocurrency is currently the most prominent application that heavily relies on RingCT protocol to provide privacy-preserving solutions. It also uses *stealth addresses* to allow the users to hide the recipient's identity. The general idea for the stealth address is to enable the spender derive new public keys (i.e., output addresses) from a long-term public key so that the newly generated keys (still owned by the recipient) cannot be linked together.

When one is interested in post-quantum solutions, the cost of all aforementioned tools increase significantly in comparison to, e.g., discrete logarithm (DL) based schemes. Taking lattice-based solutions as promising post-quantum cryptography candidates, the lattice-based analogues of these tools each require at least an order of magnitude more storage. For example, a single lattice-based range proof on a 64-bit range in [7] costs about 100KB and also does not allow efficient aggregation of multiple proofs in the way required in a RingCT system.[3] Moreover, in this case, one is inherently required to use a modulus of size more than 64 bits to make the proof work, which means the computational efficiency will significantly degrade as the values will not fit into 64-bit registers.

[2]For the purposes of this paper, a commitment scheme is always considered to be homomorphic, thus we omit saying "homomorphic".

[3]The range proof in [7] allows efficient aggregation only when the values are committed all together in a single commitment. In RingCT, however, the amounts are committed separately as they will be transferred to distinct addresses.

Table 2: Running times (in ms) of MatRiCT at 3 GHz.

Anon. level	1/10		1/100		1/1000	
#in's → #out's	1 → 2	2 → 2	1 → 2	2 → 2	1 → 2	2 → 2
Key Gen.	2	2	2	2	2	2
Transact. Gen.	242	375	360	620	1858	3514
Verification	20	23	31	40	146	223

1.1 Our Contribution

Improved ring signature. Our first contribution in this work is to introduce the shortest ring signature to date from standard lattice assumptions, namely M-SIS and M-LWE. In particular, we introduce several improvements on the sublinear-sized ring signature in [7]. Our construction, unlike the one in [7], does not require any (discrete) Gaussian sampling, and therefore it is much easier to protect against side-channel attacks. To get an advantageous use of the uniform distribution, we introduce a new technique for the application of rejection sampling on binary secrets with fixed Hamming weight (see Section 1.3).

A novel post-quantum RingCT. Our main contribution in this work is the design of a novel RingCT protocol, named MatRiCT, that is *efficient*, *scalable* and *post-quantum*. The main technical novelties of MatRiCT are sketched in Section 1.3. As shown in Table 1, in comparison to the only existing post-quantum RingCT protocol supporting multiple inputs and outputs, we achieve a dramatic improvement in transaction size, which is the main metric in determining transaction fees. Our scheme is also very efficient in terms of computational complexity even for an anonymity set as large as 1000 as shown in Table 2.

As a bonus feature, we show in Section 6 that MatRiCT easily extends to provide *auditability* (i.e., the ability of an authority to trace real spenders) in a way that does not require significant modifications to the system. Auditability is an important feature to prevent illegal use of a cryptocurrency, and is desired, e.g., for regulatory or financial enterprise applications.

Novel extractable commitment. We introduce a novel *extractable* commitment scheme from lattices, which extends the commonly used Hashed-Message Commitment (HMC) (see Section 2.2). An extractable commitment has an additional CExtract function that allows a party to recover the message stored in a commitment using a (secret) *trapdoor*. Without knowledge of the trapdoor, however, the message remains hidden. Therefore, extractable commitments are ideal tools for privacy-preserving applications where accountability, e.g., in case of misbehaviour is desired.

The main advantage of our primitive is that it can be realized with almost the same parameters as HMC, and does not mandate very aggressive parameters. To illustrate, for an $n \times m$ commitment matrix over a ring with modulus q, the GPV trapdoor [10] (see also the improved constructions and Figure 1 in [21]) requires $m = O(n \log q)$ whereas, for the same trapdoor norm level, we only require $m = O(n)$ as in standard HMC. The extraction works when the input message space is not too large (i.e., it is feasible to iterate over all messages).

Efficient group signature for moderate-sized groups. Combination of our ring signature with the extractable commitment results in a group signature (or an accountable ring signature), which

Table 3: Comparison of signature lengths (in KB) of "post-quantum" ring/group signatures. The last column represents whether ring or group signature is supported. "?" indicates that the signature length cannot be approximated using the results of the respective reference.

Ring/Group Size N :	2	8	64	4096	2^{21}	Ring/Group
[6]	581	581	581	581	581	Group
[14]	?	?	250	456	?	Ring&Group
[7]	36	41	58	103	256	Ring
This Work	**18**	**19**	**31**	**59**	**148**	Ring
This Work	**28**	**29**	**34**	**60**	**148**	Group

shares the same efficiency features as the ring signature. The signature length of our ring/group signature is very short and compared to the state-of-the-art post-quantum proposals in Table 3. More discussion on the ring/group signature is provided in the full version.

New formal definitions for RingCT-like protocols. Further, we introduce new rigorous security definitions for RingCT-like protocols. Our goal in introducing a new set of definitions is to provide an easy-to-understand model that captures the real-world scenario more closely than the previous attempts [28, 32].

Related Work. The shortest state-of-the-art ring signature from standard lattice assumptions (in the random oracle model) is due to Esgin et al. [7], which uses the same blueprint in [8], but the underlying ZKP reaches a convincing soundness level at a single protocol iteration unlike the case in [8]. The signature length growth of the proposal in [7] is very slow, and is in particular poly-logarithmic in the ring size and quasi-linear in the security parameter. Therefore, the ring signature in [7] is a very attractive choice for scalable applications. We overview our improvements over this ring signature construction in Section 1.3.

The first RingCT protocol was introduced in [23] (called RingCT 1.0, hereafter), and more formal definitions were then provided in [28] (called RingCT 2.0, hereafter). Both of these solutions are in the DL setting and the latter requires a trusted setup, which undermines the idea of a blockchain environment where there is no particular trusted authority. Very recently, another DL-based RingCT is proposed in [32] (called RingCT 3.0, hereafter), where the security model is also improved over RingCT 2.0. The RingCT correctness and security models defined in RingCT 2.0 and 3.0 have some unsatisfactory aspects. In particular, they are (in our view) complicated to understand, and do not capture the inherent stateful nature of a blockchain system. Therefore, there are gaps in some definitions. For example, our balance definition is stronger than the one in RingCT 3.0, which requires *all* input accounts to be uncorrupted. Also, we provide an indistinguishability-based anonymity definition, unlike RingCT 3.0. We further observe that the non-slanderability definition in RingCT 2.0 and 3.0 does not seem to be a crucial property in the RingCT environment (though it may be in other applications of linkable ring signatures). A more detailed comparison of our formal definitions to the prior ones is provided in Section 3 after the introduction of our definitions.

In this work, we focus on guaranteeing the confidentiality of the transaction amount and spender anonymity (not recipient anonymity) and, in common with prior formal RingCT models [28, 32], do not capture the use of stealth addresses and the recipient anonymity in our formal security model.[4] As done in Monero, our tools can be extended to provide recipient anonymity through the use of stealth addresses, which we leave as a future work.

In the post-quantum world, an initial attempt to design a lattice-based RingCT was done by Torres et al. in [31] (called LRCT v1.0, hereafter). This protocol is restricted to the single-input single-output wallets, i.e., the user spends a single account to a single output address, and therefore the balance property is easy to satisfy. Moreover, it does not involve a range proof to make sure that the amount being spent is positive. Very recently, LRCT v1.0 is extended to support multiple-input and multiple-output wallets in [30] (called LRCT v2.0, hereafter), where a range proof is used to prove balance. However, the concrete efficiency of this scheme is far behind practical expectations (see Table 1). Two crucial advantages of MatRiCT over LRCT v2.0 are that 1) the underlying ZKPs of MatRiCT reach a convincing soundness level in a single execution, and thus no protocol repetition is required, and 2) MatRiCT does not require a 64-bit range proof even though 64-bit amounts are allowed (recall that a single 64-bit range proof from lattices alone costs about 100 KB currently).

Another project to design a post-quantum privacy-preserving cryptocurrency has been initiated in [9], where lattice-based techniques are to be used. Though there is currently no concrete scheme available, the authors mention that they aim to design a ring signature of size less than 400 KB for rings of 2^{15} users and that the range proof is expected to cost a few hundred KB. Our results are far ahead of these goals (see Tables 1 and 3).

Organization of the paper. We provide an overview of our RingCT protocol in Section 1.2, followed by a sketch of our techniques in Section 1.3. Preliminaries are gathered in Section 2, and our formal definitions for RingCT-like protocols are covered in Section 3. In Section 4, we describe MatRiCT, our lattice-based RingCT protocol, in detail and then provide rigorous security proofs in Section 5. We discuss how MatRiCT can be extended to provide auditability in Section 6, where the extractable commitment is also introduced. Due to limited space, some related technical lemmas and some lemma proofs are deferred to the full version.

1.2 Overview of MatRiCT

One of the most important features of MatRiCT is that no *wide* range proof is required. The general structure of the whole system is as follows. We work over two cyclotomic rings $R_q = \mathbb{Z}_q[X]/(X^d + 1)$ and $R_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^d + 1)$ where q is a small modulus (of about 31 bits) and $\hat{q}$ is large modulus (of about 53 bits). Note that both moduli are much smaller than 64 bits in bit-length even though we allow the transaction amount to be of 64 bits.

A user secret sk is a random short vector over R_q and the user's public key is generated by using sk as the randomness of a commitment to zero. When minting a coin to represent an amount without revealing its value, we do not commit to the integer amount, but instead commit to the bits of the amount over R_q. Therefore, a

[4] In a concurrent and independent work [15] published on IACR's ePrint archive after the submission of this work, the use of stealth addresses is included in the formal RingCT model.

standard argument stating that the sum of input coins equals the sum of output coins is not sufficient for the balance property as the addition is done over $\mathbb{Z}_q$. Instead, we introduce a new balance proof as sketched in Section 1.3.

To spend some of her accounts, each of which is a pair of a public key and a coin, a user Alice proceeds as follows. She mints her coins and computes some "corrector" values to be used in the balance proof. These corrector values help Alice prove that the sum of input amounts equals the sum of output amounts, and they are binary when there is one input account and at most two output accounts. For simplicity, let us assume that is the case for now. To hide her identity, Alice gathers other accounts to be used in a ring signature. Suppose Alice wants to spend M of her accounts while hiding herself among N users, in which case she gathers $M \cdot N$ accounts (including those of her own), seen as an $M \times N$ matrix. Then, she chooses an index $\ell \in [0, N-1]$ and places her own accounts on the ℓ-th column.

After this initial setting, Alice computes an *aggregated* binary proof over $R_{\hat{q}}$ where she proves that 1) all minted output coins are commitment to bits, 2) her index ℓ is properly encoded by some bits in some ring elements, and 3) the "corrector" values are properly encoded as bits in some ring elements. Here, our scheme crucially benefits from this efficient aggregation. Alice then provides M ring signatures for her accounts to be spent, and also proves that the "corrector" commitment is of a special form such that the commitment does not contain any value (i.e., the committed message represents zero). Finally, she runs another ring signature on commitments $P_0, \ldots, P_{N-1}$ where

$$P_i = \sum(\text{output coins}) - \sum \binom{\text{input coins in}}{i\text{-th column}} + (\text{"corrector" com.}).$$

Observe that Alice knows all the secret values that constructs P_ℓ, and the corrector commitment is constructed in a way that P_ℓ is a commitment to zero over R_q *when* the sum of input (integer) amounts equals the sum of output (integer) amounts. Recall that Alice also proves corrector commitment contains no value. Finally, she computes serial numbers as commitments to her secret keys used under a new commitment key and proves that this is indeed the case. This step is to prevent Alice from double-spending.

For auditability, we allow the auditor to extract Alice's index since she already proves that her index ℓ is properly encoded in some commitment. For this, we require an extra tool: an extractable commitment compatible with our construction (see Section 6).

An important feature of our auditable RingCT system is that users can choose a specific auditor from a set of possible auditors or can even choose to have no auditors, all within the same environment. Therefore, the user chooses an auditing option i, where $i = 0$ indicates no auditor (i.e., full anonymity) and $i > 0$ indicates auditing by the i-th authority (i.e., conditional anonymity).

1.3 Our Techniques

Improved ring signature. The ring signature in [7] consists mainly of two parts: 1) a binary proof on the signer's index, and 2) a one-out-of-many proof on the set of public keys. We show that the two verification equations in the binary proof can be batched together, which reduces the number of commitments and the randomnesses to be communicated by half and thus the binary proof's cost almost by half. The general idea for the binary proof in the works [5, 7, 8, 12] is as follows.

Suppose that we want to show b is a bit. Let $B = \mathrm{Com}(b; *)$, $A = \mathrm{Com}(a; *)$ and $f = x \cdot b + a$ for some masking value a and a challenge x where Com is homomorphic commitment.[5] Then, one verification equation shows that f is well-formed by checking $xB + A = \mathrm{Com}(f; *)$. The other equation proves that the coefficient of x^2 in the product $f \cdot (x - f) = x^2\,[b(1-b)] + x\,[a(1-2b)] - a^2$ is zero by checking $\mathrm{Com}(f \cdot (x-f); *) = xC + D$ where $C = \mathrm{Com}(a(1-2b); *)$ and $D = \mathrm{Com}(-a^2; *)$ are set by the prover. Thus, the latter verification equation proves that $b(1-b) = 0$, which implies (under certain conditions) that b is binary.

In the ring signature, we do not make use of the commitments A, B, C, D, other than for showing that f "encodes" a bit b. Therefore, the prover can set $B = \mathrm{Com}(b, a(1-2b); *)$ and $A = \mathrm{Com}(a, -a^2; *)$, and the verifier can equivalently check $xB + A = \mathrm{Com}(f, f(x-f); *)$. That is, since both verification equations are effectively linear in x, they can be batched together. This ensures again that $f = xb + a$ and that the coefficient of x^2 in the product $f \cdot (x-f)$ is zero. Now we do not need to have the commitments C, D at all, and also do not need to communicate a masked randomness for a second verification. The gain in the communication cost follows from here. This idea works both in the DL setting (and thus applies to all protocols using the proof systems from [5, 12]), and in the lattice setting as we do not exploit any special property of the commitment scheme other than the standard binding property.

Second, we show that by using two sets of *compatible* parameters for the two parts of the ring signature, one can significantly reduce the signature length. Here, it is important to choose the parameters carefully as the two parts are not completely independent. In our setting, the binary proof requires a much bigger modulus than the one-out-of-many proof. This is due to both the hardness of the underlying M-SIS problem and also to make the binary proof go through in a *ring* $R_{\hat{q}}$ with zero divisors where $b(1-b) = 0$ may not imply $b \in \{0, 1\}$ (unlike in the field $\mathbb{Z}_q$). Therefore, we use a large modulus $\hat{q}$ for the binary proof, and a small modulus q for the one-out-of-many proof (and also the other parts of the protocol). In addition to reducing the proof length, this also reduces the user public key size. Since public keys play a central role in the whole blockchain system, the overall advantage is two-fold. Our binary proof additionally has the advantage that the condition on the modulus to make the binary proof go through in the ring $R_{\hat{q}}$ is much weaker than the one in [7]. Using the soundness proof in [7], one would need to set $\hat{q}$ to be of more than 70 bits.

Efficient rejection sampling for a secret vector of fixed Hamming weight. Recall that the prover's binary secrets are encoded as $f = x \cdot b + a$ where b is a secret bit, $a \in R_{\hat{q}}$ is a masking element and $x \in R_{\hat{q}}$ is a challenge received (computed) after a is sampled. For our commitment scheme to be binding, f needs to be of small norm and thus we cannot choose a randomly from $R_{\hat{q}}$. In this case, a standard technique to make sure that f does not reveal information about the secret is using rejection sampling [18]. Suppose that we sample $a \leftarrow \{-\mathcal{B}_a, \ldots, \mathcal{B}_a\}^d$ and $\|x \cdot b\|_\infty \le p$ for

[5] In the real proof, multiple binary proofs are batched by committing to all the bits b_i's together as $B = \mathrm{Com}(b_0, b_1, \ldots; *)$, which we ignore here for simplicity.

all possible x and b values where $\mathcal{B}_a \gg p \in \mathbb{Z}^+$. The idea for the rejection sampling in [18] is to make the distribution of f uniform in a box by aborting the interactive protocol (or starting over in the non-interactive case) if the maximum absolute coefficient of f is greater than $\mathcal{B}_a - p$.

Now, when $b = 0$, we know independent of the challenge x that f will be equal to a. Therefore, in this case, one may sample a directly from $\{-(\mathcal{B}_a - p), \ldots, \mathcal{B}_a - p\}^d$ in the first place to make sure that $f = x \cdot b + a$ is not rejected. Still, the distribution of f conditioned on passing the rejection sampling check is identical to the uniform distribution on $\{-(\mathcal{B}_a - p), \ldots, \mathcal{B}_a - p\}^d$, thus simulation-based security aspects remain untouched. However, the number of zero secrets affects the overall acceptance probability and thus such a rejection sampling leaks side-channel information. For example, proving knowledge of secret bits 1, 1, 1, 1, 1 is likely to take longer than proving knowledge of secret bits 0, 0, 0, 0, 0 as the latter is never rejected while the former is rejected with some non-negligible probability.

In our protocol, the user index is represented in unary, i.e., the bit sequence representing the user index has a *fixed* number of zeros and ones. Therefore, the above technique of sampling the masking value from the accepted distribution in advance does not leak additional information as the prover's goal is to prove that there are exactly k ones in the secret bit sequence for some publicly known $k \in \mathbb{Z}^+$. Hence, the technique is applicable and allows us to increase the acceptance rate significantly without needing to sample these components from a wider interval. To illustrate, when $N = 200$, the bit sequence representing the user index ℓ is the ℓ-th unit vector, i.e., has 199 zeros and a single one. Therefore, if the acceptance probability for a single secret bit is P, then the overall acceptance probability using our technique is still P instead of P^{200}, which would be the case using the previous standard technique.

We also note that the technique trivially extends to the case where the secret sequence has a fixed number of zeros and some other elements (which may not be binary) as we do not make use of the fact that nonzero secrets are equal to 1.

Novel balance proof. Suppose we want to prove that

$$\sum_{i=0}^{M-1} a_{\text{in},i} = \sum_{i=0}^{S-1} a_{\text{out},i} \tag{1}$$

for some input amounts $a_{\text{in},0}, \ldots, a_{\text{in},M-1}$ and output amounts $a_{\text{out},0}, \ldots, a_{\text{out},S-1}$ where $M, S \in \mathbb{Z}^+$. The general idea to prove (1) while hiding the amounts is to commit to each amount value using a homomorphic commitment scheme, and then show that 1) each committed value is in a valid positive range, and 2) the sum of output commitments minus the sum of the input commitments is a commitment to zero. For the lattice-based schemes, there does not exist a range proof that is significantly shorter than the generic approach: first, prove that some masked values encode bits, and then that these bits construct the committed integer. One important detail that especially has an effect for the lattice-based schemes is that (1) must hold over $\mathbb{Z}$, not just $\mathbb{Z}_q$.

