# Efficient Verifiable Partially-Decryptable Commitments from Lattices and Applications

Muhammed F. Esgin[1,2], Ron Steinfeld[1], and Raymond K. Zhao[1]

[1] Faculty of Information Technology, Monash University, Australia
[2] CSIRO's Data61, Australia
{Muhammed.Esgin,Ron.Steinfeld,Raymond.Zhao}@monash.edu

**Abstract.** We introduce *verifiable partially-decryptable commitments* (VPDC), as a building block for constructing efficient privacy-preserving protocols supporting *auditability* by a trusted party. A VPDC is an extension of a commitment along with an accompanying proof, convincing a verifier that (i) the given commitment is well-formed and (ii) a certain *part* of the committed message can be decrypted using a (secret) trapdoor known to a trusted party.

We first formalize VPDCs and then introduce a general decryption feasibility result that overcomes the challenges in *relaxed* proofs arising in the lattice setting. Our general result can be applied to a wide class of Fiat-Shamir based protocols and may be of independent interest.

Next, we show how to extend the commonly used lattice-based 'Hashed-Message Commitment' (HMC) scheme into a succinct and efficient VPDC. In particular, we devise a novel 'gadget'-based Regev-style (partial) decryption method, compatible with efficient relaxed lattice-based zero-knowledge proofs. We prove the soundness of our VPDC in the setting of adversarial proofs, where a prover tries to create a valid VPDC output that fails in decryption.

To demonstrate the effectiveness of our results, we extend a private blockchain payment protocol, MatRiCT, by Esgin et al. (ACM CCS '19) into a formally auditable construction, which we call MatRiCT-Au, with very low communication and computation overheads over MatRiCT.

**Keywords:** Lattice · Zero Knowledge · Verifiable Partially-Decryptable Commitment · Auditable RingCT · Accountable Ring Signature

## 1 Introduction

Commitment schemes and accompanying zero-knowledge proofs (ZKPs) have become crucial tools used in countless privacy-preserving protocols. For example, they are extensively used in privacy-aware blockchain applications such as Monero and Zcash cryptocurrencies to hide sensitive information such as user identities and transaction amounts. In many such privacy-preserving applications, there is a need for *auditability*, i.e., the ability of a trusted third-party to revoke the privacy or anonymity of the protocol, in order to catch or punish misbehaving entities. For instance, it is well known that the privacy features of cryptocurrencies have been exploited by cyber criminals to hide their illegal financial

activities, and some level of government oversight may be required in future to allow such activities to be traced by law authorities. Many applications where such an auditability feature is needed exist, including group signatures [7], fair exchange [1], key escrow [30] and e-voting. To enable the auditability property of the privacy protocol, we would like the protocol to use a *decryptable* commitment scheme, supporting a trapdoor decryption algorithm that enables the authority with some trapdoor to recover a message from a given commitment.[3] At the same time, to prevent malicious parties from escaping the auditability property, the protocol must support a *verifiable* decryptable commitment, which allows protocol parties to verify that a commitment is decryptable by the authority, while still hiding its contents from all other parties.

A problem similar to constructing verifiable decryptable commitments has been previously studied under the name of *verifiable encryption* [6] in the classical setting of DL-based and factoring-based public-key cryptography. The approach here is to use a public-key encryption scheme as the commitment (with the secret key known to the authority), and attach to it a zero-knowledge proof of plaintext knowledge in order to turn it into a verifiable commitment. This approach was extended to the post-quantum lattice-based setting in [22], instantiating the encryption scheme by a variant of Regev's encryption scheme [28] based on (Ring/Module)-LWE. We note that Regev's encryption scheme can also be viewed as a decryptable *short message* variant of the 'Unbounded-Message Commitment' (UMC) scheme [3] (see Appendix C). Despite allowing Regev-style decryption, UMC also has the practical efficiency drawbacks we discuss below.

The use of verifiable Regev encryption as in [22] can result in very long commitments and communication overheads in typical applications. This is because both the randomness length and commitment length of Regev-encryption commitments have an additive term proportional to the dimension of the message vector. In typical lattice-based ZKPs such as [9, 11, 12, 14], the structure of the protocol requires the prover to send commitments to a large number of messages including masking randomness values as well as auxiliary terms, in addition to the commitment of the 'real' message which needs to be decrypted by the opening authority (e.g., the payment amount, or payer/payee identity in cryptocurrencies). The protocol also requires the prover to send masked variants of the commitment randomness. Both those factors lead to long proofs with Regev encryption commitments. To illustrate, in the MatRiCT cryptocurrency protocol of [14], an aggregated binary proof is used (see [14, Section 1.2]) to significantly reduce the proof length. In this proof, it is necessary to commit to individual bits of integers by separate ring elements so that each bit can be manipulated independently. As a result, the total message dimension in a commitment over the underlying polynomial ring $R_q$ is in the order of several hundreds (the 'real' message is still a few hundreds dimensional). If one were to use a Regev-style encryption for this commitment, the commitment *alone* would cost around 100-

[3] We note here that our notion of a decryptable commitment is different from a *trapdoor* commitment. For a given commitment and a message, the latter allows a trapdoor holder to find a properly distributed opening randomness for the commitment.

200 KB. In comparison, this commitment costs only about 13 KB in MatRiCT thanks to the use of a *compressing* commitment.

To reduce the length of commitments/proofs, an alternative approach (used in [14]) to Regev-encryption commitments is to instead use lattice-based 'Hashed-Message Commitments' (HMC), where message hashing leads to a short commitment dimension independent of the total dimension of the committed messages. In HMC, message hashing is achieved by multiplying the (long) message vector by a *random* 'fat' (i.e., compressing) matrix and one relies on the hardness of (Ring/Module)-SIS to accomplish the binding property. However, in our context of *decryptable* commitments, the lack of a unique decryption for such HMC commitments (due to compression) makes them not directly suitable. Therefore, we study HMC in the *partial* decryption setting where the committed message has two parts: (i) a decryptable message (that contains the 'real' message the authority wants to recover), and (ii) a non-decryptable/auxiliary message (that contains the other auxiliary terms that need not be recovered). This way, we can achieve both of our succinctness and (partial) decryptability goals simultaneously. We note that a straightforward combination of using UMC for the decryptable part of the message and HMC for the non-decryptable message part, although it deals with the auxiliary terms, still suffers from an overhead of at least two commitments plus the large cost of a UMC commitment. In contrast to HMC, the latter UMC commitment dimension over $R_q$ is linear in the message dimension over $R_q$, which is over 100 in the context of MatRiCT discussed above.

An initial attempt to overcome the above-mentioned efficiency issues of UMC-like commitments in constructing VPDCs, was proposed in [14, Section 6.1], where a method of incorporating a lightweight Regev-style decryption trapdoor into an HMC commitment was proposed. However, although a promising direction to combine the best of both HMC and Regev encryption commitments, the work of [14] does not give a full solution to the problem, as it does not address two main technical challenges that we now explain.

Firstly, the decryption algorithm in [14] is only analyzed for *honestly-created* commitments *without* a rigorous framework. The analysis against *adversarially-created* commitments/proofs that pass the verification check, which is an important requirement in the auditability setting of VPDCs, is missing. We recall that for the underlying efficient ZKPs of opening for the HMC scheme we study in this work (see, e.g., [11, 12]), the ZKP soundness only guarantees the existence of a *relaxed* commitment opening $(\boldsymbol{m}, \boldsymbol{r}, y)$ of a commitment $C$, satisfying the relaxed opening relation

$$(yC = \mathsf{Com}_{ck}(y\boldsymbol{m}; \boldsymbol{r})) \wedge (y \in \Delta\mathcal{C}) \wedge (\boldsymbol{m} \in \mathcal{M}), \tag{1}$$

where $y$ is a short non-zero relaxation factor, $\Delta\mathcal{C}$ is the set of challenge differences and $\mathcal{M}$ is a public message space. Observe that the message opening $\boldsymbol{m}$ is proven to be in some set $\mathcal{M}$, which is important for our analysis, and for example, $\mathcal{M} = \{0,1\}^v$ for some $v \geq 1$ for the proof systems in [11, 12, 14]. Also, note that the relaxation factor $y$ is *unknown* to the decryption algorithm as it is part of the prover's secret. Thus, it is not clear how one could enable such a decryption

feature in the setting of relaxed proofs as the decryptor does not even know what to decrypt exactly. The work by Lyubashevsky and Neven [22] addresses this problem in the setting of verifiable Regev encryption. Particularly in [22], it is shown that choosing a random $y$ from the set of possible relaxation factors is in fact a good way to go, and the *expected* running time for their decryption algorithm is shown to be proportional to the number of random oracle queries made by the prover to generate the protocol transcript[4]. However, this result is specific to the Fiat-Shamir (FS) protocol[5] and the Regev-style decryption described in [22].

A second technical challenge in constructing an HMC-based partially decryptable commitment following the approach of [14] is that even if a suitable relaxation factor $y$ is known by the decryption algorithm, decrypting the commitment with the Regev-style trapdoor key does not directly yield the decryptable message, but reveals a noisy inner-product (over the underlying polynomial ring $R_q$) of the message with a known *random* vector $\boldsymbol{a}$, of the form $\langle y\boldsymbol{a}, \boldsymbol{m}\rangle + e$ for some short noise term $e$ (and $y$ the relaxation factor). This leaves the question of how to efficiently recover the message $\boldsymbol{m}$ from this noisy information. The work of [14] addressed this issue only for *small* message spaces (and honestly generated commitments) by performing an exhaustive search over all possible messages, which is very restrictive and computationally expensive. How to make decryption work *efficiently* for *exponentially large* message spaces and guarantee the decryption soundness even against adversarially constructed commitments having such a relaxed opening has since remained unaddressed.

### 1.1 Our Contributions

**Verifiable Partially-Decryptable Commitments.** In this work, we first formalize the notion of a *Verifiable Partially-Decryptable Commitment* (VPDC), which is closely related to proofs of plaintext knowledge and verifiable encryption. In particular, a VPDC extends a commitment scheme $\mathsf{C}$ and a matching Non-Interactive Zero-Knowledge Proof (NIZK) $\Sigma$ of opening for $\mathsf{C}$ by adding a trapdoor key generation algorithm $\mathsf{CAddTd}$ and a matching decryption algorithm $\mathsf{CDec}$ for $\mathsf{C}$. The VPDC ensures that any valid commitment-proof pair $(C, \pi)$ can be (partially) decrypted using the (secret) trapdoor $\mathsf{td}$ output by $\mathsf{CAddTd}$.

The above notion is similar to verifiable encryption except that $C$ is not an encryption, but rather a commitment. The differences, as pointed out in the introduction, are as follows. First, a commitment scheme in general allows for a more *succinct* encoding of a message (i.e., can be compressing unlike an encryption) and is readily compatible with many existing proof systems (see, e.g., [2, 3, 11–14]), hence has a matching NIZK already available. Second, in a VPDC, there are two message spaces: (i) a *decryptable* message space $\mathcal{D}$, whose elements can be committed *and* recovered in decryption, and (ii) an *auxiliary*

[4] We refer to [22] for methods that can be used to restrict an attacker from making a lot of random oracle queries.

[5] We call a public-coin proof made non-interactive via the Fiat-Shamir transformation as a Fiat-Shamir (FS) protocol.

message space $\mathcal{U}$, whose elements can be used to create a commitment, but are not decryptable. As a result, a VPDC eliminates the need for an additional set of requirements due to an encryption scheme, avoids potential compatibility issues and enables partial decryption while still permitting a succinct encoding of the whole message (together with additional auxiliary terms). We therefore believe VPDCs can serve as an important building block in constructing *efficient* cryptographic schemes supporting accountability, such as group signatures, fair exchange protocols, key escrow and e-voting.

**Generalized analysis of decryption feasiblity for *relaxed* ZKPs.** To address the first main technical challenge of handling relaxed ZKPs in decryption of VPDCs, we show how to abstract and generalize the decryption algorithm of [22] that works only for the specific Regev-based (UMC-like) encryption considered therein, to design an efficient decryption algorithm for *any* VPDC satisfying a few natural properties. In particular, the expected number of iterations until the decryption function terminates is about the number of random oracle queries made by the prover in generating the transcript to be decrypted as in [22]. Our general result is applicable to any VPDC whose underlying NIZK is derived via the Fiat-Shamir transform in the random oracle model from a Sigma protocol satisfying a variant of special soundness that is satisfied by all known instantiations of such Sigma protocols.

**A novel gadget-based Regev-style decryption for HMC.** Building on the above general foundations, we construct a VPDC extending one of the most commonly used lattice-based commitment schemes, namely HMC[6]. For example, the HMC scheme is an integral part of one of the most efficient post-quantum ring signatures and set membership proofs in [13], arising from [11, 14], as well as sublinear-sized arithmetic circuit satisfiability proofs in [2].

In particular, to address the second main technical challenge, we introduce an HMC-compatible trapdoor decryption method that works even when the decryptable message opening is proven to be in a set of exponential size (such as $2^{256}$). We analyze this method in the setting of adversarially-created VPDC outputs and provide decryption soundness guarantees. As opposed to the trapdoor decryption of [14], where the trapdoor decryption yields $\langle y\boldsymbol{a}, \boldsymbol{m}\rangle + e$ for a *random* vector $\boldsymbol{a}$, small noise $e$ and relaxation factor $y$ (which is hard to decrypt), our new Regev-style partial trapdoor embeds a *structured* 'gadget' vector $\bar{t}\boldsymbol{g}$ in place of $\boldsymbol{a}$ in the HMC submatrix corresponding to the decryptable message. With this, trapdoor decryption of a commitment yields $\bar{t}y\langle \boldsymbol{g}, \boldsymbol{m}\rangle + e$ for a 'large' integer $\bar{t}$, which is efficiently decryptable by exploiting the structure of the gadget vector $\bar{t}\boldsymbol{g}$ using a rounding procedure similar to standard Regev decryption. The runtime of our new trapdoor decryption is polylogarithmic in the message space size $|\mathcal{D}|$ and we prove that it works correctly even against adversarially-generated commitments and ZKPs, as long as the system modulus $q$ is sufficiently large and the message is proven to be a part of a decryptable message space $\mathcal{D}$.

[6] This distinguishes our VPDC construction from the verifiable encryption scheme of [22], that extends a UMC-type commitment scheme.

**Table 1.** Comparison between MatRiCT [14] and MatRiCT-Au (this work).

| Anonymity level | | 1/10 | | 1/100 | |
|---|---|---|---|---|---|
| # of inputs → # of outputs | | 1 → 2 | 2 → 2 | 1 → 2 | 2 → 2 |
| *Proof Size* | MatRiCT [14] | 93 | 110 | 103 | 120 |
| | **MatRiCT-Au** | **96** | **113** | **106** | **123** |
| *Spend / Verify Runtimes* | MatRiCT [14] | 242 / 20 | 375 / 23 | 360 / 31 | 610 / 40 |
| | **MatRiCT-Au** | **233 / 21** | **414 / 25** | **402 / 33** | **654 / 42** |
| *Parameters* | MatRiCT [14] | PK Size: 4.36 KB | | Moduli: $< 2^{53.0}$ | |
| | **MatRiCT-Au** | PK Size: 4.36 KB | | Moduli: $< 2^{55.3}$ | |

Our lightweight Regev-style 'partial trapdoor' also avoids the heavyweight machinery of 'full' lattice trapdoors a-la [24], and still supports SIS-style HMC commitment, compatible with efficient ZKP techniques used in [11, 14]. Using the 'full' trapdoors in [24] in our commitments (with ternary coordinate trapdoor vectors) requires SIS matrices with $n$ rows and $m \geq n \log q$ columns over the underlying ring, while the 'partial trapdoor' commitments we use, $m = 2n$ columns are sufficient (still with ternary coordinate trapdoor vectors). We save a significant factor $\approx \log q$ in both public parameter length and the length of masked messages in the ZKP protocol, for the same security level.

