MatRiCT$^+$: More Efficient Post-Quantum Private Blockchain Payments

Muhammed F. Esgin
Monash University and *CSIRO's Data61*
Australia
muhammed.esgin@monash.edu

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

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

***Abstract*—We introduce MatRiCT$^+$, a practical private blockchain payment protocol based on "post-quantum" lattice assumptions. MatRiCT$^+$ builds on MatRiCT due to Esgin et al. (ACM CCS'19) and, in general, follows the Ring Confidential Transactions (RingCT) approach used in Monero, the largest privacy-preserving cryptocurrency. In terms of the practical aspects, MatRiCT$^+$ has 2–17$\times$ shorter proofs (depending on the number of input accounts, M) and runs 3–8$\times$ faster (for a typical transaction) in comparison to MatRiCT. A significant advantage of MatRiCT$^+$ is that the proof length's dependence on M is very minimal (only $O(\log M)$), while MatRiCT has a proof length linear in M.**

To support its efficiency, we devise several novel techniques in our design of MatRiCT$^+$ to achieve compact lattice-based zero-knowledge proof systems, exploiting the algebraic properties of power-of-2 cyclotomic rings commonly used in practical lattice-based cryptography. Along the way, we design an "optimal" challenge space with minimal ℓ_1-norm and invertible challenge differences (with overwhelming probability), while supporting highly-splitting power-of-2 cyclotomic rings. We believe all these results to be widely applicable and of independent interest.

***Index Terms*—Post-Quantum, RingCT, Lattice, Zero-Knowledge, Blockchain, Ring Signature**

I. Introduction

Recent advances in quantum computing, which can be exemplified by IBM's roadmap [1], have led to significant effort being put into designing *post-quantum - PQ* (i.e., quantum-safe) cryptographic schemes. Those tailored for blockchain applications are no exception, and there are already considerations in the blockchain community to support PQ cryptography. For example, Ethereum 2.0 Serenity upgrade [2] is planned to have an option for a PQ signature. The recent improvements in the *practical* efficiency of PQ cryptographic proposals are promising and have sometimes been more than an order of magnitude. To name a few such recent results suitable for blockchain applications, the first practical PQ verifiable random function was introduced in [3], the first PQ adaptor signature and payment channel network in [4], the first practical PQ RingCT protocol in [5], substantially more efficient PQ ring signatures in [5]–[7], and more efficient lattice-based zero-knowledge proof systems, e.g., in [5], [7]–[10].

Our focus in this paper is on privacy-preserving blockchain protocols, specifically RingCT protocol [11] used in Monero. A RingCT protocol allows a user to transfer assets on blockchain while keeping sensitive information such as payee/payer identities and transaction amount confidential. The original RingCT protocol [11] has been significantly improved [12], [13], and Esgin et al. [5] recently introduced the first (and only) practical PQ RingCT protocol, named MatRiCT, based on computationally hard lattice problems.

A. Our Contributions

Our main contribution in this work is the design and analysis of a practical RingCT protocol, MatRiCT$^+$, based on "post-quantum" lattice assumptions[1]. Our construction builds on MatRiCT [5], but is significantly more efficient and incorporates new techniques, that we believe are widely applicable in other proof systems (including non-lattice-based ones). We achieve a proof length reduction of up to 17$\times$ for the number of input accounts in the range $[2, 100]$ (see Table I). Furthermore, a typical transaction in MatRiCT$^+$ can be generated in 136 ms and verified in just 3 ms, achieving 3–8$\times$ speedup over MatRiCT (see Table II). An overview of MatRiCT$^+$ is provided in Section I-C. We also note that the compression techniques used in Dilithium [14] can also be applied to MatRiCT$^+$, which would lead to a further saving of around 15% in proof length (over those in Table I).

To achieve high efficiency for MatRiCT$^+$, we introduce several novel technical tools for design of lattice-based zero-knowledge proof systems (see Sec. I-B below for more details). The first is about invertibility of challenge differences, a key component in the soundness security of compact lattice-based proofs. A second technical result is concerned with the properties of *short* elements in power-of-2 cyclotomic rings $\mathcal{R}_q$ having binary or ternary CRT (Chinese Remainder Theorem) 'slots'. These results are general and believed to be of independent interest for many lattice-based proof systems. Furthermore, we apply these results and introduce a novel zero-knowledge integer balance proof based on a CRT packing technique. Beyond being useful in many proof systems, we believe that our techniques can prove useful in other real-life applications (besides private blockchain payments) such as e-voting, e-cash systems and anonymous credentials.

[1] As in many prior works utilizing the Fiat-Shamir transformation such as [3]–[10], we analyze the security of our scheme in the ROM while relying on the hardness of "post-quantum" lattice assumptions.

TABLE I
PROOF LENGTH COMPARISON (IN KB) OF MATRICT AND MATRICT$^+$.

Anonymity level[a]	$1/N = 1/11$			
#inputs → #outputs	$2 \to 2$	$20 \to 2$	$50 \to 2$	$100 \to 2$
MatRiCT [5]	110	~ 310	~ 610	~ 1100
MatRiCT$^+$	**48**	**59**	**61**	**64**
MatRiCT [5]	PK/SN sizes: 4.36 KB, 248 B			Moduli: $\leq 2^{53}$
MatRiCT$^+$	PK/SN sizes: 3.42 KB, 32 B			Moduli: $\leq 2^{34}$

[a]Anonymity level indicates that each real account is hidden using $N-1$ decoy accounts. Monero currently uses an anonymity level of $1/11$.
PK/SN sizes are for a public key and an account serial number (in bytes).
PK size/Moduli are for the most typical case of 2 input/output accounts.
See Appendix A for further comparison.

TABLE II
RUNTIMES (IN MS) OF MATRICT AND MATRICT$^+$ AT 3 GHZ.

	Anonymity level	$1/10$	$1/20$	$1/50$
	#inputs → #outputs	$2 \to 2$	$2 \to 2$	$2 \to 2$
MatRiCT [5]	Key Gen.	2	2	2
	Transaction Gen.	375	420	471
	Verification	23	25	31
MatRiCT$^+$	Key Gen.	0.07	0.07	0.07
	Transaction Gen.	136	192	380
	Verification	3	5	10

Unlike the prior CRT-packing technique in [7], our new CRT-packing technique allows a significant reduction in communication. As noted as an open problem in [7], it is indeed a more important challenge to reduce the communication as that is the main bottleneck for lattice schemes. The main drawback in [7] is that, to use the CRT-packing technique, one needs to work with very restrictive challenge sets. In particular, while the dimension of $\mathcal{R}_q$ is d, the challenges must have degree less than d/s and large coefficients (even up to 2^{31} as given in [7, Table 5]), where s denotes the number of CRT slots the messages are stored in. On the other hand, our new techniques lead to using "optimal" challenge spaces where the challenges can have full degree, the smallest possible coefficients (in $\{-1, 0, 1\}$), small Hamming weight (e.g., 56 non-zero coefficients for $d = 256$), *and* their differences are invertible (with overwhelming probability). We also allow $\mathcal{R}_q$ to split into $d/4$ factors (for typical choices of $q \geq 2^{25}$). These lead to significantly smaller parameters and proof sizes compared to [5], [7].

B. Our Techniques and Results

New technical tools for lattice-based proofs. In lattice-based proof systems, power-of-2 cyclotomic rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$ are widely employed and exploited. When $X^d + 1$ factors into, say, r polynomials, then we have, by the CRT, $\mathcal{R}_q \cong \mathcal{R}_q^{(0)} \times \cdots \times \mathcal{R}_q^{(r-1)}$. Hence, we can represent any $f \in \mathcal{R}_q$ by its representatives, called *CRT slots*, in $\mathcal{R}_q^{(i)}$'s, which can be used to pack information compactly in lattice-based zero-knowledge proof systems. Our first set of results lead to a bound on q (stated as Lemma 4 in Sec. III-B) that guarantees that if $b \in \mathcal{R}_q$ is a short polynomial having many CRT slots equal to some small integer α, then b must be equal to α. We use this result to argue (see Cor. 3 in Sec. III-B) that certain polynomials with binary CRT slots in $\mathcal{R}_{\hat{q}}$ (for some $\hat{q} \in \mathbb{Z}^+$) are just binary integers (in $\{0, 1\}$). In our novel CRT-packing based MatRiCT$^+$ transactions, this enables us to properly encode a user index in unary representation and prevent the prover from cheating. Using our new result, we reduce the requirement on the size of $\hat{q}$ to just $\hat{q} > 2^{28.7}$, while MatRiCT requires $\hat{q} > 2^{50}$ to reach the same conclusion (see [5, Lemma 5.5]). Note that the larger the system modulus, the worse the overall efficiency. We also use the same result to argue that all the CRT slots of a polynomial (masked by some random polynomial) is the same, which is crucial in our novel balance proof (discussed below).

Moving to our second set of tools, an important component of the soundness security of many compact lattice-based proofs is that any challenge difference is invertible in the underlying ring $\mathcal{R}_q$, except for a negligible probability. Here, it is important to have as *short* challenges (in terms of ℓ_1-norm, denoted by w) as possible since, in effect, the underlying M-SIS problem becomes easier with increasing norm of the challenges and therefore, we need to accommodate for this by choosing larger parameters, which in turn degrades efficiency. In summary, the goal is to construct a challenge set $\mathcal{C}$ with the following properties: (i) $|\mathcal{C}|$ is exponentially large (say, greater than 2^{200}), (ii) challenges have as small ℓ_1-norm as possible, (iii) (non-zero) challenge differences are invertible, and (iv) $\mathcal{R}_q$ splits into many factors (the more factors the better). To this end, we prove (in Cor. 1 in Sec. III-A) an efficiently computable bound on the non-invertibility probability for challenges sampled at random from a challenge set of ternary (i.e. $\{-1, 0, +1\}$) coefficient polynomials with *low Hamming weight* while allowing $\mathcal{R}_q$ split into $d/4$ factors for typical choices of modulus q of more than, say, 25 bits.

Our result improves upon the recent results in [8] that deal with large Hamming-weight challenges, and immediately improves the communication size of the proof systems in [8]–[10]. In particular, [8] obtains the above Property (iii) and $|\mathcal{C}| \approx 2^{203}$ with weight $w = 128$, while we get Property (iii) for a much larger challenge set $|\mathcal{C}| > 2^{237}$ with $w = 56$. Both our results and those in [8] allow a similar splitting of $\mathcal{R}_q$. In practice, the modulus size due to M-SIS hardness is often quadratic in w (see Assumption 5, for example), and thus reducing $w = 128$ to $w = 56$ saves a factor of more than 5 in the modulus q. This may often be critical to keep q below 32 bits, which then significantly helps with computational complexity.

Moreover, our challenge difference invertibility results can be seen "optimal" in the sense that (i) more than $d/4$ factors of $\mathcal{R}_q$ cannot be achieved via the probabilistic approach we take for typical choices of q around 32 bits since $1/q^\delta$ is non-negligible for a power-of-2 $\delta < 4$, and (ii) even the state-of-the-art lattice-based signature scheme Dilithium [14] uses $w = 60$ for $|\mathcal{C}| \approx 2^{256}$ *without* achieving Property (iii).

Novel balance proof based on CRT-packing. Suppose we want to perform a (zero-knowledge) *balance proof* to show

the following

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

for some non-negative integers a_i's and b_i's, *without* revealing the values themselves. The standard idea to do this is to use a homomorphic commitment to encode each value as $A_i = \mathsf{Com}(a_i)$ and $B_i = \mathsf{Com}(b_i)$, and then prove in zero knowledge that $\sum_{i=0}^{S-1} B_i - \sum_{i=0}^{M-1} A_i$ is a commitment to zero. As discussed in [5, Section 1.3], this is very costly to do in the lattice setting as it leads to using very large system modulus, which in turn significantly reduces both the computational and communication efficiency. Instead, our balance proof proceeds in a unique fashion as below.

Let $a[j]$ denote the j-th bit of a non-negative integer a and suppose that the sum of the integers fits into r bits with r being a power of 2.[2] Now, if there exists some "corrector" values[3] $c_1, \ldots, c_{r-1}$ with $c_0 = c_r = 0$ such that the following holds for all $j = 0, \ldots, r-1$

$$T_j := \sum_{i=0}^{S-1} b_i[j] - \sum_{i=0}^{M-1} a_i[j] + c_j - 2c_{j+1} = 0, \tag{2}$$

then (1) follows by considering $\sum_{j=0}^{r-1} 2^j T_j$ (see also [5, Section 1.3])[4]. Therefore, we aim to prove (2) for all $j = 0, \ldots, r-1$ *in parallel* in the protocol. Specifically, we want to perform the balance proof in the *CRT domain*, where we can realize our zero-knowledge proofs of interest very efficiently. Let us explain what we mean by this.

We choose a prime modulus q such that $X^d + 1$ (for some fixed power-of-2 d) factors into exactly r irreducible polynomials $g_0, \ldots, g_{r-1}$. As before by CRT, $\mathcal{R}_q \cong \mathcal{R}_q^{(0)} \times \cdots \times \mathcal{R}_q^{(r-1)}$ and $\mathcal{R}_q^{(i)} = \mathbb{Z}_q[X]/(g_i(X))$. Then, we employ the CRT packing to encode each integer in a polynomial in $\mathcal{R}_q$ as follows. Let $f = [\![f_0, \ldots, f_{r-1}]\!]$ denote the unique polynomial $f \in \mathcal{R}_q$ such that $f \equiv f_i \pmod{q, g_i(X)}$. That is, f_i's are the *CRT slots* of f. With this setup, we compute in the protocol

$$\hat{a}_i = [\![a_i[0], \ldots, a_i[r-1]]\!], \quad \hat{b}_i = [\![b_i[0], \ldots, b_i[r-1]]\!],$$
$$\hat{c} = [\![c_0, \ldots, c_{r-1}]\!] \in \mathcal{R}_q,$$

for $c_0 = 0$. We commit to these as $A_i = \mathsf{Com}(\hat{a}_i)$, $B_i = \mathsf{Com}(\hat{b}_i)$ and $C = \mathsf{Com}(\hat{c})$. Now, we perform an aggregated CRT range proof on C and binary proof on B_i's. That is, in a single aggregated proof, we prove that B_i's encode polynomials with binary CRT slots and C encodes a polynomial with CRT slots in $[-(M-1), S-1]$. As A_i's serve as the output coins of previous transactions that already include a binary CRT proof on A_i's, the same proof on A_i's is not repeated. Here, an important advantage of our approach is that the above proofs on CRT slots can be efficiently performed. In particular, proving that $f(f-1)\cdots(f-\alpha) = 0$ in $\mathcal{R}_q$ for $\alpha \in \mathbb{Z}^+$ is sufficient to prove that all the CRT slots of f are in $[0, \alpha]$. In contrast, if we wanted to prove that *polynomial* coefficients of f are in $[0, \alpha]$ (i.e., $f(X) = a_0 + a_1 X + \cdots + a_{d-1} X^{d-1}$ and $a_i \in [0, \alpha]$), then this would be a more costly proof.

Now, we need to construct the structure of $c_j - 2c_{j+1}$ as in (2) in a verifiable fashion. For this, we employ the Galois automorphisms. What we *ideally* want to do is to find an automorphism π that performs a cyclic left-shift of the CRT slots such that $\hat{c} - 2\pi(\hat{c})$ yields the desired form. That is, π needs to satisfy the following property: if $f = [\![f_0, f_1, \ldots, f_{r-2}, f_{r-1}]\!]$ for $f_i \in \mathbb{Z}_q$, then $\pi(f) = [\![f_1, f_2, \ldots, f_{r-1}, f_0]\!]$. However, such an automorphism with a "full cycle" does not exist in the power-of-2 cyclotomic rings we are interested in. We can of course choose to work on a rather non-typical polynomial ring, where such an automorphism exists, but we aim to solve the problem in power-of-2 cyclotomic rings that are very widely used in lattice-based proofs. To this end, we introduce a low-cost "patching" technique as described below.

What we can find is a Galois automorphism σ over $\mathcal{R}_q$ that performs a cyclic left-shift of *each half* of the CRT slot vector. That is, if $f = [\![f_0, \ldots, f_{r-1}]\!]$ for $f_i \in \mathbb{Z}_q$, then $\sigma(f) = [\![f_1, \ldots, f_{r'-1}, f_0, f_{r'+1}, \ldots, f_{r-1}, f_{r'}]\!]$ for $r' := r/2$. It is important to note here that the above mapping can indeed be obtained when $f_i \in \mathbb{Z}_q$. If f_i's are non-constant polynomials, then the automorphism may change the elements inside the CRT slots. But, what we want is the case where CRT slots only shift and their values remain the same. As we will use the automorphism on C, whose message opening is proven to have integer CRT slots, the automorphism σ works in a suitable fashion.

Now, when we compute $c' := \hat{c} - 2\sigma(\hat{c})$, the CRT slots of c' that are not well-formed are indexed by $r'-1, r'$ and $r-1$. We let the verifier remove the two CRT slots of $\hat{c}$ indexed by 0 and r', by removing those two CRT slots of $C = \mathsf{Com}(\hat{c})$. This fixes the problem at $(r-1)$-th CRT slot of c'. Then, we separately send a polynomial u that is proven to encode the same integer (i.e., $c_{r'}$) in $[-(M-1), S-1]$ in all of its CRT slots. This proof is accomplished efficiently using our new result from Cor. 3. Finally, setting $\tilde{C} := C' - 2\sigma(C') + u \cdot [\![0^{r'-1}, -2, 1, 0^{r'-1}]\!]$ for $C' = [\![0, 1^{r'-1}, 0, 1^{r'-1}]\!] \cdot C$ yields a commitment of the desired form, where u patches the middle CRT slots. The communication cost of u is small in comparison to the other proof parts as it has small coefficients.

Overall, using the nice homomorphic properties of the CRT mapping, we consider the following

$$\begin{aligned} &\sum_{i=0}^{S-1} \hat{b}_i - \sum_{i=0}^{M-1} \hat{a}_i + [\![c_0 - 2c_1, \ldots, c_{r-1} - 2c_r]\!] \\ &= \Bigg[\!\!\Bigg[\sum_{i=0}^{S-1} b_i[0] - \sum_{i=0}^{M-1} a_i[0] + c_0 - 2c_1, \ldots, \\ &\qquad \sum_{i=0}^{S-1} b_i[r-1] - \sum_{i=0}^{M-1} a_i[r-1] + c_{r-1} - 2c_0 \Bigg]\!\!\Bigg]. \end{aligned} \tag{3}$$

From the above, proving that (3) is equal to zero yields that (2) holds for all $j = 0, \ldots, r-1$, which then shows that the balance is preserved. Hence, the rough idea in the protocol is

[2]The requirement of having a power-of-2 r is due to the CRT-packing. For the general balance proof idea, r can be an arbitrary positive integer.

[3]One may also call them "carry values", but they may be negative.

[4]Here, we consider the base-2 representation, but the result easily extends to other bases.

to prove that $\sum_{i=0}^{S-1} B_i - \sum_{i=0}^{M-1} A_i + \tilde{C}$ is a commitment to zero. Note that since all the CRT slot values in (3) are proven to be small, we can easily ensure that there is no wrap-around mod q.

In our actual protocol, the balance proof is more complicated than what is described above as we actually need to hide which input coins (represented by A_i's here) are being used. For this, we combine the above idea with a suitable 1-out-of-N proof (a.k.a. a ring signature).

The main advantage of our novel balance proof is that it is significantly more efficient, for example, than the state-of-the-art result in MatRiCT [5]. Setting $r = 64$, in MatRiCT, each integer amount is represented by 64 polynomials in $\mathcal{R}_q$, whereas only a *single* polynomial in $\mathcal{R}_q$ is required to represent a 64-bit amount in MatRiCT$^+$. This helps to significantly reduce the total proof length as shown in Table I as well as improving the computational efficiency as shown in Table II.

C. Overview of MatRiCT$^+$

The operations in MatRiCT$^+$ are performed over the power-of-2 cyclotomic rings $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$ and $\mathcal{R}_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^d + 1)$ for $d = 256$ and the moduli $q = 167770241 \approx 2^{27}$ and $\hat{q} = 2^{34} - 2^{26} - 2^7 + 1$. The parameters are set so that $\mathcal{R}_q$ splits into exactly $r = 64$ subrings. A user public key pk is the sum of the following two components: (i) a commitment $\mathsf{Com}(0; \mathsf{sk})$ to zero under a short randomness vector sk over $\mathcal{R}_q$, which serves as the user secret key, and (ii) hash $\mathcal{G}(\gamma)$ of a 256-bit seed γ, that serves as the serial number. To use the new balance proof idea described above, each amount is represented by a binary vector of length $r = 64$, where each bit is encoded in a CRT slot. Hence, each coin cn is a commitment to the unique polynomial (in $\mathcal{R}_q$), whose CRT slots are equal to the bits of the amount.

When spending an account $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$, Alice mints the output coins and proves that each coin encodes a polynomial with binary CRT slots. This ensure that the coins represent a unique integer in $[0, 2^r - 1]$. All these binary CRT proofs and the range CRT proof of the "corrector" commitment C are aggregated to reduce the communication costs. To hide her identity, Alice creates an $M \times N$ matrix A_{in} of input accounts, where the ℓ-th column is comprised of her own accounts to spend (i.e., Alice spends M accounts, each hidden in an anonymity set of size N). She uses the new balance proof idea from Section I-B to prove that balance is preserved.