Now, let us see how our balance proof works. Assume we want to work in base $\beta \geq 2$ and the amounts are represented by r digits. Then, we can write $a = \sum_{j=0}^{r-1} \beta^j a[j]$ for any amount a with the digits $a[j]$'s. Hence, we get

$$\begin{aligned}
&\sum_{i=0}^{M-1} a_{\text{in},i} = \sum_{i=0}^{S-1} a_{\text{out},i} \iff \sum_{i=0}^{M-1}\sum_{j=0}^{r-1} \beta^j a_{\text{in},i}[j] = \sum_{i=0}^{S-1}\sum_{j=0}^{r-1} \beta^j a_{\text{out},i}[j], \\
&\iff \sum_{j=0}^{r-1} \beta^j \sum_{i=0}^{M-1} a_{\text{in},i}[j] = \sum_{j=0}^{r-1} \beta^j \sum_{i=0}^{S-1} a_{\text{out},i}[j], \\
&\iff 0 = \sum_{j=0}^{r-1} \beta^j \left(\sum_{i=0}^{S-1} a_{\text{out},i}[j] - \sum_{i=0}^{M-1} a_{\text{in},i}[j] \right), \qquad (2) \\
&\iff 0 = \sum_{j=0}^{r-1} \beta^j \left(\sum_{i=0}^{S-1} a_{\text{out},i}[j] - \sum_{i=0}^{M-1} a_{\text{in},i}[j] + c_j - \beta c_{j+1} \right), \qquad (3)
\end{aligned}$$

for $c_0 = c_r = 0$ and any "corrector values" $c_1, \ldots, c_{r-1} \in \mathbb{Z}$. Therefore, instead of using the general idea that mandates a very large modulus, we can proceed as follows. Setting $\beta = 2$, for each amount, we commit to its bits as

$$\begin{aligned}
C_{\text{in},i} &= \text{Com}(a_{\text{in},i}[0], \ldots, a_{\text{in},i}[r-1]; *), \\
C_{\text{out},i} &= \text{Com}(a_{\text{out},i}[0], \ldots, a_{\text{out},i}[r-1]; *).
\end{aligned}$$

Then, we also create a "corrector" commitment $C = \text{Com}(c_0 - 2c_1, \ldots, c_{r-1} - 2c_r; *)$ with $c_0 = c_r = 0$. Finally, we prove that 1) $C_{\text{in},i}$'s and $C_{\text{out},i}$'s are commitments to bits, 2) $\sum_{i=0}^{S-1} C_{\text{out},i} - \sum_{i=0}^{M-1} C_{\text{in},i} + C$ is a commitment to zero, and 3) C is well-formed as above. These guarantee that 1) the opening message for any commitment to an amount represents a unique value in a positive range $[0, 2^r - 1]$, 2)

$$0 = \sum_{i=0}^{S-1} a_{\text{out},i}[j] - \sum_{i=0}^{M-1} a_{\text{in},i}[j] + c_j - 2c_{j+1} \tag{4}$$

for any $j \in \{0, \ldots, r-1\}$, and 3) C does not add any value in this representation. Therefore, we prove (3) and equivalently (1). Since a range proof already decomposes a value to its bits, from a practical perspective, we replace the reconstruction part of the range proof by the proof that shows C is well-formed. Importantly, though, we do not need to use a very large modulus since the modulus just needs to be large enough to guarantee that (4) holds over $\mathbb{Z}$, which is a very weak condition in a practical RingCT system.

The reason why we cannot simply use (2) is that, when amounts are represented by commitments to their bits, the addition of the commitments adds the corresponding bits over $\mathbb{Z}_q$ where $q \gg 2$. Therefore, for $\text{Bits}(a)$ denoting the bits of a positive integer a,

$$\text{Com}(\text{Bits}(a_1); *) + \text{Com}(\text{Bits}(a_2); *) \neq \text{Com}(\text{Bits}(a_1 + a_2); *),$$

and hence the proof does not work without the corrector C.

New extractable commitment. Our extractable commitment can be seen as a bridge between an LWE-based encryption, and a SIS-based commitment scheme with a "full trapdoor", i.e., the commitment matrix $\boldsymbol{A}$ is constructed in a way that a trusted party knows a matrix $\boldsymbol{G}$ such that $\boldsymbol{G} \cdot \boldsymbol{A} = \boldsymbol{0} \bmod q$. The disadvantage of an encryption scheme is that it does not allow compression (since there is unique decryption). As a result, it is inefficient to encrypt long messages whereas we want to have a compact commitment to long message vectors, i.e., we require a compressing commitment. The most promising candidate for this task is HMC, which can be seen as a commitment to the hash of the message while still preserving the algebraic structure. Now, if one puts a full trapdoor to HMC,

then the recovered information via annihilating the randomness part with the trapdoor would be only the hash of the message, not the message itself. Then, one still has to recover the original message from here. Further, putting a full trapdoor often requires more aggressive parameters than those sufficient for the system without the trapdoor.

These bring us to our idea of using a "mini trapdoor". Suppose that $C = \boldsymbol{A}\boldsymbol{r} + \boldsymbol{B}\boldsymbol{m}$ is a commitment to a message vector $\boldsymbol{m}$ with a randomness vector $\boldsymbol{r}$ and a uniformly random matrix $\boldsymbol{B}$. The idea now works as follows. We construct $\boldsymbol{A}$ as an LWE matrix such that $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}' \\ \boldsymbol{t}^\top \end{bmatrix}$ where $\boldsymbol{t} = \boldsymbol{A}'^\top \boldsymbol{s} + \boldsymbol{e}$ for some secret $\boldsymbol{s}$ and error $\boldsymbol{e}$ known only to the message extractor and $\boldsymbol{A}'$ is uniformly random. To extract a message from C, the extractor computes $\langle (\boldsymbol{s}, -1), C \rangle$, which is equal to $-\langle \boldsymbol{e}, \boldsymbol{r} \rangle + \langle \boldsymbol{b}, \boldsymbol{m} \rangle$ for $\boldsymbol{b} = (\boldsymbol{s}, -1)^\top \boldsymbol{B}$. Then, the idea is to let the extractor iterate through all the possible messages $\boldsymbol{m}'$ and compute $\langle (\boldsymbol{s}, -1), C \rangle - \langle \boldsymbol{b}, \boldsymbol{m}' \rangle$. For the correct message, the result will be $e' = -\langle \boldsymbol{e}, \boldsymbol{r} \rangle$, and for an incorrect message, it will be a random element in the ring R_q since $\boldsymbol{B}$ is an independent uniformly random matrix and $\boldsymbol{m} - \boldsymbol{m}' \neq \boldsymbol{0}$. Therefore, we can set the parameters so that $\| \langle (\boldsymbol{s}, -1), C \rangle - \langle \boldsymbol{b}, \boldsymbol{m}' \rangle \|_\infty$ is small only for the correct message with an overwhelming probability, which allows the extractor to recover the message. Furthermore, from M-LWE problem, $\boldsymbol{A}$ is computationally indistinguishable from random, and thus hiding property of HMC can still be used.

2 PRELIMINARIES

We denote the ring of integers modulo q by $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, which is represented by the range $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$ for an odd integer q. 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. $(\boldsymbol{a}, \boldsymbol{b})$ denotes combining the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ to form a single longer vector. For a vector $\boldsymbol{v} = (v_0, \ldots, v_{n-1})$, the norms are defined as $\|\boldsymbol{v}\| = \sqrt{\sum_{i=0}^{n-1} v_i^2}$, $\|\boldsymbol{v}\|_\infty = \max_i |v_i|$ and $\|\boldsymbol{v}\|_1 = \sum_{i=0}^{n-1} |v_i|$. The corresponding norms for a polynomial f are defined analogously on the coefficient vector of f. For a vector $\boldsymbol{f} = (f_0, \ldots, f_{s-1})$ of polynomials, $\|\boldsymbol{f}\| = \sqrt{\sum_{i=0}^{s-1} \|f_i\|^2}$, $\|\boldsymbol{f}\|_1 = \sum_{i=0}^{s-1} \|f_i\|_1$, $\|\boldsymbol{f}\|_\infty = \max_i \|f_i\|_\infty$. Uniform distribution on a set S is denoted by $\mathcal{U}(S)$. We use $a \leftarrow \mathcal{S}$ to denote sampling a from a distribution $\mathcal{S}$, or uniformly sampling from a set $\mathcal{S}$. $[a, b] = \{a, \ldots, b\}$, $[a, b) = \{a, \ldots, b-1\}$ for $a < b \in \mathbb{Z}$ and logarithms are base 2 unless specified otherwise.

We denote a commitment by a capital letter such as C (even though it is mostly a vector). We write $\mathcal{S}^{d \cdot m}$ to indicate that a total of md coefficients from a set $\mathcal{S}$ are sampled to generate m polynomials in $R = \mathbb{Z}[X]/(X^d + 1)$ of degree d. $\mathfrak{U}_\mathcal{B}$ denotes the set of polynomials in R with infinity norm at most $\mathcal{B} \in \mathbb{Z}^+$. $\mathsf{HW}(\boldsymbol{f})$ denotes the Hamming weight of the (whole) coefficient vector of $\boldsymbol{f}$. We use $S[i]$ to denote either the i-th element of an ordered set S or the i-th bit of an integer S. If an algorithm F is run on all the elements of a set S such that $o_i \leftarrow F(s_i)$ for all $s_i \in S$, we write $O \leftarrow F(S)$ where $O = \{o_0, o_1, \ldots\}$.

2.1 Module-SIS and Module-LWE problems

We define below the hard computational problems, Module-SIS (M-SIS) and Module-LWE (M-LWE) [16].

Definition 2.1 (M-SIS$_{n,m,q,\beta_{\mathrm{SIS}}}$). Given $\boldsymbol{A} \leftarrow R_q^{n \times m}$, the problem asks to find $\boldsymbol{z} \in R_q^m$ such that $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0}$ over R_q and $0 < \|\boldsymbol{z}\| \leq \beta_{\mathrm{SIS}}$.

We define M-LWE such that both the secret and the randomness are sampled from $\mathfrak{U}_\mathcal{B}$. This variant is commonly used in practical lattice schemes, e.g., [2, 6–8].

Definition 2.2 (M-LWE$_{n,m,q,\mathcal{B}}$). Let $\boldsymbol{s} \leftarrow \mathfrak{U}_\mathcal{B}^n$ be a secret key. Define $\mathsf{LWE}_{q,\boldsymbol{s}}$ as the distribution obtained by outputting $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e)$ for $\boldsymbol{a} \leftarrow R_q^n$, $e \leftarrow \mathfrak{U}_\mathcal{B}$. The problem asks to distinguish between m given samples from either $\mathsf{LWE}_{q,\boldsymbol{s}}$ or $\mathcal{U}(R_q^n, R_q)$.

2.2 Commitment Scheme

Hashed-Message Commitment (HMC) allows commitment to long message vectors without significantly increasing the commitment size (see, e.g., [2, 8]). Suppose that we commit to v-dimensional vectors over R_q for $v \geq 1$. For positive integers $n, m, \mathcal{B}, q$ with $m > n$, the instantiation of the commitment scheme is as follows.

- $\mathsf{CKeygen}(1^\lambda)$: Sample $\boldsymbol{A} \leftarrow R_q^{n \times m}$ and $\boldsymbol{B} \leftarrow R_q^{n \times v}$. Output $ck = \boldsymbol{G} = [\, \boldsymbol{A} \,\|\, \boldsymbol{B} \,] \in R_q^{n \times (m+v)}$.
- $\mathsf{Commit}_{ck}(\boldsymbol{m})$: Sample $\boldsymbol{r} \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$. Output
$$\mathrm{Com}_{ck}(\boldsymbol{m}; \boldsymbol{r}) = \boldsymbol{G} \cdot (\boldsymbol{r}, \boldsymbol{m}) = \boldsymbol{A} \cdot \boldsymbol{r} + \boldsymbol{B} \cdot \boldsymbol{m}.$$
- $\mathsf{COpen}_{ck}(C, (y, \boldsymbol{m}', \boldsymbol{r}'))$: If $\mathrm{Com}_{ck}(\boldsymbol{m}'; \boldsymbol{r}') = yC$ and $\|(\boldsymbol{r}', \boldsymbol{m}')\| \leq \gamma_{\mathrm{com}}$, return 1. Otherwise, return 0.

In common with similar lattice-based schemes, e.g., [2, 4, 6], the opening algorithm COpen does not simply check if the message-randomness pair commits to C, but rather checks whether $yC = \mathrm{Com}_{ck}(\boldsymbol{m}'; \boldsymbol{r}')$ for some *relaxation factor* $y \in R_q$. This is required for efficient lattice-based zero-knowledge proofs and the commitment scheme is still hiding and binding as below. It is also easy to see that the commitment scheme is additively homomorphic.

Lemma 2.3. *HMC defined above is*

- *computationally hiding if M-LWE$_{m-n,m,q,\mathcal{B}}$ problem is hard, and*
- *computationally strong γ_{com}-binding with respect to the same relaxation factor y if M-SIS$_{n,m+v,q,2\gamma_{\mathrm{com}}}$ is hard.*

In [7, 8], a variant of Lemma 2.3 has been shown to hold when the matrix $\boldsymbol{G}$ is in "Hermite normal form", i.e., the first $n \times n$ part of $\boldsymbol{G}$ is the identity matrix. The result extends in a similar way to our case and we discuss more on this in the full version.

3 FORMAL DEFINITIONS FOR RINGCT-LIKE CRYPTOCURRENCY PROTOCOLS

In this section, we describe our formal definitions for RingCT-like protocols. First, we introduce the notation used specifically for the security model in Table 4. The blockchain state $\mathbb{S}$ consists of two lists: 1) a list of registered accounts $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$, indicating a public key pk is paired with a coin cn, and 2) a list of all verified transactions. We assume that $\mathbb{S}$ is properly updated among all users at all times.[6] The following tuple of polynomial time algorithms define RingCT protocol.

[6] In practice, this is managed by a consensus algorithm, which is outside the scope of this work.

Table 4: Notations for the RingCT formal model.

$\mathbb{S}$	the blockchain state
act = (pk, cn)	an account comprised of a public key and a coin
$M, S \geq 1$	# of spender's input and output accounts, resp.
$N \geq 2$	# of accounts to hide a single input account
R_{in}	set of spender's real accounts
$\mathsf{K}_{\mathsf{in}} = (\mathsf{SK}_{\mathsf{in}}, \mathsf{CK}_{\mathsf{in}}, \mathsf{Amt}_{\mathsf{in}})$	set of spender's account secret keys ask = (sk, cnk, amt) with a secret key, coin key & amount
A_{in}	set of all input accounts arranged as a $M \times N$ matrix where the i-th row contains $\mathsf{R}_{\mathsf{in}}[i]$
$\mathsf{PK}_{\mathsf{out}}$	set of output public keys with $\lvert\mathsf{PK}_{\mathsf{out}}\rvert = S$
$\mathsf{CN}_{\mathsf{out}}$	set of output coins with $\lvert\mathsf{PK}_{\mathsf{out}}\rvert = S$
$\mathsf{Amt}_{\mathsf{out}}$	set of output amounts with $\lvert\mathsf{Amt}_{\mathsf{out}}\rvert = S$
$\mathsf{CK}_{\mathsf{out}}$	set of output coin keys with $\lvert\mathsf{CK}_{\mathsf{out}}\rvert = S$
$\mathsf{A}_{\mathsf{out}}$	set of output accounts with $\lvert\mathsf{A}_{\mathsf{out}}\rvert = S$
Π	the proof output
SN	set of serial numbers
tx	a transaction $\mathsf{tx} = (\mathsf{A}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{CN}_{\mathsf{out}}, \Pi, \mathsf{SN})$
V	set of all valid amounts

- $pp \leftarrow$ **Setup**(1^λ) : given the security parameter λ, output the system parameters pp, which is assumed to be an implicit input to all the remaining functions.
- (pk, sk) $\leftarrow$ **KeyGen**() : output a public-secret key pair (pk, sk).
- s $\leftarrow$ **SerialGen**(sk) : on input a secret key sk, output a serial number s associated to sk.
- (cn, cnk)/$\perp \leftarrow$ **Mint**(amt) : on input an amount amt, if amt $\in$ V, output a coin cn and its coin key cnk. Otherwise, output $\perp$. If cnk is given as an input, then **Mint** computes a deterministic function such that cn = **Mint**(amt, cnk).
- (act, $\mathbb{S}$) $\leftarrow$ **AccountGen**(pk, cn, $\mathbb{S}$) : on input a public key pk and a coin cn, register an account act = (pk, cn) to the blockchain state $\mathbb{S}$. Output act and updated state $\mathbb{S}$.
- 0/1 $\leftarrow$ **CheckAct**(pk, cn, $\mathbb{S}$) : on input a public key pk, a coin cn and the blockchain state $\mathbb{S}$, output 1 if (pk, cn) is a registered account in $\mathbb{S}$. Otherwise, output 0. In the case that the input has a set of pairs of (pk, cn), then output 1 if all (pk, cn) pairs are registered accounts in $\mathbb{S}$. Otherwise, output 0.
- (tx, $\mathsf{CK}_{\mathsf{out}}$) $\leftarrow$ **Spend**($\mathsf{A}_{\mathsf{in}}, \mathsf{R}_{\mathsf{in}}, \mathsf{K}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{Amt}_{\mathsf{out}}$) : on input A_{in}, R_{in}, K_{in}, $\mathsf{PK}_{\mathsf{out}}$, $\mathsf{Amt}_{\mathsf{out}}$ as in Table 4, mint output coins by running $(\mathsf{CN}_{\mathsf{out}}, \mathsf{CK}_{\mathsf{out}}) \leftarrow$ **Mint**$(\mathsf{Amt}_{\mathsf{out}})$. Generate the serial numbers by running SN $\leftarrow$ **SerialGen**$(\mathsf{SK}_{\mathsf{in}})$ and a proof Π. Output $(\mathsf{tx}, \mathsf{CK}_{\mathsf{out}}) = ((\mathsf{A}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{CN}_{\mathsf{out}}, \Pi, \mathsf{SN}), \mathsf{CK}_{\mathsf{out}})$.[7]
- 0/1 $\leftarrow$ **IsSpent**(SN, $\mathbb{S}$) : on input a set SN of serial numbers and the blockchain state $\mathbb{S}$, if there is a collision in SN or if a serial number appears both in SN and $\mathbb{S}$, output 1. Otherwise, output 0.
- $\emptyset/(\mathsf{A}_{\mathsf{out}}, \mathbb{S}) \leftarrow$ **Verify**(tx, $\mathbb{S}$) : on input a transaction tx as in Table 4, if **IsSpent**(SN, $\mathbb{S}$) = 1 or **CheckAct**($\mathsf{A}_{\mathsf{in}}, \mathbb{S}$) = 0, output $\emptyset$. Check the proof Π and output $\emptyset$ if not valid. Otherwise, run $(\mathsf{A}_{\mathsf{out}}, \mathbb{S}) \leftarrow$ **AccountGen**$(\mathsf{PK}_{\mathsf{out}}, \mathsf{CN}_{\mathsf{out}}, \mathbb{S})$ and add tx to $\mathbb{S}$. Output $\mathsf{A}_{\mathsf{out}} \neq \emptyset$ and updated $\mathbb{S}$.