**MatRiCT-Au: Auditable RingCT based on standard lattice assumptions.** As an application of our compact lattice-based VPDC, we show how it can enable an extension of the lattice-based RingCT-like private cryptocurrency protocol MatRiCT [14] easily and efficiently into an auditable variant we call MatRiCT-Au, where an auditor with access to a (secret) trapdoor can revoke the anonymity of certain users (e.g., in case of misbehaviour). The auditability feature can be optional (i.e., each user individually decides whether and by whom she wants to be audited) or enforced by a simple public check. Our construction allows *adversarially-generated* transactions to be audited, whereas, in [14], the discussion about auditability is incomplete, as the decryption method given there may fail in the adversarial transaction setting, potentially allowing adversaries to avoid auditability. Furthermore, the proposal in [14] requires an exhaustive-search-based approach while we can very efficiently run **Audit** function over a message space of size $> 2^{128}$. To analyze auditability formally in confidential transactions, we also extend the formal model for RingCT-like protocols in [14] to add the *auditability* property and prove formally that MatRiCT-Au is auditable. We compute concrete parameters for MatRiCT-Au and present implementation results[7]. Our evaluation demonstrates the practicality of MatRiCT-Au, and in particular there are very little communication and computation overheads introduced over the original MatRiCT protocol [14] as shown in Table 1 (see Table 5 in Appendix E for more run-time results).

We believe that our new techniques will find further applications in the settings where accountable anonymity is desired. Particularly, our extension of

[7] The source code of our MatRiCT-Au implementation is available at `https://gitlab.com/raykzhao/matrict_au`.

HMC into a VPDC with soundness against adversarially-created outputs extends the group (or accountable ring) signature in [14] so as to enable *efficient* anonymity revocation (i.e., opening of a group signature) against *cheating* signers. Interesting research directions from here would be, for example, to design efficient post-quantum e-voting, auction and anonymous credential schemes by exploiting the accountable anonymity provided by our VPDC.

### 1.2 Our Results and Techniques

**A novel gadget-based Regev-style decryption for HMC.** Suppose that we work over a cyclotomic ring $R_q = \mathbb{Z}_q[X]/(X^d+1)$, and have a *binary* secret vector $\boldsymbol{b} \in \{0,1\}^v \subset R_q^v$ that forms the decryptable message to be recovered in decryption. As explained above, in a typical application protocol, we commit to this message $\boldsymbol{b}$ together with a non-decryptable message $\boldsymbol{u}$ as $C = \mathsf{Com}_{ck}(\boldsymbol{b}, \boldsymbol{u}; \boldsymbol{r})$ under some commitment randomness $\boldsymbol{r}$. The application protocol also proves knowledge of a *relaxed* opening of $C$ (i.e., knowledge of $(y, \boldsymbol{b}', \boldsymbol{u}', \boldsymbol{r}')$ such that $yC = \mathsf{Com}_{ck}(y\boldsymbol{b}', \boldsymbol{u}'; \boldsymbol{r}')$ and $\boldsymbol{b}' \in \{0,1\}^v$). For simplicity, let us consider the case $y = 1$. After dealing with this case, we will discuss how we lift the restriction of $y = 1$ using our generalized decryption analysis results from Sec. 4.

The HMC commitment we use has the form $C = \mathsf{Com}_{ck}(\boldsymbol{b}, \boldsymbol{u}; \boldsymbol{r}) = \boldsymbol{A}\boldsymbol{r} + \boldsymbol{B}\boldsymbol{b} + \boldsymbol{C}\boldsymbol{u}$ and we recover the decrypted message as an element of $R_t$ for some $t \geq 1$. To allow trapdoor decryption of $\boldsymbol{b}$, but not $\boldsymbol{r}$ and $\boldsymbol{u}$, our trapdoor key generation algorithm embeds a Regev-style '*gadget trapdoor*' into the last row $\boldsymbol{t}_B^\top$ of matrix $\boldsymbol{B}$ and a Regev-style '*error trapdoor*' into the last row $\boldsymbol{t}_A^\top$ (resp. $\boldsymbol{t}_C^\top$) of matrix $\boldsymbol{A}$ (resp. $\boldsymbol{C}$). That is, for the 'gadget trapdoor' matrix, we have $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}' \\ \boldsymbol{t}_B^\top \end{pmatrix}$ with 'gadget trapdoor' row $\boldsymbol{t}_B^\top = \boldsymbol{s'}^\top \boldsymbol{B}' + \boldsymbol{e}_B^\top + \bar{t}\boldsymbol{g}^\top$, where $\bar{t} = \lfloor q/t \rfloor$, $\boldsymbol{e}_B$ is a short error, $\boldsymbol{s}'$ is a random secret, and $\boldsymbol{g}^\top$ is a 'gadget' vector with coordinates of the form $(2^i X^j)_{i<\tau, j<d}$ where $2^\tau \leq t$. While for the 'error trapdoor' matrices, we have $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}' \\ \boldsymbol{t}_A^\top \end{pmatrix}$ and $\boldsymbol{C} = \begin{pmatrix} \boldsymbol{C}' \\ \boldsymbol{t}_C^\top \end{pmatrix}$ with 'error trapdoor' rows $\boldsymbol{t}_A^\top = \boldsymbol{s'}^\top \boldsymbol{A}' + \boldsymbol{e}_A^\top$ and $\boldsymbol{t}_C^\top = \boldsymbol{s'}^\top \boldsymbol{C}' + \boldsymbol{e}_C^\top$, where $\boldsymbol{e}_A, \boldsymbol{e}_C$ are short errors. Let $\boldsymbol{s}^\top = (-\boldsymbol{s'}^\top, 1)$ be the trapdoor. We remark that in the prior work [14], the matrix $\boldsymbol{B}$ was a random SIS matrix with no decryption trapdoor, which led to an inefficient exhaustive search decryption over the message space.

Now, it is easy to observe that $C' := \langle \boldsymbol{s}, C \rangle = e + \langle \bar{t}\boldsymbol{g}, \boldsymbol{b} \rangle$, where $e := (\langle \boldsymbol{e}_A, \boldsymbol{r} \rangle + \langle \boldsymbol{e}_B, \boldsymbol{b} \rangle + \langle \boldsymbol{e}_C, \boldsymbol{u} \rangle)$ is a small error. Thanks to the structure of the gadget vector $\bar{t}\boldsymbol{g}^\top$, the integer coefficients of $\langle \bar{t}\boldsymbol{g}, \boldsymbol{b} \rangle$ are multiples of the large integer $\bar{t}$ and encode the bits of the decryptable message $\boldsymbol{b}$ in their binary representation. Thus, $\boldsymbol{b}$ can be recovered from $C'$ in the decryption algorithm by rounding out the small error term $e$ to a multiple of $\bar{t}$ and performing binary decomposition, whereas the non-decryptable message/randomness $\boldsymbol{u}, \boldsymbol{r}$ only contribute to the error term $e$.

To apply our gadget-based Regev-style decryption for HMC to adverserially-generated commitments with a relaxed proof of opening, we apply the general

result of Theorem 1. To apply the latter theorem, we give a generalized decryption algorithm for our Regev-style HMC trapdoor and analyse (in Theorem 2 in Sec. 5.5) its correctness and soundness against (i) 'false rejection' decryption errors (where the algorithm fails to recover a decryptable message opening, even though the latter exists), as well as (ii) 'false acceptance' decryption errors (where the algorithm recovers a different decryptable message than the one in the valid opening), respectively. For (i), to recover the decryptable message $\boldsymbol{b}$ even for adverserial commitments $C$ with a non-trivial relaxed opening $yC = \mathsf{Com}_{ck}(y\boldsymbol{b}, \boldsymbol{u}; \boldsymbol{r})$ with some short relaxation factor $y$, our decryption algorithm recovers $y\langle \boldsymbol{g}, \boldsymbol{b}\rangle \bmod t$ after rounding $\langle \boldsymbol{s}, yC\rangle$ to a multiple of $\bar{t}$, and we rely on invertibility of relaxation factors $y \bmod t$ to recover $\boldsymbol{b}$. For (ii), we show that a mildly larger choice of modulus $q$ than needed for (i) guarantees that incorrect (non-unique) decryptable messages are never returned by our decryption algorithm, even with adverserial commitments/proofs and relaxation factors $y$.

We remark that the high-level structure of our HMC gadget-based Regev-style decryption trapdoor is similar to the full LWE inversion trapdoor of [24], but there are several important technical differences due to our HMC setting that are crucial to our scheme's efficiency and security. First, our use of the gadget during decryption is in some sense 'dual' to its use in [24]: in the LWE inversion problem considered in [24], the LWE secret $\boldsymbol{s}$ is assumed to be *uniformly random* $\bmod q$ (rather than 'short'), so that trapdoor decryption yields $\boldsymbol{c} = \boldsymbol{G}\cdot\boldsymbol{s}+\boldsymbol{e'}$ for a gadget matrix $\boldsymbol{G}$ and short error vector $\boldsymbol{e'}$. Here, to efficiently recover the 'large' coordinate secret $\boldsymbol{s}$ from $\boldsymbol{c}$, the gadget matrix $\boldsymbol{G}$ is constructed to have $\log q$ powers of 2 (up to $q/2$) along each of its *columns* so that the mapping $\boldsymbol{s} \mapsto \boldsymbol{G}\cdot\boldsymbol{s}$ effectively performs *bit decomposition* of the coordinates of $\boldsymbol{s}$. This approach *expands* the dimension of $\boldsymbol{s}$ by a factor $\log q$ to allow recovery of each bit of each coordinate of $\boldsymbol{s}$ from the corresponding row of $\boldsymbol{Gs}$. Whereas in our 'dual' HMC decryption algorithm, the decryptable message $\boldsymbol{s}$ is binary (and hence 'short'), so that when our trapdoor decryption similarly yields $\boldsymbol{c} = \boldsymbol{G}\cdot\boldsymbol{s} + \boldsymbol{e'}$, we can choose the gadget matrix $\boldsymbol{G} = \boldsymbol{g}^\top$ to have powers of 2 along its *row* so that the mapping $\boldsymbol{s} \mapsto \boldsymbol{G}\cdot\boldsymbol{s}$ performs *binary reconstruction* of integers whose bits are the coordinates of $\boldsymbol{s}$. Our approach *compresses* the dimension of $\boldsymbol{s}$ to a single element over the underlying ring, and minimises the dimension of the underlying matrices/commitments. Hence, our algorithm can also be viewed as a more efficient inversion trapdoor for LWE in 'dual' *knapsack* form ($\boldsymbol{c} = \boldsymbol{Bb}$ for 'short' $\boldsymbol{b}$ and 'fat' $\boldsymbol{B}$) rather than the more usual 'primal' form ($\boldsymbol{c} = \boldsymbol{As} + \boldsymbol{e}$ for 'short' $\boldsymbol{e}$ and 'tall' $\boldsymbol{A}$) addressed in [24]. A second difference from [24] is our use of *error trapdoors* for the HMC submatrices corresponding to non-decryptable message/randomness. And thirdly, as outlined above, our decryption algorithm analysis handles the adverserial commitment case with relaxed opening proofs, whereas [24] only analyses decryption for honestly created LWE samples.

**MatRiCT-Au application.** To show the usefulness of our novel decryption method in practice, we apply it in the setting of MatRiCT [14]. In MatRiCT, a commitment $B$ encodes (i) an index in binary form that identifies the real user creating the transaction, and (ii) the bits of the transaction amount. Therefore, we can apply our novel decryption method to decrypt this commitment. Overall,

in addition to revoking the anonymity, we can enable an auditor to recover the hidden transaction amount. This is similar in spirit to traceable range proofs [19] (though the techniques are completely different).

Recently, a newer version of MatRiCT was published in [13]. Our techniques apply also to this newer version, called MatRiCT$^+$, and the overhead of extending MatRiCT$^+$ to support auditability is just an increase of about 20% in proof size. We discuss further details in Appendix G.

**Organization of the paper.** Section 2 covers preliminaries. We introduce the formal definitions of a VPDC in Section 3. Our generalized analysis of decryption runtime for relaxed ZKPs is introduced in Section 4. Then, in Section 5, we provide, along with the ordinary HMC scheme, the details of our new lattice-based VPDC, its decryption algorithm, and its adversarial soundness and runtime analyses. We discuss how VPDC can be used to construct MatRiCT-Au in Section 6 and, due to limited space, provide the full details relating to MatRiCT-Au in the appendices. Particularly, our extended formal model for RingCT-like protocols and the full description of MatRiCT-Au (including parameter setting and implementation details) are deferred to Appendices D and E, respectively. The security properties of MatRiCT-Au are discussed in Appendix F.

## 2 Preliminaries

For an odd modulus $q$, the ring of integers modulo $q$, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, is represented by the range $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$. To denote column vectors and matrices, we use bold-face lower-case letters such as $\boldsymbol{x}$ and bold-face capital letters such as $\boldsymbol{V}$, respectively (hence, $\boldsymbol{x}^\top$ denotes a row vector). $(\boldsymbol{x}, \boldsymbol{y})$ is used to denote concatenation of the two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ to form a single longer vector. For a vector $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$, we define the following norms $\|\boldsymbol{x}\| = \sqrt{\sum_{i=0}^{n-1} x_i^2}$, $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$ and $\|\boldsymbol{x}\|_1 = \sum_{i=0}^{n-1} |x_i|$. When considering a norm of a polynomial $f$, we define the same norms on the coefficient vector of $f$. For a vector $\boldsymbol{f} = (f_0, \ldots, f_{s-1})$ of polynomials, we further define $\|\boldsymbol{f}\| = \sqrt{\sum_{i=0}^{s-1} \|f_i\|^2}$, $\|\boldsymbol{f}\|_1 = \sum_{i=0}^{s-1} \|f_i\|_1$, $\|\boldsymbol{f}\|_\infty = \max_i \|f_i\|_\infty$. The Hamming weight of the (concatenated) coefficient vector of $\boldsymbol{f}$ is denoted by $\mathsf{HW}(\boldsymbol{f})$. $\mathcal{U}(S)$ denotes uniform distribution on a set $S$.

We use capital letters such as $C$ to denote commitments, and write $\mathcal{S}^{d \cdot k}$ when a total of $kd$ coefficients are sampled from a set $\mathcal{S}$ in order to generate $k$ polynomials in $R = \mathbb{Z}[X]/(X^d + 1)$ of a power-of-2 degree $d$. $\mathbb{S}_{\mathcal{B}}$ denotes the set of polynomials in $R$, where each coefficient has an absolute value bounded by $\mathcal{B} \in \mathbb{Z}^+$.

### 2.1 Security Assumptions

In our applications, we use a commitment scheme whose security relies on the following well-known lattice problems.

**Definition 1 (M-SIS$_{n,m,q,\beta_{\mathrm{SIS}}}$).** *Given $\boldsymbol{A} \leftarrow R_q^{n\times m}$ sampled uniformly at random, the Module-SIS (M-SIS) problem asks to find a* short $\boldsymbol{x} \in R_q^m$ *such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ over $R_q$ and $0 < \|\boldsymbol{x}\| \leq \beta_{\mathrm{SIS}}$.*

**Definition 2 (M-LWE$_{n,m,q,\mathcal{B}}$).** *The Module-LWE (M-LWE) problem asks to distinguish between the following two cases: (i) $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ for $\boldsymbol{A} \leftarrow R_q^{m\times n}$, a secret vector $\boldsymbol{s} \leftarrow \mathbb{S}_{\mathcal{B}}^n$ and an error vector $\boldsymbol{e} \leftarrow \mathbb{S}_{\mathcal{B}}^m$, and (ii) $(\boldsymbol{A}, \boldsymbol{t})$ for $\boldsymbol{A} \leftarrow R_q^{m\times n}$ and $\boldsymbol{t} \leftarrow R_q^m$.*

It is known that the secret $\boldsymbol{s}$ can equivalently be sampled from $\mathcal{U}(R_q^n)$.

### 2.2 Zero-Knowledge Proofs

A Relaxed NIZK $\Sigma = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ for relation $\mathsf{R}_\sigma$ and its relaxed counterpart $\mathsf{R}'_\sigma$ with $\mathsf{R}_\sigma \subseteq \mathsf{R}'_\sigma$ (parameterized by a common reference string $\sigma$) and their corresponding languages $L_\sigma = \{u : \exists r \text{ s.t. } (u, r) \in \mathsf{R}_\sigma\}$ and $L'_\sigma = \{u : \exists r \text{ s.t. } (u, r) \in \mathsf{R}'_\sigma\}$ respectively, consists of the following algorithms (here, $u$ denotes a language member and $r$ denotes a witness):

$\sigma \leftarrow \mathsf{K}(1^\lambda)$ : is the PPT common reference string generation algorithm of $\Sigma$ that outputs a common reference string $\sigma$.