Now, to prevent double-spending, Alice outputs M 256-bit seed value γ_i's used in key generation. Then, in the algorithms, $\mathcal{G}(\gamma_i)$ is subtracted from all the public keys in the i-th row of A_{in}. Modeling $\mathcal{G}$ as a random oracle, this enforces Alice to set her i-th account's public key to be $\mathsf{pk}_i = \mathsf{Com}(0; \mathsf{sk}_i) + \mathcal{G}(\gamma_i)$ in the first place and requires her to use γ_i as the serial number when she wants to spend her account $\mathsf{act}_i = (\mathsf{pk}_i, \cdot)$.

Finally, to prove ownership of M accounts, we again employ an aggregation strategy. In particular, since all of Alice's accounts are in the ℓ-th column of A_{in}, Alice linearly combines the rows of A_{in} to shrink it into a *single* row. That is, if $R_0, \ldots, R_{M-1}$ are the rows of A_{in} for public keys, then Alice computes $R = \alpha_0 R_0 + \cdots + \alpha_{M-1} R_{M-1}$ for random challenge α_i's generated using the Fiat-Shamir transformation. Alice now knows the secret key of the ℓ-th element in R and can prove this using a *single* 1-out-of-N proof (a.k.a. a ring signature). Since α_i's are random and created *after* committing to A_{in}, intuitively Alice cannot know the secret key of the ℓ-th commitment in R without knowing the secret keys of the individual ℓ-th commitments in all R_i's. This idea helps to largely remove the proof length's dependence on the number of input accounts M, and can also be employed in non-lattice settings such as in discrete logarithm based proof systems. As a result, MatRiCT$^+$ remains very efficient even for increasing M values as shown in Table I.

Organization of the manuscript: The rest of the paper is organized as follows. Section II covers the preliminaries including background on our security assumptions, rejection sampling, and cyclotomic rings and Galois automorphisms. In Section III, we introduce our new technical tools about cyclotomic rings. The formal RingCT model used is discussed in Section IV. We describe the details of MatRiCT$^+$ in Section V, where the concrete parameter setting is also covered. The proof of balance, the main challenging property to accomplish, is provided in Section VI. Some discussions and proofs are deferred to appendices.

II. Preliminaries

For $\mathcal{R} = \mathbb{Z}[X]/(X^d+1)$, $\mathbb{S}_c$ denotes the set of polynomials in $\mathcal{R}$ with infinity norm at most $c \in \mathbb{Z}^+$. We write $f \leftarrow \mathbb{D}_s$ to denote sampling a polynomial in $\mathcal{R}$ with each coefficient following a discrete Gaussian distribution centred at zero with standard deviation s. We write $[n] = \{0, \ldots, n-1\}$ and $[a, b] = \{a, \ldots, b\}$. We also write 0^n and 1^n to denote n-dimensional all-zero and all-one vectors, respectively. To sample an element x from a set X uniformly at random, we write $x \xleftarrow{\$} X$. We use the notation '$f(X) \bmod (q, g(X))$' to mean that the coefficients of the polynomial $f(X)$ are reduced mod q and the polynomial itself is reduced mod $g(X)$.

A. Security Assumptions

MatRiCT$^+$ relies on the two well-known lattice problems, namely Module-SIS (M-SIS) and Module-LWE (M-LWE).

Definition 1 (M-SIS$_{n,m,q,\beta}$)**.** *For positive integer parameters (n, m, q, β) with $m > n$, given $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m}$, the M-SIS problem asks to find a* short non-zero *vector $\boldsymbol{x} \in \mathcal{R}^m$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \in \mathcal{R}_q^n$ and $\|\boldsymbol{x}\| \leq \beta$.*

We write M-SIS$^{\infty}_{n,m,q,\beta}$ to denote the M-SIS problem where the norm bound is with respect to the ℓ_∞-norm, i.e., $\|\boldsymbol{x}\|_\infty \leq \beta$. For readability purposes, we define M-LWE in a standard case commonly used in prior works, e.g., [5]–[7].

Definition 2 (M-LWE$_{\kappa,m,q,\mathcal{B}}$)**.** *For positive integer parameters $(\kappa, m, q, \mathcal{B})$, the M-LWE problem asks to distinguish between the two cases: (i) $(\boldsymbol{A}, \boldsymbol{t}) \xleftarrow{\$} \mathcal{R}_q^{m \times \kappa} \times \mathcal{R}_q^m$, and (ii) $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ for $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m \times \kappa}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{S}_{\mathcal{B}}^{\kappa}$ and $\boldsymbol{e} \xleftarrow{\$} \mathbb{S}_{\mathcal{B}}^{m}$.*

For the practical security estimations against known attacks in this work, the parameter m in both M-SIS and M-LWE does not play an important role. Therefore, we sometimes simply omit it and write M-SIS$_{n,q,\beta}$ and M-LWE$_{\kappa,q,\mathcal{B}}$.

B. Power-of-2 Cyclotomic Rings and Galois Automorphisms

Let $r \leq d$ be powers of 2 and $q \equiv 2r+1 \pmod{4r}$ and $\delta := d/r$. Fix a primitive $2r$'th root of unity ζ in $\mathbb{Z}_q$. Then, the polynomial X^d+1 factors into r irreducible polynomials $g_i(X) := X^\delta + \zeta_i$ modulo q, where for $i \in [r]$, $\zeta_i := \zeta^{2i+1}$ are the primitive $(2r)$-th roots of unity in $\mathbb{Z}_q$. Hence, by CRT, $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1) \cong \mathcal{R}_q^{(0)} \times \cdots \times \mathcal{R}_q^{(r-1)}$ for $\mathcal{R}_q^{(i)} = \mathbb{Z}_q[X]/(g_i(X))$. Let us write $f = [\![f\{0\}, \ldots, f\{r-1\}]\!]$ if $f \equiv f\{i\} \pmod{q, g_i(X)}$. Recall that $f\{i\}$ is said to be the i'th CRT slot of f. Then, by CRT, we have for any $f, h \in \mathcal{R}_q$

$$f + h = [\![f\{0\} + h\{0\}, \ldots, f\{r-1\} + h\{r-1\}]\!],$$
$$f \cdot h = [\![f\{0\} \cdot h\{0\}, \ldots, f\{r-1\} \cdot h\{r-1\}]\!].$$

It is known (see, e.g., [8, Se. 2.2]) that the automorphism group of $\mathcal{R}_q$ is isomorphic to $\mathbb{Z}_{2d}^*$ and consists of the mappings $\{\sigma_j\}_{j \in \mathbb{Z}_{2d}^*}$, where σ_j maps $X \in \mathcal{R}_q$ to X^j and extends to all $\mathcal{R}_q$ by homomorphism. We use the fact (also exploited in the context of FHE schemes [15]) that for an appropriate ordering of the CRT slots, and a polynomial a with CRT slots that are *constant* (degree 0) polynomials, there exists an automorphism σ_j of $\mathcal{R}_q$ such that the CRT slots of $b := \sigma_j(a)$ are a cyclic rotation of the left and right CRT slot halves of f. Namely, let $b\{i\} := b(X) \pmod{X^\delta - \zeta^{2i+1}}$ and $a\{i\} := a(X) \pmod{X^\delta - \zeta^{2i+1}}$ denote the CRT slots of b and a respectively, indexed by $2i+1 \in \mathbb{Z}_{2r}^*$. Then we have (see [8, Se. 2.2]) that $\sigma_j(a(X)) \pmod{X^\delta - \zeta^{2i+1}} = \sigma_j(a(X)) \pmod{X^\delta - \zeta^{(2i+1)j^{-1} \bmod 2r}}$, i.e. $\sigma_j(a\{i\}) = b\{(2i+1) \cdot j^{-1} \bmod 2r\}$ for $(2i+1) \in \mathbb{Z}_{2r}^*$. As $\mathbb{Z}_{2r}^*$ is a product of a two cyclic groups of order r (resp. 2) generated by 3 (resp. -1), we can re-index the CRT slots writing $2i+1 = 3^t(-1)^s \bmod 2r$ for $(t,s) \in \mathbb{Z}_r \times \mathbb{Z}_2$ and defining $a\{t,s\} := a\{3^t(-1)^s \bmod 2r\}$ and $b\{t,s\} := b\{3^t(-1)^s \bmod 2r\}$. Setting $j=3$ we have $\sigma_3(a\{t,s\}) = b\{t-1 \bmod r, s\}$ for $(t,s) \in \mathbb{Z}_r \times \mathbb{Z}_2$, so applying σ_3 to a induces a cyclic left rotation by one position to the r-slot half-vector $(a\{0,s\}, a\{1,s\}, \ldots, a\{r-1,s\})$ for each $s \in \{0,1\}$. We see that for a general $a \in \mathcal{R}_q$ the rotation is 'defective'; in addition to rotating the polynomial $a\{t,s\}(X)$ in slot $\{t,s\}$ of a to slot $\{t-1 \bmod r, s\}$ of b, σ_3 also has the effect of modifying the polynomial $a\{t,s\}(X)$ into $\sigma_3(a\{t,s\}(X)) = a\{t,s\}(X^3)$. However, this 'defect' does not appear and the rotation is perfect (i.e. $\sigma_3(a\{t,s\}(X)) = a\{t,s\}(X)$) if a has degree 0 polynomials in its CRT slots $a\{t,s\}$; our protocol ensures the latter holds for the underlying message values encoded in the CRT slots, as explained in Sec. I. We also use the inverse automorphism $\sigma_3^{-1} = \sigma_{3^{-1} \bmod 2d}$ satisfying $\sigma_3^{-1}(\sigma_3(a)) = a$ for all $a \in \mathcal{R}_q$ (regardless of whether a has degree 0 CRT slots or not).

C. Rejection Sampling

In our protocol, we employ the rejection sampling technique from [16] and its recent optimized variant from [17]. While the former method (Alg. 1) does not leak any information about the secret vector $\boldsymbol{c}$, the optimized rejection sampling (Alg. 2) reveals the fact that $\langle \boldsymbol{z}, \boldsymbol{c} \rangle \geq 0$ and uses this fact to optimize parameter selection. The security in this optimized case relies on "Extended M-LWE" assumption. We refer the reader to [17] for a further discussion. We also add the infinity norm check into the rejection sampling algorithms as a shortcut to be used in the protocol.

Algorithm 1 $\mathsf{Rej}(\boldsymbol{z}, \boldsymbol{c}, \phi, T)$

1: $\sigma = \phi T$; $\mu(\phi) = e^{12/\phi + 1/(2\phi^2)}$; $u \leftarrow [0,1)$
2: **if** $u > \frac{1}{\mu(\phi)} \cdot \exp\left(\frac{-2\langle \boldsymbol{z}, \boldsymbol{c} \rangle + \|\boldsymbol{c}\|^2}{2\sigma^2}\right)$, **then return** 1
3: **if** $\|\boldsymbol{z}\|_\infty > 6\sigma$, **then return** 1
4: **else return** 0

Algorithm 2 $\mathsf{RejOp}(\boldsymbol{z}, \boldsymbol{c}, \phi, T)$

1: **if** $\langle \boldsymbol{z}, \boldsymbol{c} \rangle < 0$, **then return** 1
2: $\sigma = \phi T$; $\mu(\phi) = e^{1/(2\phi^2)}$; $u \leftarrow [0,1)$
3: **if** $u > \frac{1}{\mu(\phi)} \cdot \exp\left(\frac{-2\langle \boldsymbol{z}, \boldsymbol{c} \rangle + \|\boldsymbol{c}\|^2}{2\sigma^2}\right)$, **then return** 1
4: **if** $\|\boldsymbol{z}\|_\infty > 6\sigma$, **then return** 1
5: **else return** 0

III. New Tools for Lattice-Based Proof Systems

A. New Results on Invertibility of Challenge Differences

For the efficient zero-knowledge proofs underlying our protocols, we need to construct a challenge set $\mathcal{C}$ with the following properties: (i) $|\mathcal{C}|$ is exponentially large (say, greater than 2^{200}), (ii) challenges have as small ℓ_1-norm as possible, (iii) (non-zero) challenge differences are invertible, and (iv) $\mathcal{R}_q$ splits into many factors (the more factors the better). Previous results on invertibility of challenge differences are suboptimal in either requiring almost non-splitting choice of ring modulus q [18] or large Hamming weight challenges [8]. We give new results that allow us to efficiently compute invertibility probability bounds extending [8] to low Hamming weight challenges.

In this Section, we use the same notations and assumptions for q, r, δ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ as in Sec. II-B. Let $\mathbb{S}_1^{(\delta)}$ be the set of polynomials in $\mathbb{S}_1$ of the form $f(X) = f_0 + f_\delta X^\delta + \cdots + f_{(r-1)\delta} X^{(r-1)\delta}$. Our bounds apply to the uniformly random challenge distribution $\mathfrak{C}$ on the challenge set $\mathcal{C}$, defined as

$$\mathcal{C} = \left\{ \tilde{c}_0 + \tilde{c}_1 X + \cdots + \tilde{c}_{\delta-1} X^{\delta-1} : \tilde{c}_i \in \mathbb{S}_1^{(\delta)} \wedge \|\tilde{c}_i\|_1 = \tilde{w} \right\}. \quad (4)$$

Note that challenges $c(X) = \sum_{i=0}^{\delta-1} \tilde{c}_i(X) X^i$ in $\mathcal{C}$ have total Hamming weight $w = \delta\tilde{w}$ with non-zero coefficients in $\{-1, +1\}$, and the coefficient index set $S_i := \{j \in [d] : j = i \bmod \delta\}$ appearing in $\tilde{c}_i(X)$ has weight $\tilde{w}$ for each $i \in [\delta]$.

We first observe (implicit in [8]) that, using the fact that for $i \in [r]$, $X^\delta = \zeta_i \bmod g_i(X)$, the i'th CRT slot $c\{i\} = c(X) \bmod g_i(X)$ of a polynomial $c(X) \in \mathcal{C}$ is given by:

$$c\{i\} = \tilde{c}_{i,0} + \tilde{c}_{i,1} X + \cdots + \tilde{c}_{i,\delta-1} X^{\delta-1}, \quad (5)$$

where for $k \in [\delta]$, the coefficient of X^k in $c\{i\}$ is $\tilde{c}_{i,k} := \bar{c}_k(\zeta_i)$, where

$$\bar{c}_k(X) := c_k + c_{k+\delta}X + \cdots c_{k+(r-1)\delta}X^{r-1}, \quad (6)$$

and therefore $\bar{c}_k(X)$ and $\tilde{c}_{i,k}$ depends only on the coefficient set S_j of $\tilde{c}_j$. Since the coefficients in sets $S_j, S_{j'}$ for challenge polynomials $c(X)$ are chosen by challenge distribution $\mathfrak{C}$ independently for $j \neq j'$, the distribution of the coefficients $\tilde{c}_{i,j}$ are also independent for $j \neq j'$. It thus suffices to analyse the distribution P_2 of each $\tilde{c}_{i,j}$ for $c(X)$ sampled from $\mathfrak{C}$. We use Fourier analysis for this, using an extension of the analysis in [8]. One technical difficulty here (which is not present in the unrestricted weight challenges used in [8]) is that the coefficients *within* each set S_j are *not* independent, due to the Hamming weight restriction. To handle this difficulty, our analysis of the distribution P_2 proceeds in two steps. In the first step, we derive a bound M_1 on a related distribution P_1 where the coefficients within each set S_i are independently distributed, and then in the second step we use essentially a Rényi divergence argument to derive from M_1 a bound M_2 on our desired distribution P_2 above. The complete proof of the following Lemma is in Appendix H.

Lemma 1. *Let P_2 denote the probability distribution of the coefficient $\tilde{c}_{i,k}$ of X^k in the i'th CRT slot $c[i] = c(X) \bmod g_i(X)$ of a challenge $c(X)$ sampled from the distribution $\mathfrak{C}$, i.e. uniformly at random from the challenge set $\mathcal{C}$ in (4). Let $\eta := \frac{r^{\tilde{w}}(r-\tilde{w})!}{r!}$. Then, for all $i \in [r]$ and $k \in [\delta]$, we have:*

$$\max_y P_2(y) \leq M_2 := \frac{\eta}{q}\left(1 + 2r \sum_{j \in \mathbb{Z}_q^*/<\zeta_i>} |\hat{\mu}(j)|^{\tilde{w}}\right), \quad (7)$$

where for $j \in \mathbb{Z}_q^*/ < \zeta_i >$, we define

$$\hat{\mu}(j) := \frac{1}{r}\sum_{k \in [r]} \cos(2\pi j \zeta_i^k / q). \quad (8)$$

From the above result, the independence of the δ coefficients of each CRT slot, and the fact that a challenge difference $c(X) - c'(X)$ is non-invertible in $\mathcal{R}_q$ if and only if one of its CRT slots is 0, we immediately get the following corollary.

Corollary 1. *Let $c(X), c'(X)$ denote a pair of challenges independently sampled from distribution $\mathfrak{C}$. The probability that $c(X) - c'(X)$ is not invertible in $\mathcal{R}_q$ is upper bounded by $p := rM_2^\delta$, where M_2 is the bound from Lemma 1.*

In our protocol, we use the parameters $d = 256$, $r = 64$, $\delta = 4$, $\tilde{w} = 14$ with two different moduli $q = 167770241 \approx 2^{27}$ and $\hat{q} = 2^{34} - 2^{26} - 2^7 + 1$. As it may be of independent interest, we provide in Table III the bounds M_2 and p from Lemma 1 and Corollary 1 computed using a SageMath script for several values of $\tilde{w}$. We remark that computation of M_2 takes time $O(q)$ for computing a table of $\hat{\mu}$ and then can be more quickly used in time $O(q/r)$ to compute bounds for any value of $\tilde{w}$; for our q, the table computation took a few mins (resp. several hours for $\hat{q}$) on a 3.1 GHz Intel Core i5 laptop, whereas the latter computation with the table was completed in seconds for both $q, \hat{q}$. To speed-up parameter selection for large q, we have also written a SageMath script to quickly compute a heuristic estimate bounds for M_2 and p which we empirically found to very closely approximate the provable bounds computed above for our range of parameters. The fast heuristic bound script is easier to use for parameter selection; the slower provable bound script can be subsequently used only to verify the heuristic bounds once final parameters are fixed. We provide both our heuristic and provable bound scripts as tools for future work[5]. We remark that our heuristic bound is derived by heuristically modelling ζ_i^k in (8) as independent uniformly random elements in $\mathbb{Z}_q$ as k runs though $[r]$. Under this statistical heuristic, the variance of $\mu(j)$ is $1/(2r)$. Modelling each $\mu(j)$ as a zero-mean continuous Gaussian random variable with variance $1/(2r)$, we see that the expected value of the bias term $b := 2r\sum_{j \in \mathbb{Z}_q^*/<\zeta_i>} |\hat{\mu}(j)|^{\tilde{w}}$ in (7) is equal (up to a factor of $(q-1)/q$, which is negligibly close to 1 for our purposes) to the $\tilde{w}$'th moment of a Gaussian with variance $1/(2r)$, i.e. $b = (\tilde{w}-1)!!/(2r)^{\tilde{w}/2}$ (assuming $\tilde{w}$ is even), which gives us the heuristic bound $M_2 \approx \eta \cdot (1/q + (\tilde{w}-1)!!/(2r)^{\tilde{w}/2})$ that we use in our heuristic SageMath script.

TABLE III
CHALLENGE DIFFERENCE INVERTIBILITY BOUNDS FOR $d = 256$, $r = 64$, $\delta = 4$, WITH DIFFERENT VALUES OF q AND $\tilde{w}$.

$\tilde{w}$	10	12	14	16
$\log_2 M_2$ ($q \approx 2^{27}$)	-23.8	-25.3	-25.1	-24.4
$\log_2 p$ ($q \approx 2^{27}$)	-89.4	-95.2	-94.2	-91.4
$\log_2 M_2$ ($q \approx 2^{34}$)	-24.1	-27.2	-29.6	-30.5
$\log_2 p$ ($q \approx 2^{34}$)	-90.6	-102.8	-112.4	-116.2
$\log_2 \lvert\mathcal{C}\rvert = \delta(\tilde{w} + \log_2(\binom{r}{\tilde{w}}))$	188	214	237	259

B. New Results on Power-of-2 Cyclotomic Rings

The following lower bound on the minimum of ideals in $\mathcal{R}_q$ works for general q.