One of the most important differences of our definitions to RingCT 2.0 and 3.0 is that some of the functions take the blockchain state $\mathbb{S}$ as an input to capture the inherent stateful nature of a blockchain environment. However, this important piece is completely missing in RingCT 2.0 and 3.0, and sometimes used implicitly in the definitions without having it as an input. For the tuple of algorithms that define the protocol, we additionally have the functions **SerialGen**, **CheckAct** and **IsSpent**, which do not exist in RingCT 2.0 or 3.0. Therefore, in the correctness definitions of RingCT 2.0 and 3.0, there is no restriction on input accounts being unspent whereas there should be such a restriction (see our correctness definition further below).

We consider an account as a registered public key and coin pair on blockchain. Therefore, our **Spend** algorithm does not output *accounts* as the transaction would not have been validated at that point yet. Hence, **Verify** takes public key and coin pairs as inputs, and outputs the accounts if the input transaction is valid. On the other hand, **Spend** algorithms in RingCT 2.0 and 3.0 directly output *accounts*. Also, **Mint** algorithm in RingCT 2.0 and 3.0 take a public key as an input, but does not make use of it.

[7] $\mathsf{CK}_{\mathsf{out}}$ along with the output amounts are delivered to the recipient(s) privately.

Table 5: Structure of the list $\mathcal{L}$ used in the security model.

$\mathcal{L}$:	pk	sk	s (serial #)	cn	cnk	amt	IsCrpt

3.1 Security Definitions

Towards getting a "cleaner" model, we only use the single list $\mathcal{L}$ in Table 5 instead of five lists as in RingCT 2.0 and 3.0. The list $\mathcal{L}$ is seen as a database for which any of the following can be used as a unique identifier of a row: a public key, a secret key, a serial number, a coin or a coin key. Retrieving a row in $\mathcal{L}$ is denoted, for example, by $\mathcal{L}$[pk] for some public key pk. Then, $\mathcal{L}$[pk].cnk denotes the coin key associated with the public key pk. IsCrpt denotes the "is corrupted" tag.

Oracles. The oracles accessed by an adversary $\mathcal{A}$ are defined below.

- PkGen(i) : on the i-th query, run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow$ **KeyGen**(), $s \leftarrow$ **SerialGen**(sk_i) and output pk_i. Add $(\mathsf{pk}_i, \mathsf{sk}_i, s)$ to $\mathcal{L}$ where IsCrpt tag is set to zero and the remaining fields are left empty.
- Mint(amt) : run (cn, cnk) $\leftarrow$ **Mint**(amt), and output cn.
- ActGen(pk, amt, $\mathbb{S}$) : run (cn, cnk) $\leftarrow$ **Mint**(amt) and (act, $\mathbb{S}$) $\leftarrow$ **AccountGen**(pk, cn, $\mathbb{S}$). Insert (cn, cnk, amt) to $\mathcal{L}$[pk] and output (act, $\mathbb{S}$).
- Corrupt(act) : For act = (pk, cn), if $\mathcal{L}$[pk] cannot be found, return $\perp$, indicating failure. Otherwise, update $\mathcal{L}$[pk].IsCrpt to 1, and output $\mathcal{L}$[pk].sk, $\mathcal{L}$[pk].cnk and $\mathcal{L}$[pk].amt. Alternatively, the input may be either pk alone or cn alone. In the former case, only $\mathcal{L}$[pk].sk is returned, and in the latter, $\mathcal{L}$[cn].cnk and $\mathcal{L}$[cn].amt are returned.
- Spend($\mathsf{A}_{\mathsf{in}}, \mathsf{R}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{Amt}_{\mathsf{out}}$) : Retrieve from $\mathcal{L}$ all account secret keys K_{in} associated to R_{in}. Run $(\mathsf{tx}, \mathsf{CK}_{\mathsf{out}}) \leftarrow$ **Spend**$(\mathsf{A}_{\mathsf{in}}, \mathsf{R}_{\mathsf{in}}, \mathsf{K}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{Amt}_{\mathsf{out}})$ and $B \leftarrow$ **Verify**(tx, $\mathbb{S}$). If $B = \emptyset$ (i.e., the verification fails), return $\perp$. Otherwise, return tx and, for each $1 \leq i \leq \lvert\mathsf{PK}_{\mathsf{out}}\rvert$, update the coin, coin key and amount information in $\mathcal{L}[\mathsf{PK}_{\mathsf{out}}[i]]$ with $\mathsf{CN}_{\mathsf{out}}[i]$, $\mathsf{CK}_{\mathsf{out}}[i]$ and $\mathsf{Amt}_{\mathsf{out}}[i]$, respectively.

We let Orc denote the set of all oracles defined above together with the random oracle. With respect to the positioning of the

accounts in R_{in} inside A_{in}, we define two flavours of properties for RingCT: 1) *with shuffling* and 2) *without shuffling*. In the latter case, all the accounts in R_{in} are restricted to be in the same column, which provides a somewhat weaker level of anonymity as described in [32]. We give our definitions for the former case, and it is trivial to get the latter by imposing the aforementioned restriction on R_{in}.

Correctness. Informally, correctness requires that any user is able to spend any of her honestly generated *unspent* accounts, which has honestly generated keys and coins with a valid amount.

A RingCT protocol is said to be ϵ-*correct* if the following holds for any $pp \leftarrow \mathbf{Setup}(1^\lambda)$, any $M, N, S \in \mathbb{Z}^+$, $(\mathsf{pk}_0, \mathsf{sk}_0), \ldots, (\mathsf{pk}_{M-1}, \mathsf{sk}_{M-1}) \leftarrow \mathbf{KeyGen}(pp)$ such that $\mathbf{IsSpent}(\mathbf{SerialGen}(\mathsf{sk}_i)) = 0$ for all $i = 0, \ldots, M-1$, any $\mathsf{amt}_0, \ldots, \mathsf{amt}_{M-1}, \mathsf{amt}_{\text{out},0}, \ldots, \mathsf{amt}_{\text{out},S-1} \in \mathsf{V}$ such that $\sum_{i=0}^{M-1} \mathsf{amt}_i = \sum_{i=0}^{S-1} \mathsf{amt}_{\text{out},i}$, any set PK_{out} of arbitrarily generated output public keys and any set $\mathsf{A}_{\text{in}} \setminus \mathsf{R}_{\text{in}}$ of arbitrarily generated decoy accounts,

$$\Pr\left[\begin{array}{c}\mathbf{Verify}(\mathsf{tx}, \mathbb{S}) \neq \emptyset \ : \\ (\mathsf{tx}, \mathsf{CK}_{\text{out}}) \leftarrow \mathbf{Spend}(\mathsf{A}_{\text{in}}, \mathsf{R}_{\text{in}}, \mathsf{K}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})\end{array}\right] \geq 1 - \epsilon$$

where $\mathsf{cn}_i = \mathbf{Mint}(\mathsf{amt}_i, \mathsf{cnk}_i)$ for some cnk_i's in the domain of coin keys, $\mathsf{act}_i \leftarrow \mathbf{AccountGen}(\mathsf{pk}_i, \mathsf{cn}_i)$ for $i = 0, \ldots, M-1$, $\mathsf{R}_{\text{in}} = \{\mathsf{act}_0, \ldots, \mathsf{act}_{M-1}\}$, $\mathsf{Amt}_{\text{out}} = \{\mathsf{amt}_{\text{out},0}, \ldots, \mathsf{amt}_{\text{out},S-1}\}$, A_{in} and tx are as in Table 4, and $\mathsf{K}_{\text{in}} = \{(\mathsf{sk}_0, \mathsf{cnk}_0, \mathsf{amt}_0), \ldots, (\mathsf{sk}_{M-1}, \mathsf{cnk}_{M-1}, \mathsf{amt}_{M-1})\}$. If $\epsilon = 0$, then the protocol is said to be *perfectly correct*. If $\epsilon = \text{negl}(\lambda)$, then it is said to be *statistically correct*.

Observe from the above correctness definition that the spent input coins may not be generated honestly, but the input amounts and coin keys are in the correct domains. Indeed, the input coins spent by a user, say Alice, are the output coins of a previous transaction and thus are generated by another user, say Bob. Therefore, Alice cannot guarantee that Bob generated the coins honestly. However, Alice also receives the coin keys and amounts for these coins and can easily check if they are in the correct domains and whether the coins can be spent. Therefore, the correctness property alone does not guarantee that any received coins can be spent. This aspect can be captured in a *security* property (such as availability), which could require any coin output by a *verified* transaction to be "spendable". As in prior models RingCT 2.0 and 3.0, our model does not include such an availability property.

In the correctness definition of RingCT 3.0, the amounts are randomly sampled from $\mathbb{Z}_p$. However, in our case, the correctness requires any amount in the valid range V to be able to be spent.

Anonymity. Informally, anonymity requires that the real spender's accounts are hidden among the uncorrupted (i.e., never been queried to Corrupt) accounts as long as there are at least two sets of uncorrupted input accounts that can be successfully spent.

A RingCT protocol is said to be *anonymous* if the following holds for all PPT adversaries $\mathcal{A}$ and $pp \leftarrow \mathbf{Setup}(1^\lambda)$

$$\Pr[\mathcal{A} \text{ wins the game } \texttt{Exp:Anonymity}] \leq 1/2 + \text{negl}(\lambda),$$

where `Exp:Anonymity` is defined as follows.

(1) $(\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}}, \mathsf{R}^0_{\text{in}}, \mathsf{R}^1_{\text{in}}, \mathsf{st}) \leftarrow \mathcal{A}^{\text{Orc}}(pp)$: $\mathcal{A}$ is given pp and access to all oracles, and then outputs two target sets of accounts to be spent as $(\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}}, \mathsf{R}^0_{\text{in}}, \mathsf{R}^1_{\text{in}}, \mathsf{st})$ where
- st is some state information to be used by $\mathcal{A}$ in the next stage,
- $\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}$ and $\mathsf{Amt}_{\text{out}}$ are as in Table 4,
- $\mathsf{R}^0_{\text{in}}, \mathsf{R}^1_{\text{in}} \subset \mathsf{A}_{\text{in}}$ such that both R^0_{in} and R^1_{in} contain exactly one account from each row of A_{in}.

(2) $(\mathsf{tx}_i, \mathsf{CK}^i_{\text{out}}) \leftarrow \mathbf{Spend}(\mathsf{A}_{\text{in}}, \mathsf{R}^i_{\text{in}}, \mathsf{K}^i_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$ for $i = 0, 1$: Both sets R^0_{in} and R^1_{in} of real input accounts are spent with the arguments specified by $\mathcal{A}$ where
- K^i_{in} is the set of account secret keys of the accounts in R^i_{in} retrieved from $\mathcal{L}$ for $i = 0, 1$.

If $\mathbf{Verify}(\mathsf{tx}_i, \mathbb{S}) = \emptyset$ for some $i \in \{0, 1\}$, then set $\mathsf{tx}_0 = \mathsf{tx}_1 = \perp$.

(3) $b \leftarrow \{0, 1\}$

(4) $b' \leftarrow \mathcal{A}^{\text{Orc}}(\mathsf{tx}_b, \mathsf{CK}^b_{\text{out}}, \mathsf{Amt}^0_{\text{in}}, \mathsf{Amt}^1_{\text{in}}, \mathsf{st})$: $\mathcal{A}$ is given access to all the oracles, the state st, one of the **Spend** outputs, and the input amounts in K^0_{in} and K^1_{in}. Then, $\mathcal{A}$ outputs a guess for the real input of the **Spend** output provided.

$\mathcal{A}$ wins the game `Exp:Anonymity` if the following holds
- all public keys and coins in R^0_{in} and R^1_{in} are generated by PkGen and Mint, respectively, and all accounts in R^0_{in} and R^1_{in} are generated by ActGen,
- all public keys in PK_{out} are generated by PkGen,
- $\mathsf{tx}_0 \neq \perp$ and $\mathsf{tx}_1 \neq \perp$,
- no account (including its public key and coin) in R^0_{in} or R^1_{in} has been corrupted (i.e., queried to Corrupt),
- $(\cdot, \mathsf{R}^i_{\text{in}}, \cdot)$ has never been queried to Spend for $i = 0, 1$,
- $b' = b$.

Note that the adversary is restricted to corrupting at most $N - 2$ accounts in any row of A_{in} by making sure that R^0_{in} and R^1_{in} have all uncorrupted accounts. Further, instead of having two sub-definitions as in RingCT 3.0, we define a single anonymity experiment that covers different attack scenarios. Our definition is based on an indistinguishability argument, which makes it easier to extend the anonymity proofs of the ring signature used as a building block. Moreover, in our anonymity definition, only the accounts in R_{in} are assumed to be honestly generated, not all those in A_{in}.

Balance. Informally, balance requires that no adversary can spend a set A of accounts under his control such that the sum of output amounts is more than the sum of the amounts in A.

A RingCT protocol is said to be *balanced* if the following holds for all PPT adversaries $\mathcal{A}$ and $pp \leftarrow \mathbf{Setup}(1^\lambda)$

$$\Pr[\mathcal{A} \text{ wins the game } \texttt{Exp:Balance}] \leq \text{negl}(\lambda),$$

where `Exp:Balance` is defined as follows.

(1) $(\mathsf{tx}_1, \mathsf{Amt}^1_{\text{out}}, \mathsf{CK}^1_{\text{out}}), \cdots, (\mathsf{tx}_t, \mathsf{Amt}^t_{\text{out}}, \mathsf{CK}^t_{\text{out}}) \leftarrow \mathcal{A}^{\text{Orc}}(pp)$: The adversary $\mathcal{A}$ is given access to all the oracles Orc together with pp, and outputs a set of t transactions where
- $\mathsf{tx}_i = (\mathsf{A}^i_{\text{in}}, \mathsf{PK}^i_{\text{out}}, \mathsf{CN}^i_{\text{out}}, \Pi_i, \mathsf{SN}_i)$ for $i = 1, \ldots, t$,
- $\mathsf{Amt}^i_{\text{out}}$'s and $\mathsf{CK}^i_{\text{out}}$'s are sets of output amounts and coin keys, respectively, for *uncorrupted* output public keys with $|\mathsf{CK}^i_{\text{out}}| = |\mathsf{Amt}^i_{\text{out}}| \leq |\mathsf{PK}^i_{\text{out}}| = |\mathsf{CN}^i_{\text{out}}|$ for all $i \in \{1, \ldots, t\}$.

(2) $B_i \leftarrow \mathbf{Verify}(\mathsf{tx}_i, \mathbb{S})$ for $i = 1, \ldots, t$.

$\mathcal{A}$ wins the game `Exp:Balance` if the following holds
- for all $i \in \{1, \ldots, t\}$, all public keys and coins in A^i_{in} are generated by PkGen and Mint, respectively, and all accounts in A^i_{in} are generated by ActGen,
- $\bigcap_{i=1}^t \mathsf{SN}_i = \emptyset$,

- $B_i \neq \emptyset$ for all $i = 1, \ldots, t$,
- there exists a $j^* \in [1, t]$ such that $\sum_{i=0}^{S'-1} \mathsf{Amt}_{\mathsf{out}}^{j^*}[i] > \sum_{i=0}^{M-1} \mathsf{amt}_{\mathsf{in},i}$ where $S' = |\mathsf{Amt}_{\mathsf{out}}^{j^*}|$, $M = |\mathsf{SN}_{j^*}|$, $\mathsf{amt}_{\mathsf{in},i} = \mathcal{L}[\mathsf{s}_i].\mathsf{amt}$ for all $\mathsf{s}_i \in \mathsf{SN}_{j^*}$ if $\mathsf{s}_i \in \mathcal{L}$ and $\mathcal{L}[\mathsf{s}_i].\mathsf{IsCrpt} = 1$, and $\mathsf{amt}_{\mathsf{in},i} = 0$ otherwise,
- for any $i \in [1, t]$ and $0 \leq j < |\mathsf{PK}_{\mathsf{out}}^i|$, if $\mathcal{L}[\mathsf{pk}_{i,j}].\mathsf{IsCrpt} = 0$ for $\mathsf{pk}_{i,j} = \mathsf{PK}_{\mathsf{out}}^i[j]$, then $\mathsf{CK}_{\mathsf{out}}^i[j] = \mathcal{L}[\mathsf{pk}_{i,j}].\mathsf{cnk}$, $\mathsf{Amt}_{\mathsf{out}}^i[j] = \mathcal{L}[\mathsf{pk}_{i,j}].\mathsf{amt}$ and $\mathsf{CN}_{\mathsf{out}}^i[j] = \mathbf{Mint}(\mathsf{Amt}_{\mathsf{out}}^i[j], \mathsf{CK}_{\mathsf{out}}^i[j])$.[8] That is, for all *uncorrupted* output public keys, the corresponding output coin key, output amount and output coin provided by the adversary are correct.

In `Exp:Balance`, the output of the adversary does not include information about the output coin key or output amount for the *corrupted* output public keys. The reason that the adversary needs to output such information for *uncorrupted* output public keys is that the honest recipient checks whether the output coin key and output amount are in the correct domains and construct the coin. Clearly, the adversary may corrupt all the output public keys, in which case, he would not need to output $\mathsf{CK}_{\mathsf{out}}$ or $\mathsf{Amt}_{\mathsf{out}}$.

Attack Scenarios of the balance model.

(1) **Forgery:** The attacker tries to create a valid proof where (at least) one of the real spent accounts is not corrupted. This is captured by setting an input amount to zero if no corruption occurs with respect to a certain serial number.
(2) **Unbalanced input and output amounts:** The attacker tries to create a transaction where the sum of input amounts being spent does not match the sum of output amounts. This is captured by letting the attacker corrupt all the input accounts.
(3) **Double spending:** The attacker tries to spend an account twice with distinct serial numbers. This is captured by setting an input amount to zero if the respective serial number is not in $\mathcal{L}$.

Our balance definition is presented as a single experiment rather than having sub-definitions. For example, RingCT 3.0 has three sub-cases of balance: unforgeability, equivalence and linkability. Further, the unforgeability definition in RingCT 3.0 requires *all* input accounts in A_{in} to be uncorrupted. However, in this case, a natural forgery attack where the attacker has control over a single input account (which may not even be the real spent one) is excluded from the model. In our balance definition, on the other hand, the adversary wins the game -among other cases- if there is only a single uncorrupted account for which a valid serial number is generated by the adversary. Further, our balance definition allows an adversary to output a set of transactions (where one transaction can possibly be an input to the other), while only the linkability definition in RingCT 3.0 allows just two transactions to be output.

In RingCT 2.0/3.0 formal models (and also in LRCT v2.0), there is an additional property, *non-slanderability*. It states "it is infeasible for any malicious user to produce a valid spending that shares at least one serial number with a *previously* generated *honest* spending" [28]. Although non-slanderability could be a requirement in some applications of a linkable ring signature where the users are punished if a signature is detected to be generated twice using the same secret, we do not believe that is the case in the RingCT setting. The reason is that if someone generates a spending that has the same serial number with a *previously* generated one, then simply the second spending does not verify and thus ignored with no punishment in regards to the first spending. Hence, even if an attacker succeeds in winning the above non-slanderability game, there is no harm to honest users nor is there any gain for the attacker.