$\pi \leftarrow \mathsf{P}^{\mathcal{H}}(\sigma, u, r)$ : is the PPT prover algorithm of $\Sigma$ that, given a common reference string $\sigma$, access to a random oracle $\mathcal{H}$ and a language member $u$ and a witness $r$ with $(u, r) \in \mathsf{R}_\sigma$, outputs a proof $\pi$.

$0/1 \leftarrow \mathsf{V}^{\mathcal{H}}(\sigma, u, \pi)$ : is the PPT verification algorithm of $\Sigma$ that, given a common reference string $\sigma$, access to a random oracle $\mathcal{H}$ and a language member $u$ and proof $\pi$, outputs 0 (invalid) or 1 (valid).

We remark that our lattice-based constructions regard the commitment key as part of the CRS $\sigma$ (a similar issue arises in both DL-based Pedersen and lattice-based commitments). We refer to Appendix A.1 for the standard definitions of completeness, soundness and zero-knowledge for NIZK proofs.

Our VDPC construction is based on a NIZK obtained using the Fiat-Shamir (FS) transform [15] applied to an interactive Zero-Knowledge Sigma protocol $\Sigma_{\mathsf{I}} = (\mathsf{K}_{\mathsf{I}}, \mathsf{P}_{\mathsf{I}}, \mathsf{V}_{\mathsf{I}})$ for relations $\mathsf{R}_\sigma, \mathsf{R}'_\sigma$ (parameterised by a common reference string $\sigma$) with a challenge space $\mathcal{C}$ and public-private inputs $(u, r)$ with same notations for relations as above. We refer to Appendix A.1 for the standard definitions of completeness, special soundness and honest-verifier zero-knowledge for Sigma protocols. The FS heuristic transforms $\Sigma_{\mathsf{I}}$ into a NIZK using a random oracle $\mathcal{H}$, by letting the prove algorithm compute the verifier's challenge from the common reference string $\sigma$, public input $u$, and commitment message $w$, setting $x = \mathcal{H}(\sigma, u, w)$.

### 2.3 Commitment Schemes

A commitment scheme $\mathsf{C} = (\mathsf{CKeygen}, \mathsf{Commit}, \mathsf{COpen})$ consists of three algorithms:

$pp = (ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}(1^\lambda)$ : is a PPT key generation algorithm returning $pp$ containing a commitment key $ck$ and descriptions of message space $\mathcal{M}$ and randomness space $\mathcal{R}$. Note $pp$ is an implicit input to the remaining algorithms.

$(C, \mathsf{o}) \leftarrow \mathsf{Commit}(\mathsf{m})$ : is a PPT commitment algorithm which for message $\mathsf{m} \in \mathcal{M}$, outputs a commitment $C$ to $\mathsf{m}$ together with an opening $\mathsf{o}$.

$0/1 \leftarrow \mathsf{COpen}(C, \mathsf{o})$ : is a deterministic poly-time opening algorithm that given commitment $C$ and opening $\mathsf{o}$, checks whether $\mathsf{o}$ is a valid opening of $C$.

An opening $\mathsf{o}$ of a commitment is a tuple containing a message $\mathsf{m}$, randomness $r$, and possibly also relaxation factors used by the opening algorithm (e.g., the relaxation factor $y$ used in the lattice-based HMC commmitment in Sec. 5.1). We write $\mathsf{m}(\mathsf{o})$ to denote the message part of opening $\mathsf{o}$. We refer to Appendix A.2 for standard definitions of correctness, hiding and binding properties of commitment schemes.

## 3 VPDC: Verifiable Partially-Decryptable Commitments

A VPDC is an extension of two building blocks: (1) a (non-decryptable) commitment scheme $\mathsf{C}$, and (2) a NIZK relaxed proof of opening protocol $\Sigma$ for $\mathsf{C}$. The VPDC adds a new trapdoor key generation algorithm $\mathsf{CAddTd}$ to embed a hidden partial decryption trapdoor $\mathsf{td}$ in the commitment key of $\mathsf{C}$, such that with this trapdoor, efficient partial decryption of commitments accompanied by a valid relaxed proof of opening is possible, using the VPDC's partial decryption algorithm $\mathsf{CDec}$. In particular, for VPDC, we view the commitment scheme's message space $\mathcal{M}$ as the product of two sets $\mathcal{D}$ and $\mathcal{U}$, where $\mathcal{D}$ is the *decryptable* message space and $\mathcal{U}$ is the *auxiliary* message space. For a commitment opening $\mathsf{o}$, we let $\mu(\mathsf{o})$ denote the decryptable message part of $\mathsf{o}$.

Formally, a Verifiable Partially-Decryptable Commitment scheme $\mathsf{VPDC} = (\mathsf{C}, \Sigma, \mathsf{CAddTd}, \mathsf{CDec})$ consists of a (non-decryptable) commitment scheme $\mathsf{C} = (\mathsf{CKeygen}, \mathsf{Commit}, \mathsf{COpen})$ with message space $\mathcal{M} = \mathcal{D} \times \mathcal{U}$ (the decryptable message space $\mathcal{D}$ and auxiliary message space $\mathcal{U}$ respectively), and a *matching* NIZK relaxed proof of opening protocol $\Sigma = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ for $\mathsf{C}$, a trapdoor key generation algorithm $\mathsf{CAddTd}$ and a partial decryption algorithm $\mathsf{CDec}$.

We say that the underlying NIZK $\Sigma$ is a *matching* NIZK relaxed proof of opening for $\mathsf{C}$ if:

- On input $1^\lambda$, the CRS generation algorithm $\mathsf{K}$ returns a CRS of the form $\sigma = (pp, \sigma')$, where $pp = (ck, \mathcal{M}, \mathcal{R})$ is $\mathsf{C}$'s public parameters $pp \leftarrow \mathsf{CKeygen}(1^\lambda)$. (i.e. $\Sigma$ has $pp$ in its CRS).
- $\Sigma$ satisfies the completeness, soundness and zero-knowledge properties in Def. 6 with respect to the following commitment opening relations $\mathsf{R}_{\mathsf{C},pp} \subseteq \mathsf{R}'_{\mathsf{C},pp}$ (parameterised by the commitment key $pp$ from the CRS):

$$\mathsf{R}_{\mathsf{C},pp} = \{(C, \mathsf{o}) : \exists(\mathsf{m}, r) \in (\mathcal{M} \times \mathcal{R}) \text{ with } (C, \mathsf{o}) = \mathsf{Commit}(m; r)\}$$

and

$$\mathsf{R}'_{\mathsf{C},pp} \subseteq \mathsf{R}^{\mathsf{COpen}}_{\mathsf{C},pp} := \{(C, \mathsf{o}) : \mathsf{COpen}(C, \mathsf{o}) = 1\}\,.$$

In addition to the algorithms $(\mathsf{CKeygen}, \mathsf{Commit}, \mathsf{COpen})$ of $\mathsf{C}$ and the algorithms $(\mathsf{K}, \mathsf{P}, \mathsf{V})$ of $\Sigma$, VPDC adds two new algorithms to enable decryptability, with the following syntax:

$(ck^{\mathsf{td}}, \mathsf{td}) \leftarrow \mathsf{CAddTd}(ck, \mathcal{D}, \mathcal{U})$ : a PPT algorithm that on input a commitment key $ck$ and a description of the decryptable and auxiliary message spaces $\mathcal{D}$ and $\mathcal{U}$ such that $\mathcal{M} = \mathcal{D} \times \mathcal{U}$, outputs a 'trapdoored' commitment key $ck^{\mathsf{td}}$ and a partial decryption trapdoor $\mathsf{td}$.

$\mu' \leftarrow \mathsf{CDec}_{\mathsf{td}}(C, \pi)$ : is a probabilistic algorithm that on input a commitment $C$ with a corresponding proof $\pi$ and a trapdoor $\mathsf{td}$, outputs a message $\mu' \in \mathcal{D}$.

We now list several additional properties for a VPDC, all of which are enjoyed by our construction:

**Succinctness:** The bit length of the commitment should depend only polylogarithmically on the bit length of the auxiliary message.[8]

**Additive Homomorphism:** The commitment message and randomness spaces are subsets of modules with operations $(+, \cdot)$ over some underlying scalar ring $\mathfrak{R}$, the commitment space is a subset of a module with operations $(\oplus, \otimes)$ over $\mathfrak{R}$, and there exists a set $S \subseteq \mathfrak{R}$ of scalars, such that for all messages $\mathsf{m}_1, \mathsf{m}_2 \in \mathcal{M}$, randomness $r_1, r_2 \in \mathcal{R}$ and scalar $\alpha \in S$, we have $C = \alpha \otimes C_1 \oplus C_2$ for $(C, \cdot) := \mathsf{Commit}(\alpha \cdot \mathsf{m}_1 + \mathsf{m}_2; \alpha \cdot r_1 + r_2)$, $(C_1, \cdot) := \mathsf{Commit}(\mathsf{m}_1; r_1)$ and $(C_2, \cdot) := \mathsf{Commit}(\mathsf{m}_2; r_2)$.

**Small Integer Decryptable Message Space:** The decryptable message space $\mathcal{D} \subset \mathfrak{R}^v$ is of the form $\mathcal{D} := Z_{\mathcal{B}}^v$, where $Z_{\mathcal{B}} \subseteq \mathbb{Z}$ is a set of *integers* of small maximum absolute value $\mathcal{B} = \lambda^{o(1)}$, and $v$ is the decryptable message dimension over the underlying scalar ring $\mathfrak{R}$.

The succinctness property is essential for the efficient application of our VPDC in ZKPs. For our concrete VPDC construction, $\mathcal{U}$ is a much bigger set than $\mathcal{D}$. Thus, one can commit to auxiliary terms together with the target message to be decrypted under a single succinct commitment to save significant communication thanks to the commitment's compression feature (which is not available in encryption-based commitments). Here, we stress that *partial* decryption (as opposed to *full* decryption) is an important feature, not a drawback. If we required full decryption, then we would not be able to achieve succinctness. Similarly, the additively homomorphic property is needed to support efficient (e.g., 'Schnorr-like' [29]) ZKPs that rely on this property. Note that such a homomorphism is needed also for the non-decryptable message parts. This precludes a simple VPDC solution that would commit by hashing the auxiliary message part with a non-homomorphic collision-resistant hash function. The 'Small Integer Decryptable Message Space' property is required to efficiently support certain classes of ZK proofs needed in applications, such as the binary proofs, range proofs and 1-out-of-$N$ proofs in [11, 14, 17]. Here, the fact that the message coordinates are integers, rather than general ring elements, allows for independent manipulation of the committed decryptable message coordinates

[8] Note that succinctness cannot be achieved for the *decryptable* message.

(e.g., for computing an integer vector inner-product or independent evaluation of a quadratic function on all message coordinates) as needed in the verification of such ZK proofs. Their smallness bound (size $\mathcal{B} = \lambda^{o(1)}$) allows the length of such proofs to be kept short.

It is important to note that our VPDC model allows all of the following three properties *together* within the same environment:

- one can commit to any message in $\mathcal{M}$, where decryption is (computationally) infeasible. Such commitments are simply ordinary commitments and in this case, the commitment key $ck$ should be used.
- one can commit to any message $\mu$ in $\mathcal{D}$ together with an auxiliary message in $\mathcal{U}$, where recovery of $\mu$ is possible using $\mathsf{td}$. In this case, the commitment key $ck^{\mathsf{td}}$ should be used.
- one can commit to any message in $\mathcal{M}$, where decryption is not necessarily needed. Here, both commitment keys $ck$ or $ck^{\mathsf{td}}$ can be used and commitments created this way are easily compatible with the rest of the protocol.

We require that $\mathsf{C}$ satisfies the standard correctness, hiding and binding properties of commitment schemes in Def. 8 in Appendix A.2. Furthermore, we recall that NIZK proof $\Sigma$ is required to be a matching NIZK for $\mathsf{C}$ (see above), and so satisfies the completeness, (relaxed) soundness and zero-knowledge properties in Def. 6 in Appendix A.1 for relations $(\mathsf{R}_{\mathsf{C},pp}, \mathsf{R}'_{\mathsf{C},pp})$ defined above.

In addition, as a partially-decryptable extension of a given commitment scheme $\mathsf{C}$ and matching ZK proof $\Sigma$, we would like the VPDC's trapdoor key generation algorithm for $\mathsf{C}$ to preserve the functionality and security properties of $\mathsf{C}$ and $\Sigma$. Accordingly, we say that $\mathsf{C}$ (resp. $\Sigma$) satisfies the *VPDC trapdoor key variants* of correctness, hiding, and binding properties for $\mathsf{C}$ from Def. 8 (respectively, the trapdoor key variants of completeness, (relaxed) soundness and zero-knowledge for $\Sigma$ from Def. 6) if the properties are still satisfied when the commitment key generation calls $pp = (ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}$ in $\mathsf{C}$ (resp. its call in $\mathsf{K}$ of $\Sigma$) are followed by the trapdoor commitment key generation calls $(ck^{\mathsf{td}}, \mathsf{td}) \leftarrow \mathsf{CAddTd}(ck)$ and $pp' = (ck^{\mathsf{td}}, \mathcal{M}, \mathcal{R})$ replaces $pp$. For ease of reference, we define from hereon $(ck^{\mathsf{td}}, \mathsf{td}, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygenTd}(1^\lambda)$ as the function that runs $(ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}(1^\lambda)$ and $(ck^{\mathsf{td}}, \mathsf{td}) \leftarrow \mathsf{CAddTd}(ck)$, and returns $(ck^{\mathsf{td}}, \mathsf{td}, \mathcal{M}, \mathcal{R})$. The following commitment Key Indistinguishability property for a VPDC suffices for this purpose (see Proposition 1).

**Key Indistinguishability.** A VPDC scheme is said to satisfy *key indistinguishability* if any PPT adversary $\mathcal{A}$ wins the following game with probability $1/2 + \mathrm{negl}(\lambda)$:

1. $pp_0 = (ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}(1^\lambda)$,
2. $pp_1 \leftarrow (ck^{\mathsf{td}}, \mathcal{M}, \mathcal{R})$, where $(ck^{\mathsf{td}}, \mathsf{td}) \leftarrow \mathsf{CAddTd}(ck)$.
3. $b \xleftarrow{\$} \{0, 1\}$,
4. $b' \leftarrow \mathcal{A}(pp_b)$.
5. $\mathcal{A}$ wins the game if $b' = b$.

The following proposition is immediate from the fact that the trapdoor key $\mathsf{td}$ does not appear in the view of the adversary in the security games defining

the trapdoor key variants of the $\mathsf{C}$ and $\Sigma$ properties. Therefore, by key indistinguishability, any attack against the VPDC trapdoor key variant properties of $\mathsf{C}$ (resp. $\Sigma$) would imply a corresponding attack contradicting the assumed (non trapdoor key variant) property of $\mathsf{C}$ (resp. $\Sigma$).

**Proposition 1.** *If a VPDC scheme* $\mathsf{VPDC} = (\mathsf{C}, \Sigma, \mathsf{CAddTd}, \mathsf{CDec})$ *satisfies key indistinguishability, then* $\mathsf{C}$ *(resp.* $\Sigma$*) satisfies the* $\mathsf{VPDC}$ *trapdoor key variants of correctness, hiding, and binding properties for* $\mathsf{C}$ *from Def. 8 (respectively, the* $\mathsf{VPDC}$ *trapdoor key variants of completeness, (relaxed) soundness and zero-knowledge for* $\Sigma$ *from Def. 6).*

In some applications, it is desirable to strengthen the binding requirement for the VDPC so it holds even against attackers that are given the partial decryption trapdoor key $\mathsf{td}$ (e.g. in our blockchain application as we do not want auditors to create fake proofs). We call this requirement *trapdoor-binding*.

**Trapdoor-Binding.** A VPDC is *(computationally) trapdoor-binding* if, for $(pp, \mathsf{td}) \leftarrow \mathsf{CKeygenTd}(1^\lambda)$, the following probability (over the randomness of PPT $\mathcal{A}$ and $\mathsf{CKeygen}$) is negligible

$$\Pr[(C, \mathsf{o}, \mathsf{o}') \leftarrow \mathcal{A}(pp, \mathsf{td}) : \mathsf{m}(\mathsf{o}) \neq \mathsf{m}'(\mathsf{o}) \wedge \mathsf{COpen}(C, \mathsf{o}) = \mathsf{COpen}(C, \mathsf{o}') = 1].$$

We capture the decryptability requirements for VPDC by the *Decryption Soundness* and *Decryption Feasibility* properties defined as follows.