Lemma 2 (Adapted from [19, Sec.2], [18, Le.2.7])**.** *Let $\mathcal{R} := \mathbb{Z}[X]/(X^d+1)$ and $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^d+1)$ for d a power of 2 and q a prime such that $X^d + 1 = \prod_{j \in [r]} g_j(X) \bmod q$ with $g_j(X) = X^\delta - \zeta_j$ irreducible over $\mathbb{Z}_q$ of degree $\delta := d/r$ for $j \in [r]$. Let $Z \subset [r]$ with $|Z| = \nu$. The minimum infinity norm $\lambda_1^\infty(I_{Z,\mathcal{R}})$ over all non-zero vectors of the ideal lattice $I_{Z,\mathcal{R}} := \{a \in \mathcal{R} : a \bmod (g_j, q) = 0$ for $j \in Z\}$ is lower bounded as*

$$\lambda_1^\infty(I_{Z,\mathcal{R}}) \geq \frac{1}{\sqrt{d}} \cdot q^{\nu/r}.$$

An improvement over the above lower bound on the minimum of ideals in $\mathcal{R}_q$ that works for more special choice of q can be obtained by a generalization of the results of [18, Th.1.1] (which can be viewed as a special case of the corollary below in the case $\nu = 1$, although [18] stated their result for more general cyclotomic rings).

[5]Our SageMath scripts are available at https://gitlab.com/raykzhao/matrict_plus.

Lemma 3 (Generalization of [18, Le.3.2])**.** *Assume the same notations as in Lemma 2 with the condition $q = 1 \bmod 2r$. Let $\bar{\mathcal{R}} := \mathbb{Z}[X]/(X^r + 1)$ with $X^r + 1 = \prod_{j\in[r]}(X - \zeta_j) \bmod q$ and define an ideal lattice in $\bar{\mathcal{R}}$ as $I_{Z,\bar{\mathcal{R}}} := \{\bar{c} \in \bar{\mathcal{R}} : \bar{c} \bmod (X - \zeta_j, q) = 0$ for $j \in Z\}$. For a polynomial $c(X) \in \mathcal{R}$ and $k \in [\delta]$, let $\bar{c}_k(X) \in \bar{\mathcal{R}}$ be defined as in Eq. (6) in Sec. III-A. If $c(X)$ is non-zero in ideal lattice $I_{Z,\mathcal{R}}$ for some $Z \subseteq [r]$ with $|Z| = \nu$, then there exists $k^* \in [\delta]$ such that $\bar{c}_{k^*}(X)$ is non-zero and in ideal lattice $I_{Z,\bar{\mathcal{R}}}$.*

Proof. By definition of $I_{Z,\mathcal{R}}$, if $c(X) \in I_{Z,\mathcal{R}}$ then the CRT slots $c\{j\} := c(X) \pmod{g_j(X), q} = 0$ for $j \in Z$. On the other hand, we have from Eq. (5) in Sec. III-A that $c\{j\} = \sum_{k\in[\delta]} \bar{c}_k(\zeta_j)X^k$. It follows that $\bar{c}_k(\zeta_j) = \bar{c}_k(X) \pmod{X - \zeta_j} = 0 \bmod q$ for $k \in [\delta]$ and $j \in Z$, and hence $\bar{c}_k(X) \in I_{Z,\bar{\mathcal{R}}}$ for $k \in [\delta]$. Moreover, if $c(X) \neq 0$, then it has some non-zero coefficient c_κ for $\kappa \in [d]$. Let $k^* := \kappa \bmod \delta \in [\delta]$. From Eq. (6) in Sec. III-A, the coefficients of $\bar{c}_{k^*}$ are the c_i's with $i \equiv k^* \bmod \delta$, which includes c_κ. Hence, $\bar{c}_{k^*}(X)$ is non-zero and in ideal lattice $I_{Z,\bar{\mathcal{R}}}$, as claimed. □

Using Lemma 3, we lower bound the ideal lattice minimum $\lambda_1^\infty(I_{Z,\mathcal{R}})$ for ring $\mathcal{R}$ of degree d, by the ideal lattice minimum $\lambda_1^\infty(I_{Z,\bar{\mathcal{R}}})$ in the ring $\bar{R}$ of smaller degree r; for the latter, we use Lemma 2 to get the following.

Corollary 2. *Assume the same notations as in Lemma 3. The minimum infinity norm $\lambda_1^\infty(I_{Z,\mathcal{R}})$ over all non-zero vectors of the ideal lattice $I_{Z,\mathcal{R}} := \{a \in \mathcal{R} : a \bmod (g_j, q) = 0$ for $j \in Z\}$ is lower bounded as*

$$\lambda_1^\infty(I_{Z,\mathcal{R}}) \geq \frac{1}{\sqrt{r}} \cdot q^{\nu/r}.$$

We now apply the above bounds to obtain sufficient conditions that ensure that a polynomial with binary CRT slots that is short must be a small integer, a property required for the soundness security of our protocol.

Lemma 4. *Assume the same notations as in Lemma 2.*

Let $b \in \mathcal{R}_q$ be such that an integer *α occurs in ν CRT slots of b (i.e. $b\{j\} = b \pmod{g_j(X), q} = \alpha \in \mathbb{Z}$ occurs for ν values of j).*

Then, for any $\Delta \in \mathcal{R}_q^*$ (i.e. Δ is invertible in $\mathcal{R}_q$), if $< \|\Delta \cdot b\|_\infty \leq 2B$ and

$$q > \left(\sqrt{d} \cdot (\alpha \cdot \|\Delta\|_\infty + 2B)\right)^{r/\nu}, \tag{9}$$

then $b = \alpha \in \mathbb{Z}$. If, in addition, q also satisfies the condition $q = 1 \bmod 2r$ then the Lemma holds if

$$q > \left(\sqrt{r} \cdot (\alpha \cdot \|\Delta\|_\infty + 2B)\right)^{r/\nu}. \tag{10}$$

Proof. Suppose, towards a contradiction, that $b \neq \alpha$. Let $Z := \{j \in [r] : b \pmod{g_j, q} = \alpha\}$ denote the set of CRT slots of b equal to α. We have $|Z| = \nu$. Observe that $b - \alpha \pmod{g_j, q} = 0$ for $j \in Z$. Therefore, $b - \alpha \in I_{Z,\mathcal{R}}$, where $I_{Z,\mathcal{R}}$ is the ideal lattice defined as in Lemma 2. Moreover, using $b \neq \alpha$ and the fact that $\Delta \in \mathcal{R}_q^*$, we have that $\Delta \cdot (b - \alpha) \in \mathcal{R}_q$ is a non-zero polynomial in ideal lattice $I_{Z,\mathcal{R}}$. From Lemma 2, it follows that

$$\|\Delta \cdot (b - \alpha)\|_\infty \geq \frac{1}{\sqrt{d}} q^{\nu/r}. \tag{11}$$

Therefore, $\|\Delta \cdot b\|_\infty = \|\Delta \cdot (b - \alpha) + \alpha\Delta\|_\infty \geq \|\Delta \cdot (b - \alpha)\|_\infty - \alpha\|\Delta\|_\infty$, which, using (11) gives $\|\Delta \cdot b\|_\infty \geq \frac{1}{\sqrt{d}} q^{\nu/r} - \alpha\|\Delta\|_\infty$. The latter gives a contradiction with $\|\Delta \cdot b\|_\infty \leq 2B$ if $\frac{1}{\sqrt{d}} q^{\nu/d} - \alpha\|\Delta\|_\infty > 2B$, which is equivalent to (9), as required. In the second case of the Lemma, the additional conditions on q allow us to apply the improved bound of Corollary 2 on the minimum of the ideal lattice $I_{Z,\mathcal{R}}$, so that (11) is replaced by

$$\|\Delta \cdot (b - \alpha)\|_\infty \geq \frac{1}{\sqrt{r}} q^{\nu/r}. \tag{12}$$

The remaining argument is the same as before, leading to the condition (10). □

For $b \in \mathcal{R}_q$ with binary (resp. ternary) integer CRT slots, one integer value must occur in at least $\nu = r/2$ (resp. $\nu = r/3$) CRT slots of b. Lemma 4 then gives the following result.

Corollary 3. *Assume the same notations for $q, r, \delta, \mathcal{R}_q$ as in Lemma 2. If $b \in \mathcal{R}_q$ has integer CRT slot values in $\{0, 1\}$ (resp. $\{0, \pm 1\}$), Δ is invertible in $\mathcal{R}_q$, $\|\Delta \cdot b\|_\infty \leq 2B$ and*

$$q > \left(f \cdot (\alpha \cdot \|\Delta\|_\infty + 2B)\right)^e, \tag{13}$$

with $e = 2$ (resp. $e = 3$), and $f := \sqrt{r}$ if $q = 1 \bmod 2r$ or $f := \sqrt{d}$ otherwise, then $b \in \{0, 1\}$ (resp. $b \in \{0, \pm 1\}$).

IV. Formal Model for RingCT-like Protocols

In this section, we recall and slightly extend the formal model for RingCT-like protocols in [5], and fix the notations used specifically for the formal model in Table IV.

The blockchain state $\mathbb{B}$ is comprised of two lists: (i) a list of registered accounts $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$, indicating a public key pk is paired with a coin cn, and (ii) a list of all verified transactions. $\mathbb{B}$ is assumed to be properly updated among all users at all times, which is managed by a consensus algorithm

TABLE IV
Notations for the RingCT formal model.

$\mathbb{B}$	the blockchain state
$\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$	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
$\mathsf{K}_{\text{in}} = (\mathsf{SK}_{\text{in}}, \mathsf{CK}_{\text{in}}, \mathsf{Amt}_{\text{in}})$	set of spender's account secret keys $\mathsf{ask} = (\mathsf{sk}, \mathsf{ck}, \mathsf{amt})$ 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 $\mathsf{R}_{\text{in}}[i]$
PK_{out}	set of output public keys with $\lvert\mathsf{PK}_{\text{out}}\rvert = S$
CN_{out}	set of output coins with $\lvert\mathsf{PK}_{\text{out}}\rvert = S$
$\mathsf{Amt}_{\text{out}}$	set of output amounts with $\lvert\mathsf{Amt}_{\text{out}}\rvert = S$
CK_{out}	set of output coin keys with $\lvert\mathsf{CK}_{\text{out}}\rvert = S$
A_{out}	set of output accounts with $\lvert\mathsf{A}_{\text{out}}\rvert = S$
π	the proof output
SN	set of serial numbers
tx	a transaction $\mathsf{tx} = (\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \pi, \mathsf{SN})$
V	set of all valid amounts

TABLE V
STRUCTURE OF THE LIST $\mathcal{L}$ USED IN THE SECURITY MODEL.

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

that is outside the scope of this work. The following tuple of polynomial time algorithms define RingCT protocol.

- $pp \leftarrow$ **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.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ **KeyGen**() : output a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{sn} \leftarrow$ **SerialGen**(sk) : on input a secret key sk, output a serial number sn associated to sk.
- $(\mathsf{cn}, \mathsf{ck}) / \perp \leftarrow$ **Mint**(amt) : on input an amount amt, if $\mathsf{amt} \in \mathsf{V}$, output a coin cn and its coin key ck. Otherwise, output $\perp$. If ck is given as an input, then **Mint** computes a deterministic function such that $\mathsf{cn} =$ **Mint**$(\mathsf{amt}, \mathsf{ck})$.
- $(\mathsf{act}, \mathbb{B}) \leftarrow$ **AccountGen**$(\mathsf{pk}, \mathsf{cn}, \mathbb{B})$: on input a public key pk and a coin cn, register an account $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$ to the blockchain state $\mathbb{B}$. Output act and updated state $\mathbb{B}$.
- $0/1 \leftarrow$ **CheckAct**$(\mathsf{pk}, \mathsf{cn}, \mathbb{B})$: on input a public key pk, a coin cn and the blockchain state $\mathbb{B}$, output 1 if $(\mathsf{pk}, \mathsf{cn})$ is a registered account in $\mathbb{B}$. Otherwise, output 0. In the case that the input has a set of pairs of $(\mathsf{pk}, \mathsf{cn})$, then output 1 if all $(\mathsf{pk}, \mathsf{cn})$ pairs are registered accounts in $\mathbb{B}$. Otherwise, output 0.
- $(\mathsf{tx}, (\mathsf{CK}_{\text{out}}, \mathsf{Amt}_{\text{out}})) \leftarrow$ **Spend**$(\mathsf{A}_{\text{in}}, \mathsf{R}_{\text{in}}, \mathsf{K}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$: on input $\mathsf{A}_{\text{in}}, \mathsf{R}_{\text{in}}, \mathsf{K}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}}$ as in Table IV, mint output coins by running $(\mathsf{CN}_{\text{out}}, \mathsf{CK}_{\text{out}}) \leftarrow$ **Mint**$(\mathsf{Amt}_{\text{out}})$. Generate the serial numbers by running $\mathsf{SN} \leftarrow$ **SerialGen**$(\mathsf{SK}_{\text{in}})$ and a proof π. Output $(\mathsf{tx}, (\mathsf{CK}_{\text{out}}, \mathsf{Amt}_{\text{out}}))$ where $\mathsf{tx} = (\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \pi, \mathsf{SN})$.[6]
- $0/1 \leftarrow$ **IsSpent**$(\mathsf{SN}, \mathbb{B})$: on input a set SN of serial numbers and the blockchain state $\mathbb{B}$, if there is a collision in SN or if a serial number appears both in SN and $\mathbb{B}$, output 1. Otherwise, output 0.
- $\emptyset / (\mathsf{A}_{\text{out}}, \mathbb{B}) \leftarrow$ **Verify**$(\mathsf{tx}, \mathbb{B})$: on input a transaction tx as in Table IV, if **IsSpent**$(\mathsf{SN}, \mathbb{B}) = 1$ or **CheckAct**$(\mathsf{A}_{\text{in}}, \mathbb{B}) = 0$, output $\emptyset$. Check the proof π and output $\emptyset$ if not valid. Otherwise, run $(\mathsf{A}_{\text{out}}, \mathbb{B}) \leftarrow$ **AccountGen**$(\mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \mathbb{B})$ and add tx to $\mathbb{B}$. Output $\mathsf{A}_{\text{out}} \neq \emptyset$ and updated $\mathbb{B}$.

A. Security Definitions

The list $\mathcal{L}$ in Table V is used in the model 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 $\mathcal{L}[\mathsf{pk}]$ for some public key pk. Then, $\mathcal{L}[\mathsf{pk}].\mathsf{ck}$ 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. We define an additional SUBTOBC oracle that allows the adversary to submit transactions directly on blockchain (rather than outputting them in the game). Also, our PKGEN oracle additionally returns the serial number, which strengthens the model.

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

We denote the set of all oracles defined above together with the random oracle(s) by ORC. With respect to the positioning of the accounts in R_{in} inside A_{in}, there are two flavours of properties for RingCT: (i) *with shuffling* and (i) *without shuffling*. In the latter case, all the accounts in R_{in} are restricted to be in the same column. The definitions are given for the former case, and it is trivial to get the latter by imposing the aforementioned restriction on R_{in}.

Informally, correctness requires that every user is able to spend any of her honestly generated *unspent* accounts, which has honestly generated keys and coins with a valid amount. Further, anonymity, informally, requires that the real spender's accounts are hidden among the uncorrupted accounts as long as there are at least two sets of uncorrupted input accounts that can be successfully spent. We refer to Appendix F for the formal definitions of correctness and anonymity.

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. We simplify the balance property given in [5] by having the adversary output a single transaction in the game as he can submit the others directly to the blockchain using the new oracle SUBTOBC.

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

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

where `Exp:Balance` is defined as follows.

[6] $(\mathsf{CK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$ are delivered to the recipient(s) privately.

1) $(\mathsf{tx}, \mathsf{Amt}_{\text{out}}, \mathsf{CK}_{\text{out}}) \leftarrow \mathcal{A}^{\text{ORC}}(pp)$: The adversary $\mathcal{A}$ is given access to all the oracles ORC together with pp, and outputs a transaction where
   - $\mathsf{tx} = (\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \pi, \mathsf{SN})$,
   - $\mathsf{Amt}_{\text{out}}$'s and CK_{out}'s are sets of output amounts and coin keys, respectively, for *uncorrupted* output public keys with $|\mathsf{CK}_{\text{out}}| = |\mathsf{Amt}_{\text{out}}| \leq |\mathsf{PK}_{\text{out}}| = |\mathsf{CN}_{\text{out}}|$.
2) $B \leftarrow \textbf{Verify}(\mathsf{tx}, \mathbb{B})$.

$\mathcal{A}$ wins the game `Exp:Balance` if the following holds

- all public keys and coins in A_{in} are generated by PKGEN and MINT, respectively, and all accounts in A_{in} are generated by ACTGEN,
- $B \neq \emptyset$,
- $\sum_{i=0}^{S'-1} \mathsf{Amt}_{\text{out}}[i] > \sum_{i=0}^{M-1} \mathsf{amt}_{\text{in},i}$ where $S' = |\mathsf{Amt}_{\text{out}}|$, $M = |\mathsf{SN}|$, $\mathsf{amt}_{\text{in},i} = \mathcal{L}[\mathsf{sn}_i].\mathsf{amt}$ for all $\mathsf{sn}_i \in \mathsf{SN}$ if $\mathsf{sn}_i \in \mathcal{L}$ and $\mathcal{L}[\mathsf{sn}_i].\mathsf{IsCrpt} = 1$, and $\mathsf{amt}_{\text{in},i} = 0$ otherwise,
- for any $0 \leq j < |\mathsf{PK}_{\text{out}}|$, if $\mathcal{L}[\mathsf{pk}_j].\mathsf{IsCrpt} = 0$ for $\mathsf{pk}_j = \mathsf{PK}_{\text{out}}[j]$, then $\mathsf{CK}_{\text{out}}[j] = \mathcal{L}[\mathsf{pk}_j].\mathsf{ck}$, $\mathsf{Amt}_{\text{out}}[j] = \mathcal{L}[\mathsf{pk}_j].\mathsf{amt}$ and $\mathsf{CN}_{\text{out}}[j] = \textbf{Mint}(\mathsf{Amt}_{\text{out}}[j], \mathsf{CK}_{\text{out}}[j])$.[7] That is, for all *uncorrupted* output public keys, the corresponding output coin key, output amount and output coin provided by the adversary are correct.[8]

As described in [5], the balance property covers three attack scenarios: (i) forgery, (ii) unbalanced input and output amounts and (iii) double spending.

V. MATRICT$^+$: OUR EFFICIENT LATTICE-BASED RINGCT

In this section, we describe the details of MatRiCT$^+$, our efficient lattice-based RingCT protocol. Similar to MatRiCT, we present the algorithm descriptions for the case of $M, S \leq 2$ (i.e., there are at most two input/output accounts), and discuss the general case in the text. Again for ease of presentation, the algorithm descriptions here do not perform a sanity check on the inputs, which would need to be done in practice. The number of bits of precision required to represent the sum of amounts is denoted by r and each amount is also represented in r bits. We first fix the notations in Table VI, set $\mathcal{C}$ as in (4) and define the following additional challenge set:

$$\mathcal{C}' = \left\{ c \in \mathcal{R}_q^* : \|c\|_\infty = 1 \wedge \|c\|_1 = w_\alpha \right\}. \tag{14}$$

In general, the operations are performed over $\mathcal{R}_q$ or $\mathcal{R}_{\hat{q}}$ (or without any mod reduction). Our analysis requires that $\mathcal{R}_q$ splits into exactly r factors, i.e., $\mathcal{R}_q \cong \mathcal{R}_q^{(0)} \times \cdots \times \mathcal{R}_q^{(r-1)}$. We generate a large commitment matrix $\boldsymbol{A}' \in \mathcal{R}_q^{(n+3)\times(n+\kappa+3)}$ and use its submatrices as commitment keys such that

$$\boldsymbol{A}' := \left(\begin{array}{c} \boldsymbol{A} \\ \hline \text{—}\boldsymbol{a}_1\text{—} \\ \text{—}\boldsymbol{a}_2\text{—} \\ \text{—}\boldsymbol{a}_3\text{—} \end{array}\right) = \left(\begin{array}{c|c} \boldsymbol{B} & * \\ \hline \text{—}\boldsymbol{b}_1\text{—} & * \\ * \ * \ * & * \\ * \ * \ * & * \end{array}\right) = \left(\begin{array}{c|c} \boldsymbol{G} & * \\ \hline * \ * \ * & * \\ * \ * \ * & * \\ * \ * \ * & * \end{array}\right), \tag{15}$$

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

[8] Note that the adversary may corrupt all the output public keys, so there is no loss of generality.

TABLE VI
IDENTIFIERS OF MATRICT$^+$.