In our formal definitions, we aimed to explicitly state our assumptions so that the model can further be easily strengthened in future by removing some of them. For example, a potential extension is to remove the assumption in `Exp:Balance` that the input coins are generated honestly. This would require using the soundness of the preceding transaction proofs (in addition to the "current" one), complicating the balance analysis even further. Thus, in this work, such an assumption is included as in RingCT 2.0 and 3.0.

[8] Without loss of generality, we assume that the indices for corrupted public keys are the last ones so that the indexing matches.

Table 6: Identifiers for MatRiCT.

$R_q, R_{\hat{q}}$	Cyclotomic rings of degree d : $R_q = \mathbb{Z}_q[X]/(X^d+1)$, $R_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^d+1)$
$n, \hat{n}$	height of commitment matrices in R_q and $R_{\hat{q}}$, resp
$m, \hat{m}$	randomness vector dimensions in R_q and $R_{\hat{q}}$, resp
$N = \beta^k$	ring size of ring signature (and anonymity set size)
ℓ	spender's column index with $0 \leq \ell < N$
r	bit-length of an amount, i.e., $\mathsf{amt} \in \mathsf{V} = [0, 2^r - 1]$
$\mathcal{B}$	max. absolute coefficient of initial randomness

4 MatRiCT: LATTICE-BASED RINGCT

We describe the full details of MatRiCT in this section. We specify the functions **Setup**, **KeyGen**, **SerialGen**, **Mint**, **Spend** and **Verify**. To simplify presentation, the part concerning the corrector values in **Spend** is shown for the case $(M, S) = (1, 2)$ in Algorithm 8 and we discuss in the text how the general case can be easily accomplished. Let us go over the description of each algorithm one-by-one and fix the notation in Table 6. We assume that r-bit precision is always sufficient for the amounts (even when they are summed)[9] and that valid amounts are used to call the functions. It is trivial to return an error when that is not the case.

In general, there are 3 commitment keys $\boldsymbol{G}, \hat{\boldsymbol{G}}$ and $\boldsymbol{H}$ used in the system where $\boldsymbol{H}$ (defined over R_q) is used only in the serial number generation, $\hat{\boldsymbol{G}}$ is used for the commitments over $R_{\hat{q}}$ in binary proof and $\boldsymbol{G}$ is used elsewhere for the commitments over R_q.

Algorithm 1 $\mathsf{SamMat}(\rho', \upsilon, q', n', m', \mathrm{str})$

INPUT: ρ' for some seed $\rho' \in \{0,1\}^{256}$; $\upsilon, q', n', m' \in \mathbb{Z}^+$; str is an optional auxiliary input string.

1: $\boldsymbol{G} \leftarrow \mathsf{Sam}(\rho', \mathrm{str})$ where $\boldsymbol{G} \in R_{q'}^{n' \times (m'+\upsilon)}$
2: **return** $\boldsymbol{G}$ ▷ $\boldsymbol{G}$ can be output in the NTT domain.

[9] We do not assume here that summing multiple amounts of, say, $2^{64}-1$ is impossible. To be more explicit, r can be set as the smallest integer such that $\mathsf{MAX}_{\mathsf{amt}} \cdot \mathsf{MAX}_{\mathsf{io}} \leq 2^r - 1$ where $\mathsf{MAX}_{\mathsf{amt}}$ is the maximum amount possible and $\mathsf{MAX}_{\mathsf{io}}$ is the maximum number of input/output accounts allowed. But, the amount is still represented in r bits. Recall that our protocol does not have the disadvantage of requiring a modulus greater than $2^r - 1$, and a few more bits of precision can be added at almost no cost.

Algorithm 2 Setup(1^λ) λ is the security parameter

1: Choose int. params $k, \beta, r, n, m, \hat{n}, \hat{m}, d, q, \hat{q}$ such that $N = \beta^k$
2: Set w, p such that $|C^d_{w,p}| > 2^{256}$
3: $\rho \leftarrow \{0,1\}^{256}$
4: Pick a hash function $\mathcal{H} : \{0,1\}^* \to C^d_{w,p}$
5: **return** $pp = (\rho, \mathcal{H}, k, \beta, r, n, m, \hat{n}, \hat{m}, n_s, d, w, p, q, \hat{q})$

Algorithm 3 KeyGen(pp)

1: $\boldsymbol{G} \leftarrow \mathsf{SamMat}(\rho, 0, q, n, m, \text{“G”})$
2: $\boldsymbol{r} \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$
3: $\boldsymbol{c} = \boldsymbol{G} \cdot \boldsymbol{r}$ in R_q^n ▷ Com. to zero under $ck = \boldsymbol{G}$
4: **return** $(\mathsf{pk}, \mathsf{sk}) = (\boldsymbol{c}, \boldsymbol{r})$ ▷ Can be output in the NTT domain

Algorithm 1 generates a random matrix from a small seed ρ using an extendable output function Sam (modelled as a random oracle in the security analysis). It also has an auxiliary input string str so that different matrices from the same seed can be generated. Calling the function with the same seed ρ and same string str results in the generation of the same entries. For example, running $\boldsymbol{A} \leftarrow \mathsf{SamMat}(\rho, v_1, q, n_1, m_1, \text{str})$ and $\boldsymbol{B} \leftarrow \mathsf{SamMat}(\rho, v_2, q, n_2, m_2, \text{str})$ with $n_1 \le n_2$, $m_1 + v_1 \le m_2 + v_2$ results in two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ where $\boldsymbol{A}$ is a sub-matrix of $\boldsymbol{B}$.

Algorithm 2 sets the system parameters. Here, for ease of presentation, we assume the ring size to be fixed to some N. The range of the hash function is defined as the following challenge space

$$C^d_{w,p} = \{x \in \mathbb{Z}[X] : \deg(x) = d-1 \wedge \ \mathsf{HW}(x) = w \wedge \ \|x\|_\infty = p\}.$$

This is the same set defined in [7] and $|C^d_{w,p}| = \binom{d}{w}(2p)^w$. Thus, given d, one can easily set (w, p) such that $|C^d_{w,p}| > 2^{256}$.

Algorithm 3 generates a public-secret key pair. The secret key is random vector over R_q with infinity norm $\mathcal{B}$, and the public key is a commitment to zero with the secret key used as the randomness.

Algorithm 4 generates a serial number for a given secret key. The serial number is a commitment to zero using the secret key as the randomness under the commitment key $\boldsymbol{H}$. Observe that the height of the commitment matrix here is set to n_s.

Algorithm 5 implements minting a coin, which is computed as a commitment to the bits of an input amount. The commitment key used here is the same as the one in **KeyGen**.

Since **Spend** algorithm is very long, we split it into multiple parts, Algorithms 6, 7, 8 and 9. **Spend** starts by setting some parameters in Algorithm 8. These settings are done to accommodate different parameters while keeping the acceptance rate of the rejection sampling similar. In Step 7, we compute the corrector values c_i where the division by two is always exact by the following lemma.

Lemma 4.1. *Let A, B be two sets of non-negative integers and $a = (a[0], \ldots, a[r-1])$ be the representation of a non-negative integer a in base $\beta \ge 2$. If $\sum_{a \in A} a = \sum_{b \in B} b$, then for any $\beta \ge 2$, there exists $c_0, \ldots, c_r \in [-(|B|-1), |A|-1]$ with $c_0 = 0$ such that*

$$\sum_{a \in A} a[i] - \sum_{b \in B} b[i] = \beta c_{i+1} - c_i.$$

Further, $c_r = 0$ if the result of the sum is of at most r digits in base β.

Algorithm 4 SerialGen(sk) for a secret key $\mathsf{sk} \in R_q^m$

1: $\boldsymbol{H} \leftarrow \mathsf{SamMat}(\rho, 0, q, n_s, m, \text{“H”})$
2: $\boldsymbol{c} = \boldsymbol{H} \cdot \boldsymbol{r}$ in $R_q^{n_s}$ where $\boldsymbol{r} = \mathsf{sk} \in R_q^m$ ▷ Com. to zero under $\boldsymbol{H}$
3: **return** $\mathsf{s} = \boldsymbol{c}$ ▷ s can be output in the NTT domain

Algorithm 5 Mint(amt) for $\mathsf{amt} \in [0, 2^r - 1]$

1: $\boldsymbol{G} \leftarrow \mathsf{SamMat}(\rho, r, q, n, m, \text{“G”})$
2: $\boldsymbol{r} \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$, $(b_0, \ldots, b_{r-1}) \leftarrow \mathsf{Bits}(\mathsf{amt})$
3: $C = \mathrm{Com}_{ck}(b_0, \ldots, b_{r-1}; \boldsymbol{r})$ in R_q^n where $ck = \boldsymbol{G}$
4: **return** $(\mathsf{cn}, \mathsf{cnk}) = (C, \boldsymbol{r})$

Proof of Lemma 4.1. Let $\beta \ge 2$. For any set A of non-negative integers and any $0 \le i < r$ where r is the maximum number of digits needed to represent elements in A, we can write

$$\left(\sum_{a \in A} a\right)[i] - c_i + \beta c_{i+1} = \sum_{a \in A} a[i] \tag{5}$$

for the carries $c_1, \ldots, c_r$ and c_0 since there is no carry for the least significant bit. Now, fix A, B as two sets of non-negative integers where the sum over A equals the sum over B. Clearly, we have

$$\left(\sum_{a \in A} a\right)[i] = \left(\sum_{b \in B} b\right)[i] \tag{6}$$

for any $0 \le i < r$. Using (5) and (6), for any $0 \le i < r$, we get

$$\begin{aligned}\sum_{a \in A} a[i] - \sum_{b \in B} b[i] &= \left(\sum_{a \in A} a\right)[i] - c''_i + \beta c''_{i+1} - \left(\sum_{b \in B} b\right)[i] + c'_i - \beta c'_{i+1} \\ &= -(c''_i - c'_i) + \beta(c''_{i+1} - c'_{i+1}),\end{aligned}$$

for some carries $c'_0, c''_0, \ldots, c'_r, c''_r$ where $c'_0 = c''_0 = 0$. Defining $c_i := c''_i - c'_i$ with $c_0 = 0$ concludes the first part. Note that due to c''_i, c'_i being carry values, $c''_i \in [0, |A|-1]$ and $c'_i \in [0, |B|-1]$, and thus $c_i \in [-(|B|-1), |A|-1]$. When neither of the sums exceed r digits, $c'_r = c''_r = 0$, and thus $c_r = 0$. □

After computation of the corrector values, the spender mints the output coins, and runs an aggregated binary proof using Algorithm 6. The idea for the binary proof is the same as in [7], but we apply our efficient rejection sampling technique for binary secrets of fixed Hamming weight and our binary proof proves a slightly different relation given in Lemma 5.5. In general, Algorithm 6 takes t sequences of bits where each sequence has s_j elements, and the masking values for each sequence is sampled from $\mathfrak{U}_{\mathcal{B}_j}$. Also, each sequence has a flag Bool_j to indicate whether the sequence has a fixed Hamming weight, in which case the masking values for zero bits are sampled directly from the accepted distribution of rejection sampling. Note that for the case of Hamming weight equal to 1, there is always exactly one element in $\{a_1^{(j)}, \ldots, a_{s_j}^{(j)}\}$ not sampled from the accepted distribution.

Defining $\delta_{i,j}$ as the Kronecker's delta, Step 14 of Algorithm 8 computes the unary representation of the spender's index ℓ in base β, which has a fixed Hamming weight. Therefore, the flag Bool_j is set to True for these sequences. Then, we also add the corrector values and the output amount bits to the array $\boldsymbol{b}$, which is then input to the binary proof. The most common cases for the number

Algorithm 6 BinaryCommit ▷ Commitment Step of Binary Proof

INPUT: $t \in \mathbb{Z}^+$; $\{(s_j, \text{Bool}_j, (b_0^{(j)}, \ldots, b_{s_j-1}^{(j)}), \mathcal{B}_j)\}_{j=0}^{t-1}$ where $s_j \in \mathbb{Z}^+$, $\text{Bool}_j \in \{\text{True}, \text{False}\}$, $b_i^{(j)} \in \{0,1\}$; $\mathcal{B}, \hat{\mathcal{B}}_{\text{big}} \in \mathbb{Z}^+$.

OUTPUT: $(\boldsymbol{r}_a, \boldsymbol{r}_b), (A, B), \{(a_0^{(j)}, \ldots, a_{s_j-1}^{(j)})\}_{j=0}^{t-1}$ where $\boldsymbol{r}_a, \boldsymbol{r}_b \in R_{\hat{q}}^{\hat{m}}$, $A, B \in R_{\hat{q}}^{\hat{n}}$ and $a_i^{(j)} \in R_{\hat{q}}$.

1: $ck = \hat{\boldsymbol{G}} \leftarrow \text{SamMat}(\rho, v, \hat{q}, \hat{n}, \hat{m}, \text{“Gbig”})$ for $v = 2 \cdot \left(\sum_{j=0}^{t-1} s_j\right)$
2: $\boldsymbol{r}_b \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot \hat{m}}$ and $\boldsymbol{r}_a \leftarrow \{-\hat{\mathcal{B}}_{\text{big}}, \ldots, \hat{\mathcal{B}}_{\text{big}}\}^{d \cdot \hat{m}}$
3: **for** $j = 0, \ldots, t-1$ **do** ▷ Iterate over each bit sequence
4: **if** $\text{Bool}_j = \text{True}$ **then** ▷ The case of HW being 1.
5: **if** $b_0^{(j)} = 0$, **then** $i^* = -1$ ▷ Out of $[1, s_j - 1]$
6: **else** $i^* \leftarrow \{1, \ldots, s_j - 1\}$ and $a_{i^*}^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
7: **for** $i = 1, \ldots, s_j - 1$ and $i \neq i^*$ **do** ▷ i starts from 1.
8: **if** $b_i^{(j)} = 0$, **then** $a_i^{(j)} \leftarrow \{-(\mathcal{B}_j - p), \ldots, \mathcal{B}_j - p\}^d$
9: **else** $a_i^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
10: **end for**
11: $a_0^{(j)} = -\sum_{i=1}^{s_j-1} a_i^{(j)}$
12: **else**
13: **for** $i = 0, \ldots, s_j - 1$ **do** ▷ i starts from 0.
14: $a_i^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
15: **end for**
16: **end if**
17: **end for**
18: $\boldsymbol{b} = \left(b_0^{(0)}, \ldots, b_{s_{t-1}-1}^{(t-1)}\right)$, $\boldsymbol{a} = \left(a_0^{(0)}, \ldots, a_{s_{t-1}-1}^{(t-1)}\right)$
19: $\boldsymbol{c} = \left(a_0^{(0)}(1 - 2b_0^{(0)}), \ldots, a_{s_{t-1}-1}^{(t-1)}(1 - 2b_{s_{t-1}-1}^{(t-1)})\right)$
20: $\boldsymbol{d} = \left(-(a_0^{(0)})^2, \ldots, -(a_{s_{t-1}-1}^{(t-1)})^2\right)$
21: $B = \text{Com}_{ck}(\boldsymbol{b}, \boldsymbol{c}; \boldsymbol{r}_b)$, $A = \text{Com}_{ck}(\boldsymbol{a}, \boldsymbol{d}; \boldsymbol{r}_a)$ in $R_{\hat{q}}^{\hat{n}}$ with $ck = \hat{\boldsymbol{G}}$
22: **return** $(\boldsymbol{r}_a, \boldsymbol{r}_b), (A, B), \{(a_0^{(j)}, \ldots, a_{s_j-1}^{(j)})\}_{j=0}^{t-1}$

of input/output accounts are $(M, S) = (1, 2)$ and $(M, S) = (2, 2)$. For the former case, the corrector values are binary as they are simply the carries in the sum of two output amounts. Therefore, the steps given in Algorithm 8 are sufficient. In the latter case, we can prove that the corrector values are differences of some bits $c_{\text{in},i}$, $c_{\text{out},i}$, which are the carries from the sum of two inputs and the sum of two outputs, respectively.

In general, however, the corrector values can fall in a larger interval $[-(M-1), S-1]$, and in that case, one needs to prove that they are indeed in that interval. This can be done using a standard range proof, where the range width is only $M + S - 1$, which is expected to be very small. In fact, we do not need to prove that they fall exactly in $[-(M-1), S-1]$, but can alternatively prove that they are in a range of width 2^l for $l = \lceil \log(M + S - 1) \rceil$. There are standard methods to "shift" the range at no cost (see, e.g., [7]). As mentioned in Section 1.3, as long as (4) is ensured to hold over $\mathbb{Z}$, the corrector values can be set freely. The final part of Algorithm 6 is committing to all the values as outlined in Section 1.3.

Steps 26 and 27 of Algorithm 8 are used to prove that the corrector commitment C is well-formed, i.e., does not contain any value with respect to the representation of the amounts. After that, the spender

Algorithm 7 RingCommit ▷ Commitment Step of Ring Sign.

INPUT: $\text{GenSerial} \in \{\text{True}, \text{False}\}$; $(P_0, \ldots, P_{N-1})$ where $P_i \in R_q^n$; $(p_{0,0}, \ldots, p_{N-1,k-1})$ where $p_{i,j} \in R_q$; $\mathcal{B}, \mathcal{B}_{\text{big,k}} \in \mathbb{Z}^+$.

OUTPUT: $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, F_0, \ldots, E_{k-1}, F_{k-1})$ where $\boldsymbol{\rho}_j \in R_q^m$, $E_j \in R_q^n$ and $F_j \in R_q^{n_s}$. F_j's are omitted when GenSerial = False.

1: $ck = \boldsymbol{G} \leftarrow \text{SamMat}(\rho, 0, q, n, m, \text{“G”})$
2: **if** GenSerial = True, **then** $\boldsymbol{H} \leftarrow \text{SamMat}(\rho, 0, q, n_s, m, \text{“H”})$
3: $\boldsymbol{\rho}_0 \leftarrow \{-\mathcal{B}_{\text{big,k}}, \ldots, \mathcal{B}_{\text{big,k}}\}^{d \cdot m}$
4: **for** $j = 0, \ldots, k-1$ **do**
5: $\boldsymbol{\rho}_j \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$ **if** $j \neq 0$
6: $R_j = \text{Com}_{ck}(\boldsymbol{0}; \boldsymbol{\rho}_j)$ in R_q^n
7: $E_j = \sum_{i=0}^{N-1} p_{i,j} P_i + R_j$ in R_q^n
8: **if** GenSerial = True, **then** $F_j = \boldsymbol{H} \cdot \boldsymbol{\rho}_j$ in $R_q^{n_s}$
9: **end for**
10: **if** GenSerial = True, **then**
return $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, F_0, \ldots, E_{k-1}, F_{k-1})$
11: **return** $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, \ldots, E_{k-1})$

runs M ring signatures to prove ownership of an account from each row of A_{in}. Here, she also computes a serial number for each account spent. Finally, another ring signature is run to prove that the balance is preserved by showing $\sum_{i=0}^{S-1} \mathsf{cn}_{\text{out},i} - \sum_{i=0}^{M-1} \mathsf{cn}_{i,\ell} + C$ is a commitment to zero for the same index $\ell \in [0, N-1]$. Note that all ring signatures are run using the same vector $\boldsymbol{p}$, and thus the indices of the spender's accounts are the same in all rows (notice also that **Verify**, Algorithm 10, uses the same $f_{j,i}$'s in the verification of the ring signatures at Steps 22 and 28).