**Decryption Soundness.** A VPDC scheme is said to satisfy *Decryption Soundness* if any PPT adversary wins the following `Exp:Soundness` game with $\mathrm{negl}(\lambda)$ probability.

1. $P := (ck^{\mathsf{td}}, \mathsf{td}, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygenTd}(1^\lambda)$
2. $(C, \pi) \leftarrow \mathcal{A}(P)$,
3. $b \leftarrow \mathsf{V}_{ck^{\mathsf{td}}}(C, \pi)$,
4. $\mu' \leftarrow \mathsf{CDec}_{\mathsf{td}}(C, \pi)$.

$\mathcal{A}$ wins the game if $b = 1$ and one of the following conditions holds

(i) There exists no opening $\mathsf{o}$ such that $\mathsf{COpen}(C, \mathsf{o}) = 1$, or

(ii) There exists an opening $\mathsf{o}$ such that $\mathsf{COpen}(C, \mathsf{o}) = 1$ and $\mu(\mathsf{o}) \neq \mu'$.

**Decryption Feasibility.** A VPDC scheme is said to satisfy *Decryption Feasibility* if, for any $\alpha \geq 1$ and any PPT adversary $\mathcal{A}$, if $b = 1$ in Step 3 of game `Exp:Soundness` above, the running time of $\mathsf{CDec}_{\mathsf{td}}(C, \pi)$ in Step 4 of game `Exp:Soundness` is at most $\alpha \cdot T_{\mathcal{A}} \cdot \mathrm{poly}(\lambda)$, except with probability $\leq \frac{1}{\alpha} + \mathrm{poly}\left(\frac{T_{\mathcal{A}}}{2^\lambda}\right)$, where $T_{\mathcal{A}}$ is the runtime of $\mathcal{A}$.

*Remark 1 (Decryption Soundness).* The decryption soundness property captures the informal requirement that it should be infeasible for an attacker to output a maliciously-created commitment and proof $(C, \pi)$ that passes the $\mathsf{V}$ verification check, but where $C$ cannot be decrypted into the correct decryptable message $\mu$ using the trapdoor decryption algorithm $\mathsf{CDec}_{\mathsf{td}}$. The latter may occur either because of the non-existence of a decryptable message opening (case i), or because of the existence multiple decryptable message openings that may cause a 'false accept' decryption error (case ii).

*Remark 2 (Decryption Feasibility).* The decryption feasibility requirement captures the property that the decryption algorithm does not run for 'too long'. Here, a too long decryption time corresponds to exceeding the attacker runtime by a super-polynomial factor. Jumping ahead to our construction, similarly to [22], this attacker time corresponds to the number of queries made by the attacker to a certain random oracle. Such a runtime as given in our decryption feasibility definition (arising from our results generalizing those of [22]) is currently the best one can achieve for relaxed proofs, where the relaxation factor is unknown to the decryptor.

There are examples of VPDC-like constructions in the literature satisfying some but not all of our desired properties. For example, the proofs of plaintext knowledge in general are an example, where the commitment is an encryption scheme and $\mathcal{U} = \emptyset$. For a lattice-based construction, one may see the discussions in [22, Section 3.3] and [9]. The "extractable" commitment scheme in [14] is another (weaker) example, where the decryption soundness holds only against *honest* provers and decryption runtime is linear in $|\mathcal{D}|$. The main motivation for our VPDC notion is that we want the additional properties of succinctness (i.e., $\mathsf{C}$ should be compressing, which is not possible for encryption) and decryption feasibility and soundness against *cheating* provers. These properties are achieved by our concrete construction $\mathsf{VPDC}_{\mathsf{HMC}}$ (Section 5).

## 4 Generalized Decryption Feasibility for Relaxed Proofs

In this section, we study the decryption algorithms for relaxed NIZK proofs and show a general result on the Partial-Decryption Feasibility of any VPDC in which the underlying NIZK $\Sigma$ is derived from a suitable interactive Sigma protocol $\Sigma_{\mathsf{I}}$ using the Fiat-Shamir (FS) transform. Our result generalizes previous results of [22]. The discussion is kept abstract in this section to preserve the generality of our results. Our concrete lattice-based instantiation of decryption algorithms is given in the next section.

The questions we focus on are as follows. If one is given a valid transcript $\mathsf{tr} = (C, w, x, z)$ for $\Sigma$ and a trapdoor $\mathsf{td}$ that enables recovering a message from a *well-formed* commitment of the form $\bar{x}C$ for an *unknown* relaxed opening factor $\bar{x}$ (also known as a relaxation factor) and a known commitment $C$, how should one precisely design the overall decryption algorithm? Moreover, what is the *expected* number of iterations until the decryption algorithm terminates?

To answer these questions, we prove the general result in Theorem 1 below for the generic decryption algorithm given in Algorithm 1. This approach first allows us to put our decryption methodology into a general framework. Then, we identify the connections between the components of Algorithm 1 that must be satisfied so that the decryption runs in polynomial time. As the result can be applied to *any* suitable functions $F, \mathsf{Rec}, V'$ in Algorithm 1, we can use this result to analyze the run-time of different decryption methods. Besides the Partial-Decryption Feasibility, there is, of course, also the Partial-Decryption Soundness

aspect that depends on the concrete instantiation of $\mathsf{CDec}$, which will be analyzed in the next section.

Our Partial-Decryption Feasibility result applies to VPDCs in which the underlying NIZK $\Sigma$ is derived via the FS transform from an interactive Sigma protocol with the following mild variant of the special soundness property, that we call *existential special-soundness*. This property relaxes the standard PPT efficiency requirement for extractor $\mathcal{E}$, but requires that the extracted witness contains a component (relaxation factor $\bar{x} = x - x'$ in Schnorr-like protocols) depending on only the two input transcript challenges via a poly-time computable function $F$ (the latter syntactic requirement is used in our decryption compatibility definition below). Therefore, the existential special-soundness is directly implied by the standard special-soundness property for a large class of known Schnorr-like Sigma protocols, in which $\mathcal{E}$ is efficient, and $F(x, x') = x - x'$.

**Definition 3 (Existential Special-Soundness).** *We say that a Sigma protocol $\Sigma_{\mathsf{I}} = (\mathsf{K}_{\mathsf{I}}, \mathsf{P}_{\mathsf{I}}, \mathsf{V}_{\mathsf{I}})$ for relations $\mathsf{R}_{\mathsf{C},pp}, \mathsf{R}'_{\mathsf{C},pp}$ (parameterised by a common reference string pp) with a challenge space $\mathcal{C}$ and public-private inputs $(C, \mathsf{o})$ (see Def. 7), satisfies* existential special-soundness *if the following holds.*

- ***Existential special-soundness:*** *There exists an* extractor $\mathcal{E}$ *and a deterministic poly-time algorithm $F$ such that, given $(pp, \sigma') \leftarrow \mathsf{K}_{\mathsf{I}}(1^\lambda)$ and two accepting protocol transcripts $\mathsf{tr} = (C, w, x, z)$ and $\mathsf{tr}' = (C, w, x', z')$ with $x \neq x'$, computes an extracted witness of the form $\bar{\mathsf{o}} = (\bar{x}, \bar{\mathsf{o}}')$ where $\bar{x} = F(x, x')$, satisfying $(C, \bar{\mathsf{o}}) \in \mathsf{R}'_{\mathsf{C},pp}$ with probability $1 - \mathrm{negl}(\lambda)$ over the choice of $\sigma$.*

The following definition captures the properties of a decryption algorithm $\mathsf{CDec}$ that are sufficient to ensure it terminates in feasible time, if the underlying Sigma protocol $\Sigma_{\mathsf{I}}$ satisfies existential special-soundness.

**Definition 4 (Compatible $\mathsf{CDec}$).** *Let $\mathsf{VPDC} = (\mathsf{C}, \Sigma, \mathsf{CAddTd}, \mathsf{CDec})$ with $\Sigma$ a matching NIZK relaxed proof of opening for $\mathsf{C}$, and $\Sigma$ is obtained from a Sigma protocol $\Sigma_{\mathsf{I}}$ using the Fiat-Shamir transform.*

*We say that $\mathsf{CDec}$ is* compatible *with $\Sigma$ if it satisfies the following properties:*

$\mathbf{P}_1$ : $\mathsf{CDec}$ *has a structure as in Algorithm 1, where* $\mathsf{Rec}$ *is a PPT algorithm.*

$\mathbf{P}_2$ : *On input* $\mathsf{td}$ *(generated by running $(ck^{\mathsf{td}}, \mathsf{td}, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygenTd}(1^\lambda)$) and any* $\mathsf{tr}$*, if, for some loop iteration, Step 3 of* $\mathsf{CDec}$ *computes a "good" $\bar{x}$ (such that there exists an opening of the form $\bar{\mathsf{o}} = (\bar{x}, \bar{\mathsf{o}}')$ satisfying $(C, \bar{\mathsf{o}}) \in \mathsf{R}'_{\mathsf{C},pp}$), then* $\mathsf{CDec}$ *terminates in this loop iteration, i.e.* $\mathsf{Rec}$ *recovers a message $m'$ deemed "valid" by $V'(m') = 1$.*

The function $\mathsf{Rec}$ in Alg. 1 is a procedure that recovers the message from a well-formed commitment and will be instantiated depending on our decryption method. One may imagine it being similar to the decryption of Regev encryption scheme. However, the $\mathsf{Rec}$ function always returns a message $m'$ that may simply be useless. Therefore, there is an additional check $V'$ to make sure that the given message is "valid" (where "valid" is protocol-dependent).

Theorem 1 below shows that the only task required to use our results is to design a compatible decryption algorithm as in Def. 4 (as existential special-soundness is implied by special-soundness). In essence, this task itself reduces to

**Algorithm 1** $\mathsf{CDec}_{\mathsf{td}}(\mathsf{tr})$

**INPUT:** $\mathsf{tr} = (C, \pi = (w, x, z))$ $\Sigma_{\mathsf{I}}$ protocol transcript; $\mathsf{td}$ trapdoor
**OUTPUT:** $(m', x')$ such that $V'(m') = 1$ for some validity check $V'$

**loop**
  $x' \xleftarrow{\$} \mathcal{C}$ ▷ Choose a random challenge
  $\bar{x} = F(x, x')$ ▷ $F(x, x') = x - x'$ for 2-sound FS proofs
  $m' = \mathsf{Rec}(\bar{x}, \mathsf{tr}, \mathsf{td})$ ▷ Tries to decrypt a well-formed commitment
  **if** $V'(m') = 1$ **then** ▷ Check if the recovered message is "valid"
    **return** $(m', x')$
  **end if**
**end loop**

making sure that the message recovery algorithm $\mathsf{Rec}$ returns a "valid" message from any given "good" $\bar{x}$ for the relaxed relation $\mathsf{R}'_{\mathsf{C},pp}$. In the next section, we will show how our VPDC allows the recovery of the *same* message used to create the proof transcript.

To make it easier to read Theorem 1, let us interpret it in the case of 'Schnorr-like' FS proofs that work as follows. For a homomorphic commitment $\mathsf{Com}$ (we use additive notation as it is the case in the lattice setting), let $C = \mathsf{Com}(r)$ be an input commitment whose opening the prover wants to prove knowledge of. The prover computes a 'masking' commitment $w = \mathsf{Com}(\rho)$ for some masking value $\rho$. Then, she computes a challenge $x \leftarrow \mathcal{H}(pp, C, w)$, followed by a response $z = \rho + x \cdot r$, where $r$ is the prover's witness. The verification $V$ in this case checks $w + x \cdot C \stackrel{?}{=} \mathsf{Com}(z)$. It is easy to see from here that this proof has the '2-soundness' property, i.e., a *knowledge extractor* can extract a (relaxed) opening of $C$ given two 'rewinded' accepting transcripts with distinct challenges. In particular, given accepting $(C, w, x, z)$ and $(C, w, x', z')$ with $x \neq x'$, we have $\bar{x}C = \mathsf{Com}(\bar{z})$ for $\bar{x} := x - x'$ and $\bar{z} := z - z'$. Therefore, the concrete functions in this case are $F(x, x') = x - x'$ and $(C, (\bar{x}, \bar{z})) \in \mathsf{R}'_{\mathsf{C},pp}$ iff $\bar{x}C = \mathsf{Com}(\bar{z})$, i.e., $\mathsf{R}'_{\mathsf{C},pp}$ in Theorem 1 corresponds to the *relaxed* commitment opening relation $\mathsf{COpen}(C, (\bar{x}, \bar{m}, \bar{r}))$ where $\bar{z} = (\bar{m}, \bar{r})$. Here, $(\bar{x}, \bar{z})$ serves as an *extracted* witness/opening for $C$. It is easy to see that the existential special-soundness property follows from the special-soundness of the 'Schnorr-like' protocol. Although in the setting of the Schnorr proof of knowledge of discrete-log [29], one may further recover an *exact* opening of $u$ by computing $\bar{z}/\bar{x}$, this approach does not work in the lattice variants [20, 21] as $\bar{z}/\bar{x}$ must be *short* (relative to the system modulus $q$), which cannot be guaranteed unless some costly measures are implemented[9].

**Theorem 1.** *Let* $\mathsf{VPDC} = (\mathsf{C}, \Sigma, \mathsf{CAddTd}, \mathsf{CDec})$ *with* $\Sigma$ *a matching NIZK relaxed proof of opening for* $\mathsf{C}$*, and* $\Sigma$ *is obtained from a Sigma protocol* $\Sigma_{\mathsf{I}}$ *using the Fiat-Shamir transform with random oracle* $\mathcal{H} : \{0,1\}^* \to \mathcal{C}$*.*

*If* $\Sigma_{\mathsf{I}}$ *satisfies Existential Special Soundness (Def. 3),* $\mathsf{CDec}$ *is compatible with* $\Sigma$ *(Def. 4) and* $|\mathcal{C}| \geq 2^{\lambda}$*, then* $\mathsf{VPDC}$ *satisfies Decryption Feasibility.*

[9] For example, *exact* lattice proofs (see, e.g., [9]) require around 40-50 KB in comparison to 2-3 KB for *relaxed* lattice proofs.

*Concretely, let $\hat{H}$ and $\hat{D}$ be the random coins of $\mathcal{H}$ and* CDec*, respectively, and $T$ be the number of loop iterations in the execution of* CDec *in Step 6 of game* ***Exp:Soundness*** *when $b = 1$. Then, for any $\mathcal{A}$ that makes at most $q_H - 1$ queries to $\mathcal{H}$ and any positive $\alpha$,*

$$\Pr_{\hat{H},\hat{D}}\left[T \geq \alpha \cdot q_H\right] \leq \frac{1}{\alpha} + 2 \cdot \sqrt{\frac{q_H}{\alpha \cdot |\mathcal{C}|}} + \frac{q_H}{|\mathcal{C}|}. \tag{2}$$

*Proof (Theorem 1).* The proof follows essentially the same blueprint as in the proof of [22, Lemma 3.2], but we show precisely where the properties in the theorem statement are needed.

Let $pp$ be some public parameters. For a given $\mathsf{tr} = (C, w, x, z)$ of $\Sigma_{\mathsf{I}}$, define the set of "good" challenges $\mathcal{G}_{\mathsf{tr}}$ as follows

$$\mathcal{G}_{\mathsf{tr}} = \{\, x' \in \mathcal{C} \;:\; \exists z' \;:\; \mathsf{V}_{\mathsf{I}}(C, w, x', z') = 1 \,\}. \tag{3}$$

Here, $\mathsf{V}_{\mathsf{I}}$ denotes the verification algorithm of $\Sigma_{\mathsf{I}}$. Let $G$ be the event that $\mathcal{A}$ produces $\mathsf{tr}$ with $|\mathcal{G}_{\mathsf{tr}}| > f$ for $f = \left\lceil \sqrt{\frac{|\mathcal{C}|}{\alpha q_H}} \right\rceil$.

**Claim:** For any valid $\mathsf{tr} = (C, w, x, z)$, if CDec chooses $x'$ with $x' \in \mathcal{G}_{\mathsf{tr}} \setminus \{x\}$, then CDec terminates.