Notation	Explanation
$\mathcal{R}_q, \mathcal{R}_{\hat{q}}$	cyclotomic rings of dim. d with moduli q and $\hat{q}$
ℓ	the spender's column index
r	bitlength of an amount and # of factors of $\mathcal{R}_q$
$a[i]$	i-th bit of $a \in \mathbb{Z}$
$[\![c_0, \ldots, c_{r-1}]\!]$	the CRT inverse of values $c_i \in \mathcal{R}_q^{(i)}$
$\sigma(c)$	automorphism: if $c = [\![c_0, \ldots, c_{r-1}]\!]$ & $c'_i = \sigma(c_i)$, $\sigma(c) = [\![c'_1, \ldots, c'_{r/2-1}, c'_0, c'_{r/2+1}, \ldots, c'_{r-1}, c'_{r/2}]\!]$
$n, \bar{n}, \hat{n}$	M-SIS module ranks
$\kappa, \hat{\kappa}$	M-LWE module ranks
Expand	expansion function (modelled as random oracle)

Algorithm 3 MatRiCT$^+$ Routines

procedure $\mathsf{SamMat}(\rho, q, n, m, \text{str})$: ▷ str is optional
  $\boldsymbol{C} \leftarrow \mathsf{Expand}(\rho, \text{str})$ where $\boldsymbol{C} \in \mathcal{R}_q^{n\times m}$
  return $\boldsymbol{C}$ ▷ $\boldsymbol{C}$ can be output in the CRT domain.
end procedure

procedure SETUP(1^λ):
  Choose int. params $r, n, \kappa, \bar{n}, \hat{n}, \hat{\kappa}, d, q, \hat{q}, w$
  Set $\mathcal{C}, \mathcal{C}'$ as in (4) and (14), respectively
  $\rho \xleftarrow{\$} \{0,1\}^{256}$
  Pick functions $\mathcal{H} : \{0,1\}^* \to \mathcal{C}$, $\mathcal{H}' : \{0,1\}^* \to \{0,1\}^{256}$ and $\mathcal{G} : \{0,1\}^* \to \mathcal{R}_q^{\bar{n}}$
  return $pp = (\rho, \mathcal{H}, \mathcal{H}', \mathcal{G}, r, n, \kappa, \bar{n}, \hat{n}, \hat{\kappa}, d, q, \hat{q}, w)$
end procedure

procedure KEYGEN(pp):
  $\boldsymbol{G} \leftarrow \mathsf{SamMat}(\rho, q, \bar{n}, \bar{n} + \kappa, \text{“A”})$
  $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{r} + \mathcal{G}(\gamma) \in \mathcal{R}_q^{\bar{n}}$ for $\boldsymbol{r} \xleftarrow{\$} \mathbb{S}_1^{\bar{n}+\kappa}$ & $\gamma \xleftarrow{\$} \{0,1\}^{256}$
  return $(\mathsf{pk}, \mathsf{sk}, \mathsf{sn}) = (\boldsymbol{c}, \boldsymbol{r}, \gamma)$
end procedure

procedure MINT(a): $0 \leq a < 2^r$
  $(\boldsymbol{B}^\top \| \boldsymbol{b}_1^\top)^\top \leftarrow \mathsf{SamMat}(\rho, q, n+1, n+\kappa+1, \text{“A”})$
  $\hat{a} = [\![a[0], \ldots, a[r-1]]\!] \in \mathcal{R}_q$
  $\boldsymbol{t} = \boldsymbol{B}\boldsymbol{r} \in \mathcal{R}_q^n$ for $\boldsymbol{r} \xleftarrow{\$} \mathbb{S}_1^{n+\kappa+1}$
  $t_1 = \langle \boldsymbol{b}_1, \boldsymbol{r} \rangle + \hat{a} \in \mathcal{R}_q$
  return $(\mathsf{cn}, \mathsf{ck}, \hat{a}) = ((\boldsymbol{t}, t_1), \boldsymbol{r}, \hat{a})$
end procedure

where $\boldsymbol{A} \in \mathcal{R}_q^{n\times(n+\kappa+3)}$, $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3 \in \mathcal{R}_q^{n+\kappa+3}$, $\boldsymbol{B} \in \mathcal{R}_q^{n\times(n+\kappa+1)}$, $\boldsymbol{b}_1 \in \mathcal{R}_q^{n+\kappa+1}$, and $\boldsymbol{G} \in \mathcal{R}_q^{\bar{n}\times(\bar{n}+\kappa)}$ (for $\bar{n} \leq n$). We also generate $\boldsymbol{G}' \in \mathcal{R}_{\hat{q}}^{\hat{n}\times(\hat{n}+\hat{\kappa}+2N+4)}$.

Alg. 3 presents some of the routines of MatRiCT$^+$. In particular, SamMat samples a random matrix by expanding a small seed ρ using an expansion function Expand (modelled as random oracle). Then, **Setup** sets the system parameters, random oracles and samples a random seed ρ.

KeyGen in Alg. 3 creates a public-secret key pair, where the secret key is a *short* random vector over $\mathcal{R}$. Then, to create the public key, we compute $\boldsymbol{G}\boldsymbol{r} + \mathcal{G}(\gamma)$ for some random serial number γ (i.e., **SerialGen** is computed inside **KeyGen**). The reason for the additional $\mathcal{G}(\gamma)$ term is to prevent double-spending (see further below for an intuitive discussion). In

particular, for linkability of the ring signature, we employ an idea similar to [20], but our approach is simpler as we do not require non-slanderability. As discussed in [5], non-slanderability is not a crucial requirement in the RingCT setting (though it may be in other settings).

Mint in Alg. 3 samples a *short* randomness $\boldsymbol{r}$ that serves as a coin key and the "binding part" $\boldsymbol{t}$ of a coin is computed as $\boldsymbol{Br}$. This part is quite standard, but the way we encode an amount is unique. In particular, we place each bit of the amount into a CRT slot and then compute the polynomial $\hat{a} \in \mathcal{R}_q$ whose CRT slot vector is equal to the bit vector of the input amount. The ordering of the CRT slots is set in a way so that the automorphism $\sigma(f(X)) = f(X^3)$ performs a cyclic-rotation of each half of the CRT slot vector as shown in Table VI. On the other hand, σ^{-1} is defined to be the inverse operation of σ such that $\sigma(\sigma^{-1}(f)) = f$ for all $f \in \mathcal{R}_q$. In particular, for $\sigma(f(X)) = f(X^3)$, we have $\sigma^{-1}(f(X)) = f(X^k)$ for $k = 3^{-1} \bmod 2d$.

The most important and involved procedure of MatRiCT$^+$ is **Spend**, which is split into Algorithms 4, 5 and 6. We let $\delta_{i,\ell} = 1$ if $i = \ell$, and $\delta_{i,\ell} = 0$ otherwise. Let us provide an intuitive summary of the underlying zero-knowledge proofs performed in spending.

1) Prove knowledge of an opening of each output coin (Steps 5-6 in Alg. 4 & Step 1 in Alg. 6),
2) Prove that each output coin encodes a polynomial whose CRT slots are binary (Steps 19-23 in Alg. 4),
3) Prove knowledge of an opening of the "corrector" commitment C and an opening of $\sigma(C)$ (Steps 15-17 in Alg. 4 & Steps 3-4 in Alg. 6),
4) Prove that the "corrector" commitment C encodes a polynomial whose CRT slots are in the range $[-(M-1), S-1]$ (Steps 19-23 in Alg. 4),[9]
5) Run an aggregate 1-out-of-N proof (a.k.a. a linkable ring signature) proving knowledge of an opening of some (undisclosed) element P_ℓ in $\{P_0, \ldots, P_{N-1}\}$, where each P_j is constructed by linearly combining the public keys in the j-th column of A_{in} (Steps 24-27 in Alg. 4 & Alg. 5),
   - This proof outputs masked values that can be used to "select" the elements in the ℓ-th column of A_{in} without disclosing them,
6) Prove knowledge of an opening of the sum of input coins in column ℓ of A_{in} without disclosing ℓ as above (Steps 13-14 in Alg. 4 & Step 2 in Alg. 6),
7) Prove that

$$\sum(\text{output coins}) - \sum \begin{pmatrix}\text{input coins in the} \\ \ell\text{-th column of } \mathsf{A}_{\mathsf{in}}\end{pmatrix} + C - 2\sigma(C)$$

   is a commitment to zero without disclosing ℓ as before, where C is "patched" using f_N and f_{N+1} (Step 18 in Alg. 4, Steps 4, 13 in Alg. 5 & Steps 14, 15 in Alg. 7).

Most of the remaining steps of the **Spend** function (apart from those referred to above) just creates some elements (such as norm bounds, commitment matrices or randomnesses) to be used in these proofs. However, an important operation is performed in Step 24 of Alg. 4 to prevent double-spending. Intuitively, the idea here is that a public key pk must have an "offset" $\mathcal{G}(\mathsf{sn})$ in its creation so that $\hat{\mathsf{pk}} = \mathsf{pk} - \mathcal{G}(\mathsf{sn})$ can become a commitment with a short known opening. Unless the public key is generated as in **Mint**, the spender will not be able to prove knowledge of an opening of $\hat{\mathsf{pk}}$ as $\mathcal{G}$ is modelled as a random oracle. That is, every PPT spender is bound to

[9]The algorithms describe the case $M, S \leq 2$.

Algorithm 4 Spend-I

INPUT: a transaction message μ; $M, S \in \mathbb{Z}^+$; $\mathsf{A}_{\mathsf{in}} = (\mathsf{act}_{0,0}, \ldots, \mathsf{act}_{M-1,N-1})$ where $\mathsf{act}_{i,j} = (\mathsf{pk}_{i,j}, \mathsf{cn}_{i,j})$ is an account; $\ell \in [N]$; $(\mathsf{ask}_0, \ldots, \mathsf{ask}_{M-1})$ where $\mathsf{ask}_i = (\mathsf{sk}_i, \mathsf{sn}_i, \mathsf{ck}_i, \mathsf{amt}_{\mathsf{in},i}) \in \mathbb{S}_1^{\bar{n}+\kappa} \times \{0,1\}^{256} \times \mathbb{S}_1^{n+\kappa+1} \times [2^r]$ and $\mathsf{SN} := (\mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1})$; $\mathsf{PK}_{\mathsf{out}} = (\mathsf{pk}_{\mathsf{out},0}, \ldots, \mathsf{pk}_{\mathsf{out},S-1})$ where $\mathsf{pk}_{\mathsf{out},i} \in \mathcal{R}_q^{\bar{n}}$; $\mathsf{Amt}_{\mathsf{out}} = (\mathsf{amt}_{\mathsf{out},0}, \ldots, \mathsf{amt}_{\mathsf{out},S-1})$ where $\mathsf{amt}_{\mathsf{out},i} \in [2^r]$.

Create output coins

1: $\mathsf{IN} := (pp, \mu, \mathsf{A}_{\mathsf{in}}, \mathsf{SN}, \mathsf{PK}_{\mathsf{out}})$
2: $\boldsymbol{A}' \leftarrow \mathsf{SamMat}(\rho, q, n+3, n+\kappa+3, \text{"A"})$ as in (15)
3: $\mathcal{B} = w\sqrt{d\,((S+1.5M)(n+\kappa+1) + 2(n+\kappa+3) + (\hat{n}+\hat{\kappa}))}$
4: **for** $i = 0, \ldots, S-1$ **do**
5: $\quad(\mathsf{cn}_{\mathsf{out},i}, \mathsf{ck}_{\mathsf{out},i}, \hat{a}_i) \leftarrow$ **Mint**$(\mathsf{amt}_{\mathsf{out},i})$, $\mathsf{cn}_{\mathsf{out},i} := (\boldsymbol{t}^{(i)}, t_1^{(i)})$
6: $\quad\boldsymbol{w}_{\mathsf{out},i} = \boldsymbol{B}\boldsymbol{y}_{\mathsf{out},i}$ for $\boldsymbol{y}_{\mathsf{out},i} \leftarrow \mathbb{D}_{\mathcal{B}}^{n+\kappa+1}$
7: **end for**
8: $\mathsf{CN}_{\mathsf{out}} := (\mathsf{cn}_{\mathsf{out},0}, \ldots, \mathsf{cn}_{\mathsf{out},S-1})$, $\mathsf{CK}_{\mathsf{out}} := (\mathsf{ck}_{\mathsf{out},0}, \ldots, \mathsf{ck}_{\mathsf{out},S-1})$

Corrector well-formedness and CRT range proofs

9: **for** $i = 0, \ldots, r-2$ **do** ▷ $c_0 = c_r = 0$
10: $\quad c_{i+1} = (c_i + \sum_{j=0}^{S-1} \mathsf{amt}_{\mathsf{out},j}[i] - \sum_{j=0}^{M-1} \mathsf{amt}_{\mathsf{in},j}[i])/2$
11: **end for**
12: $c = [\![c_0, \ldots, c_{r'-1}, 0, c_{r'+1}, \ldots, c_{r-1}]\!]$ for $r' := r/2$
13: $\boldsymbol{y}_0 \leftarrow \mathbb{D}_{\mathcal{B}}^{n+\kappa+1}$ and retrieve $a_0, \ldots, a_{N+1}$ in Alg. 5
14: $\boldsymbol{w}_0 = \boldsymbol{B}\boldsymbol{y}_0 - \sum_{j=0}^{N-1} a_j \sum_{i=0}^{M-1} \boldsymbol{t}_{\mathsf{in},j}^{(i)}$ for $\mathsf{cn}_{i,j} = (\boldsymbol{t}_{\mathsf{in},j}^{(i)}, v_{\mathsf{in},j}^{(i)})$
15: $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{\rho}$ for $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{S}_1^{n+\kappa+3}$
16: $C = \langle \boldsymbol{a}_3', \boldsymbol{\rho} \rangle + c$ for $\boldsymbol{a}_3' = \boldsymbol{a}_3 \cdot [\![0, 1^{r'-1}, 0, 1^{r'-1}]\!]$
17: $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{y}_i$ for $\boldsymbol{y}_i \leftarrow \mathbb{D}_{\mathcal{B}}^{n+\kappa+3}$ and $i = 1, 2$
18: $w = \langle \boldsymbol{a}_3', \boldsymbol{y}_1 \rangle - 2\sigma(\langle \boldsymbol{a}_3', \boldsymbol{y}_2 \rangle) + \langle \boldsymbol{b}_1, \sum_{i=0}^{S-1} \boldsymbol{y}_{\mathsf{out},i} - \boldsymbol{y}_0 \rangle + \sum_{j=0}^{N-1} a_j \sum_{i=0}^{M-1} v_{\mathsf{in},j}^{(i)} - (a_N - a_{N+1}) \cdot [\![0^{r'-1}, -2, 1, 0^{r'-1}]\!]$
19: $v_2 = \langle \boldsymbol{a}_2, \boldsymbol{\rho} \rangle + g_2$ for g_2 in (18)
20: $\mathsf{COM} := (\mathsf{CN}_{\mathsf{out}}, \{\boldsymbol{w}_{\mathsf{out},i}\}_{i=0}^{S-1}, \boldsymbol{w}_0, \boldsymbol{v}, v_2, C, w, \boldsymbol{w}_1, \boldsymbol{w}_2)$
21: $\alpha_0, \ldots, \alpha_{S-1}, \alpha_0', \ldots, \alpha_{M-2}' \leftarrow \mathsf{Expand}(\alpha, \text{"Ch"})$ for $\alpha \leftarrow \mathcal{H}'(\mathsf{IN}, \mathsf{COM})$, where $\alpha_i \in \mathcal{R}_q^*$ & $\alpha_i' \in \mathcal{C}'$
22: $v_0 = \langle \boldsymbol{a}_1, \boldsymbol{y}_1 \rangle + \hat{g}_0$ for $\hat{g}_0$ in (21)
23: $v_1 = \langle \boldsymbol{a}_1, \boldsymbol{\rho} \rangle + \langle \boldsymbol{a}_2, \boldsymbol{y}_1 \rangle + \hat{g}_1$ for $\hat{g}_1$ in (21)

Linear combinations for aggregate 1-out-of-N proof

24: $\hat{\mathsf{pk}}_{i,j} = \mathsf{pk}_{i,j} - \mathcal{G}(\mathsf{sn}_i)$ for $i \in [M]$ & $j \in [N]$
25: $P_j = \sum_{i=0}^{M-2} \alpha_i' \hat{\mathsf{pk}}_{i,j} + \hat{\mathsf{pk}}_{M-1,j}$ in $\mathcal{R}_q^{\bar{n}}$ for $j = 0, \ldots, N-1$
26: $\boldsymbol{s} = \sum_{i=0}^{M-2} \alpha_i' \mathsf{sk}_i + \mathsf{sk}_{M-1}$
27: Continue with running $\mathsf{RSign}(\mu, (P_0, \ldots, P_{N-1}), \ell, \boldsymbol{s})$

Algorithm 5 $\mathsf{RSign}(\mu, (P_0, \ldots, P_{N-1}), \ell, \mathsf{sk})$ ▷ Ring Sign.

INPUT: a message μ; commitments $P_j \in \mathcal{R}_q^{\bar{n}}$ for $j = 0, \ldots, N-1$; $\ell \in [N]$; and $\mathsf{sk} = \boldsymbol{s} \in \mathbb{S}_\beta^{\bar{n}+\kappa}$.

ASSUME: $\|\boldsymbol{s}\|_\infty \leq \beta := w_\alpha \min\{M-1, \sqrt{2(M-1)}\}+1$

1: $\mathcal{B}_a = \sqrt{2w}$, $\phi_a = 10$, $\phi_{rs} = 13$, $\mathcal{B}_{\text{rs}} = \beta w \sqrt{d(\bar{n}+\kappa)}$
2: $\mathcal{B}_{\text{g1}} = (\phi_a \mathcal{B}_a)^2 d/2$, $\mathcal{B}_{\text{g0}} = (\phi_a \mathcal{B}_a)^2 d(N-1)/3$
3: $\boldsymbol{G}' \leftarrow \mathsf{SamMat}(\rho, \hat{q}, \hat{n}, \hat{n}+\hat{\kappa}+2N+4, \text{“G”})$
4: Set $k, b_{1-k} \in \{0,1\}$ s.t. $b_0 - b_1 = c_{32} \in \{\pm 1, 0\}$ & $b_k = 0$
5: $\boldsymbol{b} = (\delta_{\ell,0}, \ldots, \delta_{\ell,N-1}, b_0, b_1)$
6: $a_1, \ldots, a_{N+1} \leftarrow \mathbb{D}_{\phi_a \mathcal{B}_a}$ and $a_0 = -\sum_{j=1}^{N-1} a_j$
7: $\boldsymbol{a} = (a_0, \ldots, a_{N+1})$, $\boldsymbol{d} = \left(-a_0^2, \ldots, -a_{N+1}^2\right)$
8: $\boldsymbol{c} = \boldsymbol{a} \circ (1^{N+1} - 2 \cdot \boldsymbol{b})$, for $\circ$ denoting entry-wise product
9: $B = \boldsymbol{G}' \begin{pmatrix} \boldsymbol{r}_b \\ \boldsymbol{b} \\ \boldsymbol{c} \end{pmatrix}$, $A = \boldsymbol{G}' \begin{pmatrix} \boldsymbol{r}_a \\ \boldsymbol{a} \\ \boldsymbol{d} \end{pmatrix}$ for $\boldsymbol{r}_b \xleftarrow{\$} \mathbb{S}_1^{\hat{n}+\hat{\kappa}}, \boldsymbol{r}_a \leftarrow \mathbb{D}_{\mathcal{B}}^{\hat{n}+\hat{\kappa}}$
10: $E = \boldsymbol{G}\boldsymbol{y} - \sum_{j=0}^{N-1} a_j P_j$ for $\boldsymbol{y} \xleftarrow{\$} \mathbb{D}_{\phi_{rs}\mathcal{B}_{\text{rs}}}^{\bar{n}+\kappa}$
11: $x = \mathcal{H}(\mathsf{IN}, \mathsf{COM}, v_0, v_1, A, B, E)$
12: $f_j = x\delta_{\ell,j} + a_j$ for $j = 0, \ldots, N-1$
13: $f_N = x b_0 + a_N$, $f_{N+1} = x b_1 + a_{N+1}$
14: $\boldsymbol{f}_1 = (f_1, \ldots, f_{N+1})$ ▷ f_0 is excluded
15: **if** $\mathsf{Rej}(\boldsymbol{f}_1, x(\delta_{\ell,1}, \ldots, \delta_{\ell,N-1}, b_0, b_1), \phi_a, \mathcal{B}_a)$, **then** Restart
16: $g_0 = f_0(x - f_0)$, $\boldsymbol{g}_1 = (f_1(x-f_1), \ldots, f_{N+1}(x - f_{N+1}))$
17: **if** $\|g_0\|_\infty > \mathcal{B}_{\text{g0}}$ or $\|\boldsymbol{g}_1\|_\infty > \mathcal{B}_{\text{g1}}$, **then** Restart
18: $\boldsymbol{z}_b = \boldsymbol{r}_a + x\boldsymbol{r}_b$ ▷ dim: $\hat{n}+\hat{\kappa}$
19: $\boldsymbol{z} = \boldsymbol{y} + x\boldsymbol{s}$ ▷ dim: $\bar{n}+\kappa$
20: **if** $\mathsf{Rej}(\boldsymbol{z}, x\boldsymbol{s}, \phi_{rs}, \mathcal{B}_{\text{rs}})$, **then** Restart
21: Run **Spend**-II

Algorithm 6 **Spend**-II

1: $\boldsymbol{z}_{\text{out},i} = \boldsymbol{y}_{\text{out},i} + x\boldsymbol{r}_{\text{out},i}$ for $\boldsymbol{r}_{\text{out},i} := \mathsf{ck}_{\text{out},i}$ & $i \in [S]$
——— Opening of $C, \sigma(C)$ and sum of input coins
2: $\boldsymbol{z}_0 = \boldsymbol{y}_0 + x\sum_{i=0}^{M-1} \boldsymbol{r}_i$ for $\boldsymbol{r}_i = \mathsf{ck}_i$ ▷ dim: $n+\kappa+1$
3: $\boldsymbol{z}_1 = \boldsymbol{y}_1 + x\boldsymbol{\rho}$ ▷ dim: $n+\kappa+3$
4: $\boldsymbol{z}_2 = \boldsymbol{y}_2 + \sigma^{-1}(x)\boldsymbol{\rho}$ ▷ dim: $n+\kappa+3$
5: **if** $\mathsf{RejOp}((\{\boldsymbol{z}_{\text{out},i}\}_{i\in[S]}, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_0, \boldsymbol{z}_b), (\{x\boldsymbol{r}_{\text{out},i}\}_{i\in[S]}, x\boldsymbol{\rho}, \sigma^{-1}(x)\boldsymbol{\rho}, x\sum_{i=0}^{M-1} \boldsymbol{r}_i, x\boldsymbol{r}_b), 1, \mathcal{B})$, **then** Restart
6: **return** $\pi = (C, \boldsymbol{v}, v_1, v_2, B, \alpha, x, \boldsymbol{z}_{\text{out},0}, \ldots, \boldsymbol{z}_{\text{out},S-1}, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{f}_1, \boldsymbol{z}_b, \boldsymbol{z})$, $\mathsf{A_{in}}$, $\mathsf{CN_{out}}$, $\mathsf{PK_{out}}$, SN, $(\mathsf{CK_{out}}, \mathsf{Amt_{out}})$

publishing the value γ in **Mint** as the serial number and no PPT spender can find another γ' such that $\mathcal{G}(\gamma) = \mathcal{G}(\gamma')$.