The main part of the ring signature is summarized in Algorithm 7, which follows the same blueprint as the one-out-of-many proof in [7], but again proves a slightly different relation given in Lemma 5.6. Additionally, when the ring signature is used to prove knowledge of a user secret key, Algorithm 7 also outputs elements $F_0, \ldots, F_{k-1}$ to be used in verification of the serial number. $p_{i,j} \in \boldsymbol{p}$ input to Algorithm 7 are defined as in Equation (28) of [7] and their exact computation for $k \in \{1, 2\}$ is given in the full version. The final step of Algorithm 8 is hashing all the information up to that step where the hash function is modelled as a random oracle.

The second part of **Spend** (Algorithm 9) is comprised of the spender's masked responses of the underlying ZKP. Each bit input to the binary proof is masked by the corresponding $a \in R_{\hat{q}}$, and the rejection sampling technique from [18] is applied. The general idea works as follows. If we have a vector $\boldsymbol{y} = \boldsymbol{s} + \boldsymbol{v}$ where $\boldsymbol{s}$ is the secret-dependent part and $\boldsymbol{v} \leftarrow \{-B, \ldots, B\}^t$ for some $B, t \in \mathbb{Z}^+$, then rejection happens when $\|\boldsymbol{y}\|_\infty > B - \|\boldsymbol{s}\|_\infty$. To have a small rejection probability, we set $B = c \cdot \|\boldsymbol{s}\|_\infty \cdot t$ for some constant c.

Additionally, **Spend** also restarts if the norm of some $f_{j,0}$ or $\boldsymbol{g}$ is "unexpectedly large". This is done in order to use tighter bounds when computing M-SIS hardness. The bound on $f_{j,0}$ comes from the fact that $a_{j,0}$ is the sum of uniformly sampled elements and thus its distribution converges to a Gaussian distribution. It is hard to formally bound the probability of having a rejection due to Step 20, and thus the bound T_g on $\|\boldsymbol{g}\|$ is computed experimentally so that

Algorithm 8 **Spend**-I

INPUT: $M, S \in \mathbb{Z}^+$; $\mathsf{A}_{\mathsf{in}} = (\mathsf{act}_{0,0}, \ldots, \mathsf{act}_{M-1,N-1})$ where $\mathsf{act}_{i,j} = (\mathsf{pk}_{i,j}, \mathsf{cn}_{i,j})$ is an account; $\ell \in [0, N-1]$; $(\mathsf{ask}_{0,\ell}, \ldots, \mathsf{ask}_{M-1,\ell})$ where $\mathsf{ask}_{i,\ell} = (\boldsymbol{r}_{i,\ell}, \mathsf{cnk}_{i,\ell}, \mathsf{amt}_{\mathsf{in},i}) \in R_q^m \times R_q^m \times \mathbb{Z}^+$; $\mathsf{PK}_{\mathsf{out}} = (\mathsf{pk}_{\mathsf{out},0}, \ldots, \mathsf{pk}_{\mathsf{out},S-1})$ where $\mathsf{pk}_{\mathsf{out},i} \in R_q^n$; $(\mathsf{amt}_{\mathsf{out},0}, \ldots, \mathsf{amt}_{\mathsf{out},S-1})$ where $\mathsf{amt}_{\mathsf{out},i} \in [0, 2^r - 1]$.

$\mathcal{B}_a = \lceil 20 \cdot pkd \rceil$, $\mathcal{B}_r = \lceil p(S+1)rd \rceil$
$T_g = d^3 \left(\mathcal{B}_a^4 k\beta(\beta+1) + \mathcal{B}_r^4 r(S+1) \right) / (4d)$
$\mathcal{B}_{\mathsf{big}} = \lceil 1.2 \cdot (M+S+1)\mathcal{B}pwmd \rceil$, $\hat{\mathcal{B}}_{\mathsf{big}} = \lceil 8 \cdot (M+S+1)\mathcal{B}pw\hat{m}d \rceil$
$\mathcal{B}_{\mathsf{big,k}} = \lceil 1.2 \cdot (M+S+1)\mathcal{B}(pw)^k md \rceil$
$\mathcal{B}'_{\mathsf{big,k}} = \lceil 2.4 \cdot (M+S+1)\mathcal{B}(pw)^k md \rceil$
for $i = 0, \ldots, r-2$ **do** ▷ $c_0 = c_r = 0$
  $c_{i+1} = \left(c_i + \sum_{j=0}^{S-1} \mathsf{amt}_{\mathsf{out},j}[i] - \sum_{j=0}^{M-1} \mathsf{amt}_{\mathsf{in},j}[i] \right) / 2$
end for
for $i = 0, \ldots, S-1$ **do**
  $(\mathsf{cn}_{\mathsf{out},i}, \mathsf{cnk}_{\mathsf{out},i}) \leftarrow \mathbf{Mint}(\mathsf{amt}_{\mathsf{out},i})$
end for
$\mathsf{CN}_{\mathsf{out}} = (\mathsf{cn}_{\mathsf{out},0}, \ldots, \mathsf{cn}_{\mathsf{out},S-1})$
$\mathsf{CK}_{\mathsf{out}} = (\mathsf{cnk}_{\mathsf{out},0}, \ldots, \mathsf{cnk}_{\mathsf{out},S-1})$
$\boldsymbol{b} = \{(\beta, \mathrm{True}, (\delta_{\ell_j,0}, \ldots, \delta_{\ell_j,\beta-1}), \mathcal{B}_a)\}_{j=0}^{k-1}$
$\boldsymbol{b} = \boldsymbol{b} \cup (r-1, \mathrm{False}, (c_1, \ldots, c_{r-1}), \mathcal{B}_r)$
for $j = 0, \ldots, S-1$ **do**
  $\boldsymbol{b} = \boldsymbol{b} \cup (r, \mathrm{False}, \mathsf{Bits}(\mathsf{amt}_{\mathsf{out},j}), \mathcal{B}_r)$
end for
$ck = \boldsymbol{G} \leftarrow \mathsf{SamMat}(\rho, r, q, n, m, \text{“G”})$
$\boldsymbol{r}_c \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$, $\boldsymbol{r}_d \leftarrow \{-\mathcal{B}_{\mathsf{big}}, \ldots, \mathcal{B}_{\mathsf{big}}\}^{d \cdot m}$
$(\boldsymbol{r}_a, \boldsymbol{r}_b), (A, B), (a_{0,0}, \ldots, a_{k-1,\beta-1}, a_{c,1}, \ldots, a_{c,r-1}, a_{\mathsf{out},0}^{(0)}, \ldots, a_{\mathsf{out},r-1}^{(S-1)})$
$\leftarrow \mathsf{BinaryCommit}(k+1+S, \boldsymbol{b}, \mathcal{B}, \hat{\mathcal{B}}_{\mathsf{big}})$ ▷ $a_{c,0} = a_{c,r} = 0$
for $i = 0, \ldots, S-1$ **do**
  $\boldsymbol{r}_g^{(i)} \leftarrow \{-\mathcal{B}_{\mathsf{big}}, \ldots, \mathcal{B}_{\mathsf{big}}\}^{d \cdot m}$
  $G_i = \mathrm{Com}_{ck}(a_{\mathsf{out},0}^{(i)}, \ldots, a_{\mathsf{out},r-1}^{(i)}; \boldsymbol{r}_g^{(i)})$ in R_q^n
end for
$C = \mathrm{Com}_{ck}(c_0 - 2c_1, \ldots, c_{r-1} - 2c_r; \boldsymbol{r}_c)$ in R_q^n
$D = \mathrm{Com}_{ck}(a_{c,0} - 2a_{c,1}, \ldots, a_{c,r-1} - 2a_{c,r}; \boldsymbol{r}_d)$ in R_q^n
Compute $\boldsymbol{p} = (p_{0,0}, \ldots, p_{N-1,k-1})$ as in Equation (28) of [7]
for $i = 0, \ldots, M-1$ **do**
  $\mathsf{s}_i = \mathbf{SerialGen}(\boldsymbol{r}_{i,\ell})$ ▷ Not recomputed if restarted
  $(\boldsymbol{\rho}_0^{(i)}, \ldots, \boldsymbol{\rho}_{k-1}^{(i)}), (E_0^{(i)}, F_0^{(i)}, \ldots, E_{k-1}^{(i)}, F_{k-1}^{(i)}) \leftarrow$
    $\mathsf{RingCommit}(\mathrm{True}, (\mathsf{pk}_{i,0}, \ldots, \mathsf{pk}_{i,N-1}), \boldsymbol{p}, \mathcal{B}, \mathcal{B}_{\mathsf{big,k}})$
end for
for $j = 0, \ldots, N-1$ **do**
  $P_j = \sum_{i=0}^{S-1} \mathsf{cn}_{\mathsf{out},i} - \sum_{i=0}^{M-1} \mathsf{cn}_{i,j} + C$ in R_q^n
end for
$(\boldsymbol{\rho}_0^{(M)}, \ldots, \boldsymbol{\rho}_{k-1}^{(M)}), (E_0^{(M)}, \ldots, E_{k-1}^{(M)}) \leftarrow$
  $\mathsf{RingCommit}(\mathrm{False}, (P_0, \ldots, P_{N-1}), \boldsymbol{p}, \mathcal{B}, \mathcal{B}'_{\mathsf{big,k}})$
$x = \mathcal{H}(A, B, C, D, E_0^{(0)}, \ldots, E_{k-1}^{(M)}, F_0^{(0)}, \ldots, F_{k-1}^{(M)}, G_0, \ldots, G_{S-1}, \mathsf{s}_0, \ldots, \mathsf{s}_{M-1}, \mathsf{A}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{CN}_{\mathsf{out}})$

Algorithm 9 **Spend**-II ▷ No mod q or $\hat{q}$ in this function!

OUTPUT: $\mathsf{CK}_{\mathsf{out}}, \mathsf{A}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{CN}_{\mathsf{out}}, \mathsf{SN}$ and Π as below where $F_1^{(0)}, \ldots, F_{k-1}^{(M)} \in R_q^{n_s}$; $B \in R_{\hat{q}}^{\hat{n}}$; $C, E_1^{(0)}, \ldots, E_{k-1}^{(M)} \in R_q^n$; $x \in C_{w,p}^d$; $\boldsymbol{f}_1 \in R_q^{k(\beta-1)}$; $\boldsymbol{f}_r \in R_q^{r-1+Sr}$; $\boldsymbol{z}_b \in R_{\hat{q}}^{\hat{m}}$; $\boldsymbol{z}_c, \boldsymbol{z}_{\mathsf{out},0}, \ldots, \boldsymbol{z}_{\mathsf{out},S-1}, \boldsymbol{z}^{(0)}, \ldots, \boldsymbol{z}^{(M)} \in R_q^m$

for $j = 0, \ldots, k-1$ and $i = 0, \ldots, \beta-1$ **do**
  $f_{j,i} = x\delta_{\ell_j,i} + a_{j,i}$ ▷ masked spender index repr.
end for
for $i = 1, \ldots, r-1$ **do**
  $f_{c,i} = xc_i + a_{c,i}$ ▷ masked corrector values
end for
for $j = 0, \ldots, S-1$ and $i = 0, \ldots, r-1$ **do**
  $f_{\mathsf{out},i}^{(j)} = x \cdot \mathsf{amt}_{\mathsf{out},j}[i] + a_{\mathsf{out},i}^{(j)}$ ▷ masked out. amt. bits
end for
$\boldsymbol{f}_1 = (f_{0,1}, \ldots, f_{k-1,\beta-1})$ ▷ $f_{j,0}$'s are excluded
if $\|\boldsymbol{f}_1\|_\infty > \mathcal{B}_a - p$, **then** Go to Step 20 of Alg. 8
$\boldsymbol{f}_r = \left(f_{c,1}, \ldots, f_{c,r-1}, f_{\mathsf{out},0}^{(0)}, \ldots, f_{\mathsf{out},r-1}^{(S-1)} \right)$
if $\|\boldsymbol{f}_r\|_\infty > \mathcal{B}_r - p$, **then** Go to Step 20 of Alg. 8
for $j = 0, \ldots, k-1$ **do**
  if $\|f_{j,0}\| > \mathcal{B}_a \sqrt{d(\beta-1)}$, **then** Go to Step 20 of Alg. 8
end for
$\boldsymbol{g} = \left(f_{0,0}(x - f_{0,0}), \ldots, f_{k-1,\beta-1}(x - f_{k-1,\beta-1}) \right)$
$\boldsymbol{g} = \boldsymbol{g} \cup \left(f_{c,1}(x - f_{c,1}), \ldots, f_{c,r-1}(x - f_{c,r-1}) \right)$
$\boldsymbol{g} = \boldsymbol{g} \cup \left(f_{\mathsf{out},0}^{(0)}(x - f_{\mathsf{out},0}^{(0)}), \ldots, f_{\mathsf{out},r-1}^{(S-1)}(x - f_{\mathsf{out},r-1}^{(S-1)}) \right)$
if $\|\boldsymbol{g}\| > \sqrt{T_g}$, **then** Go to Step 20 of Alg. 8
$\boldsymbol{z}_b = x\boldsymbol{r}_b + \boldsymbol{r}_a$ ▷ $\hat{m}$-dimensional
if $\|\boldsymbol{z}_b\|_\infty > \hat{\mathcal{B}}_{\mathsf{big}} - \mathcal{B}pw$, **then** Go to Step 20 of Alg. 8
$\boldsymbol{z}_c = x\boldsymbol{r}_c + \boldsymbol{r}_d$
for $i = 0, \ldots, S-1$ **do**
  $\boldsymbol{z}_{\mathsf{out},i} = x\boldsymbol{r}_{\mathsf{out},i} + \boldsymbol{r}_g^{(i)}$ where $\boldsymbol{r}_{\mathsf{out},i} = \mathsf{cnk}_{\mathsf{out},i}$
end for
if $\|(\boldsymbol{z}_c, \boldsymbol{z}_{\mathsf{out},0}, \ldots, \boldsymbol{z}_{\mathsf{out},S-1})\|_\infty > \mathcal{B}_{\mathsf{big}} - \mathcal{B}pw$, **then** Go to Step 20 of Alg. 8
for $i = 0, \ldots, M-1$ **do**
  $\boldsymbol{z}^{(i)} = x^k \boldsymbol{r}_{i,\ell} - \sum_{j=0}^{k-1} x^j \boldsymbol{\rho}_j^{(i)}$
  if $\|\boldsymbol{z}^{(i)}\|_\infty > \mathcal{B}_{\mathsf{big,k}} - \mathcal{B}(pw)^k$, **then** Go to Step 20 of Alg. 8
end for
$\boldsymbol{z}^{(M)} = x^k \boldsymbol{r}_{M,\ell} - \sum_{j=0}^{k-1} x^j \boldsymbol{\rho}_j^{(M)}$
  where $\boldsymbol{r}_{M,\ell} = \sum_{i=0}^{S-1} \boldsymbol{r}_{\mathsf{out},i} - \sum_{i=0}^{M-1} \mathsf{cnk}_{i,\ell} + \boldsymbol{r}_c$
if $\|\boldsymbol{z}^{(M)}\|_\infty > \mathcal{B}'_{\mathsf{big,k}} - (M+S+1)\mathcal{B}(pw)^k$, **then** Go to Step 20 of Alg. 8
return $\mathsf{CK}_{\mathsf{out}}, \mathsf{A}_{\mathsf{in}}, \mathsf{PK}_{\mathsf{out}}, \mathsf{CN}_{\mathsf{out}}, \mathsf{SN} = (\mathsf{s}_0, \ldots, \mathsf{s}_{M-1})$ and
$\Pi = (B, C, E_1^{(0)}, \ldots, E_{k-1}^{(M)}, F_1^{(0)}, \ldots, F_{k-1}^{(M)}, x, \boldsymbol{f}_1, \boldsymbol{f}_r, \boldsymbol{z}_b, \boldsymbol{z}_c, \boldsymbol{z}^{(0)}, \ldots, \boldsymbol{z}^{(M)}, \boldsymbol{z}_{\mathsf{out},0}, \ldots, \boldsymbol{z}_{\mathsf{out},S-1})$

the chance of restarting due to Step 20 is less than 1%.[10] However, this does not raise a security concern as the same bound is also checked by the verifier and thus ensured to hold for any accepting transcript. The masked randomnesses are similarly computed as in the one-out-of-many proof of [7] and the same rejection sampling idea as above is used. The output part $\mathsf{CK}_{\mathsf{out}}$ of Algorithm 9 is

[10] Observe here that T_g is a factor $4d$ smaller than the theoretical bound in Lemma 5.2.