The claim follows from the following facts. If the assumption of the claim holds, then there exist $(C, w, x, z)$ (as the input) and $(C, w, x', z')$ such that $\mathsf{V}_{\mathsf{I}}(C, w, x, z) = \mathsf{V}_{\mathsf{I}}(C, w, x', z') = 1$ by the definition of $\mathcal{G}_{\mathsf{tr}}$. Then, by the existential special soundness of $\Sigma_{\mathsf{I}}$, there exists $(\bar{x}, \bar{\mathsf{o}}')$ such that $\bar{x} = F(x, x')$ and $(C, (\bar{x}, \mathsf{o}')) \in \mathsf{R}'_{\mathsf{C},pp}$. Now, by the property $\mathbf{P}_2$ of CDec, the claim follows.

As a result, the probability that CDec terminates in one iteration is at least $\frac{|\mathcal{G}_{\mathsf{tr}}|-1}{|\mathcal{C}|}$. Therefore, we have

$$\mathrm{Exp}_{\hat{D}}\left[\, T \mid \mathcal{A}^{\mathcal{H}} \text{ outputs } \mathsf{tr} \,\right] \leq \frac{|\mathcal{C}|}{|\mathcal{G}_{\mathsf{tr}}| - 1}, \quad \text{and also} \tag{4}$$

$$\mathrm{Exp}_{\hat{D}}\left[\, T \mid \mathcal{A}^{\mathcal{H}} \text{ outputs } \mathsf{tr} \;\wedge\; G \,\right] \leq \frac{|\mathcal{C}|}{f}. \tag{5}$$

We say that "$\mathcal{A}^{\mathcal{H}}$ outputs $\mathsf{tr}_i$" if $\mathcal{A}^{\mathcal{H}}$ outputs $\mathsf{tr} = (C, w, x, z)$ such that the output of $\mathcal{A}$'s $i$-th random oracle query is $x$. As in [22], without loss of generality, we consider an adversary $\mathcal{A}$ that (1) makes $q_H$ random oracle queries, (2) uses one of the random oracle outputs in his output transcript and (3) only makes random oracle queries for transcripts $\mathsf{tr}_i$ with $|\mathcal{G}_{\mathsf{tr}_i}| > f$ as we are conditioning on $G$. Then, similar to [22], we have the following

$$\mathrm{Exp}_{\hat{H},\hat{D}}\left[T \mid G\right] = \sum_{i=1}^{q_H} \Pr_{\hat{H}}\left[\mathcal{A}^{\mathcal{H}} \text{ outputs } \mathsf{tr}_i \mid G\right] \mathrm{Exp}_{\hat{D}}\left[T \mid \mathcal{A}^{\mathcal{H}} \text{ outputs } \mathsf{tr}_i \;\wedge\; G\right]. \tag{6}$$

For each random oracle query made by $\mathcal{A}$ for a transcript $\mathsf{tr}_i$, the probability (over $\hat{H}$) that $\mathcal{A}$ outputs $\mathsf{tr}_i$ is at most the probability that the random oracle query output is in $\mathcal{G}_{\mathsf{tr}_i}$, as otherwise there exists no response $z$ such that $\mathsf{V}_{\mathsf{I}}(C, w, x, z) =$

1. Therefore, each $\mathsf{tr}_i$ can be output with probability at most $\frac{|\mathcal{G}_{\mathsf{tr}_i}|}{|\mathcal{C}|}$. Then, using this fact and (4), we get

$$\mathrm{Exp}_{\hat{H},\hat{D}}[\,T \mid G\,] \leq \sum_{i=1}^{q_H} \frac{|\mathcal{G}_{\mathsf{tr}_i}|}{|\mathcal{C}|} \frac{|\mathcal{C}|}{|\mathcal{G}_{\mathsf{tr}_i}|-1} \leq q_H \cdot \max_{i=1,\ldots,q_H} \left(\frac{|\mathcal{G}_{\mathsf{tr}_i}|}{|\mathcal{G}_{\mathsf{tr}_i}|-1}\right) \leq \frac{q_H(f+1)}{f}.$$

For any random oracle query, the probability that $\mathcal{A}$ outputs a transcript with $|\mathcal{G}_{\mathsf{tr}}| \leq f$ is at most $f/|\mathcal{C}|$. Therefore, we have

$$\Pr_{\hat{H},\hat{D}}[\,\neg G\,] \leq \frac{f \cdot q_H}{|\mathcal{C}|}. \tag{7}$$

Using now Markov's inequality and (7), we get

$$\begin{aligned}\Pr_{\hat{H},\hat{D}}[T \geq \alpha q_H] &= \Pr_{\hat{H},\hat{D}}[T \geq \alpha q_H \mid G\,] \Pr_{\hat{H},\hat{D}}[\,G\,] + \Pr_{\hat{H},\hat{D}}[T \geq \alpha q_H \mid \neg G\,] \Pr_{\hat{H},\hat{D}}[\,\neg G\,] \\ &\leq \frac{\mathrm{Exp}_{\hat{H},\hat{D}}[\,T \mid G\,]}{\alpha \cdot q_H} + \Pr_{\hat{H},\hat{D}}[\,\neg G\,] \leq \frac{f+1}{\alpha \cdot f} + \frac{f \cdot q_H}{|\mathcal{C}|} = \frac{1}{\alpha} \cdot \left(1 + \frac{1}{f}\right) + \frac{f \cdot q_H}{|\mathcal{C}|}.\end{aligned}$$

Plugging in the value of $f = \left\lceil \sqrt{\frac{|\mathcal{C}|}{\alpha q_H}} \right\rceil$ proves the result. □

# 5 HMC-based VPDC from Lattices

## 5.1 Instantiation of (ordinary) HMC

We start by describing the (ordinary) Hashed-Message Commitment (HMC) scheme $\mathsf{C}$ underlying our lattice-based VPDC. Let $n, m, q$ be positive integers with $m > n$. If we want to commit to a $v_1$-dimensional 'real' message over $R_q$ for $v_1 \geq 1$ together with a $v_2$-dimensional auxiliary message for $v_2 \geq 0$, then HMC is instantiated as follows.

- $\mathsf{CKeygen}(1^\lambda)$ : Sample $\boldsymbol{A} \leftarrow R_q^{n \times m}$, $\boldsymbol{B} \leftarrow R_q^{n \times v_1}$ and $\boldsymbol{C} \leftarrow R_q^{n \times v_2}$. Output $ck = \boldsymbol{G} = [\,\boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C}\,] \in R_q^{n \times (m+v_1+v_2)}$, message space $\mathcal{M} = \mathcal{D} \times \mathcal{U}$ with $\mathcal{D} := \mathbb{S}_\alpha^{v_1}$ and $\mathcal{U} := \mathbb{S}_\beta^{v_2}$, and $\mathcal{R} := \mathbb{S}_\mathcal{B}^m$ for some $\alpha, \beta, \mathcal{B} \geq 1$.
- $\mathsf{Commit}_{ck}(\boldsymbol{m}, \boldsymbol{u})$ : Sample $\boldsymbol{r} \leftarrow \mathbb{S}_\mathcal{B}^m$. Output $C$ and $\mathsf{o} = (1, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r})$, where

$$C = \mathsf{Com}_{ck}(\boldsymbol{m}, \boldsymbol{u}; \boldsymbol{r}) = \boldsymbol{G} \cdot (\boldsymbol{r}, \boldsymbol{m}, \boldsymbol{u})^\top = \boldsymbol{A} \cdot \boldsymbol{r} + \boldsymbol{B} \cdot \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{u}.$$

- $\mathsf{COpen}_{ck}(C, (y, \boldsymbol{m}', \boldsymbol{u}', \boldsymbol{r}'))$ : If $yC = \mathsf{Com}_{ck}(y\boldsymbol{m}', \boldsymbol{u}'; \boldsymbol{r}')$, $\|(\boldsymbol{r}', y\boldsymbol{m}', \boldsymbol{u}')\| \leq \gamma_{\mathrm{com}}$, and $\dim(\boldsymbol{m}') = v_1$, $\dim(\boldsymbol{u}') = v_2$ and $\dim(\boldsymbol{r}') = m$ over $R_q$, return 1. Otherwise, return 0.

One can easily observe that HMC is additively homomorphic. Moreover, note that the opening algorithm is relaxed, where an additional *relaxation factor* $y \in R_q$ is involved. This relaxation is needed to obtain *efficient* lattice-based ZKPs. For classical commitment schemes such as Pedersen commitment, the

relaxation factor is always 1. The same is true for *honestly-created* lattice-based commitments. However, *efficient* lattice-based ZKPs do not always prove that this is the case. For example, for the *exact* proof of knowledge of a commitment opening with $y = 1$ as in [9], the proof length is more than 40 KB while a *relaxed* variant leads to a proof length of only a few KBs. Therefore, the relaxation factor can be a non-trivial value when created by a cheating prover (that still succeeds in the ZKP verification). For HMC used within our VPDC below, we say that the trapdoor-binding property is *satisfied w.r.t. to the same relaxation factor* if the relaxation factors in $\mathsf{o}$ and $\mathsf{o}'$ in the binding definition in Section 3 are restricted to be the same. This same-relaxation trapdoor-binding property is sufficient for our applications as well as many prior ones, e.g., [11, 12, 14], and can be based on a harder variant of the MSIS problem than the general trapdoor-binding property (see Lemma 1).

We remark that in some applications, the $\mathsf{COpen}$ algorithm checks a slightly different relation than above, of the form $yC = \mathsf{Com}_{ck}(\boldsymbol{m}', \boldsymbol{u}'; \boldsymbol{r}')$, where the relaxation factor $y$ does not multiply the decryptable message. However, in this paper, we need the stronger variant in the above definition. Despite the relaxation factor, HMC defined above is still (computationally) binding and hiding as we will discuss in Lemma 1.

### 5.2 Instantiation of NIZK

Our lattice-based VPDC can be instantiated with any suitable Schnorr-like lattice-based relaxed NIZK proofs of opening for HMC commitments derived from a Sigma protocol via the Fiat-Shamir transform, such as the one-shot relaxed binary proof protocols in [11,13,14]. For compatibility with such protocols, we define the challenge space

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

The same set is also defined in [11] and $|\mathcal{C}^d_{w,p}| = \binom{d}{w}(2p)^w$. Thus, given $d$, it is easy to set $(w, p)$ such that $|\mathcal{C}^d_{w,p}| > 2^{256}$. Throughout the manuscript, we assume that $(d, w, p)$ is set so that $|\mathcal{C}^d_{w,p}|$ is exponentially large. We also let $\Delta\mathcal{C}^d_{w,p}$ denote the set of differences of challenges in $\mathcal{C}^d_{w,p}$ except for the zero element. We design our VPDC to work with the following definition of relaxed "well-formedness" of a commitment relation. We refer the reader to Lemma 2 in the Sec. 6 for a concrete example of such a relaxed NIZK protocol $\Sigma$ in our cryptocurrency protocol application.

**Definition 5 ($\gamma$-valid commitment opening relation $\mathsf{R}'_{\mathsf{C},pp}$).** *We say that $\mathsf{o} := (y, (\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r}))$ is a $\gamma$-valid opening of a commitment $C$ with a decryptable message space $\mathcal{D}$, denoted by $(C, \mathsf{o}) \in \mathsf{R}'_{\mathsf{C},pp}$, if the following holds:*

- $y \in \Delta\mathcal{C}^d_{w,p}$,
- $\boldsymbol{m} \in \mathcal{D}$,
- $yC = \mathsf{Com}_{ck}(y\boldsymbol{m}, \boldsymbol{u}; \boldsymbol{r})$,
- $\|(y\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r})\| \leq \gamma$ *for* $\gamma \in \mathbb{R}^+$,

– $\dim(\boldsymbol{m}) = v_1$, $\dim(\boldsymbol{u}) = v_2$ *and* $\dim(\boldsymbol{r}) = m$ *over* $R_q$.

The above relation definition is very similar to $\mathsf{COpen}$ except that we additionally have the first two requirements. For our proof systems (as in Schnorr-like proofs), the commitment $w$ of the Fiat-Shamir protocol as given in Section 4 is uniquely determined by the rest of the proof output. Hence, it need not be included in the non-interactive proof transcript and therefore its notation is omitted in the rest of the paper.

### 5.3 VPDC Trapdoor for HMC

Now, we present our gadget-based Regev-style VPDC trapdoor algorithm $\mathsf{CAddTd}$ for our lattice-based VPDC based on the HMC commitment described in Sec. 5.1.

Our trapdoor $\boldsymbol{s}$ is designed to allow partial decryption of the latter HMC commitment, i.e., to recover the decryptable binary message $\boldsymbol{m} \in \{0,1\}^{v_1}$ from the commitment $C = \mathsf{Com}_{ck}(\boldsymbol{m}, \boldsymbol{u}; \boldsymbol{r}) = \boldsymbol{G} \cdot \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{m} \\ \boldsymbol{u} \end{pmatrix}$, where $\boldsymbol{G} = [\, \boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C} \,] \in R_q^{n \times (m+v_1+v_2)}$ is the commitment key matrix, and $\boldsymbol{r}$ and $\boldsymbol{u}$ are the short non-decryptable commitment randomness and auxiliary message, respectively. Our trapdoor $\boldsymbol{s}$ is embedded into the matrix $\boldsymbol{G}$ such that $\boldsymbol{s}^\top \cdot \boldsymbol{G} \approx [\boldsymbol{0}, \bar{t}\boldsymbol{g}^\top, \boldsymbol{0}] \in R_q^{m+v_1+v_2}$ (with the approximate equality up to a 'short' error vector) where $\bar{t}\boldsymbol{g}^\top$ (with $\bar{t} = \lfloor q/t \rfloor$) is a large gadget vector of the form $\bar{t} \cdot (1, 2, 2^2, \ldots, 2^{\tau-1}, X, X \cdot 2, \ldots, X \cdot 2^{\tau-1}, \ldots)$. This means that VPDC partial decryption of the commitment $C$ can be carried out by computing $\boldsymbol{s}^\top C \approx \bar{t}\boldsymbol{g}^\top \boldsymbol{m}$. The (approximately) 0 entries in $\boldsymbol{s}^\top \cdot \boldsymbol{G}$ annihilate the non-decryptable $\boldsymbol{r}$ and $\boldsymbol{u}$ vectors in decryption (these vectors only contribute to the short error terms in the approximate equality), whereas the gadget vector $\bar{t}\boldsymbol{g}^\top$ entry of $\boldsymbol{s}^\top \cdot \boldsymbol{G}$ 'selects' the decryptable message $\boldsymbol{m}$ and compresses its dimension by reconstructing and packing groups of $\tau$ bits in $\boldsymbol{m}$ into the integer coefficients of $1, X, X^2, \ldots$ in the ring element $\boldsymbol{g}^\top \boldsymbol{m}$. To achieve the desired trapdoor condition $\boldsymbol{s}^\top [\, \boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C} \,] \approx [\boldsymbol{0}, \bar{t}\boldsymbol{g}^\top, \boldsymbol{0}]$, for the 'selection' gadget, we embed a Regev-style LWE decryption '*gadget trapdoor*' into the last row $\boldsymbol{t}_B^\top$ of matrix $\boldsymbol{B}$, setting $\boldsymbol{t}_B^\top \approx \boldsymbol{s'}^\top \boldsymbol{B'} + \bar{t}\boldsymbol{g}^\top$, where $\boldsymbol{B'}$ consists of the top $n-1$ rows of $\boldsymbol{B}$ and $\boldsymbol{s'} \in R_q^{n-1}$ is random, and we use the form $\boldsymbol{s} = (-\boldsymbol{s'}, 1)$ for the trapdoor. For the annihilating 0 entries of $\boldsymbol{s}^\top \cdot \boldsymbol{G}$, we embed a Regev-style '*error trapdoor*' into the last rows $\boldsymbol{t}_A^\top$ (resp. $\boldsymbol{t}_C^\top$) of matrices $\boldsymbol{A}$ (resp. $\boldsymbol{C}$), setting them to $\approx \boldsymbol{s'}^\top \boldsymbol{A'}$ (resp. $\approx \boldsymbol{s'}^\top \boldsymbol{C'}$), where $\boldsymbol{A'}$ (resp. $\boldsymbol{C'}$) denote the top $n-1$ rows of $\boldsymbol{A}$ (resp. $\boldsymbol{C}$). Due to the errors in the above approximate equalities, to an attacker not knowing the secret trapdoor $\boldsymbol{s'}$, the trapdoor rows of matrix $\boldsymbol{G}$ are indistinguishable from uniformly random rows, assuming the hardness of the rank-$(n-1)$ M-LWE problem with respect to the secret $\boldsymbol{s'}$.

We now summarise our new gadget-based Regev-style HMC VPDC construction and start with instantiating $\mathsf{CAddTd}$.