It's easy to observe that a user may easily create two distinct accounts with the same serial number. However, this does not benefit the user since as soon as one of the accounts is spent, the other becomes unspendable. This stems from the fact that trying to spend the latter account requires using an already used serial number, which will be rejected by the verifiers.

On the other hand, if an attacker (somehow) gets to know an honest user account's serial number, then he can create a new account with that serial number to prevent the honest user from spending her account. This can be easily prevented as we discuss in Appendix D and is also not an attack against balance, but is an attack against availability (see also [5, Appendix B]).

Below we define some terms computed in the algorithms. The verifier uses the middle expressions to compute h_i's whereas the spender uses the last expressions. For $i = 0, \ldots, S-1$, let

$$h_S = \langle \boldsymbol{a}'_3, \boldsymbol{z}_1 \rangle - xC = \langle \boldsymbol{a}'_3, \boldsymbol{y}_1 \rangle - xc, \tag{16}$$

$$h_i = \langle \boldsymbol{b}_1, \boldsymbol{z}_{\text{out},i} \rangle - xt_1^{(i)} = \langle \boldsymbol{b}_1, \boldsymbol{y}_{\text{out},i} \rangle - x\hat{a}_i. \tag{17}$$

Let us treat x as a variable and define the following equalities

$$h_S(h_S + x)(h_S - x) = g_0^{(S)} + x \cdot g_1^{(S)} + x^2 \cdot g_2 - x^3 \cdot (c(c-1)(c+1)), \tag{18}$$

$$h_i(h_i + x) = g_0^{(i)} + x \cdot g_1^{(i)} + x^2 \cdot (\hat{a}_i(\hat{a}_i - 1)), \tag{19}$$

for some polynomials $g_2, g_0^{(i)}, g_1^{(i)}$ and $i \in [S]$, all independent of x. Therefore, for any $\alpha_0, \ldots, \alpha_{S-1} \in \mathcal{R}$, we have

$$h_S(h_S + x)(h_S - x) + \sum_{i=0}^{S-1} \alpha_i h_i(h_i + x) = \hat{g}_0 + x \cdot \hat{g}_1 + x^2 \left(\sum_{i=0}^{S-1} \alpha_i \hat{a}_i(\hat{a}_i - 1) + g_2 \right) - x^3(c(c-1)(c+1)), \tag{20}$$

where

$$\hat{g}_0 := \sum_{i=0}^{S-1} \alpha_i g_0^{(i)} + g_0^{(S)}, \text{ and } \hat{g}_1 := \sum_{i=0}^{S-1} \alpha_i g_1^{(i)} + g_1^{(S)}. \tag{21}$$

The 1-out-of-N proof (or ring signature) described in Alg. 5 is a simplified version of that in [5] with some additional steps. As we focus on medium-sized ring signatures (with ring size $N \leq 100$), we opt to use this simplified version. If a larger ring size is desired, the logarithmic-sized ring signature in [5] can be used. For $w_\alpha = 39$ as in our concrete parameter setting, we computed empirically the probability that a particular coefficient of $\boldsymbol{s}$ exceeds $\beta := w_\alpha \cdot \sqrt{2(M-1)} + 1$ and found that it is less than 2^{-94} for $M \leq 100$. Note that the actual "bad case" probability of $\Pr[\|x\boldsymbol{s}\| > \mathcal{B}_{\text{rs}}]$ is even much lower than $\Pr[\|\boldsymbol{s}\|_\infty > \beta]$. We performed a similar computation for $\boldsymbol{r}' := x\sum_{i=0}^{M-1} \boldsymbol{r}_i$ in $\boldsymbol{z}_0$, and found that the probability of a particular coefficient of $\boldsymbol{r}'$ exceeding $w\sqrt{1.5M}$ in absolute value is less than 2^{-101} for $w = 56$ and $M \leq 100$. For values of $M \gg S$, we can apply RejOp in Step 5 of Alg. 6 separately on $\boldsymbol{z}_0$ and the other vectors to reduce the proof size by sampling the other vectors with a smaller standard deviation.

It is easy to see that we can construct a value $c_{32} \in \{-1, 0, 1\}$ by the difference of two bits $b_0, b_1 \in \{0, 1\}$. In fact, one of the b_i's can always be set to zero. This is precisely what is done in Step 4 of Alg. 5.

The verification function in Alg. 7 computes the "missing" components that are not output in the proof, checks that masked randomness openings are short and that the hash

inputs are consistent. The algorithms **AccountGen**, **IsSpent** and **CheckAct** are assumed to be available and used implicitly in Alg. 7 as described in Section IV.

Let us come back to the general case when we have M input and S output accounts (without necessarily having $M, S \leq 2$). In this case, "corrector" values will fall in the range $[-(M-1), S-1]$ and we therefore need to prove that the "corrector" commitment C encodes a polynomial whose CRT slots are in the range $[-(M-1), S-1]$. In fact, as discussed in [5], it is also sufficient to prove the CRT slots are in some range U with $[-(M-1), S-1] \subseteq U \subseteq [-(q-1)/4, (q-1)/4]$. Hence, we can choose a set U of a power-of-2 width. Let $t = \log_2 |U|$. Then, the spender proves that commitments $C_0, \ldots, C_{t-1}$ encodes t polynomials with binary CRT slots. Then, the spender and verifier construct $C := \sum_{i=0}^{t-1} 2^i C_i - [\![M-1, \ldots, M-1]\!]$. By construction, C satisfies the desired property. Similarly, to construct an encoding of c_{32} to "patch" C' in Alg. 7, we can use t polynomials $f_N, \ldots, f_{N+t-1}$.

We discuss in Appendix C how to extend MatRiCT$^+$ to provide recipient anonymity and auditability. In particular, MatRiCT$^+$ supports the same *auditability* feature as MatRiCT, where an authority can revoke the anonymity of a user, e.g., in case of illegal activity. We can either allow the system to enforce auditability or leave it optional for the user to decide. This property is important to introduce some level of accountability into applications where it is desirable.

A. Parameter Setting and Implementation

Our parameter setting is done to satisfy the requirements listed in Assumption 1 in Section VI. We first set $r = 64$ as 64-bit precision is sufficient in practice. Then, as in MatRiCT, we focus on $M = S = 2$ for the concrete parameter setting as this is the most typical transaction scenario. We also set $N = 11$ as in Monero, but other anonymity set sizes can of course be supported.

We need $\mathcal{R}_q$ to split into exactly $r = 64$ factors, and also want $\mathcal{R}_{\hat{q}}$ to split as much as possible while ensuring challenge difference invertibility. We achieve this by restricting the moduli $q, \hat{q}$ to primes with $q, \hat{q} \equiv 2r+1 \pmod{4r}$. Then, we set $(d, w) = (256, 56)$ so that (i) any challenge difference is invertible (except for a negligible probability of $< 2^{-94}$) by the results in Section III, and (ii) $|\mathcal{C}|$ is sufficiently large (in particular, we get $|\mathcal{C}| > 2^{237}$). Then, we set $w_\alpha = 39$ so that $|\mathcal{C}'| > 2^{192}$. Both challenge sets $\mathcal{C}, \mathcal{C}'$ have enough entropy. Finally, we set $(n, \bar{n}, \kappa) = (4, 4, 4)$, $(\hat{n}, \hat{\kappa}) = (5, 5)$, $q = 167770241$ and $\hat{q} = 2^{34} - 2^{26} - 2^7 + 1$. With this parameter setting, the root Hermite factor (RHF), a standard metric to estimate M-SIS/LWE hardness in practice, is between 1.0043 and 1.0049 for all M-SIS/LWE assumptions in Assumption 1. It is a common practice to choose a RHF around 1.0045 for 128-bit post-quantum security (see, e.g., [5]–[7], [9]). The RHF calculations for M-SIS and M-LWE were done according to the methodology described in [21, Section 3.2.4]. For M-SIS$^\infty$ in Sub-Assumption 7, we used Dilithium's scripts[10]. The latter gave the largest RHF among our assumptions, but we believe the Dilithium script underestimates this assumption's hardness and can be improved using "Asymmetric MSIS" analysis as in [22] since the largest bound $\mathcal{B}_{\text{g0}}$ is only for a single coordinate of the vector over $\mathcal{R}_{\hat{q}}$ while all the other

[10] https://github.com/pq-crystals/security-estimates

Algorithm 7 Verify

INPUT: μ, $M, S \in \mathbb{Z}^+$, A_{in}, PK_{out}, SN as in Alg. 4; $\mathsf{CN}_{\text{out}} = (\mathsf{cn}_{\text{out},0}, \ldots, \mathsf{cn}_{\text{out},S-1})$; $\pi = (C, \boldsymbol{v}, v_1, v_2, B, \alpha, x, \boldsymbol{z}_{\text{out},0}, \ldots, \boldsymbol{z}_{\text{out},S-1}, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{f}_1, z_b, \boldsymbol{z})$.

Opening proof of output coins

$\boldsymbol{A}' \leftarrow \mathsf{SamMat}(\rho, q, n+3, n+\kappa+3, \text{"A"})$ as in (15)
Set $\phi_a, \mathcal{B}_a, \mathcal{B}_{\text{g1}}, \mathcal{B}_{\text{g0}}, \mathcal{B}, \phi_{rs}, \mathcal{B}_{\text{rs}}$ as in Alg. 4 and 5
if $\|(\{\boldsymbol{z}_{\text{out},i}\}_{i\in[S]}, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_0, \boldsymbol{z}_b)\|_\infty > 6\mathcal{B}$, **then return** 0
for $i = 0, \ldots, S-1$ **do**
  if $\|\boldsymbol{z}_{\text{out},i}\| > 1.2\mathcal{B}\sqrt{d(n+\kappa+1)}$, **then return** 0
  $\boldsymbol{w}_{\text{out},i} = \boldsymbol{B}\boldsymbol{z}_{\text{out},i} - x\boldsymbol{t}^{(i)}$ for $\mathsf{cn}_{\text{out},i} = (\boldsymbol{t}^{(i)}, t_1^{(i)})$
  $h_i = \langle \boldsymbol{b}_1, \boldsymbol{z}_{\text{out},i} \rangle - x t_1^{(i)}$ ▷ for bin CRT proof of $\mathsf{cn}_{\text{out},i}$
end for

Corrector well-formedness and CRT range proofs

if $\|\boldsymbol{z}_0\| > 1.2 \cdot \mathcal{B}\sqrt{d(n+\kappa+1)}$, **then return** 0
if $\|\boldsymbol{z}_i\| > 1.2 \cdot \mathcal{B}\sqrt{d(n+\kappa+3)}$ for some $i \in \{1, 2\}$, **then return** 0
$C' = C \cdot [\![0, 1^{r'-1}, 0, 1^{r'-1}]\!]$, $\boldsymbol{a}_3' = \boldsymbol{a}_3 \cdot [\![0, 1^{r'-1}, 0, 1^{r'-1}]\!]$
$\boldsymbol{w}_0 = \boldsymbol{B}\boldsymbol{z}_0 - \sum_{j=0}^{N-1} f_j \sum_{i=0}^{M-1} \boldsymbol{t}_{\text{in},j}^{(i)}$ for $\mathsf{cn}_{i,j} = (\boldsymbol{t}_{\text{in},j}^{(i)}, v_{\text{in},j}^{(i)})$
$\boldsymbol{w}_1 = \boldsymbol{A}\boldsymbol{z}_1 - x\boldsymbol{v}$, $\boldsymbol{w}_2 = \boldsymbol{A}\boldsymbol{z}_2 - \sigma^{-1}(x)\boldsymbol{v}$
$u = (f_N - f_{N+1}) \cdot [\![0^{r'-1}, -2, 1, 0^{r'-1}]\!]$
$w = \langle \boldsymbol{a}_3', \boldsymbol{z}_1 \rangle - 2\sigma(\langle \boldsymbol{a}_3', \boldsymbol{z}_2 \rangle) + \langle \boldsymbol{b}_1, \sum_{i=0}^{S-1} \boldsymbol{z}_{\text{out},i} - \boldsymbol{z}_0 \rangle - x \sum_{i=0}^{S-1} t_1^{(i)} + \sum_{j=0}^{N-1} f_j \sum_{i=0}^{M-1} v_{\text{in},j}^{(i)} - xC' - u + x2\sigma(C')$
if $\alpha \neq \mathcal{H}'(\mathsf{IN}, \mathsf{COM})$ for $\mathsf{IN}, \mathsf{COM}$ as in Alg. 4, **then return** 0
$\alpha_0, \ldots, \alpha_{S-1}, \alpha_0', \ldots, \alpha_{M-2}' \leftarrow \mathsf{Expand}(\alpha, \text{"Ch"})$ for $\alpha \leftarrow \mathcal{H}'(\mathsf{IN}, \mathsf{COM})$, where $\alpha_i \in \mathcal{R}_q$ & $\alpha_i' \in \mathcal{C}'$
$h_S = \langle \boldsymbol{a}_3', \boldsymbol{z}_1 \rangle - xC'$
$v_0 = h_S(h_S + x)(h_S - x) + \sum_{i=0}^{S-1} \alpha_i h_i (h_i + x) - xv_1 - x^2 v_2 + \langle \boldsymbol{a}_1, \boldsymbol{z}_1 \rangle + x\langle \boldsymbol{a}_2, \boldsymbol{z}_1 \rangle$

Aggregate 1-out-of-N Proof

if $\|\boldsymbol{f}_1\|_\infty > 6 \cdot \phi_a \mathcal{B}_a$, **then return** 0
if $\|\boldsymbol{z}\| > 1.2 \cdot \phi_{rs}\mathcal{B}_{\text{rs}}\sqrt{d(\bar{n}+\kappa)}$ or $\|\boldsymbol{z}\|_\infty > 6 \cdot \phi_{rs}\mathcal{B}_{\text{rs}}$, **then return** 0
$f_0 = x - \sum_{j=0}^{N-1} f_j$ for $\boldsymbol{f}_1 = (f_1, \ldots, f_{N+1})$
Define $\boldsymbol{f} := (f_0, \ldots, f_{N+1})$ and $g_0, \boldsymbol{g}_1$ as in Alg. 5
$\boldsymbol{g} = (f_0(x - f_0), \ldots, f_{N+1}(x - f_{N+1}))$
if $\|g_0\|_\infty > \mathcal{B}_{\text{g0}}$ or $\|\boldsymbol{g}_1\|_\infty > \mathcal{B}_{\text{g1}}$, **then return** 0
$\boldsymbol{G}' \leftarrow \mathsf{SamMat}(\rho, \hat{q}, \hat{n}, \hat{n} + \hat{\kappa} + 2N + 4, \text{"G"})$
$A = \boldsymbol{G}' \begin{pmatrix} \boldsymbol{z}_b \\ \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix} - xB$ in $\mathcal{R}_{\hat{q}}^{\hat{n}}$
Compute $P_0, \ldots, P_{N-1}$ as in Alg. 4 using SN in the input
$E = \boldsymbol{G}\boldsymbol{z} - \sum_{j=0}^{N-1} f_j P_j$
if $x \neq \mathcal{H}(\mathsf{IN}, \mathsf{COM}, v_0, v_1, A, B, E)$, **then return** 0
return 1

coordinates have a smaller upperbound. Also, the choice of q is in fact slightly smaller than the maximum of $\beta_{\text{SIS}}, \beta'_{\text{SIS}}$ in Assumption 1. However, $(q, 0, \ldots, 0)$ is not an acceptable solution in our case due to the additional infinity norm checks (in particular, we still have $q > 12\mathcal{B}$ and $q > 12\phi_{rs}\mathcal{B}_{\text{rs}}$).

We implemented MatRiCT$^+$ in C/C++ on a standard desktop machine with i7-7700K CPU.[11] We employ (partial) NTT multiplication for polynomial arithmetic over $\mathcal{R}_q$ and $\mathcal{R}_{\hat{q}}$, and use SHAKE-256 to implement $\mathcal{H}$ and $\mathcal{H}'$. AES-256 (with AES-NI instructions) in counter mode is used to realize Expand. We adapt the FACCT sampler [23] in order to sample from discrete Gaussian $\mathbb{D}$. Since the FACCT sampler can only support standard deviation being an integer multiple of $\sqrt{1/(2\ln 2)}$, we round the standard deviations $\mathcal{B}$, $\phi_a\mathcal{B}_a$, $\phi_{rs}\mathcal{B}_{\text{rs}}$ up to the nearest integer multiples of $\sqrt{1/(2\ln 2)}$. To avoid timing leakage in Rej and RejOp, we adapt the constant-time Bernoulli sampling technique with bias $\exp(x)$ from [23]. A sample of concrete runtimes of MatRiCT$^+$ is given in Table II, where the runtimes are the average number of cycles (of 1000 runs) divided by $3 \cdot 10^6$. We refer to Appendix C-C for further details on the implementation and evaluation.

VI. Security Proofs

The correctness property of MatRiCT$^+$ follows via straightforward investigation and we provide the anonymity proof in Appendix G. To make the balance discussion more intuitive, we use the following notations in terms of the two commitment schemes "UMC" [24] and "HMC" (see Appendix B):

$$\mathsf{UMC}(m_1;\, \boldsymbol{r}) = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{b}_1 \end{pmatrix} \boldsymbol{r} + \begin{pmatrix} \boldsymbol{0} \\ m_1 \end{pmatrix}, \text{ and}$$
$$\mathsf{HMC}(m;\, \boldsymbol{r}) = \boldsymbol{G}\boldsymbol{r} + m \cdot \boldsymbol{g},$$

where $\boldsymbol{g}$ is a uniformly random vector in $\mathcal{R}_q^{\bar{n}}$. We denote the set of non-zero differences of challenges in $\mathcal{C}$ by $\Delta\mathcal{C}$. We list all the assumptions below and write "Sub-Assumption i" to refer to a particular one.

Assumption 1.

1) Any $y \in \Delta\mathcal{C}$ is invertible in $\mathcal{R}_q$ and in $\mathcal{R}_{\hat{q}}$,
2) $\mathcal{R}_q$ splits into exactly r factors,
3) ($\hat{q} > \max\{d(2 + 12\phi_a\mathcal{B}_a)^2, 2N^2\}$) or ($\hat{q} > \max\{r(2 + 12\phi_a\mathcal{B}_a)^2, 2N^2\}$ and $\hat{q} = 1 \bmod 2r$),
4) $q/2 > \max\{2M, 2S\}$,
5) M-SIS$_{n,q,\beta_{\text{SIS}}}$ is hard for $\beta_{\text{SIS}} = 8w \cdot 1.2\mathcal{B}\sqrt{d(n + \kappa + 3)}$,
6) M-SIS$_{\bar{n},q,\beta'_{\text{SIS}}}$ is hard for $\beta'_{\text{SIS}} = 2.4 \cdot \phi_{rs}\mathcal{B}_{rs}\sqrt{d(\bar{n} + \kappa)}$,
7) M-SIS$^{\infty}_{\hat{n},\hat{q},\mathcal{B}_{max}}$ is hard for $\mathcal{B}_{max} = 4w \cdot \max\{\mathcal{B}_{g0}, \mathcal{B}_{g1}, 6\mathcal{B}\}$,
8) M-LWE$_{\kappa,q,1}$ is hard,
9) M-LWE$_{\hat{\kappa},\hat{q},1}$ is hard.

A. Balance

The balance property is proven under three cases. The adversary tries to (i) double-spend, which is prevented thanks to the serial numbers; (ii) create a transaction with unmatching input/output amounts, which is prevented by our novel balance proof; (iii) create a transaction without owning M accounts, which is prevented by the aggregated one-out-of-many proof (or the ring signature).