Algorithm 10 Verify

INPUT: $M, S \in \mathbb{Z}^+$; $\mathsf{A}_{\text{in}} = (\text{act}_{0,0}, \ldots, \text{act}_{M-1,N-1})$ where $\text{act}_{i,j} = (\text{pk}_{i,j}, \text{cn}_{i,j})$ is an account; $\mathsf{PK}_{\text{out}} = (\text{pk}_{\text{out},0}, \ldots, \text{pk}_{\text{out},S-1})$; $\mathsf{CN}_{\text{out}} = (\text{cn}_{\text{out},0}, \ldots, \text{cn}_{\text{out},S-1})$; $\Pi = (B, C, E_1^{(0)}, \ldots, E_{k-1}^{(M)}, F_1^{(0)}, \ldots, F_{k-1}^{(M)}, x, \boldsymbol{f}_1, \boldsymbol{f}_r, z_b, z_c, \boldsymbol{z}^{(0)}, \ldots, \boldsymbol{z}^{(M)}, z_{\text{out},0}, \ldots, z_{\text{out},S-1})$; $\mathsf{SN} = (\text{s}_0, \ldots, \text{s}_{M-1})$.
OUTPUT: True/False

if $\|\boldsymbol{f}_1\|_\infty > \mathcal{B}_a - p$, **then return** False
if $\|\boldsymbol{f}_r\|_\infty > \mathcal{B}_r - p$, **then return** False
Parse $\boldsymbol{f}_1 = (f_{0,1}, \ldots, f_{k-1,\beta-1})$ as in Alg. 9
Parse $\boldsymbol{f}_r = (f_{c,1}, \ldots, f_{c,r-1}, f_{\text{out},0}^{(0)}, \ldots, f_{\text{out},r-1}^{(S-1)})$ as in Alg. 9
for $j = 0, \ldots, k-1$ **do**
    $f_{j,0} = x - \sum_{i=1}^{\beta-1} f_{j,i}$
    if $\|f_{j,0}\| > \mathcal{B}_a\sqrt{d(\beta-1)}$, **then return** False
end for
Compute $\boldsymbol{g}$ as in Alg. 9
if $\|\boldsymbol{g}\| > \sqrt{T_g}$, **then return** False
if $\|z_b\|_\infty > \hat{\mathcal{B}}_{\text{big}} - \mathcal{B}pw$, **then return** False
if $\|(z_c, z_{\text{out},0}, \ldots, z_{\text{out},S-1})\|_\infty > \mathcal{B}_{\text{big}} - \mathcal{B}pw$, **then return** False
if $\|(\boldsymbol{z}^{(0)}, \ldots, \boldsymbol{z}^{(M-1)})\|_\infty > \mathcal{B}_{\text{big,k}} - \mathcal{B}(pw)^k$, **then return** False
if $\|\boldsymbol{z}^{(M)}\|_\infty > \mathcal{B}'_{\text{big,k}} - (M+S+1)\mathcal{B}(pw)^k$, **then return** False
$\boldsymbol{f} = (f_{0,0}, \ldots, f_{k-1,\beta-1}) \cup \boldsymbol{f}_r$ ▷ $f_{j,0}$'s are included.
$A = \text{Com}_{ck}(\boldsymbol{f}, \boldsymbol{g}; z_b) - xB$ in $R_{\hat{q}}^{\hat{n}}$ for $ck = \hat{G}$
$D = \text{Com}_{ck}(f_{c,0} - 2f_{c,1}, \ldots, f_{c,r-1} - 2f_{c,r}; z_c) - xC$ in R_q^n where $f_{c,0} = f_{c,r} = 0$ and $ck = \boldsymbol{G}$ here and in the rest
for $i = 0, \ldots, S-1$ **do**
    $G_i = \text{Com}_{ck}(f_{\text{out},0}^{(i)}, \ldots, f_{\text{out},r-1}^{(i)}; z_{\text{out},i}) - x\text{cn}_{\text{out},i}$ in R_q^n
end for
for $l = 0, \ldots, M-1$ **do**
    $E_0^{(l)} = \left[\sum_{i=0}^{N-1}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)\text{pk}_{l,i}\right] - \text{Com}_{ck}(\boldsymbol{0}; \boldsymbol{z}^{(l)}) - \sum_{j=1}^{k-1} E_j^{(l)} x^j$ in R_q^n
    where $i = (i_0, \ldots, i_{k-1})$ in base β
    $F_0^{(l)} = x^k \text{s}_l - \boldsymbol{H} \cdot \boldsymbol{z}^{(l)} - \sum_{j=1}^{k-1} F_j^{(l)} x^j$ in $R_q^{n_s}$
end for
for $j = 0, \ldots, N-1$ **do**
    $P_j = \sum_{i=0}^{S-1} \text{cn}_{\text{out},i} - \sum_{i=0}^{M-1} \text{cn}_{i,j} + C$ in R_q^n
end for
$E_0^{(M)} = \left[\sum_{i=0}^{N-1}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)P_i\right] - \text{Com}_{ck}(\boldsymbol{0}; \boldsymbol{z}^{(M)}) - \sum_{j=1}^{k-1} E_j^{(M)} x^j$ in R_q^n
if $x \neq \mathcal{H}(A, B, C, D, E_0^{(0)}, \ldots, E_{k-1}^{(M)}, F_0^{(0)}, \ldots, F_{k-1}^{(M)}, G_0, \ldots, G_{S-1}, \text{s}_0, \ldots, \text{s}_{M-1}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}})$, **then return** False
return True

transmitted to the recipient privately along with the corresponding output amounts and is not revealed publicly.

The verification (Algorithm 10) of a proof performs the same norm checks as in Algorithm 9, computes the "missing" components not output by **Spend** and then checks whether the hash output matches. The missing components are those that are uniquely determined by the rest and thus need not be transferred.

We remark that for our concrete parameters, $\boldsymbol{f}_1$, $f_{j,0}$'s and $\boldsymbol{f}_r$ remain exactly the same whether we see them as elements in $R_{\hat{q}}$ or R_q since their infinity norm is smaller than $q/2 < \hat{q}/2$. Rigorous security proofs of MatRiCT are given in Section 5.

In common with RingCT 1.0 and 2.0, MatRiCT follows the paradigm that all real spent accounts are in the same column of A_{in}. Our protocol could be modified to support anonymity with shuffling (i.e., allowing spent accounts to be in different columns) using techniques from RingCT 3.0. In particular, one would need to re-randomize the input coins, add another balance proof component and include these re-randomized coins and proof elements in the proof output, which would come at the cost of longer transactions.

4.1 Implementation and Parameters

In our implementation, we target any anonymity level $1/N$ for $N \leq 1000$, 64-bit precision for amounts (i.e., $r = 64$) and the most common transaction settings where there are at most two input/output accounts (i.e., $M, S \leq 2$). For all these settings, the following parameters are sufficient: $\mathcal{B} = 1$, $(d, w, p) = (64, 56, 8)$, $q = 2^{31} - 2^{18} + 2^3 + 1$, $\hat{q} = (2^{27} - 2^{11} + 1) \cdot (2^{26} - 2^{12} + 1)$, $k = 1$, $n_s = 1$, $(n, m) = (18, 38)$ and $(\hat{n}, \hat{m}) = (32, 65)$. With these parameters, a single public costs 4.36 KB and a single serial number costs 248 bytes. The rationale behind the parameter setting is as follows.

First, our experimental analysis shows that $d = 64$ is the best choice to optimize the proof length. Having set $d = 64$, we get $(w, p) = (56, 8)$ to have about 256-bit $\mathcal{H}$ output. To measure the practical security of our scheme, we follow the same methodology in [7] and aim for a "root Hermite factor" of $\delta \approx 1.0045$ for both M-LWE and M-SIS. For M-LWE security, we use the commonly used "LWE estimator" in [1], which tells us that $\delta \approx 1.0045$ provides 128-bit post-quantum security. The choice of $\mathcal{B} = 1$ is also commonly practiced in recent lattice-based constructions, e.g., [2, 6–8].

From our security assumption $\text{M-LWE}_{m-n-n_s, m, q, \mathcal{B}}$ in Section 5, we can see that the efficiency of our scheme degrades with increasing n_s as M-LWE gets easier. Indeed, having a small n_s does not affect the anonymity or balance properties. Therefore, we can simply set $n_s = 1$. We discuss the implications of this choice in the full version. Then, we set the remaining parameters to ensure M-LWE and M-SIS requirements in Section 5 are satisfied.

q is chosen to allow R_q to split into 4 factors while having $y = x - x'$ (challenge differences) invertible in R_q for any $x, x' \in C_{w,p}^d$. This follows from the results of [20] as recalled in the full version. The other modulus $\hat{q}$ is chosen to have two "NTT-friendly" prime factors p_1 and p_2 so that both R_{p_1} and R_{p_2} fully splits, allowing efficient polynomial multiplication using NTT. All these primes p_1, p_2 and q are chosen to have a form similar to $2^{k_1} - 2^{k_2} + 1$, which enables the fast modulo reduction technique in [27] for the input smaller than $2^{k_1 - k_2}$ times the modulus. By using this technique, we only apply one modulo reduction at the end when computing $\sum x_i$, $x_i \in R_q$ or $x_i \in R_{\hat{q}}$, such as in the commitments.

To reduce the number of NTT transformations, SamMat samples uniformly at random directly from the NTT domain[11]. Also, all commitment outputs (including pk and cn) are in the NTT domain (i.e., without any inverse NTT during commitment computations). However, since the secrets (notably a, $\boldsymbol{r}$, and $\boldsymbol{\rho}$) are involved in the norm checks of **Spend** in Algorithm 9, and norm checks are also required for the output $\boldsymbol{f}$ and $\boldsymbol{z}$ of **Spend** during **Verify**, we keep

[11] What we mean by NTT domain for R_q is the four factors it splits into.

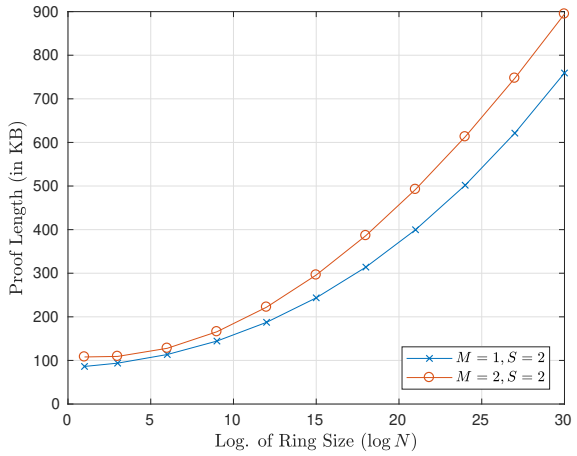

Figure 1: Proof length growth (including the cost of serial numbers) with anonymity set (ring) size.

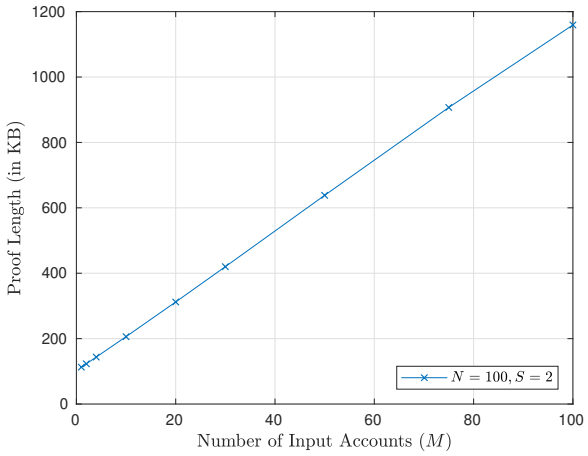

Figure 2: Proof length growth with the number of input accounts. Proof length also includes the cost of serial numbers.

these elements in their normal domain and perform NTT when computing the commitments. Therefore, only forward NTT is needed and we eliminate all the inverse NTT from the implementation.

To accelerate the norm checks and avoid unnecessary overhead during the rejection of **Spend** in Algorithm 9, we adapt the early-evaluation rejection technique in [24]. In particular, we check the infinity or Euclidean norm and restart immediately during each ring element computation of $\boldsymbol{f}_r, \boldsymbol{g}$, and all $\boldsymbol{z}$'s. However, for $\boldsymbol{f}_1$, we need to hide what index may give a rejection due to the application of our rejection sampling technique for fixed Hamming weight binary secrets. Therefore, a restart happens only after $\boldsymbol{f}_1$ is completely iterated over. In addition, since T_g is larger than 64 bits, we use the GMP library [11] to compute $\|\boldsymbol{g}\|$ and make the comparison.

To implement the NTT efficiently, we adapt the techniques discussed in [26] for both factors R_{p_1} and R_{p_2} of $R_{\hat{q}}$ during the NTT butterfly computations, notably the lazy Montgomery reduction. However, for multiplication in R_q, since the input would be reduced to $[0, 2q-1]$ in the lazy reduction, the intermediate value during multiplication reduction may exceed 64 bits for the input less than $4q^2$. Thus, we use the full Montgomery reduction for R_q instead. In addition, we also adapt the constant-time comparison techniques similar to [26] in our NTT implementation and uniform samplers (e.g., $\{-\mathcal{B}, \ldots, \mathcal{B}\}$ or $\{-\mathcal{B}_{\text{big}}, \ldots, \mathcal{B}_{\text{big}}\}$) for the secrets.

In our implementation, we use the AES-NI hardware instructions on Intel CPUs [13] to implement the pseudorandom generator and use the SHAKE-256 [22] to implement the hash function $\mathcal{H}$. We compile our implementation by using GCC 8.3.0 with the compiler switch `-O3 -march=native` during the benchmarks.

The computational evaluation of our construction is given in Table 2 in Section 1, where the running times are the average number of cycles in 1000 runs divided by $3 \cdot 10^6$. Asymptotically, the proof generation and verification times are $O(M \cdot N)$ as M ring signatures are run, each with $O(N)$ computation. Further, we show in Figure 1 that MatRiCT proof length scales very slowly with anonymity set size N. As in RingCT 3.0, the proof length scale linearly with the number of input accounts as shown in Figure 2. Asymptotically, the proof length grows poly-logarithmically in N (due to the use of an improved variant of the ring signature from [7]) and linear in M, i.e., $|\Pi| = O(M \cdot \log^c N)$ for a small constant c. We refer to [7] for more details on the ring signature length growth.

5 SECURITY PROOFS

The correctness of MatRiCT follows from the completeness of the underlying ZKP, and MatRiCT is perfectly correct. The settings of $A, D, E_0^{(l)}, F_0^{(l)}, E_0^{(M)}$ for all $0 \le l \le M-1$ are all done analogous to the one-out-of-many proof in [7]. All the norm checks will be successful as they are all also done in **Spend** algorithm. Also, the underlying one-out-of-many proof allows decoy public commitments not to be well-formed as in [7, 8], and therefore the given correctness requirements are satisfied.

5.1 Anonymity

Lemma 5.1. *(Anonymity) Let $\mathcal{A}$ be a PPT adversary, $\text{Adv}_{\mathcal{A}}^{LWE}$ be the advantage of $\mathcal{A}$ over solving M-LWE$_{m-n-n_s,m,q,\mathcal{B}}$ and $\text{Adv}_{\mathcal{A}}^{LWE_2}$ be the advantage of $\mathcal{A}$ over solving M-LWE$_{\hat{m}-\hat{n},\hat{m},\hat{q},\mathcal{B}}$. The advantage of $\mathcal{A}$ against* `Exp:Anonymity` *without shuffling is at most*

$$\text{Adv}_{\mathcal{A}}^{Ano} \le \text{Adv}_{\mathcal{A}}^{LWE_2} + k(M+1) \cdot \text{Adv}_{\mathcal{A}}^{LWE}.$$

Proof of Lemma 5.1. The proof uses the simulation of the underlying ZKP of our construction where the indistinguishability is either due to an M-LWE assumption or rejection sampling. We use the following succession of games.

Game$_0$: This is identical to `Exp:Anonymity` without shuffling.

Game$_1$: First, the challenger simulates the response where the rejection sampling is applied. In Algorithm 9, it replaces all the coordinates of $\boldsymbol{f}_1$ by uniformly random elements in $\mathfrak{U}_{\mathcal{B}_a-p}$, all the coordinates of $\boldsymbol{z}_b$ by uniformly random elements in $\mathfrak{U}_{\hat{\mathcal{B}}_{\text{big}}-\mathcal{B}pw}$, all the coordinates of $\boldsymbol{z}_c$ by uniformly random elements in $\mathfrak{U}_{\mathcal{B}_{\text{big}}-\mathcal{B}pw}$, all the coordinates of $\boldsymbol{z}^{(i)}$ by uniformly random elements in $\mathfrak{U}_{\mathcal{B}_{\text{big,k}}-\mathcal{B}(pw)^k}$ for all $0 \le i \le M-1$, and all the coordinates of $\boldsymbol{z}^{(M)}$ by uniformly

random elements in $\mathfrak{U}_{\mathcal{B}'_{\mathrm{big,k}}-(M+S+1)\mathcal{B}(pw)^k}$. This game is perfectly indistinguishable from the previous game due to rejection sampling.

$$\mathrm{Adv}^{\mathbf{Game}_0}_{\mathcal{A}} - \mathrm{Adv}^{\mathbf{Game}_1}_{\mathcal{A}} = 0.$$

Game$_2$: In Algorithm 6, the challenger replaces B by a uniformly random element in $R_q^{\hat{n}}$. This game is computationally indistinguishable from the previous game by M-LWE$_{\hat{m}-\hat{n},\hat{m},\hat{q},\mathcal{B}}$ hardness as in the hiding property of the commitment scheme.

$$\left|\mathrm{Adv}^{\mathbf{Game}_1}_{\mathcal{A}} - \mathrm{Adv}^{\mathbf{Game}_2}_{\mathcal{A}}\right| \leq \mathrm{Adv}^{\mathrm{LWE}_2}_{\mathcal{A}}.$$

Game$_3$: In Algorithm 8, the challenger replaces C by a uniformly random element in R_q^n. This game is computationally indistinguishable from the previous game by M-LWE$_{m-n,m,q,\mathcal{B}}$ hardness.

$$\left|\mathrm{Adv}^{\mathbf{Game}_2}_{\mathcal{A}} - \mathrm{Adv}^{\mathbf{Game}_3}_{\mathcal{A}}\right| \leq \mathrm{Adv}^{\mathrm{LWE}}_{\mathcal{A}}.$$

Game$_4$: In Algorithm 4, the challenger replaces serial number s_i's by a uniformly random element in $R_q^{n_s}$ and the public keys $\mathsf{pk}_{i,\ell}$'s in R_{in} (i.e., the ℓ-th column of A_{in}) by a uniformly random element in R_q^n for $i = 0, \ldots, M-1$. This game is computationally indistinguishable from the previous game by M-LWE$_{m-n-n_s,m,q,\mathcal{B}}$ hardness due to the following observation.

Let $\boldsymbol{G}' := \begin{pmatrix} \boldsymbol{G} \\ \boldsymbol{H} \end{pmatrix}$. We have $\begin{pmatrix} \mathsf{pk}_{i,\ell} \\ \mathsf{s}_i \end{pmatrix} = \boldsymbol{G}' \cdot \boldsymbol{r}_{i,\ell}$ where $\boldsymbol{r}_{i,\ell}$ is the secret key corresponding to the public key $\mathsf{pk}_{i,\ell}$. Since $\boldsymbol{G}'$ has the same distribution as a commitment key ck output by CKeygen, the hiding property argument for the commitment also holds with respect to the combined matrix $\boldsymbol{G}' \in R_q^{(n+n_s)\times m}$. Also, note that no Corrupt or Spend is allowed to be queried for these public keys, and the distribution of the secret keys $\boldsymbol{r}_{i,\ell}$ is identical to that in M-LWE definition since the public keys in R_{in} are assumed to be generated honestly by querying PkGen.

$$\left|\mathrm{Adv}^{\mathbf{Game}_3}_{\mathcal{A}} - \mathrm{Adv}^{\mathbf{Game}_4}_{\mathcal{A}}\right| \leq M \cdot \mathrm{Adv}^{\mathrm{LWE}}_{\mathcal{A}}.$$

Game$_5$: In Algorithm 7, the challenger replaces R_j by a uniformly random element in R_q^n, and F_j by a uniformly random element in $R_q^{n_s}$ for all $1 \leq j \leq k-1$ (if GenSerial = False, then only R_j is replaced and the argument still works). This game is computationally indistinguishable from the previous game by M-LWE$_{m-n-n_s,m,q,\mathcal{B}}$ hardness due to a similar discussion as above.

$$\left|\mathrm{Adv}^{\mathbf{Game}_4}_{\mathcal{A}} - \mathrm{Adv}^{\mathbf{Game}_5}_{\mathcal{A}}\right| \leq (M+1)(k-1) \cdot \mathrm{Adv}^{\mathrm{LWE}}_{\mathcal{A}}.$$

Game$_6$: In Algorithm 7, the challenger replaces E_j by a uniformly random element in R_q^n for all $1 \leq j \leq k-1$. This game is perfectly indistinguishable from the previous game as R_j is uniformly random in R_q^n and independent of the summation in Step 7 of Algorithm 7.

$$\left|\mathrm{Adv}^{\mathbf{Game}_5}_{\mathcal{A}} - \mathrm{Adv}^{\mathbf{Game}_6}_{\mathcal{A}}\right| = 0.$$

Note that **Mint** is completely independent of all the inputs to **Spend** except for output amounts, which is already known to $\mathcal{A}$. Also, output coin keys $\mathsf{CK}_{\mathrm{out}}$ are always generated independently and uniformly at random. Therefore, in **Game**$_6$, the output of **Spend** is independent of R_{in}, K_{in} and $\mathsf{Amt}_{\mathrm{in}}$, and thus also independent of b. Hence, $\mathcal{A}$ has probability $1/2$ of outputting $b' = b$ in **Game**$_6$. □

5.2 Balance

To calculate the binding requirements of the commitment scheme, the norms of some proof components are bounded in the following sequences of lemmas, whose proofs are provided in the full version.