– $\mathsf{CAddTd}(ck)$ : Let $ck = [\,\boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C}\,] \in R_q^{n\times(m+v_1+v_2)}$ where $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{A}'\\ \boldsymbol{a}^\top\end{bmatrix}$ for $\boldsymbol{A}' \in R_q^{(n-1)\times m}$ and $\boldsymbol{a} \in R_q^m$, $\boldsymbol{B} = \begin{bmatrix}\boldsymbol{B}'\\ \boldsymbol{b}^\top\end{bmatrix}$ for $\boldsymbol{B}' \in R_q^{(n-1)\times v_1}$ and $\boldsymbol{b} \in R_q^{v_1}$, and $\boldsymbol{C} = \begin{bmatrix}\boldsymbol{C}'\\ \boldsymbol{c}^\top\end{bmatrix}$ for $\boldsymbol{C}' \in R_q^{(n-1)\times v_2}$ and $\boldsymbol{c} \in R_q^{v_2}$. Sample $\boldsymbol{s}' \leftarrow R_q^{n-1}$, $\boldsymbol{e}_0 \leftarrow \mathbb{S}_{\mathcal{B}_e}^m$, $\boldsymbol{e}_1 \leftarrow \mathbb{S}_{\mathcal{B}_e}^{v_1}$, and $\boldsymbol{e}_2 \leftarrow \mathbb{S}_{\mathcal{B}_e}^{v_2}$, and set $\boldsymbol{A}^{\mathsf{td}} = \begin{bmatrix}\boldsymbol{A}'\\ \boldsymbol{t}_0^\top\end{bmatrix}$, $\boldsymbol{B}^{\mathsf{td}} = \begin{bmatrix}\boldsymbol{B}'\\ \boldsymbol{t}_1^\top\end{bmatrix}$, $\boldsymbol{C}^{\mathsf{td}} = \begin{bmatrix}\boldsymbol{C}'\\ \boldsymbol{t}_2^\top\end{bmatrix}$ where $\boldsymbol{t}_0 = \boldsymbol{A}'^\top\boldsymbol{s}' + \boldsymbol{e}_0$, $\boldsymbol{t}_1 = \boldsymbol{B}'^\top\boldsymbol{s}' + \boldsymbol{e}_1 + \bar{t}\boldsymbol{g}$ and $\boldsymbol{t}_2 = \boldsymbol{C}'^\top\boldsymbol{s}' + \boldsymbol{e}_2$, with

$$\boldsymbol{g}^\top := (2^0X^0, \ldots, 2^{\tau-1}X^0, 2^0X^1, \ldots, 2^{\tau-1}X^1, \ldots, 2^0X^{d'}, \ldots, 2^{\ell-1}X^{d'}) \in R^{v_1},$$

$\tau := \lceil \frac{v_1}{d} \rceil$, $d' := \lfloor \frac{v_1}{\tau} \rfloor$ (note that $d' \le d$) and $\ell := v_1 \bmod \tau \in \{0, \ldots, \tau-1\}$. Output $(ck^{\mathsf{td}}, \mathsf{td}) = ([\,\boldsymbol{A}^{\mathsf{td}} \,\|\, \boldsymbol{B}^{\mathsf{td}} \,\|\, \boldsymbol{C}^{\mathsf{td}}\,], \boldsymbol{s})$ where $\boldsymbol{s} = \begin{pmatrix}-\boldsymbol{s}'\\ 1\end{pmatrix}$.

The following lemma follows from the hiding/binding properties of standard HMC commitments, and the M-LWE based key indistinguishability property of $\mathsf{CAddTd}$. The proof is given in Appendix B.

**Lemma 1.** *Let the ring $R_q$ split into $s$ fields $\mathbb{F}_{p_1}, \ldots, \mathbb{F}_{p_s}$ with $p = \min\{p_1, \ldots, p_s\}$. If $\frac{n\cdot s}{p^{m-n+1}}$ is negligible, then HMC under 'trapdoored' commitment keys as output by $\mathsf{CAddTd}$ defined above is*

- *correct if $\gamma_{com} \ge \sqrt{\mathcal{B}^2md + (\gamma_y\alpha d)^2v_1 + \beta^2v_2d}$,*
- *computationally trapdoor $\gamma_{\mathrm{com}}$-binding with respect to the same relaxation factor (resp. $\gamma_{\mathrm{com}}$-binding) if M-SIS$_{n-1,m+v_1+v_2,q,2\gamma_{\mathrm{com}}}$ is hard (resp. if M-SIS$_{n-1,m+v_1+v_2,q,2\sqrt{d}\gamma_Y\cdot\gamma_{\mathrm{com}}}$ is hard, where $\gamma_Y := max_{y\in Y}\|y\|$, and $Y$ is the set of valid relaxation factors accepted by $\mathsf{COpen}$; for our VPDC, it suffices to use $Y := \Delta\mathcal{C}_{w,p}^d$ as in Def. 5).*
- *computationally hiding if M-LWE$_{m-n,m,q,\mathcal{B}}$ and M-LWE$_{n-1,m+v_1+v_2,q,\mathcal{B}}$ problems are hard.*

*Additionally, if M-LWE$_{m-n,m,q,\mathcal{B}}$ and M-LWE$_{n-1,m+v_1+v_2,q,\mathcal{B}}$ problems are hard, any commitment vector is computationally indistinguishable from a uniformly random element in $R_q^n$.*

Note that there are two main differences in Lemma 1 compared to the assumptions required for standard HMC (see [14, Lemma 2.3], [8, Lemma 3.4]): (i) the module rank of M-SIS is reduced by 1 (from $n$ to $n-1$), and (ii) the hardness of M-LWE$_{n-1,m+v_1+v_2,q,\mathcal{B}}$ is additionally required. As mentioned before, binding w.r.t. the same relaxation factor is sufficient for many applications (including ours) since the reduction creates a challenge commitment with a known *exact* opening (i.e., $y = 1$) and recovers another *relaxed* opening by rewinding the adversary. The former exact opening can be multiplied by the relaxation factor of the latter to solve an M-SIS problem.

**Algorithm 2** $\mathsf{CDecGR}(C, x, \mathsf{td}, v_1)$

**INPUT:** a commitment $C \in R_q^n$; a challenge $x \in \mathcal{C}_{w,p}^d$; trapdoor $\mathsf{td} = \boldsymbol{s} \in R_q^n$; the dimension $v_1$ such that $\mathcal{D} = \{0,1\}^{v_1}$
**OUTPUT:** $(\boldsymbol{m}', x') \in \mathcal{D} \times \mathcal{C}_{w,p}^d$

**loop**
  $x' \leftarrow \mathcal{C}_{w,p}^d$
  $y' = x - x'$ ▷ $y' = 1$ is assumed to be tried first
  $C' = \langle \boldsymbol{s}, y'C \rangle$
  $C'' = \mathsf{Rnd}_{\bar{t}}(C')$ where $\bar{t} = \lfloor q/t \rfloor$
  $\bar{m}' = (\bar{t})^{-1} \cdot C'' \in R$ ▷ Note that $C''$ is a multiple of $\bar{t}$ in $R$
  $m'' = (y')^{-1} \cdot \bar{m}' \in R_t$ ▷ If $y'$ is not invertible in $R_t$, restart from Step 2
  $\boldsymbol{m}' = \mathsf{BD}_{\tau,v_1}(m'')$
  $e' = C' - C''$
  **if** $(\|e'\|_\infty < \|e\|_{bnd,\infty})$ **and** $(m'' \in [0, \ldots, 2^\tau - 1]^d)$ **then**
    **return** $(\boldsymbol{m}', x')$
  **end if**
**end loop**

### 5.4 Gadget-based Regev-style Decryption for HMC

We now present the decryption algorithm $\mathsf{CDecGR}$ for our lattice-based VPDC. When a commitment key with a trapdoor is used to generate a proof, the ZKPs we use prove knowledge of an opening $(y, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r})$ of a commitment $C$ such that

$$yC = \mathsf{Com}_{ck^{\mathsf{td}}}(y\boldsymbol{m}, \boldsymbol{u}; \boldsymbol{r}) = \boldsymbol{A}^{\mathsf{td}}\boldsymbol{r} + \boldsymbol{B}^{\mathsf{td}} y\boldsymbol{m} + \boldsymbol{C}^{\mathsf{td}}\boldsymbol{u}. \tag{8}$$

Note that the opening message is also multiplied by the relaxation factor $y$. From here, we can try to eliminate the randomness $\boldsymbol{r}$ and the auxiliary message $\boldsymbol{u}$ by multiplying both sides by the secret trapdoor $\boldsymbol{s}$. However, the decryptor does not know what $y$ is. For an honest user, we simply have $y = 1$, but for adversarially-generated proofs, that may not be the case. Thankfully, we can use our new results from Section 4 to overcome this problem. Let us first present the full procedure in Algorithm 2. In this algorithm, the decrypted message is encoded as an element of $R_t$ for some positive integer $t$, and we define the integer $\bar{t} := \lfloor q/t \rfloor$. We also use the following two functions. The function $\mathsf{BD}_{\tau,v_1}(m'')$ performs bit decomposition of the coefficients of the $R_t$-encoded message $m'' = m''_0 + m''_1 X + \cdots + m''_{d-1} X^{d-1}$ and returns the resulting binary vector message $\boldsymbol{m}' = (m'_0, \ldots, m'_{v_1-1}) \in \{0,1\}^{v_1}$. Namely, for $j \in \{0, \ldots, v_1 - 1\}$, it sets $m'_j$ to the $k$-th bit of the coefficient $m''_{\lfloor j/\tau \rfloor}$ where $k := j \bmod \tau$. The function $\mathsf{Rnd}_{\bar{t}}(C')$ rounds each coefficient of $C' \in R$ to the nearest integer multiple of $\bar{t}$.

As mentioned in Sec. 4, an important task is to prove that the message returned by the decryption algorithm (Alg. 2) is "valid". We prove this in Theorem 2 below so that, for a commitment $C$ with a valid NIZK relaxed proof of opening and a sufficiently large $q$, the message output by Alg. 2 is the same as the one used to generate the commitment $C$. In the theorem below, we show the decryption feasibility (which relies on the results from Sec. 4) and also the decryption soundness of our construction.

**Theorem 2 (HMC Decryption).** *Let* $\mathsf{VPDC}_{\mathsf{HMC}} = (\mathsf{C}, \Sigma, \mathsf{CAddTd}, \mathsf{CDecGR})$ *denote our lattice-based VPDC construction with HMC commitment scheme* $\mathsf{C}$*,* $\Sigma$ *a matching NIZK relaxed proof of* $\gamma$*-valid opening relation* $\mathsf{R}'_{\mathsf{C},pp}$ *as in Def. 5, with* $\mathcal{D} := \{0,1\}^{v_1}$*. Suppose that* $\Sigma$ *is obtained from a Sigma protocol* $\Sigma_{\mathsf{I}}$ *using the Fiat-Shamir transform with random oracle* $\mathcal{H} : \{0,1\}^* \to \mathcal{C}$*,* $\Sigma_{\mathsf{I}}$ *satisfies Existential Special Soundness (Def. 3), that for any fixed* $x \in \mathcal{C}^d_{w,p}$*,* $x - x' \in \Delta\mathcal{C}^d_{w,p}$ *is invertible in* $R_t$ *except with negligible probability* $p_{\text{ni}}$ *over the uniformly random choice of* $x' \in \mathcal{C}^d_{w,p}$ *and* $|\mathcal{C}^d_{w,p}| \geq 2^\lambda$*.*

*For an adversary* $\mathcal{A}$ *against soundness game* ***Exp:Soundness*** *making* $q_H - 1$ *queries to its random oracle, let*

$$\|e\|_{bnd,\infty} := \sqrt{(m + v_1 + v_2)d}\mathcal{B}_e\gamma + 2pw(2^\tau - 1) + t/2. \tag{9}$$

*Suppose that* $t \geq 2^\tau$ *and*

$$\bar{t} := \lfloor q/t \rfloor > 4pw\|e\|_{bnd,\infty} + t(1/2 + 2pw). \tag{10}$$

*Then the following holds:*

1. ***Decryption Feasibility:*** *The scheme* $\mathsf{VPDC}_{\mathsf{HMC}}$ *satisfies Decryption Feasibility. Concretely, the number of iterations* $T$ *of the loop over* $x'$ *in* $\mathsf{CDecGR}$ *is upper bounded by* $\alpha \cdot q_H$*, except with probability at most* $\frac{1}{\alpha} + 2\sqrt{\frac{q_H}{\alpha \cdot |\mathcal{C}^d_{w,p}|}} + \frac{q_H}{|\mathcal{C}^d_{w,p}|} + \alpha q_H p_{\text{ni}}$*.*
2. ***Decryption soundness:*** *The scheme* $\mathsf{VPDC}_{\mathsf{HMC}}$ *satisfies Decryption Soundness. Concretely, we have*

$$Adv_{\textit{Exp:Soundness}}(\mathcal{A}) \leq q_H/|\mathcal{C}^d_{w,p}|.$$

*Proof (Thm. 2).* **Decryption Feasibility:** To prove the run-time claim, we apply Theorem 1. For this, we need to show that the assumptions of Theorem 1 are satisfied. First, by our assumption on $\Sigma_{\mathsf{I}}$, the existential special-soundness property is satisfied. For the compatibility of $\mathsf{CDecGR}$ and $\Sigma_{\mathsf{I}}$, the property $\mathbf{P}_1$ is satisfied by structure of the algorithm $\mathsf{CDecGR}$. For the property $\mathbf{P}_2$, we show that if $y'$ in some iteration of the loop in Step 3 of Algorithm $\mathsf{CDecGR}$ is 'good' in the sense that $y'$ is invertible in $R_t$ and there exists a $\gamma$-valid opening $(y', \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r})$ of $C$ as in Def. 5, then decryption will terminate and return $\boldsymbol{m}' = \boldsymbol{m}$. Let us denote by $E_0$ the bad event that $y'$ is not invertible in $R_t$. We first observe that $E_0$ occurs with negligible probability, i.e. $\Pr[E_0] \leq \alpha q_H p_{\text{ni}}$ over at most $\alpha q_H$ iterations of $\mathsf{CDecGR}$, since at each iteration $y' = x - x'$ where $x'$ sampled uniformly from $\mathcal{C}^d_{w,p}$ independently of $x$. Now we show that $\mathbf{P}_2$ holds if $E_0$ does not occur. Indeed, by $\gamma$-validity of $(y', \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r})$, we have $y'C = \boldsymbol{A}^{\mathsf{td}}\boldsymbol{r} + \boldsymbol{B}^{\mathsf{td}}y'\boldsymbol{m} + \boldsymbol{C}^{\mathsf{td}}\boldsymbol{u}$ and $\boldsymbol{m} \in \{0,1\}^{v_1}$.