Theorem 1. *(Balance) If the assumptions given in Assumption 1 hold, then MatRiCT$^+$ is balanced without shuffling.*

Proof. We study the balance property in 3 cases. In all cases, we consider an algorithm $\mathcal{S}$ that runs a successful adversary $\mathcal{A}$ against the balance game `Exp:Balance`. Note that every spender performs the zero-knowledge proofs described in Section V. Therefore, using a standard rewinding argument and the extractor of the underlying zero-knowledge proofs (see Appendix E for more details), the algorithm $\mathcal{S}$ can compute $(y, \ell, \hat{\boldsymbol{s}}, \hat{a}_{\text{out},i}, \bar{\boldsymbol{z}}_{\text{out},i}, \hat{\boldsymbol{r}}_0, \hat{a}_{\text{in}}, c_1, \ldots, c_{r-1})$ with the following properties

$$y \cdot \mathsf{cn}_{\text{out},i} = \mathsf{UMC}(y\hat{a}_{\text{out},i};\, \bar{\boldsymbol{z}}_{\text{out},i}) \text{ for } i \in [S-1], \tag{22}$$
$$yP_\ell = \mathsf{HMC}(0;\, \hat{\boldsymbol{s}}), \tag{23}$$
$$\sum_{i=0}^{M-1} v_{\text{in},\ell}^{(i)} = \langle \boldsymbol{b}_1, \hat{\boldsymbol{r}}_0 \rangle + \hat{a}_{\text{in}}, \tag{24}$$
$$\sum_{i=0}^{S-1} \hat{a}_{\text{out},i} - \hat{a}_{\text{in}} + [\![c_0 - 2c_1, \ldots, c_{r-1} - 2c_r]\!] = 0, \tag{25}$$

where

1) $y \in \Delta\mathcal{C}$,
2) $c_1, \ldots, c_{r-1} \in [-(M-1), S-1]$ and $c_0 = c_r = 0$,
3) $\hat{a}_{\text{out},i} = [\![b_{\text{out},i}^{(0)}, \ldots, b_{\text{out},i}^{(r-1)}]\!]$ for some $b_{\text{out},i}^{(j)} \in \{0, 1\}$,
4) $\|\bar{\boldsymbol{z}}_{\text{out},i}\|_\infty \leq 12\mathcal{B}$ and $\|\bar{\boldsymbol{z}}_{\text{out},i}\| \leq 2.4\mathcal{B}\sqrt{d(n + \kappa + 1)}$,
5) $\ell \in \{0, \ldots, N-1\}$,
6) $\|\hat{\boldsymbol{s}}\|_\infty \leq 12 \cdot \phi_{rs}\mathcal{B}_{\text{rs}}$ and $\|\hat{\boldsymbol{s}}\| \leq 2.4 \cdot \phi_{rs}\mathcal{B}_{\text{rs}}\sqrt{d(\bar{n} + \kappa)}$.

Case 1 (Double-spend): Let Q be the maximum number of any oracle queries $\mathcal{A}$ makes to any random oracle. Let $\mathcal{E}_{\text{2xspend}}$ be the event that $\mathcal{A}$ wins `Exp:Balance` in the following way: there exists $\mathsf{sn}_{i^*} \in \mathsf{SN}$ with $0 \leq i^* \leq M-1$ such that $\mathsf{sn}_{i^*} \notin \mathcal{L}$. Suppose that $\Pr[\mathcal{E}_{\text{2xspend}}]$ is non-negligible and assumptions in the theorem statement hold. In this case, $\mathcal{S}$ runs $\mathcal{A}$ in a modified game with a different simulation of the random oracle $\mathcal{G}$ that works as follows: for each new i'th query γ_i, instead of responding with an independent uniformly random element of $\mathcal{R}_q^{\bar{n}}$, $\mathcal{S}$ responds with $\mathsf{HMC}(0;\, \boldsymbol{z}_i)$ with an independent $\boldsymbol{z}_i \xleftarrow{\$} \mathbb{S}_1^{\bar{n}+\kappa}$. The attacker's view in this modified game is computationally indistinguishable from its view in the original attack, by the assumed hardness of M-LWE$_{\kappa,q,1}$, using a hybrid argument over all the Q queries made by $\mathcal{A}$ to $\mathcal{G}$ in the game.

Using this modified game, the $\mathcal{S}$ runs $\mathcal{A}$ with until $\mathcal{E}_{\text{2xspend}}$ happens three times with respect to distinct $\mathcal{H}$ outputs and the same $\mathcal{H}$ inputs, and then runs the extractor of the 1-out-of-N proof to obtain (23). This equation is equivalent to

$$y \cdot \left(\sum_{i=0}^{M-2} \alpha'_i \hat{\mathsf{pk}}_{i,\ell} + \hat{\mathsf{pk}}_{M-1,\ell} \right) = \mathsf{HMC}(0;\, \hat{\boldsymbol{s}}), \tag{26}$$

[11] The implementation source code is available at https://gitlab.com/raykzhao/matrict_plus.

where $\hat{\mathsf{pk}}_{i,\ell} = \mathsf{pk}_{i,\ell} - \mathcal{G}(\mathsf{sn}_i)$ for $i \in [M]$ and sn_i's are the serial numbers $\mathcal{A}$ returns in his output transaction tx. By the assumption that public keys are generated by PKGEN, we have

$$\mathsf{pk}_{i,\ell} = \mathsf{HMC}(0; \boldsymbol{r}_i) + \mathcal{G}(\gamma_i), \tag{27}$$

for some short vector $\boldsymbol{r}_i$'s and some 256-bit string γ_i's. Since $\mathsf{sn}_{i^*} \notin \mathcal{L}$, then $\gamma_{i^*} \neq \mathsf{sn}_{i^*}$. Hence, we have

$$\hat{\mathsf{pk}}_{i^*,\ell} = \mathsf{HMC}(0; \boldsymbol{r}_{i^*}) + \mathcal{G}(\gamma_{i^*}) - \mathcal{G}(\mathsf{sn}_{i^*}). \tag{28}$$

Defining $\alpha'_{M-1} := 1$, expanding (26) using (27), and moving HMC commitments to the right-hand side of (26), we get

$$y\left(\sum_{i=0}^{M-1} \alpha'_i(\mathcal{G}(\gamma_i) - \mathcal{G}(\mathsf{sn}_i))\right) - \mathsf{HMC}(0; \hat{\boldsymbol{z}}) = \boldsymbol{0}, \tag{29}$$

for $\hat{\boldsymbol{z}} := \hat{\boldsymbol{s}} - y\sum_{i=0}^{M-1} \alpha'_i \boldsymbol{r}_i$. By definition of the modified $\mathcal{G}$ oracle simulation, we have $\mathcal{G}(\gamma_i) = \mathsf{HMC}(0; \boldsymbol{z}_i)$ and $\mathcal{G}(\mathsf{sn}_i) = \mathsf{HMC}(0; \boldsymbol{z}'_i)$ for some $\boldsymbol{z}_i, \boldsymbol{z}'_i$. Therefore, we have $\mathsf{HMC}(0; \boldsymbol{v}) = \boldsymbol{0}$, where $\boldsymbol{v} := y\left(\sum_{i=0}^{M-1} \alpha'_i(\boldsymbol{z}_i - \boldsymbol{z}'_i)\right) - \hat{\boldsymbol{z}}$. Since the norm bound on $\hat{\boldsymbol{s}}$ is much bigger than the bounds for the remaining terms in $\boldsymbol{v}$, we simply take $\|\boldsymbol{v}\| \leq 2.4\phi_{rs}\mathcal{B}_{\mathrm{rs}}\sqrt{d(\bar{n}+\kappa)}$. Let $\mathcal{E}^{\boldsymbol{v}\neq 0}_{\mathrm{2xspend}}$ (resp. $\mathcal{E}^{\boldsymbol{v}=0}_{\mathrm{2xspend}}$) denote the subevents that partition $\mathcal{E}_{\mathrm{2xspend}}$ depending on whether $\boldsymbol{v} \neq 0$ (resp. $\boldsymbol{v} = 0$). Therefore, if $\mathcal{E}^{\boldsymbol{v}\neq 0}_{\mathrm{2xspend}}$ occurs, then $\boldsymbol{v}$ is a solution to M-SIS$_{\bar{n},q,\beta'_{\mathrm{SIS}}}$, so $\Pr[\mathcal{E}^{\boldsymbol{v}\neq 0}_{\mathrm{2xspend}}]$ is negligible by hardness of M-SIS$_{\bar{n},q,\beta'_{\mathrm{SIS}}}$. We can therefore assume that $\Pr[\mathcal{E}^{\boldsymbol{v}=0}_{\mathrm{2xspend}}]$ is non-negligible. If $\mathcal{E}^{\boldsymbol{v}=0}_{\mathrm{2xspend}}$ occurs, we conclude that

$$\boldsymbol{z}_{i^*} = (y\alpha'_{i^*})^{-1}\left(\hat{\boldsymbol{z}} + y\alpha'_{i^*}\boldsymbol{z}'_{i^*} - y\alpha'_i\sum_{i\neq i^*}\alpha'_i(\boldsymbol{z}_i - \boldsymbol{z}'_i)\right). \tag{30}$$

Note that since $\mathsf{sn}_{i^*} \neq \gamma_{i^*}$ we know that $\boldsymbol{z}'_{i^*} \neq \boldsymbol{z}_{i^*}$. Furthermore, we can assume that none of the γ_i and sn_i for $i \neq i^*$ are equal to γ_{i^*} using that fact that the γ_i's are sampled at random and also as we can assume without loss of generality $\mathsf{sn}_i \neq \gamma_{i^*}$ for $i \neq i^*$ because in that case the adversary consumes γ_{i^*} and prevents it from being spent. Therefore, $\mathcal{S}$ can compute $\boldsymbol{z}_{i^*}$ in terms of the other $\boldsymbol{z}_i$'s and $\boldsymbol{z}'_i$'s. This directly gives an M-LWE$_{\kappa,q,1}$ solver: Given an M-LWE instance $\boldsymbol{t} = \mathsf{HMC}(0; \boldsymbol{z})$, the M-LWE solver runs $\mathcal{S}$ in the above game with the following modification: the M-LWE solver guesses at the beginning of the game which among all the Q_{KG} queries made by the PKGEN oracle to $\mathcal{G}$ is the query γ_{i^*} corresponding to the above double spending event $\mathcal{E}^{\boldsymbol{v}=0}_{\mathrm{2xspend}}$, and programs the random oracle as $\mathcal{G}(\gamma^*_i) := \boldsymbol{t}$ (i.e. implicitly setting $\boldsymbol{z}_{i^*} := \boldsymbol{z}$) when answering that $\mathcal{G}$ query. All other $\mathcal{G}$ queries are answered with independent $\boldsymbol{z}_i$ as before. With non-negligible probability $\Pr[\mathcal{E}^{\boldsymbol{v}=0}_{\mathrm{2xspend}}]/Q_{KG}$, the M-LWE solver's guess for i^* is correct, and $\mathcal{E}^{\boldsymbol{v}=0}_{\mathrm{2xspend}}$ occurs, so the M-LWE solver computes the M-LWE solution $\boldsymbol{z}$ using (30), contradicting the assumed hardness of M-LWE$_{\kappa,q,1}$.

Remark 1. *The above proof implies that $\mathcal{A}$ must use the honestly-created serial numbers in its output, i.e., $\hat{\mathsf{pk}}_{i,\ell} = \mathsf{HMC}(0; \mathsf{sk}_{i,\ell})$ for short secret keys $\mathsf{sk}_{i,\ell}$'s. This also gives that for a transaction with serial number sn_i's, we have $\mathcal{L}[\mathsf{sn}_i].\mathsf{cn} = \mathsf{cn}_{i,\ell}$ for $i \in [M]$. We use this in the latter cases.*

Case 2 (Unbalanced Amounts): Let $\mathcal{E}_{\mathrm{unbal}}$ be the event that $\mathcal{A}$ wins `Exp:Balance` in the following way: for all $\mathsf{sn}_i \in \mathsf{SN}$ where $0 \leq i \leq M-1$, $\mathsf{sn}_i \in \mathcal{L}$ and $\mathcal{L}[\mathsf{sn}_i].\mathsf{IsCrpt} = 1$. Suppose that the assumptions in the theorem hold. $\mathcal{S}$ runs $\mathcal{A}$ until $\mathcal{E}_{\mathrm{unbal}}$ occurs three times with respect to distinct $\mathcal{H}$ outputs and the same $\mathcal{H}$ inputs. Then, $\mathcal{S}$ computes (22), (24) (25). Since the input coins are assumed to be generated by MINT, the PPT adversary cannot compute any other opening of $\sum_{j=0}^{M-1} v^{(j)}_{\mathrm{in},\ell}$, but the case where $\hat{a}_{\mathrm{in}}$ is the sum of M polynomials with binary CRT slots by Sub-Assumption 5. Hence, all CRT slots of $\hat{a}_{\mathrm{in}}$ are in $[0, M-1]$.

Now, using the fact that all CRT slots of all $\hat{a}_{\mathrm{out},i}$'s are binary, all CRT slots of $\hat{a}_{\mathrm{in}}$ are in $[0, M-1]$ and $c_1, \ldots, c_{r-1}$ are in $[-(M-1), S-1]$, we conclude that (25) in each CRT slot holds *without* mod q by Sub-Assumption 4. Further, by Remark 1, $v^{(i)}_{\mathrm{in},\ell}$'s correspond to the input coins $\mathsf{cn}_{i,\ell} = \mathcal{L}[\mathsf{sn}_i].\mathsf{cn}$ for $i \in [M]$. Denote by $b_{i,j}$ the j-th bit of the i-th input amount for $j \in [r]$ and $i \in [M]$, i.e., $\hat{a}_{\mathrm{in}} = \sum_{i=0}^{M-1}[\![b_{i,0}, \ldots, b_{i,r-1}]\!]$. Using the fact that (25) holds without mod q in each CRT slot, we get for $j \in [r]$

$$\sum_{i=0}^{S-1} b^{(j)}_{\mathrm{out},i} - \sum_{i=0}^{M-1} b_{i,j} + c_j - 2c_{j+1} = 0. \tag{31}$$

By the definition of `Exp:Balance`, the sum of the amounts in $\mathsf{Amt}_{\mathrm{out}}$ (i.e., those corresponding to *uncorrupted* output public keys) cannot exceed the sum of the amounts in all output coins. From here, we look at the following sum where $\mathsf{amt}_{\mathrm{in},i} = \mathcal{L}[\mathsf{sn}_i].\mathsf{amt}$ for $\mathsf{sn}_i = \mathsf{SN}[i]$

$$\sum_{i=0}^{S'-1}\mathsf{Amt}_{\mathrm{out}}[i] - \sum_{i=0}^{M-1}\mathsf{amt}_{\mathrm{in},i} \leq \sum_{i=0}^{S-1}\sum_{j=0}^{r-1} 2^j b^{(j)}_{\mathrm{out},i} - \sum_{i=0}^{M-1}\sum_{j=0}^{r-1} 2^j b_{i,j}$$
$$= \sum_{j=0}^{r-1} 2^j\left(\sum_{i=0}^{S-1} b^{(j)}_{\mathrm{out},i} - \sum_{i=0}^{M-1} b_{i,j}\right) = \sum_{j=0}^{r-1} 2^j\left(2c_{j+1} - c_j\right) = 0,$$

because $c_0 = c_r = 0$. The above gives $\sum_{i=0}^{S'-1}\mathsf{Amt}_{\mathrm{out}}[i] \leq \sum_{i=0}^{M-1}\mathsf{amt}_{\mathrm{in},i}$, yielding a contradiction with the winning assumption of $\mathcal{A}$ in `Exp:Balance`.

Case 3 (forgery): Let $\mathcal{E}_{\mathrm{forge}}$ be the event that $\mathcal{A}$ wins `Exp:Balance` in the following way: there exists i^* with $0 \leq i^* \leq M-1$ such that $\mathsf{sn}_{i^*} \in \mathcal{L}$ and $\mathcal{L}[\mathsf{sn}_{i^*}].\mathsf{IsCrpt} = 0$.

Let Q_P be the number of PKGEN queries $\mathcal{A}$ makes. $\mathcal{S}$ picks $\phi \xleftarrow{\$} [1, Q_P]$ and sets $\mathsf{pk}_\phi = \mathsf{HMC}(1; \boldsymbol{r}_\phi) + \mathcal{G}(\mathsf{sn}_\phi)$ for $\boldsymbol{r}_\phi \xleftarrow{\$} \mathbb{S}_1^{\bar{n}+\kappa}$ and $\mathsf{sn}_\phi \xleftarrow{\$} \{0,1\}^{256}$. Note that pk_ϕ is computationally indistinguishable from any public key generated by **KeyGen** due to the hiding property of HMC (i.e., by M-LWE$_{\kappa,q,1}$ assumption). For other PKGEN queries, $\mathcal{S}$ runs **KeyGen** and returns the public key to $\mathcal{A}$. Now, $\mathcal{S}$ runs $\mathcal{A}$ until

$\mathcal{E}_{\text{forge}}$ happens three times under the same **Spend** input and for the same random oracle input with three different outputs given by $\mathcal{S}$, where

- the index i^* is the same for all $\mathcal{E}_{\text{forge}}$ events, and
- $\mathsf{pk}_\phi \in \mathsf{A}_{\text{in}}$ and it is not corrupted.

From here, $\mathcal{S}$ runs the extractor of the underlying zero-knowledge proofs of tx as given in Appendix E and obtains (23), which is equivalent to

$$y \cdot \left(\sum_{i=0}^{M-2} \alpha'_i \hat{\mathsf{pk}}_{i,\ell} + \hat{\mathsf{pk}}_{M-1,\ell} \right) = \boldsymbol{G}\hat{\boldsymbol{s}} = \mathsf{HMC}(0;\, \hat{\boldsymbol{s}}). \quad (32)$$

By Remark 1 and the assumption that all public keys are generated by PKGEN, $\mathcal{S}$ knows the openings of all $\hat{\mathsf{pk}}_{i,j}$'s. Therefore, $\mathcal{S}$ knows $\mathsf{sk}_{0,\ell}, \ldots, \mathsf{sk}_{M-1,\ell} \in \mathbb{S}_1^{\bar{n}+\kappa}$ such that

$$\hat{\mathsf{pk}}_{i,\ell} = \mathsf{HMC}(0;\, \mathsf{sk}_{i,\ell}), \text{ for } i = 0, \ldots, M-1. \quad (33)$$

Now note that with probability $1/N$, $\hat{\mathsf{pk}}_\phi := \mathsf{pk}_\phi - \mathcal{G}(\mathsf{sn}_\phi)$ is equal to one of the elements in $\{\hat{\mathsf{pk}}_{0,\ell}, \ldots, \hat{\mathsf{pk}}_{M-1,\ell}\}$, say $\hat{\mathsf{pk}}_\phi = \hat{\mathsf{pk}}_{\psi,\ell}$ for $\psi \in [0, M-1]$. Note again that by Remark 1, the adversary cannot use any other serial number than sn_ϕ. Therefore, $\mathcal{S}$ can compute the following

$$y\alpha_\psi \hat{\mathsf{pk}}_{\psi,\ell} = \mathsf{HMC}(0;\, \hat{\boldsymbol{z}}), \quad (34)$$

where $\hat{\boldsymbol{z}} := \hat{\boldsymbol{s}} - y \left(\sum_{i=0, i\neq\psi}^{M-1} \alpha'_i \mathsf{sk}_{i,\ell} \right)$ for $\alpha_{M-1} = 1$. By the construction of $\hat{\mathsf{pk}}_\phi = \hat{\mathsf{pk}}_{\psi,\ell}$, we also have

$$y\alpha'_\psi \hat{\mathsf{pk}}_{\psi,\ell} = \mathsf{HMC}(y\alpha'_\psi;\, y\alpha'_\psi \boldsymbol{r}_\phi). \quad (35)$$

Since y and α'_ψ are invertible, $y\alpha'_\psi$ is non-zero in (35). Hence, (34) and (35) yields a binding collision, i.e., gives a solution to M-SIS. Similar to **Case 1**, bounding $\|\hat{\boldsymbol{z}}\|$ by $\|\hat{\boldsymbol{s}}\|$ for simplicity, $\hat{\boldsymbol{z}}$ gives a solution to M-SIS$_{\bar{n},q,\beta'_{\text{SIS}}}$ for $\beta'_{\text{SIS}} = 2.4 \cdot \phi_{rs} \mathcal{B}_{\text{rs}} \sqrt{d(\bar{n}+\kappa)}$, which yields a contradiction with Sub-Assumption 6. □

ACKNOWLEDGMENTS

We thank Tim Ruffing for helping with the concrete runtimes of Omniring and Monero from [13]. This work was supported in part by Australian Research Council Discovery Grant DP180102199.

REFERENCES

[1] J. Gambetta, "Ibm's roadmap for scaling quantum technology," 2020, https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/, accessed on Dec 2, 2020.

[2] V. Buterin, "Understanding serenity, part i: Abstraction," 2015, https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction/, accessed on Dec 3, 2020.

[3] M. F. Esgin, V. Kuchta, A. Sakzad, R. Steinfeld, Z. Zhang, S. Sun, and S. Chu, "Practical post-quantum few-time verifiable random function with applications to algorand," Cryptology ePrint Archive, Report 2020/1222, 2020, ia.cr/2020/1222 (to appear at FC'21).