Lemma 5.2. *The following holds for the vector $\boldsymbol{g}$ in Algorithms 9 and 10*

$$\|\boldsymbol{g}\|^2 \leq d^3\left(\mathcal{B}_a^4 k\beta(\beta+1) + \mathcal{B}_r^4 r(S+1)\right).$$

Lemma 5.3. *The extracted opening $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{r}}_a)$ of A for A defined in Algorithms 8 and 10 satisfies the following*

$$\|(\hat{\boldsymbol{a}}, \hat{\boldsymbol{r}}_a)\| \leq 2p\sqrt{dw}\left(T_g + \hat{\mathcal{B}}_{big}^2 \hat{m} d\right)^{1/2}.$$

Lemma 5.4. *The extracted opening $(\hat{\boldsymbol{c}}, \hat{\boldsymbol{r}}_c)$ of C for C defined in Algorithms 8 and 10 satisfies the following*

$$\|(\hat{\boldsymbol{c}}, \hat{\boldsymbol{r}}_c)\| \leq 2\left(9r\mathcal{B}_r^2 d + \mathcal{B}_{big}^2 m d\right)^{1/2}.$$

Further, the same bound as above also holds for the Euclidean norm of an extracted opening of $\mathsf{cn}_{\mathrm{out},i}$ for any $0 \leq i \leq S-1$.

We then summarize the relations proved by our binary ZKP and one-out-of-many ZKP. An improved soundness proof for the binary ZKP is given in the full version.

Lemma 5.5. *Assume that the following holds*

- $\hat{q}/2 > \max\left\{2pwd\mathcal{B}_f(p+\mathcal{B}_f), 2pw\mathcal{B}_a^2 d\beta\right\}$ *for* $\mathcal{B}_f = \max\{\mathcal{B}_a, \mathcal{B}_r\}$,
- *HMC is* γ_{bin}*-binding for* $\gamma_{\mathrm{bin}} = 2p\sqrt{dw}\left(T_g + \hat{\mathcal{B}}_{big}^2 \hat{m} d\right)^{1/2}$.

For an input commitment $B \in R_{\hat{q}}^{\hat{n}}$, a commitment key ck and proof output $(A, x, \boldsymbol{f}_1, \boldsymbol{f}_r, z_b)$, our binary proof proves knowledge of $(y, \boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})$ such that

- $y \in \Delta C_{w,p}^d$,
- $yB = \mathrm{Com}_{ck}(y\boldsymbol{b}, \hat{\boldsymbol{c}}; \hat{\boldsymbol{r}})$,
- *All coordinates b_i of $\boldsymbol{b}$ is in $\{0,1\}$,*
- *$\hat{\boldsymbol{c}}$ is uniquely determined by $y, \boldsymbol{f}_1, \boldsymbol{f}_r, x$ and $\boldsymbol{b}$,*
- *For the first $k\beta$ coordinates $b_{0,0}, \ldots, b_{k-1,\beta-1}$ of $\boldsymbol{b}$, $\sum_{i=0}^{\beta-1} b_{j,i} = 0$, i.e., there is only a single 1 in $\{b_{i,0}, \ldots, b_{i,\beta-1}\}$ for all $0 \leq i \leq k-1$,*
- $\|(y\boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})\| \leq \gamma_{\mathrm{bin}}$.

Lemma 5.6. *Assume that $q > (2p\sqrt{K})^K$ and $q \equiv 2K+1 \bmod 4K$ for some $1 < K \leq d$ where K is a power of 2. On input a commitment key ck and a set of commitments $(P_0, \ldots, P_{N-1})$, the underlying one-out-of-many proof of our ring signature proves knowledge of $(y, \ell, \hat{\boldsymbol{r}})$ such that*

- $\ell \in \{0, \ldots, N-1\}$,
- $yP_\ell = \mathrm{Com}_{ck}(\boldsymbol{0}; \hat{\boldsymbol{r}})$,
- *y is a product of κ elements in $\Delta C_{w,p}^d$ for $\kappa = k(k+1)/2$, and $\|y\| \leq \sqrt{d} \cdot (2p)^\kappa w^{\kappa-1}$,*
- $\|\hat{\boldsymbol{r}}\| \leq (k+1) \cdot d \cdot (2p)^{\kappa'} w^{\kappa'-1}\sqrt{md} \cdot \max\{\mathcal{B}_{big,k}, \mathcal{B}'_{big,k}\}$.

Further, the proof is k'-special sound where $k' = \max\{k+1, 3\}$.

Proof. The first three properties and the fact that one-out-of-many proof is k'-special sound directly follow from the results of [7], and the remaining property is shown in the full version. □

Lemma 5.7. *(Balance) Assume that* $q > \left(2p\sqrt{K}\right)^K$ *and* $q \equiv 2K+1 \bmod 4K$ *for some* $1 < K \leq d$ *where* K *is a power of 2. Let* $\kappa = k(k+1)/2$ *and* θ *be a positive real number such that the Euclidean norm of any product of* $\kappa - 1$ *elements in* $\Delta\mathcal{C}^d_{w,p}$ *is at most* θ. *If M-LWE*$_{m-n-n_s,m,q,\mathcal{B}}$, *M-SIS*$_{n,m+r,q,2\gamma}$ *and M-SIS*$_{\hat{n},\hat{m}+\upsilon,\hat{q},2\gamma_{\text{bin}}}$ *are hard where* $\upsilon = 2\left(k(\beta-1)+r-1+Sr\right)$,

$\gamma_{\text{bin}} = 2p\sqrt{dw}\left(T_g + \hat{\mathcal{B}}^2_{big}\hat{m}d\right)^{1/2}$ *and*

$$\gamma = \max\left\{\begin{array}{c}(k+1)\cdot d\cdot(2p)^{\kappa'}w^{\kappa'-1}\sqrt{md}\cdot\max\{\mathcal{B}_{big,k},\mathcal{B}'_{big,k}\},\\ \theta\sqrt{d}\cdot(S+1)\cdot 2\left(9r\mathcal{B}^2_r d+\mathcal{B}^2_{big}md\right)^{1/2}\end{array}\right\},$$

then no PPT adversary can win Exp:Balance *without shuffling with non-negligible probability.*

Proof of Lemma 5.7. First, due to the M-SIS assumptions, HMC is γ-binding when instantiated with parameters $n, m, q, \mathcal{B}$ and γ_{bin}-binding when instantiated with parameters $\hat{n}, \hat{m}, \hat{q}, \mathcal{B}$. We separate the proof into three cases.

Case 1 (forgery): Let $\mathcal{E}_{\text{forge}}$ be the event that $\mathcal{A}$ wins the game in a way that there exists $\mathsf{s}_{i^*} \in \mathsf{SN}_{j^*}$ with $0 \leq i^* \leq M-1$ and $1 \leq j^* \leq t$ such that $\mathsf{s}_{i^*} \in \mathcal{L}$ and $\mathcal{L}[\mathsf{s}_{i^*}].\mathsf{IsCrpt} = 0$. In this case, the proof follows as in the unforgeability proof of the ring signature in [7], which is sketched below (see also the proof of Theorem 3 in [8] for more details).

$\mathcal{A}'$ creates an invalid public key pk_ℓ for PkGen(ℓ) query such that $\mathsf{pk}_\ell = \mathrm{Com}_{ck}(1,0,\ldots,0;\boldsymbol{r})$ for $\boldsymbol{r} \leftarrow \{-\mathcal{B},\ldots,\mathcal{B}\}^{d\cdot m}$. pk_ℓ is computationally indistinguishable from a valid public key by M-LWE$_{m-n,m,q,\mathcal{B}}$ hardness assumption. $\mathcal{A}'$ runs $\mathcal{A}$ until $\mathcal{E}_{\text{forge}}$ occurs $k' = \max\{k+1,3\}$ times in total with respect to distinct $\mathcal{H}$ outputs and the same $\mathcal{H}$ inputs where

- the indices j^* and i^* are the same for all k' $\mathcal{E}_{\text{forge}}$ events,
- $\mathsf{pk}_\ell \in \mathsf{A}^{j^*}_{\text{in}}$ and it is not corrupted.

k'-special soundness of the underlying one-out-of-many proof holds when HMC is γ- and γ_{bin}-binding, which is satisfied if M-SIS$_{n,m+r,q,2\gamma}$ and M-SIS$_{\hat{n},\hat{m}+\upsilon,\hat{q},2\gamma_{\text{bin}}}$ are hard. Therefore, there exists a PPT extractor that recovers an opening $(\mathbf{0},\boldsymbol{s})$ of a public key pk_ψ in the i^* row of $\mathsf{A}^{j^*}_{\text{in}}$ such that $y\cdot\mathsf{pk}_\psi = \mathrm{Com}_{ck}(\mathbf{0};\boldsymbol{s})$ where $y \in \Delta\mathcal{C}^d_{w,p}$ is some relaxation factor with $\|y\| = \sqrt{d}(2p)^\kappa w^{\kappa-1} \ll \gamma$ and $\|\boldsymbol{s}\| \leq \gamma$ by Lemma 5.6. With probability $1/(M\cdot N)$, $\mathsf{pk}_\ell = \mathsf{pk}_\psi$. Hence, $y\cdot\mathsf{pk}_\ell = \mathrm{Com}_{ck}(y,0,\ldots,0;y\boldsymbol{r}) = \mathrm{Com}_{ck}(\mathbf{0};\boldsymbol{s}) = y\cdot\mathsf{pk}_\psi$. Since $(y,0,\ldots,0,y\boldsymbol{r}) \neq (\mathbf{0},\boldsymbol{s})$, this violates the γ-binding property of the commitment scheme and also gives a solution to M-SIS$_{n,m,q,2\gamma}$, which gives a contradiction.

Case 2 (double-spend): Let $\mathcal{E}_{\text{2xspend}}$ be the event that $\mathcal{A}$ wins the game in a way that there exists $\mathsf{s}_{i^*} \in \mathsf{SN}_{j^*}$ with $0 \leq i^* \leq M-1$ and $1 \leq j^* \leq t$ such that $\mathsf{s}_{i^*} \notin \mathcal{L}$. Assume that the assumptions in the statement of the lemma hold and $\mathcal{E}_{\text{2xspend}}$ happens. We show that this gives a contradiction. Since the transactions output by $\mathcal{A}$ are valid, we have from Algorithm 10

$$\boldsymbol{G}\cdot\boldsymbol{z}^{(i^*)} = \sum_{i=0}^{N}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)\mathsf{pk}_{i^*,i} - \sum_{j=0}^{k-1} E_j^{(i^*)}x^j, \tag{7}$$

$$\boldsymbol{H}\cdot\boldsymbol{z}^{(i^*)} = x^k\mathsf{s}_{i^*} - \sum_{j=0}^{k-1} F_j^{(i^*)}x^j, \tag{8}$$

where $\mathsf{pk}_{i^*,i}$ is an honestly generated public key for all $i \in [0,N-1]$. Again using the extractor of the underlying ZKP as in Case 1, $\mathcal{A}'$, who runs $\mathcal{A}$ multiple times, obtains a witness $\boldsymbol{s}$ such that

$$y\cdot\mathsf{pk}_{i^*,\ell} = \boldsymbol{G}\cdot\boldsymbol{s}, \tag{9}$$

$$y\cdot\mathsf{s}_{i^*} = \boldsymbol{H}\cdot\boldsymbol{s}, \tag{10}$$

for some $0 \leq \ell \leq N-1$ where y is a product of κ elements in $\Delta\mathcal{C}^d_{w,p}$ with $\|y\| \ll \gamma$ and $\|\boldsymbol{s}\| \leq \gamma$ by Lemma 5.6. Since $\mathsf{pk}_{i^*,\ell}$ is an honestly generated public key, we also have

$$\begin{array}{r}\mathsf{pk}_{i^*,\ell} = \boldsymbol{G}\cdot\boldsymbol{r}_\ell\\ \mathsf{s}_\ell = \boldsymbol{H}\cdot\boldsymbol{r}_\ell\end{array} \quad\Longrightarrow\quad y\cdot\mathsf{pk}_{i^*,\ell} = \boldsymbol{G}\cdot y\boldsymbol{r}_\ell \tag{11}$$

for some $\boldsymbol{r}_\ell \in R_q^m$ with $\|\boldsymbol{r}_\ell\|_\infty = \mathcal{B}$ where $\mathsf{s}_\ell = \mathcal{L}[\mathsf{pk}_{i^*,\ell}].\mathsf{s}$. Using (9), right side of (11) and γ-binding of HMC with respect to $\boldsymbol{G}$, we get $\boldsymbol{s} = y\boldsymbol{r}_\ell$. Then, from (10), we get

$$y\cdot\mathsf{s}_{i^*} = \boldsymbol{H}\cdot y\boldsymbol{r}_\ell \quad\Longrightarrow\quad \mathsf{s}_{i^*} = \boldsymbol{H}\cdot\boldsymbol{r}_\ell \tag{12}$$

since y is invertible because all its factors are invertible by the assumption on q and [20, Corollary 1.2]. From right side of (12) and left side of (11), we conclude that $\mathsf{s}_{i^*} = \mathsf{s}_\ell \in \mathcal{L}$, which gives a contradiction.

Case 3 (unbalanced amounts): Let $\mathcal{E}_{\text{unbalanced}}$ be the event that $\mathcal{A}$ wins the game in a way that for all $\mathsf{s}_i \in \mathsf{SN}_{j^*}$ where $0 \leq i \leq M-1$, $\mathsf{s}_i \in \mathcal{L}$ and $\mathcal{L}[\mathsf{s}_i].\mathsf{IsCrpt} = 1$. Assume that the assumptions in the statement of the lemma hold and there exists a PPT $\mathcal{A}'$ who runs $\mathcal{A}$.

As in Case 1, $\mathcal{A}'$ runs $\mathcal{A}$ until $\mathcal{E}_{\text{unbalanced}}$ occurs $k' = \max\{k+1,3\}$ times with respect to distinct $\mathcal{H}$ outputs and the same $\mathcal{H}$ inputs where the index j^* is the same for all k' events. Then, it uses the extractor of the underlying ZKP of the j^*-th transaction to obtain the following, for all $i \in [0,S-1]$,

$$\bar{x}\cdot C = \mathrm{Com}_{ck}(\bar{x}c_0 - \bar{x}2c_1,\ldots,\bar{x}c_{r-1} - \bar{x}2c_r;\boldsymbol{r}_c), \tag{13}$$

$$\bar{x}\cdot\mathsf{cn}_{\text{out},i} = \mathrm{Com}_{ck}(\bar{x}b^{(i)}_{\text{out},0},\ldots,\bar{x}b^{(i)}_{\text{out},r-1};\boldsymbol{r}_{\text{out},i}), \tag{14}$$

$$y\cdot P_\ell = \mathrm{Com}_{ck}(0;\boldsymbol{r}),\ \text{for } P_\ell = \sum_{j=0}^{S-1}\mathsf{cn}_{\text{out},j} - \sum_{j=0}^{M-1}\mathsf{cn}_{j,\ell} + C \tag{15}$$

where

- $c_0 = c_r = 0$ and $c_1,\ldots,c_{r-1} \in [-(M-1),(S-1)]$,[12]
- $\bar{x} \in \Delta\mathcal{C}^d_{w,p}$,
- y is a product of κ elements in $\Delta\mathcal{C}^d_{w,p}$ where one of its factors is $\bar{x}$ by Lemma 5.6,
- $\|\boldsymbol{r}\| \leq \gamma$ by Lemma 5.6,
- $\|(\bar{x}c_0 - \bar{x}2c_1,\ldots,\bar{x}c_{r-1} - \bar{x}2c_r,\boldsymbol{r}_c)\| \leq \gamma/((S+1)\theta\sqrt{d})$, $\|\bar{x}b^{(i)}_{\text{out},0},\ldots,\bar{x}b^{(i)}_{\text{out},r-1},\boldsymbol{r}_{\text{out},i}\| \leq \gamma/((S+1)\theta\sqrt{d})$ by Lemma 5.4,
- $b^{(i)}_{\text{out},j} \in \{0,1\}$ for all $i \in \{0,\ldots,S-1\}$ and all $j \in \{0,\ldots,r-1\}$.

Multiplying (13) and (14) by $y' = y/\bar{x}$, we get

$$y\cdot C = \mathrm{Com}_{ck}(yc_0 - y2c_1,\ldots,yc_{r-1} - y2c_r;\ y'\boldsymbol{r}_c), \tag{16}$$

$$y\cdot\mathsf{cn}_{\text{out},i} = \mathrm{Com}_{ck}(yb^{(i)}_{\text{out},0},\ldots,yb^{(i)}_{\text{out},r-1};\ y'\boldsymbol{r}_{\text{out},i}), \tag{17}$$

[12] Here, we assume the general case where the spender proves that c_i's are in $[-(M-1),(S-1)]$, and need not be necessarily binary.

where $\|(yc_0 - y2c_1, \ldots, yc_{r-1} - y2c_r, y'\boldsymbol{r}_c)\| \leq \gamma/(S+1)$, and $\|yb^{(i)}_{\text{out},0}, \ldots, yb^{(i)}_{\text{out},r-1}, y'\boldsymbol{r}_{\text{out},i}\| \leq \gamma/(S+1)$.