Multiplying the $\gamma$-valid relation by $\boldsymbol{s}^\top$, defining $\langle \boldsymbol{e}_0, \boldsymbol{r}\rangle + \langle \boldsymbol{e}_1, y'\boldsymbol{m}\rangle + \langle \boldsymbol{e}_2, \boldsymbol{u}\rangle := e$, and using $\boldsymbol{s}^\top \cdot \boldsymbol{A}^{\mathsf{td}} = \boldsymbol{e}_0^\top$, $\boldsymbol{s}^\top \cdot \boldsymbol{B}^{\mathsf{td}} = \bar{t}\boldsymbol{g}^\top + \boldsymbol{e}_1^\top$ and $\boldsymbol{s}^\top \cdot \boldsymbol{C}^{\mathsf{td}} = \boldsymbol{e}_2^\top$, we have $\langle \boldsymbol{s}, y'C\rangle = \bar{t}y'\langle \boldsymbol{g}, \boldsymbol{m}\rangle + e$ over $R_q$. Writing $y'\langle \boldsymbol{g}, \boldsymbol{m}\rangle = (y'\langle \boldsymbol{g}, \boldsymbol{m}\rangle \bmod t) + t\lfloor \frac{y'\langle \boldsymbol{g}, \boldsymbol{m}\rangle}{t} \rceil$, we get the following equality over $R_q$:

$$\langle \boldsymbol{s}, y'C\rangle = \bar{t}(y'\langle \boldsymbol{g}, \boldsymbol{m}\rangle \bmod t) + e + \tilde{e}, \tag{11}$$

where $\tilde{e} := \bar{t}t \cdot \lfloor \frac{y'\langle \boldsymbol{g},\boldsymbol{m}\rangle}{t} \rceil \bmod q$. We have $\|\bar{t}(y'\langle \boldsymbol{g},\boldsymbol{m}\rangle \bmod t)\|_\infty \leq \lfloor q/t \rfloor (t-1)/2 \leq q/2 - \bar{t}/2$. Hence, if $\|e+\tilde{e}\|_\infty < \bar{t}/2$, there is no wraparound mod $q$ on the right hand side of (11), and since $\bar{t}y'\langle \boldsymbol{g},\boldsymbol{m}\rangle$ is a multiple of $\bar{t}$ in $R$, the rounded polynomial $C'' = \mathsf{Rnd}_{\bar{t}}(C')$ (recall $C' = \langle \boldsymbol{s}, y'C\rangle \bmod q$) will be equal to $\bar{t}(y'\langle \boldsymbol{g},\boldsymbol{m}\rangle \bmod t)$ and decryption will succeed and return $\boldsymbol{m}$. It remains to show that $\|e+\tilde{e}\|_\infty < \bar{t}/2$. By the Schwartz inequality, $\|e\|_\infty \leq \|(\boldsymbol{e}_0,\boldsymbol{e}_1,\boldsymbol{e}_2)\| \cdot \|(\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})\| \leq \sqrt{(m+v_1+v_2)d}\mathcal{B}_{\mathrm{e}}\gamma$ using $\|(\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})\| \leq \gamma$ by $\gamma$-validity. Also, writing $\bar{t} = q/t - \epsilon$ for $0 \leq \epsilon < 1$, we have $\|\tilde{e}\|_\infty = \|(q/t-\epsilon)t \cdot \lfloor \frac{y'\langle \boldsymbol{g},\boldsymbol{m}\rangle}{t} \rceil \bmod q\|_\infty = \|\epsilon t \cdot \lfloor \frac{y'\langle \boldsymbol{g},\boldsymbol{m}\rangle}{t} \rceil\|_\infty \leq \|t \lfloor \frac{y'\langle \boldsymbol{g},\boldsymbol{m}\rangle}{t} \rceil\|_\infty \leq t/2 + \|y'\langle \boldsymbol{g},\boldsymbol{m}\rangle\|_\infty$, and $\|y'\langle \boldsymbol{g},\boldsymbol{m}\rangle\|_\infty \leq \|y'\|_1 \|\langle \boldsymbol{g},\boldsymbol{m}\rangle\|_\infty \leq (2pw)(2^\tau - 1)$ using $\|y'\|_1 \leq 2pw$ and $\|\langle \boldsymbol{g},\boldsymbol{m}\rangle\|_\infty \leq 2^\tau - 1$ since $\boldsymbol{m} \in \{0,1\}^{v_1}$. Overall, we have $\|e+\tilde{e}\|_\infty \leq \|e\|_\infty + \|\tilde{e}\|_\infty \leq \sqrt{(m+v_1+v_2)d}\mathcal{B}_{\mathrm{e}}\gamma + (2pw)(2^\tau-1) + t/2 := \|e\|_{bnd,\infty}$, which is less than $\bar{t}/2$ by condition (10), as required.

**Decryption Soundness:** To show the decryption soundness claim, let $E_1$ denote the event that $\mathcal{A}$ wins and case (i) in `Exp:Soundness` occurs, i.e., $\mathsf{V}(C,x,\boldsymbol{z}) = 1$ but a $\gamma$-valid opening $(C, y, (\boldsymbol{m},\boldsymbol{u},\boldsymbol{r}))$ of $C$ with $y = x - x''$ and $x'' \in \mathcal{C}^d_{w,p}$ does *not* exist. Similarly, let $E_2$ be the event that $\mathcal{A}$ wins and case (ii) in `Exp:Soundness` occurs, i.e., $\mathsf{V}(C,x,\boldsymbol{z}) = 1$ and a $\gamma$-valid opening $(C, y, (\boldsymbol{m},\boldsymbol{u},\boldsymbol{r}))$ of $C$ with $y = x - x''$ and $x'' \in \mathcal{C}^d_{w,p}$ exists, but $\mathsf{CDecGR}$ returns the wrong message $\boldsymbol{m'} \neq \boldsymbol{m}$. We show that $\Pr[E_1] + \Pr[E_2] \leq \frac{q_H}{|\mathcal{C}^d_{w,p}|}$.

We first claim that $\Pr[E_1] \leq \frac{q_H}{|\mathcal{C}^d_{w,p}|}$. Indeed, for each $\mathcal{H}$-query of $\mathcal{A}$ of the form $(pp, C, \cdot)$, we have that for any query answer $x' \neq x$, there does not exist a $\boldsymbol{z}'$ such that $\mathsf{V}(C,x',\boldsymbol{z}') = 1$ (otherwise, by existential special-soundness of the protocol $\boldsymbol{\Sigma}_{\mathsf{I}}$, a $\gamma$-valid opening $(C, y, (\boldsymbol{m},\boldsymbol{u},\boldsymbol{r}))$ of $C$ with $y = x - x'$ would exist, a contradiction with $E_1$). It follows that $\Pr[E_1]$ is upper bounded by the probability that $\mathcal{A}$ receives the special challenge $x$ for which a $\boldsymbol{z}$ exists in one of the $\leq q_H$ queries to $\mathcal{H}$. Since the special challenge is returned with probability $1/|\mathcal{C}^d_{w,p}|$ in each query, the claimed bound on $\Pr[E_1]$ follows.

Next, we claim that $\Pr[E_2] = 0$. On the one hand, if $E_2$ occurs, then the existence of the $\gamma$-valid opening $(C, y, (\boldsymbol{m},\boldsymbol{u},\boldsymbol{r}))$ of $C$ with $y = x - x''$ means that $yC = \boldsymbol{A}^{\mathsf{td}}\boldsymbol{r} + \boldsymbol{B}^{\mathsf{td}} y\boldsymbol{m} + \boldsymbol{C}^{\mathsf{td}}\boldsymbol{u}$. Similarly to (11), multiplying the latter by $\boldsymbol{s}^\top$ gives us the following relation over $R_q$:

$$\langle \boldsymbol{s}, yC\rangle = \bar{t}(y\langle \boldsymbol{g},\boldsymbol{m}\rangle \bmod t) + e + \tilde{e}, \tag{12}$$

where $e := \langle \boldsymbol{e}_0, \boldsymbol{r}\rangle + \langle \boldsymbol{e}_1, y\boldsymbol{m}\rangle + \langle \boldsymbol{e}_2, \boldsymbol{u}\rangle$ and $\tilde{e} := \bar{t}t \cdot \lfloor \frac{y\langle \boldsymbol{g},\boldsymbol{m}\rangle}{t} \rceil \bmod q$. The same bound $\|e\|_{bnd,\infty}$ on $\|e+\tilde{e}\|$ applies by the same argument as in the run-time proof, based on Schwartz inequality. On the other hand, let $y' = x - x'$ be the value chosen in the iteration of the loop in $\mathsf{CDecGR}$ for which the message $\boldsymbol{m'}$ is returned. Then $\bar{m}' = \bar{t}(y'\langle \boldsymbol{g},\boldsymbol{m'}\rangle \bmod t) \in R$, and we get the following relation over $R_q$:

$$\langle \boldsymbol{s}, yC\rangle = \bar{t}(y'\langle \boldsymbol{g},\boldsymbol{m'}\rangle \bmod t) + e', \tag{13}$$

where $\|e'\|_\infty < \|e\|_{bnd,\infty}$ by the decryption check of $\mathsf{CDecGR}$.

We now multiply (12) by $y'$ and subtract (13) multiplied by $y$. Let $b_1 := y'(y\langle \boldsymbol{g},\boldsymbol{m}\rangle \bmod t) \in R$, $b_2 := y(y'\langle \boldsymbol{g},\boldsymbol{m'}\rangle \bmod t) \in R$. Note that $b_1 - b_2 =$

$y'y\langle \boldsymbol{g}, \boldsymbol{m}-\boldsymbol{m}'\rangle \bmod t$. Writing $b_1 - b_2 = (y'y\langle \boldsymbol{g}, \boldsymbol{m}-\boldsymbol{m}'\rangle \bmod t) + t\lfloor \frac{b_1-b_2}{t} \rceil$ gives the following relation over $R_q$:

$$\bar{t}(y'y\langle \boldsymbol{g}, \boldsymbol{m}-\boldsymbol{m}'\rangle \bmod t) = y'(e+\tilde{e}) - ye' - \tilde{e}', \tag{14}$$

where $\tilde{e}' := \bar{t}t\lfloor \frac{b_1-b_2}{t} \rceil \bmod q$. We claim that the relation (14) leads to a contradiction, so that $\Pr[E_2] = 0$. To see this, first observe that the relation actually holds over $R$, not just $R_q$. Indeed, there is no wraparound mod $q$ in the left hand side of (14), since the left hand side norm is at most $\|\lfloor q/t \rfloor \cdot t/2\|_\infty < q/2$. Since $\boldsymbol{m}-\boldsymbol{m}' \neq 0$, and $\boldsymbol{m}-\boldsymbol{m}' \in \{-1,0,1\}^{v_1}$, we have $\|\langle \boldsymbol{g}, \boldsymbol{m}-\boldsymbol{m}'\rangle\|_\infty \leq 2^\tau - 1 < t$ so $\langle \boldsymbol{g}, \boldsymbol{m}-\boldsymbol{m}'\rangle \neq 0 \bmod t$. Note that $y$ is non-zero in $R$ (since $\gamma$-validity of $(C, y, (\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r}))$ implies $y \in \Delta\mathcal{C}^d_{w,p}$) and $y'$ is also non-zero in $R$ (since it caused termination of $\mathsf{CDecGR}$ and hence is invertible in $R_t$) and $R$ is an integral domain, the left hand side of (14) is a non-zero multiple of $\bar{t}$ in $R$. But the norm of the error terms on the right-hand side of (14) is bounded as follows. First, $\|y'(e+\tilde{e}) - ye'\|_\infty < \|y'\|_1\|e+\tilde{e}\|_\infty + \|y\|_1\|e'\|_\infty \leq 4pw\|e\|_{bnd,\infty}$ using $\|y'\|_1 \leq 2pw$, $\|y\|_1 \leq 2pw$, $\|e+\tilde{e}\|_\infty < \|e\|_{bnd,\infty}$ and $\|e'\|_\infty < \|e\|_{bnd,\infty}$. Also, $\|\tilde{e}'\|_\infty = \|\bar{t}t\lfloor \frac{b_1-b_2}{t} \rceil \bmod q\|_\infty \leq t\lfloor \frac{b_1-b_2}{t} \rceil \leq t(\frac{1}{2}+2pw)$ using $|\bar{t}t \mod q| < t$, and $\|\lfloor \frac{b_1-b_2}{t} \rceil\|_\infty \leq \frac{1}{2} + 2pw$ using $\|b_1\|_\infty \leq \|y'\|_1\|y\langle \boldsymbol{g}, \boldsymbol{m}\rangle \bmod t\|_\infty \leq (2pw)(t/2) \leq pw$, and similarly, $\|b_2\|_\infty \leq pw$. Overall, the norm of the right-hand side of (14) is bounded as $\|y'(e+\tilde{e}) - ye' - \tilde{e}'\|_\infty < 4pw\|e\|_{bnd,\infty} + t(\frac{1}{2} + 2pw)$, which is smaller than $\bar{t}$ by condition (10). So, the left-hand side cannot be a non-zero multiple of $\bar{t}$ in $R$, implying the claimed contradiction. This completes the proof that $\Pr[E_2] = 0$ and the claimed soundness bound. □

### 5.5 Generalized Decryption

Our gadget-based Regev-style decryption trapdoor presented in the previous section, which handles a binary decryptable message space $\mathcal{D} = \{0,1\}^{v_1}$, can be readily generalised to handle more general decryptable message spaces $\mathcal{D} = (\{0,\ldots,\beta-1\}[X]^{<\delta})^{v_1}$ whose coordinates are polynomials in the ring $R$ of degree $< \delta$ with $\beta$-ary coefficients for some positive integers $\delta, \beta > 1$. This generalisation can naturally be achieved via the appropriate generalisation of the reconstruction gadget vector $\boldsymbol{g}$, by setting

$$\boldsymbol{g}^\top := (\beta^0 X^0, \ldots, \beta^{\tau-1}X^0, \beta^0 X^\delta, \ldots, \beta^{\tau-1}X^\delta, \ldots, \beta^0 X^{d'\delta}, \ldots, \beta^{\ell-1}X^{d'\delta}) \in R^{v_1},$$

with $\tau := \left\lceil \frac{v_1}{\lfloor d/\delta \rfloor} \right\rceil$, $d' := \lfloor \frac{v_1}{\tau} \rfloor \leq \lfloor d/\delta \rfloor$. The decryption soundness result, Thm 2, directly extends to this generalised case with the term $2^\tau$ replaced by $\beta^\tau$.

### 5.6 Succinctness of Our HMC-based VPDC

An HMC commitment $C$ as defined in Section 5.1 costs $nd \log q$ bits, i.e.,

$$\mathsf{bitlen}(C) = nd \log q. \tag{15}$$

We show that we can choose parameters such that succinctness of the VPDC is satisfied, i.e., $\mathsf{bitlen}(C) = \log^{O(1)}(\mathsf{bitlen}(\boldsymbol{u}))$, where $\mathsf{bitlen}(\boldsymbol{u})$ is the bit length of *honestly generated* auxiliary messages, assuming the following very mild assumptions: (i) $v_1/d = O(\log \lambda)$ and (ii) $\gamma = (\lambda\|\boldsymbol{u}\|)^{O(1)}$, a condition that is typically satisfied by the soundness extractor of the associated ZKP. The auxiliary message space is defined as $\mathcal{U} := \mathbb{S}_{\mathcal{B}}^{v_2}$ with $\mathcal{B} = \lambda^{O(1)}$. Then $\mathsf{bitlen}(\boldsymbol{u}) := dv_2 \log(2\mathcal{B})$. Set $d = \Theta(\lambda)$, $v_1 = \Theta(\lambda)$ (with $v_1/d = O(\log \lambda)$), $v_2 = \lambda^{O(1)}$, $p = O(1)$, $w = \Theta(\lambda)$ (so that $|\mathcal{C}_{w,p}^d| \geq (d/w-1)^w \geq 2^\lambda$), $\mathcal{B}_e = \Theta(1)$ and $n, m = \Theta(\log \gamma)$. As a result, we have $t = O(2^\tau) = O(2^{v_1/d}) = \lambda^{O(1)}$.

In general, we require two conditions to be satisfied: decryption soundness requirements and M-SIS security requirements (note that M-LWE security affects the number of columns of the commitment matrix, and thus not the commitment size). Let us analyze these two aspects.

**(1) Partial-Decryption Soundness and Feasibility (Thm. 2)**:

$$q > \Omega(tpw\gamma B_e\sqrt{d(m+v_1+v_2)} + t^2pw) = \Omega(\lambda^{O(1)}\gamma\log\gamma). \tag{16}$$

**(2) M-SIS security (Lemma 1):** The hardness of $\text{M-SIS}_{n,m,q,\beta_{\text{SIS}}}$ requires (see [11, Section 1.2]):

$$nd\log q \geq \Omega(\lambda\log^2(\beta_{\text{SIS}})) = \Omega(\lambda\log^2(\gamma)). \tag{17}$$

Note that $\beta_{\text{SIS}} = 2\gamma$ as given in Lemma 1. We can satisfy both conditions with some $\log q = \Theta(\log(\lambda\gamma))$ (ignoring $\log\log\gamma$). With this choice, we get commitment length $\mathsf{bitlen}(C) = nd\log q = \Theta(\lambda\log^2(\gamma))$. To show it is polylog in $\mathsf{bitlen}(\boldsymbol{u})$, it suffices to show that $\log\gamma = \log^{O(1)}(\mathsf{bitlen}(\boldsymbol{u}))$. Assuming that $\gamma = (\lambda\|\boldsymbol{u}\|)^{O(1)}$, we have $\log\gamma \leq O(\log(\lambda\|\boldsymbol{u}\|)) = O(\log\lambda + \log(dv_2) + \log\mathcal{B}) = \log^{O(1)}(\mathsf{bitlen}(\boldsymbol{u}))$, as required.