[4] M. F. Esgin, O. Ersoy, and Z. Erkin, "Post-quantum adaptor signatures and payment channel networks," in *ESORICS (2)*, ser. LNCS, vol. 12309. Springer, 2020, pp. 378–397.

[5] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu, "MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol," in *ACM Conference on Computer and Communications Security*. ACM, 2019, pp. 567–584, (Full version at ia.cr/2019/1287).

[6] M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu, "Short lattice-based one-out-of-many proofs and applications to ring signatures," in *ACNS*, ser. LNCS. Springer, 2019, pp. 67–88, (Full version at ia.cr/2018/773).

[7] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu, "Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications," in *CRYPTO (1)*, ser. LNCS, vol. 11692. Springer, 2019, pp. 115–146, (Full version at ia.cr/2019/445).

[8] T. Attema, V. Lyubashevsky, and G. Seiler, "Practical product proofs for lattice commitments," in *CRYPTO (2)*, ser. LNCS, vol. 12171. Springer, 2020, pp. 470–499.

[9] M. F. Esgin, N. K. Nguyen, and G. Seiler, "Practical exact proofs from lattices: New techniques to exploit fully-splitting rings," in *ASIACRYPT (2)*, ser. LNCS, vol. 12492. Springer, 2020, pp. 259–288.

[10] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Practical lattice-based zero-knowledge proofs for integer relations," in *CCS*. ACM, 2020, pp. 1051–1070.

[11] S. Noether, "Ring signature confidential transactions for monero," Cryptology ePrint Archive, Report 2015/1098, 2015, ia.cr/2015/1098.

[12] T. H. Yuen, S. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *Financial Cryptography and Data Security*, ser. LNCS, vol. 12059. Springer, 2020, pp. 464–483.

[13] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling private payments without trusted setup," in *CCS*. ACM, 2019, pp. 31–48, (ia.cr/2019/580, 9-Apr-2020 version).

[14] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals–Dilithium: Digital signatures from module lattices," in *CHES*, vol. 2018-1, 2018.

[15] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," in *EUROCRYPT*, ser. LNCS, vol. 7237. Springer, 2012, pp. 465–482.

[16] V. Lyubashevsky, "Lattice signatures without trapdoors," in *EUROCRYPT*. Springer, 2012, pp. 738–755, (Full version).

[17] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Shorter lattice-based zero-knowledge proofs via one-time commitments," Cryptology ePrint Archive, Report 2020/1448, 2020, https://eprint.iacr.org/2020/1448.

[18] V. Lyubashevsky and G. Seiler, "Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs," in *EUROCRYPT (1)*, ser. LNCS, vol. 10820. Springer, 2018, pp. 204–224.

[19] D. Stehlé and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," in *EUROCRYPT*, ser. LNCS, vol. 6632. Springer, 2011, pp. 27–47.

[20] X. Wang, Y. Chen, and X. Ma, "Adding linkability to ring signatures with one-time signatures," in *ISC*, ser. LNCS, vol. 11723. Springer, 2019, pp. 445–464.

[21] M. F. Esgin, "Practice-oriented techniques in lattice-based cryptography," Ph.D. dissertation, Monash University, 5 2020, https://doi.org/10.26180/5eb8f525b3562.

[22] J. Zhang, Y. Yu, S. Fan, Z. Zhang, and K. Yang, "Tweaking the asymmetry of asymmetric-key cryptography on lattices: Kems and signatures of smaller sizes," in *Public Key Cryptography (2)*, ser. LNCS, vol. 12111. Springer, 2020, pp. 37–65.

[23] R. K. Zhao, R. Steinfeld, and A. Sakzad, "FACCT: fast, compact, and constant-time discrete gaussian sampler over integers," *IEEE Trans. Computers*, vol. 69, no. 1, pp. 126–137, 2020.

[24] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert, "More efficient commitments from structured lattice assumptions," in *SCN*, ser. LNCS, vol. 11035. Springer, 2018, pp. 368–385.

[25] Z. Liu, K. Nguyen, G. Yang, H. Wang, and D. S. Wong, "A lattice-based linkable ring signature supporting stealth addresses," in *ESORICS (1)*, ser. LNCS, vol. 11735. Springer, 2019, pp. 726–746.

[26] S. Bai, A. Langlois, T. Lepoint, D. Stehlé, and R. Steinfeld, "Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance," in *ASIACRYPT (1)*, ser. LNCS, vol. 9452. Springer, 2015, pp. 3–24.

TABLE VII
PROOF LENGTH COMPARISON (IN KB) AMONG RINGCT PROPOSALS INCLUDING PRE-QUANTUM ONES.

Anonymity level	$1/N = 1/11$				
#in's $\rightarrow$ #out's	$2 \rightarrow 2$	$20 \rightarrow 2$	$50 \rightarrow 2$	$100 \rightarrow 2$	PQ?[a]
Monero [11]	1.72	9.03	21.22	41.53	No
RingCT 3.0 [12]	1.41	2.16	3.22	4.84	No
Omniring [13]	0.78	0.84	0.91	0.97	No
MatRiCT [5]	110.00	310.00	610.00	1100.00	Yes
MatRiCT$^+$	**47.40**	**59.48**	**60.60**	**63.59**	Yes

[a] 'PQ' refers to being plausibly post-quantum (in ROM).

APPENDIX A
COMPARISON WITH OTHER RINGCT PROPOSALS

In this section, we provide a more detailed comparison between MatRiCT$^+$ and other (pre-quantum) RingCT proposals. Our goal here is to highlight the gap that remains between the post-quantum and pre-quantum proposals. As far as the current state of affairs is concerned, it is well-understood that lattice-based schemes cannot reach the communication efficiency of ECDLP-based proposals. A clear example of this is the ordinary signatures, where the gap between ECDSA/Schnorr signature and Dilithium [14] –a signature finalist in NIST's Post-Quantum Cryptography standardization process relying on the same lattice assumption as MatRiCT$^+$– is about $42\times$ (we refer to this $42\times$ gap as "inherent gap"). Therefore, we do not expect our proposal, MatRiCT$^+$, to reach the communication efficiency levels of RingCT 3.0 [12] or Omniring [13], but we discuss that the gap in the RingCT case is now (with the introduction of MatRiCT$^+$) substantially reduced.

Table VII summarizes the proof length comparison. Here, the focus is on the increasing number of inputs as the anonymity level and the number of output accounts does not vary much in Monero. However, in the last 7 days at the time of writing[12], Monero recorded several hundreds of transactions with $30+$ inputs and close to 200 transactions with $100+$ inputs. Therefore, we believe that it is important to consider increasing number of inputs.

We can first notice that MatRiCT$^+$ produces proofs of similar lengths as Monero when the number of input accounts is around 150. Moreover, we observe that the gap of around 60–$70\times$ between Omniring and MatRiCT$^+$ remains similar in Table VII, which means MatRiCT$^+$ scales similarly for increasing M as the shortest (pre-quantum) RingCT proposal. MatRiCT$^+$ scales even better than RingCT 3.0, where the gap falls even to one-third of the "inherent gap".

In terms of verification efficiency, which is the more important computational aspect since every miner in the blockchain runs verification, we have the following verification times for Monero and Omniring reported from [13] with ring sizes $N = 8, 16, 64$. Monero: $18, 28, 87$ ms, and Omniring: $13, 13, 23$ ms. That is, MatRiCT$^+$ verification is more than $4\times$ faster than both Omniring and Monero in the most typical case of $N \approx 10$. The transaction generation times are similar between Omniring and MatRiCT$^+$ (for $N \leq 64$), while Monero is about $2\times$ faster than MatRiCT$^+$.

[12] https://pooldata.xmrlab.com/, (Accessed 21 April 2021)

APPENDIX B
LATTICE-BASED COMMITMENT SCHEMES

We make use of two lattice-based commitment schemes, namely 'Hashed-Message Commitment' (HMC) (see, e.g., [6], [7]) and 'Unbounded-Message Commitment' (UMC) [24]. The two commitment schemes are very similar in overall computations required, but have different advantages/disadvantages.

In HMC, the key generation function samples two uniformly random matrices $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times(n+\kappa)}$ and $\boldsymbol{B} \xleftarrow{\$} \mathcal{R}_q^{n\times v}$, which become the public commitment keys. To commit to $\boldsymbol{m} \in \mathcal{R}_q^v$, we sample a *short* randomness $\boldsymbol{r}$ according to some distribution (often uniform on a narrow range or discrete Gaussian with small standard deviation) and compute the commitment $C = \boldsymbol{Ar} + \boldsymbol{Bm}$. The advantage of HMC is that the commitment dimension does not increase with the message dimension; however, HMC is binding only when the input message $\boldsymbol{m}$ is of small norm.

In UMC, the key generation function samples random matrices $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times(n+\kappa+v)}$ and $\boldsymbol{B} \xleftarrow{\$} \mathcal{R}_q^{v\times(n+\kappa+v)}$, which become the public commitment keys. Then, to commit to $\boldsymbol{m} \in \mathcal{R}_q^v$, we sample again a *short* randomness $\boldsymbol{r}$ and compute the commitment $\boldsymbol{t} = \boldsymbol{Ar}$ and $\boldsymbol{t}_m = \boldsymbol{Br} + \boldsymbol{m}$. One may see $\boldsymbol{t}$ as the "binding part" and $\boldsymbol{t}_m$ as the "message encoding part". The advantage of UMC is that we can now commit to any message (regardless of its norm) and the message encodings can be independently manipulated. However, the commitment dimension (which is $n + v$) increases with the message dimension v. In MatRiCT$^+$, we exploit the best of both schemes to design an efficient construction overall.

We refer to [21, Section 3.2.2] for a good collective discussion of both schemes and also to, e.g., [6], [7], [24] for more details. Both schemes are well-known to be computationally hiding based on M-LWE and computationally binding based on M-SIS. We also note that the commitment matrices can be equivalently sampled in a more structured fashion with some entries set to be 0 or 1. This saves some computation in practice and we employ this strategy in the implementation. However, in order not to clutter the presentation, we do not present the algorithms this way. The security aspects are not affected by this change (see [21, Section 3.2.2]).

APPENDIX C
MORE ON MATRICT$^+$

A. Adding Recipient Anonymity

We can employ a standard method to add recipient anonymity to MatRiCT$^+$, which can be seen as the lattice analog of the technique used in Monero. In particular, when spending to a user with a long-term public pk (generated exactly as in **KeyGen**), the spender creates a fresh public-secret key pair $(\mathsf{pk}', \mathsf{sk}') \leftarrow$ **KeyGen**() and sets the output public key to be $\mathsf{pk}_o = \mathsf{pk} + \mathsf{pk}'$. Since sk' is sampled independently and pk' is computationally indistinguishable

from a random element in $\mathcal{R}_q^{\bar{n}}$ by M-LWE$_{\kappa,q,1}$, pk_o does not leak any information about pk. Hence, recipient anonymity is achieved. The spender forwards sk' to the recipient along with the output coin opening. This approach has already been formally analyzed in the lattice setting in [25].

B. Adding Auditability

We can use the same idea in [5, Section 6] to extend MatRiCT$^+$ to provide auditability. In particular, we would need to insert a Regev-style trapdoor into the commitment matrix $\boldsymbol{G}'$ as described in [5, Section 6.1] so that the commitment B can be decryptable using the trapdoor (known to some authority). The advantage in our case is that the commitment B now only encodes the user index together with a few extra bits, while a similar commitment in MatRiCT [5] encodes the user index as well as all the output coin bits (i.e., extra $S \cdot r$ bits). Combined with the fact that MatRiCT requires larger parameters, the components in B that we do not want to recover in decryption have a much bigger norm in MatRiCT, which then puts stronger requirements on the parameters for the auditable version. In particular, MatRiCT$^+$ requires only $\hat{q} \approx 2^{34}$ while MatRiCT already has $\hat{q} \approx 2^{53}$ (without even considering auditability requirements). Hence, the existing $\hat{q}$ in MatRiCT$^+$ is already sufficient, but we additionally need to adjust the parameters to make sure that M-LWE$_{\hat{n}-1,\hat{q},1}$ is hard (to make sure that trapdoor remains hidden), which means we just need to increment $\hat{n}$ to 6.

Note that the auditability property can be enforced by making sure that verification checks whether a certain commitment matrix (that has an unknown trapdoor) is used. Therefore, we can either have the auditability optional or have it enforced.

C. More on Implementation and Evaluation of MatRiCT$^+$

As in [8], [9], we sample the fresh randomnesses over $\mathbb{S}_1$ as follows to make the sampling computationally easy. Each coefficient of a randomness polynomial in $\mathbb{S}_1$ is sampled from the binomial distribution on $\{-1, 0, 1\}$, where 0 is sampled with probability $6/16$ and each of ± 1 is sampled with probability $5/16$.

We set $\boldsymbol{A} = [\, \boldsymbol{I}_n \,\|\, * \,]$, $\boldsymbol{B} = [\, \boldsymbol{I}_n \,\|\, * \,]$, $\boldsymbol{G}' = [\, \boldsymbol{I}_{\hat{n}} \,\|\, * \,]$ and $\boldsymbol{G} = [\, \boldsymbol{I}_{\bar{n}} \,\|\, * \,]$, where $*$ denotes entries chosen uniformly at random from its respective domain. Similarly, $\boldsymbol{a}_i = (0^n, \boldsymbol{e}_i, *)$ for $i = 1, 2, 3$, and $\boldsymbol{b}_1 = (0^n, \boldsymbol{e}_1, *)$, where $\boldsymbol{e}_1 = (1, 0, 0)$, $\boldsymbol{e}_2 = (0, 1, 0)$ and $\boldsymbol{e}_3 = (0, 0, 1)$. Moreover, instead of computing $\boldsymbol{a}'_3 = \boldsymbol{a}_3 \cdot [\![0, 1^{r'-1}, 0, 1^{r'-1}]\!]$, we set $\boldsymbol{a}_3$ to have a zero in the two CRT slots indexed by 0 and r'.

MatRiCT$^+$ accompanied by the ring signature in [5] (instead of the simplified version given in this paper) achieves a proof length logarithmic in M, polylogarithmic in N and linear in S. It seems hard to avoid the linear dependence on S since each output coin opening needs to be masked separately as the openings may be transferred to different recipients. Transaction generation and verification runtimes are linear in M, N and S, which is inherent as the computation must depend on all input, output and decoy accounts. A sample of concrete proof lengths and the public key length of MatRiCT$^+$ are provided in Table I, where the bit-length computation of a discrete Gaussian sample is done by calculating its entropy.

Appendix D
Discussion on Serial Numbers

As discussed in Section V, the generation of an account's serial number in MatRiCT$^+$ does not depend on any secret information and anyone can "claim" any serial number. This way, an attacker can try to prevent an honest user, say Alice, from being able to spend her account. For this, the attacker would need to learn Alice's serial number sn *and* record a valid transaction with sn on blockchain *before* Alice's transaction with the serial number sn is recorded on blockchain. First, such an attack does *not* lead to a violation of the security properties in our security model. This would be an attack against *availability* as an honest user is prevented from spending her coins (i.e., a potential "denial-of-spending" attack).

If we want to prevent such an attack surface, then there is indeed an easy solution as described in [20]. In particular, instead of choosing a serial number sn at random, we set sn to be the public key of a one-time signature (OTS). Then, in **Spend**, we ask the spender to sign the hash of the transaction output using OTS. Since only the owner of the OTS secret key can perform this signing, the attackers are prevented from claiming the serial numbers of others. We refer to [20] for a more formal treatment.

In practice, Alice can keep the serial number sn secret until she wants to use it in a blockchain transaction. Therefore, the attacker has a very limited window to launch such a denial-of-spending attack, i.e., the attack needs to be done in the period between the time Alice submits her transaction and when it gets recorded on blockchain. Given this limited time frame, using a short DL-based signature such as ECDSA to serve as OTS above is likely to be a sufficient solution. Here, the attacker would need to break a new DLP instance for each user he wants to target and, in fact, the cost of such a (quantum) attack is likely to be well-above the transaction amount Alice is prevented from spending in most of the typical transactions. Hence, the attack may not even be worthwhile, let alone being feasible.

In general, any OTS scheme (which may just be an ordinary signature) can be used as a tool to circumvent the above attacks. Using a scheme with minimal total size of a public key and a signature would lead to the minimal communication overhead.

Appendix E
Overall Witness Extraction of the Underlying Zero-Knowledge Proof

In this section, we present a description of the overall witness extraction procedure of the *interactive* zero-knowledge proof underlying MatRiCT$^+$. The difference in the interactive protocol is that the prover sends the components $\boldsymbol{w}_{\text{out},i}, \boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, w, v_0, A, E$ in its move to the verifier, who then checks whether they are consistent as in Alg. 7 (rather

than computing them and checking the hash). We use the assumptions listed in Assumption 1.

Assume that the witness extractor E is given 3 accepting protocol transcripts with the same $(\mathsf{IN}, \mathsf{COM}, v_0, v_1, v_2, A, B, E)$, distinct challenges x, x' and x'' and some responses denoted analogous to the challenges (e.g., $\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}''$). As discussed in the 3 special soundness of [7, Theorem 2] and [5, Appendix E], E computes

$$f_j = xb_j + \hat{a}_j, \tag{36}$$
$$f'_j = x'b_j + \hat{a}_j, \tag{37}$$
$$f''_j = x''b_j + \hat{a}_j, \tag{38}$$

for $j = 0, \ldots, N+1$, where $b_j := (f_j - f'_j)(x - x')^{-1}$ and $\hat{a}_j := f_j - xb_j$, using Sub-Assumptions 1 and 7. We aim to show that $b_j \in \{0, 1\} \subset \mathcal{R}$ for all $j = 0, \ldots, N+1$. Again, following the steps of [7, Theorem 2] and [5, Appendix E], we have

$$b_j(b_j - 1) = 0 \text{ in } \mathcal{R}_{\hat{q}}, \tag{39}$$

for $j = 0, \ldots, N+1$. Note that $\|(x - x')b_j\|_\infty = \|f_j - f'_j\|_\infty \leq 12 \cdot \phi_a \mathcal{B}_a$ for $j = 1, \ldots, N+1$. By Sub-Assumption 3 and Corollary 3, we conclude that $b_j \in \{0, 1\}$ for all $j = 1, \ldots, N+1$. We still need to prove $b_0 \in \{0, 1\}$.

Now, by construction, we have

$$x = \sum_{j=0}^{N-1} f_j = \sum_{j=0}^{N-1} (xb_j + \hat{a}_j) = x \sum_{j=0}^{N-1} b_j + \sum_{j=0}^{N-1} \hat{a}_j. \tag{40}$$

Similarly, we have $x' = x' \sum_{j=0}^{N-1} b_j + \sum_{j=0}^{N-1} \hat{a}_j$. Therefore, we get $x - x' = (x - x') \sum_{j=0}^{N-1} b_j$, which implies (by Sub-Assumption 1)

$$\sum_{j=0}^{N-1} b_j = 1, \tag{41}$$

which implies $b_0 = 1 - \sum_{j=0}^{N-1} b_j$. Using this equality together with (39) for $j = 0$, we have

$$\left(1 - \sum_{j=1}^{N-1} b_j\right) \left(\sum_{j=1}^{N-1} b_j\right) = 0 \text{ in } \mathcal{R}_{\hat{q}}. \tag{42}$$

Since the left hand side of the above is just an integer less than N^2, which is less than $\hat{q}/2$ by Sub-Assumption 3, the above equality holds over $\mathbb{Z}$ and thus $\sum_{j=1}^{N-1} b_j \in \{0, 1\}$. Hence, $b_0 \in \{0, 1\}$ by (41). This concludes that $b_j \in \{0, 1\}$ for all $j = 0, \ldots, N+1$.

Combining this fact and (41), there exists a *single* index $\ell \in \{0, \ldots, N-1\}$ such that $b_\ell = 1$, and $b_j = 0$ for all $j \neq \ell$ and $j \in [N]$. From here, it's easy to see using (36) and (37) that

$$f_j - f'_j = \begin{cases} 0 & \text{if } j \neq \ell, \\ x - x' & \text{if } j = \ell. \end{cases} \tag{43}$$

for $j = 0, \ldots, N-1$. With the above, we see that all the summations of the form $\sum_{j=0}^{N-1} (f_j - f'_j) A_j$ (for some term A_j) reduces to $(x - x')A_\ell$, which we will use below.