Since the input coins are generated honestly, we also have

$$\mathsf{cn}_{i,\ell} = \mathrm{Com}_{ck}(b_{i,0}, \ldots, b_{i,r-1}; \boldsymbol{r}_i) \tag{18}$$

where $\|\boldsymbol{r}_i\|_\infty = \mathcal{B}$ and $b_{i,j} \in \{0,1\}$ for all $i \in \{0,\ldots,M-1\}$ and all $j \in \{0,\ldots,r-1\}$. Substituting (16), (17) and (18) into (15), we get

$$\begin{aligned}\mathrm{Com}_{ck}(0;\boldsymbol{r}) = &\sum_{i=0}^{S-1}\left(\mathrm{Com}_{ck}(yb^{(i)}_{\text{out},0}, \ldots, yb^{(i)}_{\text{out},r-1}; y'\boldsymbol{r}_{\text{out},i})\right)\\ &- \sum_{i=0}^{M-1}\left(\mathrm{Com}_{ck}(yb_{i,0}, \ldots, yb_{i,r-1}; y\boldsymbol{r}_i)\right)\\ &+ \mathrm{Com}_{ck}(yc_0 - y2c_1, \ldots, yc_{r-1} - y2c_r; y'\boldsymbol{r}_c).\end{aligned}$$

Observe that the input of the commitment on the left hand side has Euclidean norm at most γ. Similarly, after using the homomorphic properties of the commitment scheme, the input of the commitment on the right hand side has norm at most γ (here we neglect the norm of $(yb_{i,0}, \ldots, yb_{i,r-1}, y\boldsymbol{r}_i)$ as that is much smaller in comparison). Then, using γ-binding property of HMC, we get

$$0 = y\sum_{i=0}^{S-1} b^{(i)}_{\text{out},j} - y\sum_{i=0}^{M-1} b_{i,j} + yc_j - y2c_{j+1} \tag{19}$$

for all $j \in \{0,\ldots,r-1\}$ with $c_0 = c_r = 0$. By the assumption on q and Lemma [20, Corollary 1.2], y is invertible in R_q, and we have

$$0 = \sum_{i=0}^{S-1} b^{(i)}_{\text{out},j} - \sum_{i=0}^{M-1} b_{i,j} + c_j - 2c_{j+1}, \tag{20}$$

where with $c_0 = c_r = 0$. Since HMC is γ-binding, we have $q > \gamma \gg \max\{4M, 4S\}$. Hence, (20) holds over R. Since all the values are just integers, (20) in fact holds over $\mathbb{Z}$.

By the definition of `Exp:Balance`, the sum of the amounts in $\mathsf{Amt}^{j^*}_{\text{out}}$ (i.e., the amounts corresponding to *uncorrupted* output public keys) can be at most the sum of the amounts in all output coins. Using this fact, we look at the following sum where $\mathsf{amt}_{\text{in},i} = \mathcal{L}[\mathsf{s}_i].\mathsf{amt}$ for $\mathsf{s}_i = \mathsf{SN}_{j^*}[i]$

$$\begin{aligned}\sum_{i=0}^{S'-1}\mathsf{Amt}^{j^*}_{\text{out}}[i] - \sum_{i=0}^{M-1}\mathsf{amt}_{\text{in},i} &\leq \sum_{i=0}^{S-1}\sum_{j=0}^{r-1} 2^j b^{(i)}_{\text{out},j} - \sum_{i=0}^{M-1}\sum_{j=0}^{r-1} 2^j b_{i,j}\\ &= \sum_{j=0}^{r-1} 2^j\left(\sum_{i=0}^{S-1} b^{(i)}_{\text{out},j} - \sum_{i=0}^{M-1} b_{i,j}\right) = \sum_{j=0}^{r-1} 2^j\left(2c_{j+1} - c_j\right)\\ &= \sum_{j=0}^{r-1} 2^{j+1}c_{j+1} - \sum_{j=0}^{r-1} 2^j c_j = 2^r c_r - c_0 = 0, \quad \text{since } c_0 = c_r = 0.\end{aligned}$$

The above implies $\sum_{i=0}^{S'-1}\mathsf{Amt}^{j^*}_{\text{out}}[i] \leq \sum_{i=0}^{M-1}\mathsf{amt}_{\text{in},i}$, giving a contradiction with the winning assumption of $\mathcal{A}$ in `Exp:Balance`. □

Remark 1. *Note that in Lemma 5.7, the factor $\theta\sqrt{d}$ can be taken to be 1, when $k = 1$. This is due to fact that in this case, $\kappa = 1$, and thus $y = \bar{x}$. Hence, there is no need to do cross multiplication to have (13), (14) and (15) multiplied by the same relaxation factor y.*

6 EXTENSION TO AUDITABLE RINGCT

6.1 Extractable Commitment Scheme

We extend HMC to allow message extraction. All previous commitment algorithms, CKeygen, Commit and COpen, remain the same, and we introduce how to put a trapdoor to a commitment key.

- CAddTrapdoor(ck) : Let $ck = [\boldsymbol{A} \| \boldsymbol{B}] \in R_q^{n\times(m+\upsilon)}$ where $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{A}'\\ \boldsymbol{a}^\top\end{bmatrix}$ for $\boldsymbol{A}' \in R_q^{(n-1)\times m}$ and $\boldsymbol{a} \in R_q^m$. Sample $\boldsymbol{s}' \leftarrow R_q^{n-1}$, $\boldsymbol{e} \leftarrow \mathfrak{U}^m_{\mathcal{B}_e}$, and set $\boldsymbol{A}^{\text{td}} = \begin{bmatrix}\boldsymbol{A}'\\ \boldsymbol{t}^\top\end{bmatrix}$ where $\boldsymbol{t} = \boldsymbol{A}'^\top\boldsymbol{s}' + \boldsymbol{e}$. Output $(ck^{\text{td}}, \text{td}) = (\boldsymbol{A}^{\text{td}}, (\boldsymbol{s}, \boldsymbol{e}))$ where $\boldsymbol{s} = (\boldsymbol{s}', -1)$.

Let $\Delta C^d_{w,p}$ be the set of differences of all challenges in $C^d_{w,p}$ except for the zero element. When a commitment key with a trapdoor is used to generate a proof, the ZKPs we use proves knowledge of an opening $(y, \boldsymbol{m}, \boldsymbol{r})$ of a commitment C such that

$$yC = \mathrm{Com}_{ck}(y\boldsymbol{m}; \boldsymbol{r}) = \boldsymbol{A}^{\text{td}}\boldsymbol{r} + \boldsymbol{B}y\boldsymbol{m}. \tag{21}$$

From here, we can try to eliminate the randomness by multiplying both sides by the secret key $\boldsymbol{s}$. However, the message extractor does not know what y is. For an honest user, we simply have $y = 1$ and we restrict our discussion in this paper to this case. However, we note that, for a similar Fiat-Shamir protocol, it has been shown in [19] that a valid approach in general is actually trying random $y \in \Delta C^d_{w,p}$, and the expected number of iterations until an acceptable y is reached is the same as the number of random oracle queries made to generate the proof. We believe that the same technique (which is also used in [6]) and [19, Lemma 3.2] can be applied in our case and we leave its detailed investigation to future work.

Now, suppose that $y = 1$. We can rewrite (21) as $C = \boldsymbol{A}^{\text{td}}\boldsymbol{r} + \boldsymbol{B}\boldsymbol{m}$. From here, the extraction proceeds as outlined in Section 1.3 and the full procedure is provided in Algorithm 11. We prove in Lemma 6.1 that, for a commitment C with a valid zero-knowledge proof of opening, the message output by Algorithm 11 is the same as the one used to create the commitment C for sufficiently large q.

Algorithm 11 CExtractSM(C, td)

INPUT: $C \in R_q^n$ a commitment; td $= (\boldsymbol{s}, \boldsymbol{e})$ trapdoor

1: **for** $\boldsymbol{m}' \in \mathcal{M}$ **do** ▷ where $|\mathcal{M}| = \text{poly}(\lambda)$
2: $\quad e' = \langle \boldsymbol{s}, C\rangle - \langle \boldsymbol{b}, \boldsymbol{m}'\rangle$ where $\boldsymbol{b} = \boldsymbol{s}^\top\boldsymbol{B}$
3: $\quad$ **if** $\|e'\|_\infty < q/8$, **then return** $\boldsymbol{m}'$
4: **end for**

Lemma 6.1. *Let $ck = \boldsymbol{G} = [\boldsymbol{A}^{\text{td}} \| \boldsymbol{B}] \in R_q^{n\times(m+\upsilon)}$ be a commitment key with a trapdoor* td $= (\boldsymbol{s}, \boldsymbol{e})$ *as in* CAddTrapdoor. *Assume that $(C, (\boldsymbol{m}, \boldsymbol{r}))$ satisfy $C = \mathrm{Com}_{ck}(\boldsymbol{m}; \boldsymbol{r})$, $\|\boldsymbol{r}\|_\infty \leq \mathcal{B}_r$ and $\boldsymbol{m} \in \mathcal{M}$ with $|\mathcal{M}| = s$, and $\boldsymbol{m}' =$* CExtractSM$(C,$ td$)$. *If $q > 8\mathcal{B}_e\mathcal{B}_r md$, then $\boldsymbol{m} = \boldsymbol{m}'$ except for a probability at most $s \cdot 2^{-d}$.*

Proof of Lemma 6.1. Observe that $\boldsymbol{s}^\top \cdot \boldsymbol{A}^{\text{td}} = \boldsymbol{s}'^\top\boldsymbol{A}' - \boldsymbol{t}^\top = -\boldsymbol{e}^\top$. Let $\boldsymbol{b} = \boldsymbol{s}^\top \cdot \boldsymbol{B}$. Since $C = \mathrm{Com}_{ck}(\boldsymbol{m}; \boldsymbol{r})$ for $ck = [\boldsymbol{A}^{\text{td}} \| \boldsymbol{B}]$, we have

$$\begin{aligned}&\langle \boldsymbol{s}, C\rangle = \langle -\boldsymbol{e}, \boldsymbol{r}\rangle + \boldsymbol{s}^\top \cdot \boldsymbol{B} \cdot \boldsymbol{m} = \langle -\boldsymbol{e}, \boldsymbol{r}\rangle + \langle \boldsymbol{b}, \boldsymbol{m}\rangle,\\ \iff\quad &\langle \boldsymbol{b}, \boldsymbol{m}\rangle = \langle \boldsymbol{s}, C\rangle + \langle \boldsymbol{e}, \boldsymbol{r}\rangle.\end{aligned} \tag{22}$$

Since $\boldsymbol{m}'$ is the output of CExtractSM(C, td), we further have $e' = \langle \boldsymbol{s}, C\rangle - \langle \boldsymbol{b}, \boldsymbol{m}'\rangle$ and $\|e'\|_\infty < q/8$. Now, consider the following

$$\begin{aligned}\langle \boldsymbol{b}, \boldsymbol{m} - \boldsymbol{m}'\rangle &= \langle \boldsymbol{b}, \boldsymbol{m}\rangle - \langle \boldsymbol{b}, \boldsymbol{m}'\rangle = \langle \boldsymbol{s}, C\rangle + \langle \boldsymbol{e}, \boldsymbol{r}\rangle - \langle \boldsymbol{s}, C\rangle + e' \\ &= \langle \boldsymbol{e}, \boldsymbol{r}\rangle + e'.\end{aligned}$$

Therefore, we have

$$\begin{aligned}\|\langle \boldsymbol{b}, \boldsymbol{m} - \boldsymbol{m}'\rangle\|_\infty &= \|\langle \boldsymbol{e}, \boldsymbol{r}\rangle + e'\|_\infty \le \|\langle \boldsymbol{e}, \boldsymbol{r}\rangle\|_\infty + \|e'\|_\infty \\ &\le \mathcal{B}_e \mathcal{B}_r md + q/8 < q/8 + q/8 < q/4.\end{aligned}$$

Since $\boldsymbol{B}$ is chosen independently and uniformly at random, $\boldsymbol{b} = \boldsymbol{s}^\top \cdot \boldsymbol{B}$ is uniformly random, and thus when $\boldsymbol{m} \neq \boldsymbol{m}'$, $\langle \boldsymbol{b}, \boldsymbol{m} - \boldsymbol{m}'\rangle$ is also uniformly random in R_q. So, the above holds with probability about 2^{-d}. Thus, using a union bound on all $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{m} = \boldsymbol{m}'$ except for a probability at most $s \cdot 2^{-d}$. □

6.2 Adding Auditability

From the tools developed so far, it is now easy to add auditability to our RingCT construction. As part of **Spend**, the spender proves knowledge of an index $\ell \in [0, N-1]$ and the secret keys of the accounts in the ℓ-th column of A_{in}. Further, as given in Lemma 5.5, the binary proof part proves knowledge of $(y, \boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})$ such that $y \in \Delta C^d_{w,p}$, $yB = \mathrm{Com}_{ck}(y\boldsymbol{b}, \hat{\boldsymbol{c}}; \hat{\boldsymbol{r}}) \in R_{\hat{q}}^{\hat{n}}$ and the first $k\beta$ elements of $\boldsymbol{b}$ represents an index $\ell \in [0, N-1]$. Hence, we know that

$$yB = \boldsymbol{A}\hat{\boldsymbol{r}} + \boldsymbol{B}\begin{pmatrix} y\boldsymbol{b} \\ \boldsymbol{c}\end{pmatrix} = \boldsymbol{A}\hat{\boldsymbol{r}} + \boldsymbol{B}_0 y\boldsymbol{m} + \boldsymbol{B}_1\hat{\boldsymbol{m}} = [\,\boldsymbol{A} \,\|\, \boldsymbol{B}_1\,]\begin{pmatrix}\hat{\boldsymbol{r}} \\ \hat{\boldsymbol{m}}\end{pmatrix} + \boldsymbol{B}_0 y\boldsymbol{m},$$

where $\boldsymbol{m}$ is the part (the first $k\beta$ elements of $\boldsymbol{b}$) we want to recover and $\hat{\boldsymbol{m}}$ is the remaining message opening part. Therefore, restricting to $y = 1$, we can put a trapdoor for the concatenated matrix $[\,\boldsymbol{A} \,\|\, \boldsymbol{B}_1\,]$ and use Algorithm 11 to extract $\boldsymbol{m}$, which reveals the real spender's identity. The message space size here is equal to the anonymity set size N. Therefore, the extraction time (as in **Spend**) is linear in N. We know by Lemma 6.1 that for an appropriately chosen $\hat{q}$, the extracted index will be the same as the one used in the proof. A formal definition of auditability, which can be established similar to traceability in group signatures [3], is left as a future work.

Note that multiple trapdoors can be put for the same matrix. If no auditing is desired, the last row of the commitment matrix remains as a uniformly random vector. If a user selects auditing option $i > 0$, then a vector released by the i-th authority is used in the last row of the commitment matrix.

ACKNOWLEDGMENTS

Ron Steinfeld and Joseph K. Liu were supported in part by ARC Discovery Project grant DP180102199.

REFERENCES

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology* 9, 3 (2015), 169–203.
[2] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. 2018. More efficient commitments from structured lattice assumptions. In *SCN*. Springer, 368–385.
[3] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. 2003. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT (LNCS)*. Springer, 614–629.
[4] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. 2015. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS*. Springer, 305–325.
[5] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. 2015. Short accountable ring signatures based on DDH. In *ESORICS*. Springer, 243–265.
[6] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. 2018. Lattice-Based Group Signatures and Zero-Knowledge Proofs of Automorphism Stability. In *ACM CCS*. ACM, 574–591.
[7] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. 2019. Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications. In *CRYPTO (1) (LNCS)*. Springer, 115–146. (Full version at https://eprint.iacr.org/2019/445).
[8] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. 2019. Short Lattice-Based One-out-of-Many Proofs and Applications to Ring Signatures. In *ACNS (LNCS)*. Springer, 67–88. (Full version at https://eprint.iacr.org/2018/773).
[9] Abelian Foundation. 2018. Abelian Coin (ABE) – A Quantum-Resistant Cryptocurrency Balancing Privacy and Accountability. (2018). https://www.abelianfoundation.org/wp-content/uploads/2018/08/Abelian-Whitepaper-CB20180615.pdf (June 15, 2018 version).
[10] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*. ACM, 197–206.
[11] Torbjrn Granlund and Gmp Development Team. 2015. *GNU MP 6.0 Multiple Precision Arithmetic Library*. Samurai Media Limited, United Kingdom.
[12] Jens Groth and Markulf Kohlweiss. 2015. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*. Springer, 253–280.
[13] Shay Gueron. 2009. Intel's New AES Instructions for Enhanced Performance and Security. In *FSE (LNCS)*. Springer, 51–66.
[14] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. 2018. Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In *ACM CCS*. ACM, 525–537.
[15] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. 2019. Omniring: Scaling Up Private Payments Without Trusted Setup - Formal Foundations and Constructions of Ring Confidential Transactions with Log-size Proofs. Cryptology ePrint Archive, Report 2019/580. (2019). https://eprint.iacr.org/2019/580.
[16] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography* 75, 3 (2015), 565–599.
[17] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. 2004. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract). In *ACISP (LNCS)*. Springer, 325–335.
[18] Vadim Lyubashevsky. 2009. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*. Springer, 598–616.
[19] Vadim Lyubashevsky and Gregory Neven. 2017. One-shot verifiable encryption from lattices. In *EUROCRYPT*. Springer, 293–323.
[20] Vadim Lyubashevsky and Gregor Seiler. 2018. Short, Invertible Elements in Partially Splitting Cyclotomic Rings and Applications to Lattice-Based Zero-Knowledge Proofs. In *EUROCRYPT*. Springer, 204–224.
[21] Daniele Micciancio and Chris Peikert. 2012. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *EUROCRYPT (LNCS)*. Springer, 700–718.
[22] NIST. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf. (2015). Accessed: 2019-05-15.
[23] Shen Noether. 2015. Ring Signature Confidential Transactions for Monero. Cryptology ePrint Archive, Report 2015/1098. (2015). https://eprint.iacr.org/2015/1098.
[24] Prasanna Ravi, Sourav Sen Gupta, Anupam Chattopadhyay, and Shivam Bhasin. 2019. Improving Speed of Dilithium's Signing Procedure. Cryptology ePrint Archive, Report 2019/420. (2019). https://eprint.iacr.org/2019/420.
[25] Ronald Rivest, Adi Shamir, and Yael Tauman. 2001. How to leak a secret. *ASIACRYPT* (2001), 552–565.
[26] Michael Scott. 2017. A Note on the Implementation of the Number Theoretic Transform. In *IMACC (LNCS)*. Springer, 247–258.
[27] Gregor Seiler. 2018. Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography. Cryptology ePrint Archive, Report 2018/039. (2018).
[28] Shifeng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. 2017. RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero. In *ESORICS (LNCS)*. Springer, 456–474.
[29] Zcash Team. 2019. Frequently Asked Questions. (2019). Retrieved April 23, 2019 from https://z.cash/support/faq/#quantum-computers
[30] Wilson A. A. Torres, Veronika Kuchta, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Jacob Cheng. 2019. Lattice RingCT v2.0 with Multiple Input and Multiple Output Wallets. In *ACISP (LNCS)*. Springer, 156–175.
[31] Wilson A. A. Torres, Ron Steinfeld, Amin Sakzad, Joseph K Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. 2018. Post-Quantum One-Time Linkable Ring Signature and Application to Ring Confidential Transactions in Blockchain (Lattice RingCT v1. 0). In *ACISP*. Springer, 558–576.
[32] Tsz Hon Yuen, Shi feng Sun, Joseph K. Liu, Man Ho Au, Muhammed F. Esgin, Qingzhao Zhang, and Dawu Gu. 2019. RingCT 3.0 for Blockchain Confidential Transaction: Shorter Size and Stronger Security. Cryptology ePrint Archive, Report 2019/508. (2019). https://eprint.iacr.org/2019/508.