Next, as in the soundness proof of [7, Theorem 3], the extractor E recovers $\hat{\boldsymbol{s}} := \boldsymbol{z} - \boldsymbol{z}'$ with $\|\hat{\boldsymbol{s}}\|_\infty \leq 12 \cdot \phi_{rs} \mathcal{B}_{\text{rs}}$ and $\|\hat{\boldsymbol{s}}\| \leq 2.4 \cdot \phi_{rs} \mathcal{B}_{\text{rs}} \sqrt{d(\bar{n} + \kappa)}$ such that

$$(x - x')P_\ell = \boldsymbol{G}\hat{\boldsymbol{s}}. \tag{44}$$

We use the above opening to get a contraction with Sub-Assumption 6 in the proof of Theorem 1. Now, using the "weak opening" idea from [8], by Sub-Assumption 5, E further computes $(\hat{\boldsymbol{r}}_0, \hat{a}_{\text{in}}, \hat{c}, \hat{g}_2, \hat{\boldsymbol{\rho}}, \hat{\boldsymbol{r}}_{\text{out},i}, \hat{a}_{\text{out},i})$ for $i \in [S]$ such that

$$\mathsf{cn}_{\text{out},i} = (\boldsymbol{t}^{(i)}, t_1^{(i)}) = (\boldsymbol{B}\hat{\boldsymbol{r}}_{\text{out},i}, \langle \boldsymbol{b}_1, \hat{\boldsymbol{r}}_{\text{out},i} \rangle + \hat{a}_{\text{out},i}), \tag{45}$$
$$\sum_{j=0}^{M-1} \boldsymbol{t}_{\text{in},\ell}^{(j)} = \boldsymbol{B}\hat{\boldsymbol{r}}_0, \qquad \text{[by (43)]} \tag{46}$$
$$\sum_{j=0}^{M-1} v_{\text{in},\ell}^{(j)} = \langle \boldsymbol{b}_1, \hat{\boldsymbol{r}}_0 \rangle + \hat{a}_{\text{in}}, \tag{47}$$
$$\boldsymbol{v} = \boldsymbol{A}\hat{\boldsymbol{\rho}}, \tag{48}$$
$$C' = \langle \boldsymbol{a}'_3, \hat{\boldsymbol{\rho}} \rangle + \hat{c}, \tag{49}$$
$$v_2 = \langle \boldsymbol{a}_2, \hat{\boldsymbol{\rho}} \rangle + \hat{g}_2, \tag{50}$$

and

$$\bar{\boldsymbol{z}}_0 := \boldsymbol{z}_0 - \boldsymbol{z}'_0 = (x - x')\hat{\boldsymbol{r}}_0, \tag{51}$$
$$\bar{\boldsymbol{z}}_1 := \boldsymbol{z}_1 - \boldsymbol{z}'_1 = (x - x')\hat{\boldsymbol{\rho}}, \tag{52}$$
$$\bar{\boldsymbol{z}}_2 := \boldsymbol{z}_2 - \boldsymbol{z}'_2 = \sigma^{-1}(x - x')\hat{\boldsymbol{\rho}}, \tag{53}$$
$$\bar{\boldsymbol{z}}_{\text{out},i} := \boldsymbol{z}_{\text{out},i} - \boldsymbol{z}'_{\text{out},i} = (x - x')\hat{\boldsymbol{r}}_{\text{out},i}, \tag{54}$$

for $i = 0, \ldots, S-1$ and

$$\boldsymbol{z}_1 = \hat{\boldsymbol{y}}_1 + x\hat{\boldsymbol{\rho}}, \tag{55}$$
$$\boldsymbol{z}'_1 = \hat{\boldsymbol{y}}_1 + x'\hat{\boldsymbol{\rho}}, \tag{56}$$
$$\boldsymbol{z}''_1 = \hat{\boldsymbol{y}}_1 + x''\hat{\boldsymbol{\rho}}, \tag{57}$$
$$\boldsymbol{z}_{\text{out},i} = \hat{\boldsymbol{y}}_{\text{out},i} + x\hat{\boldsymbol{r}}_{\text{out},i}, \tag{58}$$
$$\boldsymbol{z}'_{\text{out},i} = \hat{\boldsymbol{y}}_{\text{out},i} + x'\hat{\boldsymbol{r}}_{\text{out},i}, \tag{59}$$
$$\boldsymbol{z}''_{\text{out},i} = \hat{\boldsymbol{y}}_{\text{out},i} + x''\hat{\boldsymbol{r}}_{\text{out},i}, \tag{60}$$

for some $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_{\text{out},i}$'s (not necessarily short). Observe that the above has the same structure with the responses in an honest protocol run, but we just don't have the property that the vectors on the right hand side are short. Thus, by the construction of $h_0, \ldots, h_S$ and using (45), (49) and (50), we end up with an expression similar to (20). Particularly, we get

$$\begin{aligned} &- x^3 \left[\hat{c}(\hat{c} - 1)(\hat{c} + 1)\right] \\ &+ x^2 \left[\sum_{i=0}^{S-1} \alpha_i \hat{a}_{\text{out},i}(\hat{a}_{\text{out},i} - 1) + g_2 - \hat{g}_2\right] + x[*] + [*] = 0, \end{aligned}$$

where $*$ denotes some terms that are not important and g_2 is some term as in (18). The equation above also holds for the other challenges x' and x''. Hence, by Sub-Assumption 1, we get $\hat{c}(\hat{c}-1)(\hat{c}+1) = 0$ and $\sum_{i=0}^{S-1} \alpha_i \hat{a}_{\text{out},i}(\hat{a}_{\text{out},i} - 1) + g_2 - \hat{g}_2 = 0$. Now, it is important to observe that $\hat{a}_{\text{out},i}$'s, $\hat{g}_2$, and g_2 are all *independent* of α_i's as they are part of commitments computed *before* receiving the challenges α_i's (or running the random oracle $\mathcal{H}'$). Hence, except with a negligible probability, the

only way they can sum up to zero is if $\hat{a}_{\text{out},i}(\hat{a}_{\text{out},i}-1)=0$ for all $i=0,\dots,S-1$ (and $g_2-\hat{g}_2=0$).

Overall, we conclude that $\hat{a}_{\text{out},i}$ has binary CRT slots for all $i=0,\dots,S-1$ and that all CRT slots of $\hat{c}$ are in $\{-1,0,1\}$. In fact, by construction, the verifier removed the CRT slots indexed by 0 and r' of $\hat{c}$, so they're equal to 0.

Now, we get to the crucial part that is unique to our work. Let's look at the difference of two responses of Step 15 of Alg. 7. We have

$$0=\langle \boldsymbol{a}'_3,\bar{\boldsymbol{z}}_1\rangle-2\sigma(\langle \boldsymbol{a}'_3,\bar{\boldsymbol{z}}_2\rangle)+\langle \boldsymbol{b}_1,\sum_{i=0}^{S-1}\bar{\boldsymbol{z}}_{\text{out},i}-\bar{\boldsymbol{z}}_0\rangle - y\sum_{i=0}^{S-1}t_1^{(i)}+y\sum_{i=0}^{M-1}v_{\text{in},j}^{(\ell)}-yC'-\bar{u}+y2\sigma(C'), \quad (61)$$

where $\bar{u}=u-u'=y(b_N-b_{N+1})\cdot[\![0^{r'-1},-2,1,0^{r'-1}]\!]$ and $y=x-x'$. Define $c_{r'}:=b_N-b_{N+1}\in\{-1,0,1\}$, which means $y^{-1}\bar{u}=[\![0^{r'-1},-2c_{r'},c_{r'},0^{r'-1}]\!]$. Multiplying (61) by y^{-1} and substituting Equations (45) to (54), the inner product expressions cancel out and we get

$$\sum_{i=0}^{S-1}\hat{a}_{\text{out},i}-\hat{a}_{\text{in}}+\hat{c}-2\sigma(\hat{c})+[\![0^{r'-1},-2c_{r'},c_{r'},0^{r'-1}]\!]=0.$$

By Sub-Assumption 2, all the terms in the above equation has exactly r CRT slots. Denote $\hat{c}=[\![0,c_1,\dots,c_{r'-1},0,c_{r'+1},\dots,c_{r-1}]\!]$. Then, we have $\hat{c}-2\sigma(\hat{c})+[\![0^{r'-1},-2c_{r'},c_{r'},0^{r'-1}]\!]=[\![c_0-2c_1,\dots,c_{r-1}-2c_r]\!]$ for $c_0=c_r=0$ and $c_1,\dots,c_{r-1}\in\{-1,0,1\}$. Hence,

$$\sum_{i=0}^{S-1}\hat{a}_{\text{out},i}-\hat{a}_{\text{in}}+[\![c_0-2c_1,\dots,c_{r-1}-2c_r]\!]=0, \quad (62)$$

where $c_0=c_r=0$ and $c_1,\dots,c_{r-1}\in\{-1,0,1\}$.

In the general case where we don't necessarily have $M=S=2$, the only change to the above arguments is that the CRT slot c_i's of $\hat{c}$ are proven to be in $[-(M-1),S-1]$.

Appendix F
Formal Correctness and Anonymity of RingCT

We use the correctness and anonymity definitions from [5].

Correctness: A RingCT protocol is said to be ϵ-*correct* if the following holds for any $pp\leftarrow$ **Setup**(1^λ), any $M,N,S\in\mathbb{Z}^+$, $(\mathsf{pk}_0,\mathsf{sk}_0),\dots,(\mathsf{pk}_{M-1},\mathsf{sk}_{M-1})\leftarrow$ **KeyGen**(pp) such that **IsSpent**(**SerialGen**$(\mathsf{sk}_i))=0$ for all $i=0,\dots,M-1$, any $\mathsf{amt}_0,\dots,\mathsf{amt}_{M-1},\mathsf{amt}_{\text{out},0},\dots,\mathsf{amt}_{\text{out},S-1}\in\mathsf{V}$ 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,

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

where $\mathsf{cn}_i=$ **Mint**$(\mathsf{amt}_i,\mathsf{ck}_i)$ for some ck_i's in the domain of coin keys, $\mathsf{act}_i\leftarrow$ **AccountGen**$(\mathsf{pk}_i,\mathsf{cn}_i)$ for $i=0,\dots,M-1$, $\mathsf{R}_{\text{in}}=\{\mathsf{act}_0,\dots,\mathsf{act}_{M-1}\}$, $\mathsf{Amt}_{\text{out}}=\{\mathsf{amt}_{\text{out},0},\dots,\mathsf{amt}_{\text{out},S-1}\}$, A_{in} and tx are as in Table IV, and $\mathsf{K}_{\text{in}}=\{(\mathsf{sk}_0,\mathsf{ck}_0,\mathsf{amt}_0),\dots,(\mathsf{sk}_{M-1},\mathsf{ck}_{M-1},\mathsf{amt}_{M-1})\}$. 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*.

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

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

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 IV,
   - $\mathsf{R}^0_{\text{in}},\mathsf{R}^1_{\text{in}}\subset\mathsf{A}_{\text{in}}$ such that both R^0_{in} and R^1_{in} contain exactly one account from each row of A_{in}.
2) $(\mathsf{tx}_i,\mathsf{CK}^i_{\text{out}})\leftarrow$ **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 **Verify**$(\mathsf{tx}_i,\mathbb{B})=\emptyset$ 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 $i=0,1$,
- $b'=b$.

Appendix G
Anonymity Proof of MatRiCT$^+$

We analyze the anonymity of MatRiCT$^+$ in the setting where Alg. 1 is always used as the rejection sampling method. The reason for this is that the analysis with optimized rejection sampling Alg. 2 requires major changes in the security model as a more extended view of the transactions is needed because the coin keys appear in (at most) two transactions (not just one). We leave this more detailed investigation as a future work. The only additional 'hint' given to the adversary in the case of using Alg. 2 is the sign of $\langle(\{\boldsymbol{z}_{\text{out},i}\}_{i\in[S]},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_0,\boldsymbol{z}_b),(\{x\boldsymbol{r}_{\text{out},i}\}_{i\in[S]},x\boldsymbol{\rho},\sigma^{-1}(x)\boldsymbol{\rho},x\sum_{i=0}^{M-1}\boldsymbol{r}_i,x\boldsymbol{r}_b)\rangle$, where $\boldsymbol{\rho},\boldsymbol{r}_b$ are freshly sampled

for each transaction and $\boldsymbol{r}_{\text{out},i}, \boldsymbol{r}_i$ are used in at most two transactions.

Theorem 2. *(Anonymity) Let $\mathcal{A}$ be a PPT adversary, $\text{Adv}_{\mathcal{A}}^{LWE}$ be the advantage of $\mathcal{A}$ over solving M-LWE$_{\kappa,q,1}$, $\text{Adv}_{\mathcal{A}}^{LWE_2}$ be the advantage of $\mathcal{A}$ over solving M-LWE$_{\hat{\kappa},\hat{q},1}$, and $\varepsilon(\phi) \leq 2^{-100}/\mu(\phi)$ for μ defined in Alg. 1 be the statistical distance between the resulting distribution after rejection sampling with parameter ϕ and the corresponding simulated distribution. The advantage of $\mathcal{A}$ against* `Exp:Anonymity` *without shuffling of MatRiCT$^+$ using rejection sampling in Alg. 1 is at most*

$$\text{Adv}_{\mathcal{A}}^{Ano} \leq (M+1)\cdot\text{Adv}_{\mathcal{A}}^{LWE_2}+\text{Adv}_{\mathcal{A}}^{LWE}+\varepsilon(\phi_a)+\varepsilon(\phi_{rs})+\varepsilon(1).$$

Proof of Theorem 2. The proof uses the simulation of the underlying zero-knowledge proofs, where the indistinguishability is either due to an M-LWE assumption or rejection sampling. Let us consider the following games.

$\mathbf{Game}_0$: is identical to `Exp:Anonymity` without shuffling.

$\mathbf{Game}_1$: First, the challenger simulates the responses where the rejection sampling is applied. In Algorithm 5 and 6, it replaces all the polynomials in $\boldsymbol{f}_1$ by random samples of $\mathbb{D}_{\phi_a\mathcal{B}_a}$, all the polynomials in $\boldsymbol{z}_b, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2$ by random samples of $\mathbb{D}_{\mathcal{B}}$, all the polynomials in $\boldsymbol{z}$ by random samples of $\mathbb{D}_{\phi_{rs}\mathcal{B}_{\text{rs}}}$. This game is statistically indistinguishable from the previous game due to rejection sampling.

$$\left|\text{Adv}_{\mathcal{A}}^{\mathbf{Game}_0} - \text{Adv}_{\mathcal{A}}^{\mathbf{Game}_1}\right| \leq \varepsilon(\phi_a) + \varepsilon(\phi_{rs}) + \varepsilon(1).$$

$\mathbf{Game}_2$: In Algorithm 5, the challenger replaces B by a uniformly random element in $\mathcal{R}_{\hat{q}}^{\hat{n}}$. This game is computationally indistinguishable from $\mathbf{Game}_1$ by M-LWE$_{\hat{\kappa},\hat{q},1}$ hardness.

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

$\mathbf{Game}_3$: In Algorithm 4, the challenger replaces $(\boldsymbol{v}, v_1, v_2, C)$ by a uniformly random element in $\mathcal{R}_q^{n+3}$. This game is computationally indistinguishable from the previous game by M-LWE$_{\kappa,q,1}$ hardness.

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

$\mathbf{Game}_4$: The challenger replaces serial numbers by uniformly random elements in $\{0,1\}^{256}$ and the public keys $\mathsf{pk}_{i,\ell}$'s in R_{in} (i.e., the ℓ-th column of A_{in}) by uniformly random elements in $\mathcal{R}_q^{\bar{n}}$ for $i = 0, \ldots, M-1$. This game is computationally indistinguishable from $\mathbf{Game}_3$ by M-LWE$_{\kappa,q,1}$ hardness.

$$\left|\text{Adv}_{\mathcal{A}}^{\mathbf{Game}_3} - \text{Adv}_{\mathcal{A}}^{\mathbf{Game}_4}\right| = M \cdot \text{Adv}_{\mathcal{A}}^{\text{LWE}_2}.$$

Note that $\mathsf{CK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \mathsf{Amt}_{\text{out}}$ and PK_{out} are all independent of R_{in} and K_{in}. So, in $\mathbf{Game}_4$, the output of **Spend** is independent of R_{in} and K_{in}, and thus independent of b. Hence, $\mathcal{A}$ has probability $1/2$ of outputting $b' = b$ in $\mathbf{Game}_4$. □

Appendix H
Proof of Lemma 1

Proof. We first notice that P_2 can be rewritten as the distribution of the random variable $Y_2 := \sum_{j\in[w]} h_j \zeta_i^{k_j}$ over $\mathbb{Z}_q$, with $(\boldsymbol{h} = (h_1, ..., h_w), \boldsymbol{k} = (k_1, ..., k_w))$ sampled from a distribution D_2 as follows: h_j's are identically and independently distributed (iid) uniformly on $\{-1, 1\}$, and $\boldsymbol{k}$ is sampled uniformly at random from the set of all w-tuples from $[r]$ with *distinct* coordinates (i.e. $k_j \neq k_{j'}$ for $j \neq j'$).

As outlined above, our first step (below) is to compute a bound M_1 on a slightly different distribution P_1 of random variable $Y_1 := \sum_{j\in[w]} h_j \zeta_i^{k_j}$ defined similarly to Y_2 except that in the distribution D_1 of $(\boldsymbol{k}, \boldsymbol{h})$, the k_j's are sampled iid from the uniform distribution on $[r]$ (i.e. without the distinct coordinate requirement). As our second step, we note that the support $\text{Supp}(D_2)$ of D_2 is a subset of the support $\text{Supp}(D_1)$ of D_1, and for every $(\boldsymbol{k}, \boldsymbol{h})$ in the support of D_2, we have $D_2(\boldsymbol{k}, \boldsymbol{h}) = \eta \cdot D_1(\boldsymbol{k}, \boldsymbol{h})$, where $\eta := \frac{|\text{Supp}(D_1)|}{|\text{Supp}(D_2)|} = \frac{r^{\tilde{w}}(r-\tilde{w})!}{r!}$. Since Y_1, Y_2 are computed as the same function of $(\boldsymbol{k}, \boldsymbol{h})$, it follows that $P_2(y) \leq \eta P_1(y)$ for all y in the support of P_2 (this can be seen as a simple application of the probability preservation property of Rényi divergence of order ∞, used in different contexts in [26]), and hence $M_2 \leq \eta \cdot M_1$.

To complete our first step and prove the Lemma, it therefore suffices to bound M_1. For this, we follow a Fourier analysis approach. As in the proof of [8, Le. 3.2], we rewrite $P_1 : \mathbb{Z}_q \to [0, 1]$ in terms of its Fourier transform $\hat{P}_1$ over $\mathbb{Z}_q$ (with respect to the orthogonal Fourier basis $\{\chi_j(x) = \exp(-2\pi\imath jx/q)\}_{j\in\mathbb{Z}_q}$, where $\imath := \sqrt{-1}$) to get:

$$P_1(y) = \frac{1}{q} + \frac{1}{q}\sum_{j\in\mathbb{Z}_q^*} \hat{P}_1(j) \cdot \exp(-2\pi\imath jy/q). \tag{63}$$

As the coordinates of $\boldsymbol{h}$ and $\boldsymbol{k}$ are iid, $\hat{P}_1$ is the $\tilde{w}$-fold self-convolution of the distribution μ of each term $h_j\zeta_i^{k_j}$ in Y_1. So from the convolution property of the Fourier transform, we have $\hat{P}_1(j) = \hat{\mu}(j)^{\tilde{w}}$, where $\mu(0) = 0$, and for $s \in \{-1, 1\}$ and $k \in [r]$, $\mu(s\zeta_i^k) = \Pr[h_j = s] \cdot \Pr[k_j = k] = (1/2) \cdot (1/r) = \frac{1}{2r}$. Since $\zeta_i^r = -1$, $v = s\zeta_i^k$ runs through all elements of the group $< \zeta_i >$ of $2r$'th roots of unity in $\mathbb{Z}_q^*$ as (s, k) run through $\{-1, +1\} \times [r]$, so μ is the uniform distribution on $< \zeta_i >$. Computing the Fourier transform $\hat{\mu}$ of μ, we get for each $j \in \mathbb{Z}_q^*$ that

$$\begin{aligned}\hat{\mu}(j) &:= \frac{1}{2r}\sum_{v\in<\zeta_i>} \exp(2\pi\imath jv/q) = \frac{1}{2r}\sum_{k\in[2r]} \exp(2\pi\imath j\zeta_i^k/q)\\ &= \frac{1}{r}\sum_{k\in[r]} \cos(2\pi j\zeta_i^k/q),\end{aligned}$$

where we have used $\zeta_i^r = -1$, $\cos(.)$ is even, and $\sin(.)$ odd.

So we have (this part is similar to [8, Le.3.3]):

$$\begin{aligned}P_1(y) &= \frac{1}{q}\left(1 + \sum_{j\in\mathbb{Z}_q^*} \hat{\mu}(j)^{\tilde{w}} \exp(-2\pi\imath jy/q)\right)\\ &\leq \frac{1}{q}\left(1 + \sum_{j\in\mathbb{Z}_q^*} |\hat{\mu}(j)|^{\tilde{w}}\right) = \frac{1}{q}\left(1 + 2r\sum_{j\in\mathbb{Z}_q^*/<\zeta_i>} |\hat{\mu}(j)|^{\tilde{w}}\right),\end{aligned}$$

where the inequality uses the triangle inequality (taking magnitude) and the equality uses the fact that $\hat{\mu}(j) = \hat{\mu}(j')$ for

j, j' in the same coset of $< \zeta_i >$ in $\mathbb{Z}_q^*$ and that the size of each coset is $2r$. The last bound is M_1, as claimed. $\square$