## 6 Extending MatRiCT to Auditable Setting

Having dealt with the core task of constructing and analysing a VPDC, we now explain how our new VPDC construction can be applied to extend a privacy-preserving confidential transaction blockchain protocol MatRiCT [14] to the auditable setting. Unlike the auditability feature of the original MatRiCT protocol, where auditing may fail against adversarially-created transactions, our auditable MatRiCT variant, called MatRiCT-Au, takes advantage of our VPDC to efficiently provide auditability soundness guarantees against adverserial transactions. More specifically, we show that only minor modifications to MatRiCT are sufficient to add the auditability feature. As the whole MatRiCT protocol is quite involved, in this section, we only briefly review MatRiCT, focusing on the specific parts of the protocol which we modify. For the full algorithmic descriptions and formal analysis of MatRiCT-Au, we refer the reader to Appendices E and F.

MatRiCT follows the blueprint of RingCT-like [26] private blockchain payment protocols, in which there are two main entities: (i) *spenders/payers*, who

create transactions together with a proof of validity, and (ii) *verifiers*, who check that the proof and transaction is valid. The goal of the private payment protocol is to enable users to conduct transactions on blockchain while hiding sensitive transaction information such as the payer/payee identities and the payment amount. Once such information is concealed from the verifiers, it gets harder to validate transactions as we cannot, for example, simply check that the transaction amount is positive and the total balance of the transaction is zero (i.e., the amount spent equals the amount received). To this end, the payers create a NIZK proof showing that they are not creating an invalid transaction, for example, by proving that the balance is preserved.

To hide the payer identity, MatRiCT makes use of a 1-out-of-$N$ NIZK proof (or a ring signature), where the identity of the real payer is hidden within a set of possible payers. This involves committing to the unary representation of an index $\ell \in [0, N-1]$. To hide the payment amount, a commitment to the bits of the transaction amount is used. In fact, the bits representing the user index and those representing the transaction amount are committed in a single commitment. To enable an authority to recover these two critical data pieces, we apply our new VPDC from Section 5 for this commitment, so that the VPDC decryption algorithm recovers (i) the real payer's index among $N$ users, and (ii) the transaction amount. Let us investigate more details.

In MatRiCT, an HMC commitment $B = \mathsf{Com}_{ck}(\boldsymbol{b}, \boldsymbol{u}; \boldsymbol{r})$ is computed, where $\boldsymbol{b}$ is a binary vector over $R_{\hat{q}}$ for some $\hat{q} \in \mathbb{Z}^+$ (i.e., $\boldsymbol{b} = (b_0, b_1, \ldots)$ such that $b_i \in \{0, 1\} \subset R_{\hat{q}}$), $\boldsymbol{u}$ is some short auxiliary message and $\boldsymbol{r}$ is some short randomness. Here, the binary vector $\boldsymbol{b}$ is comprised of three components: (i) the unary representation of an index $\ell$ that identifies the real payer (i.e., spender) index among $N$ parties in a ring signature or a 1-out-of-$N$ proof, (ii) the bits in the binary representation of all output amounts, and (iii) the bits in the so-called "corrector values". Our target is to recover the first two components in decryption so that the authority can learn the two hidden data pieces mentioned above. Note that the payer indeed proves in zero-knowledge that she owns the $\ell$-th public key and that certain bits (with known indices) in $\boldsymbol{b}$ construct the output coins. Hence, recovering $\boldsymbol{b}$ guarantees that the real payer index and the output amounts (and thus the transaction amount) are revealed.

As it is expensive to perform an *exact* binary proof on $B$, MatRiCT performs a *relaxed* binary proof on $B$. Let us recall a simplified version of the relation proven in MatRiCT for the commitment $B$, which also applies to our variant MatRiCT-Au. We refer to Lemma 4 for the full relation (that depends on the system parameters chosen).

**Lemma 2.** *Assume that $\hat{q}$ is sufficiently large and that HMC is $\gamma_{\text{bin}}$-binding for some $\gamma_{\text{bin}}$ that depends on the system parameters. For an input commitment $B \in R_{\hat{q}}^{\hat{n}}$ and a commitment key $ck = \hat{\boldsymbol{G}} = [\, \boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C} \,]$ defined over $R_{\hat{q}}$, our binary proof proves knowledge of $(y, \boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})$ such that*

- $y \in \Delta\mathcal{C}_{w,p}^{d}$, $\hat{\boldsymbol{c}} \in R_{\hat{q}}^{v_2}$ *and* $\hat{\boldsymbol{r}} \in R_{\hat{q}}^{\hat{m}}$ *for some* $v_2, \hat{m} \geq 1$,
- $yB = \mathsf{Com}_{ck}(y\boldsymbol{b}, \hat{\boldsymbol{c}}; \hat{\boldsymbol{r}}) = \boldsymbol{A}\hat{\boldsymbol{r}} + \boldsymbol{B}y\boldsymbol{b} + \boldsymbol{C}\hat{\boldsymbol{c}}$,

- *All coordinates* $b_i$ *of* $\boldsymbol{b}$ *are in* $\{0,1\}$*, i.e.,* $\boldsymbol{b} \in \{0,1\}^{v_1} \subset R_{\hat{q}}^{v_1}$*, where* $v_1 = k\beta + Sr + \lceil \log(M+S-1) \rceil (r-1)$ *for parameters* $k, \beta$ *satisfying* $N = \beta^k$*,* $M, S$ *denoting the number of input/output accounts and* $r$ *denoting the bit length of each amount,*
- $\|(y\boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})\| \leq \gamma_{\mathrm{B}}$ *for some* $\gamma_{\mathrm{B}} \in \mathbb{R}^+$ *with* $\gamma_{\mathrm{B}} < \hat{q}$*.*

The above relation is effectively what we study in Def. 5 with $\mathcal{D} = \{0,1\}^{v_1}$. So, the extension we need to make over MatRiCT is to let the spender use the VPDC from Section 5 for the commitment $B$. In particular, the spender just needs to update $ck = \hat{\boldsymbol{G}}$ with a 'trapdoored' commitment key $ck^{\mathsf{td}}$ generated by $\mathsf{CAddTd}$. This simply means that the spender replaces the last row of $ck = \hat{\boldsymbol{G}}$ with trapdoor rows published by an authority. This way, the authority in possession of the corresponding trapdoor $\mathsf{td}$ can execute $\mathsf{CDecGR}$ (Alg. 2) to recover the vector $\boldsymbol{b}$, which in turn reveals the real payer index $\ell \in [0, N-1]$ and the transaction amount, which is equal to the sum of the output amounts. In particular, since the commitment algorithm for our VPDC remains exactly the same as the standard HMC commitment algorithm used in MatRiCT, our VPDC can be directly plugged in and used with the same efficient NIZK proof of transaction well-formedness used in MatRiCT.

For the concrete parameter setting in MatRiCT with $N = 100$, we have $\dim(\boldsymbol{b}) = v_1 = 291$ as $(k, \beta) = (1, N)$ and $(M, S, r) = (1, 2, 64)$ with the first $k\beta = 100$ bits having exactly a single '1'. As a result, there are more than $2^{191}$ possibilities for $\boldsymbol{b}$ (i.e., $|\mathcal{D}| > 2^{191}$). Hence, it is infeasible to do an exhaustive search over $\mathcal{D}$ (as done in [14]) for decryption. As our new decryption's runtime is *polylogarithmic* in $|\mathcal{D}|$, we can efficiently execute it. In particular, as we discuss in Appendix E.2, to guarantee auditability soundness against adverserially created commitments based on our VPDC security bounds while maintaining the same security level as MatRiCT against best-known lattice attacks, we only need to increase (i) the system modulus $\hat{q}$ to a 55-bit value from a 53-bit value and (ii) the commitment matrix dimensions slightly. As shown in Table 5, the decryption can be run very fast despite the exponentially large message space.

It is important to note here that MatRiCT and MatRiCT-Au crucially relies on an *aggregate* binary proof for compactness, where many messages are committed together inside a single commitment $B$. Therefore, the additional succinctness feature of VPDC plays an important role. If one were to replace this HMC commitment with an encryption (or an encryption-like commitment as in [3]), the proof/commitment length would significantly increase (as also discussed in the introduction) due to the large input message dimension (several hundreds) over the polynomial ring $R_{\hat{q}}$.

In Appendix D, we first extend the formal model from [14] for RingCT-like protocols to include an auditability feature, which allows us to study the security properties (namely, balance, anonymity and auditability) of MatRiCT-Au rigorously. Then, we describe MatRiCT-Au in full details in Appendix E, and show that addition of a trapdoor as in $\mathsf{CAddTd}$ is effectively the only modification required over MatRiCT. If auditability is not desired, which is decided by each user individually, MatRiCT-Au runs exactly as MatRiCT (with possibly slightly

different parameters). We instantiated MatRiCT-Au concretely (Appendix E.2) and implemented it in C/C++ (Appendix E.3). We compare MatRiCT and MatRiCT-Au in Table 1 with more run-time results in Table 5. Our results show that the overhead of MatRiCT-Au over MatRiCT is very small in both communication and computation.

**Acknowledgements.** This research was supported in part by ARC Discovery Project grants DP180102199 and DP220101234.

## References

1. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line TTP. In: IEEE S&P. pp. 77–85. IEEE Computer Society (1998)
2. Baum, C., Bootle, J., Cerulli, A., del Pino, R., Groth, J., Lyubashevsky, V.: Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 10992, pp. 669–699. Springer (2018)
3. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: SCN. pp. 368–385. Springer (2018)
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: EUROCRYPT. pp. 614–629. LNCS, Springer (2003)
5. Benhamouda, F., Camenisch, J., Krenn, S., Lyubashevsky, V., Neven, G.: Better zero-knowledge proofs for lattice encryption and their application to group signatures. In: ASIACRYPT. pp. 551–572. Springer (2014)
6. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: CRYPTO. LNCS, vol. 2729, pp. 126–144. Springer (2003)
7. Chaum, D., Van Heyst, E.: Group signatures. In: EUROCRYPT. pp. 257–265. Springer (1991)
8. Esgin, M.F.: Practice-Oriented Techniques in Lattice-Based Cryptography. Ph.D. thesis, Monash University (5 2020). https://doi.org/10.26180/5eb8f525b3562
9. Esgin, M.F., Nguyen, N.K., Seiler, G.: Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In: ASIACRYPT (2). LNCS, vol. 12492, pp. 259–288. Springer (2020)
10. Esgin, M.F., Steinfeld, R., Liu, D., Ruj, S.: Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. Cryptology ePrint Archive (2022)
11. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: CRYPTO (1). pp. 115–146. LNCS, Springer (2019), (Full version at `ia.cr/2019/445`)
12. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: ACNS. pp. 67–88. LNCS, Springer (2019), (Full version at `ia.cr/2018/773`)
13. Esgin, M.F., Steinfeld, R., Zhao, R.K.: MatRiCT$^+$: More efficient post-quantum private blockchain payments. Cryptology ePrint Archive, Report 2021/545 (2021), `ia.cr/2021/545` (to appear at IEEE S&P 2022)
14. Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In: ACM CCS. pp. 567–584. CCS '19, ACM (2019), (Full version at `ia.cr/2019/1287`)

15. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO. pp. 186–194. Springer (1986)
16. Granlund, T., Team, G.D.: GNU MP 6.0 Multiple Precision Arithmetic Library. Samurai Media Limited, United Kingdom (2015)
17. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: EUROCRYPT. pp. 253–280. Springer (2015)
18. Gueron, S.: Intel's new AES instructions for enhanced performance and security. In: FSE. pp. 51–66. LNCS, Springer (2009)
19. Li, W., Wang, Y., Chen, L., Lai, X., Zhang, X., Xin, J.: Fully auditable privacy-preserving cryptocurrency against malicious auditors (2019), ia.cr/2019/925
20. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: ASIACRYPT. pp. 598–616. Springer (2009)
21. Lyubashevsky, V.: Lattice signatures without trapdoors. In: EUROCRYPT. pp. 738–755. Springer (2012), (Full version)
22. Lyubashevsky, V., Neven, G.: One-shot verifiable encryption from lattices. In: EUROCRYPT. pp. 293–323. Springer (2017), (Full version at ia.cr/2017/122)
23. Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: EUROCRYPT. pp. 204–224. Springer (2018)
24. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT. pp. 700–718. LNCS, Springer (2012)
25. NIST: SHA-3 standard: Permutation-based hash and extendable-output functions. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf (2015), accessed: 2019-05-15
26. Noether, S.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015), ia.cr/2015/1098
27. Ravi, P., Gupta, S.S., Chattopadhyay, A., Bhasin, S.: Improving speed of dilithium's signing procedure. In: CARDIS. Lecture Notes in Computer Science, vol. 11833, pp. 57–73. Springer (2019)
28. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009), preliminary version in STOC 2005
29. Schnorr, C.: Efficient identification and signatures for smart cards. In: CRYPTO. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
30. Young, A.L., Yung, M.: Auto-recoverable auto-certifiable cryptosystems. In: EUROCRYPT. LNCS, vol. 1403, pp. 17–31. Springer (1998)

# A Further Preliminaries

## A.1 Zero-Knowledge Proofs

We use the following definition for syntax and properties of Relaxed Non-Interactive Zero-Knowledge (NIZK) proofs. Our definition is adapted from [22]. It allows for a relaxed soundness relation $\mathsf{R}'$ as distinct from the completeness relation $\mathsf{R}$, to model practical relaxed lattice-based proofs. It also captures both NIZKs in the standard Common Reference String (CRS) and random-oracle models, although our VPDC construction will focus on the latter random-oracle construction, which also uses a common reference string to encode the underlying commitment key $ck$. Our definition also allows for NIZK languages/relations

that depend on the CRS $\sigma$, which corresponds to the dependence of commitment opening relation on the commitment key. We regard the commitment key as part of the CRS in our construction (this issue arises in both discrete-log and lattice-based instantiations of commitments).

A Relaxed NIZK $\Sigma = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ for relation $\mathsf{R}_\sigma$ and its relaxed counterpart $\mathsf{R}'_\sigma$ with $\mathsf{R}_\sigma \subseteq \mathsf{R}'_\sigma$ (parameterised by a common reference string $\sigma$) and their corresponding languages $L_\sigma = \{u : \exists r \text{ s.t. } (u, r) \in \mathsf{R}_\sigma\}$ and $L'_\sigma = \{u : \exists r \text{ s.t. } (u, r) \in \mathsf{R}'_\sigma\}$ respectively, consists of the following algorithms (here, $u$ denotes a language member and $r$ denotes a witness):

$\sigma \leftarrow \mathsf{K}(1^\lambda)$ : is the PPT common reference string generation algorithm of $\Sigma$ that outputs a common reference string $\sigma$.

$\pi \leftarrow \mathsf{P}^{\mathcal{H}}(\sigma, u, r)$ : is the PPT prover algorithm of $\Sigma$ that, given a common reference string $\sigma$, access to a random oracle $\mathcal{H}$ and a language member $u$ and a witness $r$ with $(u, r) \in \mathsf{R}_\sigma$, outputs a proof $\pi$.

$0/1 \leftarrow \mathsf{V}^{\mathcal{H}}(\sigma, u, \pi)$ : is the PPT verification algorithm of $\Sigma$ that, given a common reference string $\sigma$, access to a random oracle $\mathcal{H}$ and a language member $u$ and proof $\pi$, outputs 0 (invalid) or 1 (valid).

**Definition 6.** *Given a relaxed NIZK $\Sigma = (\mathsf{K}, \mathsf{P}, \mathsf{V})$ for relations $\mathsf{R}_\sigma, \mathsf{R}'_\sigma$, we define the following properties:*

**Completeness.** $\Sigma$ *satisfies* completeness *if, for $\sigma \leftarrow \mathsf{K}(1^\lambda)$ and $\mathcal{H} \leftarrow \mathcal{HS}p$ from the set of random oracles $\mathcal{HS}p$ and all $(u, r) \in \mathsf{R}_\sigma$, if $\pi \leftarrow \mathsf{P}^{\mathcal{H}}(\sigma, u, r)$ then $\mathsf{V}(\sigma, u, \pi) = 1$ with probability $1 - \mathrm{negl}(\lambda)$.*

**Soundness.** $\Sigma$ *satisfies* soundness *if, for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[(u^* \notin L'_\sigma) \wedge \mathsf{V}^{\mathcal{H}}(u^*, \pi^*) = 1 \ : \ \begin{array}{l} \sigma \leftarrow \mathsf{K}(1^\lambda),\ \mathcal{H} \leftarrow \mathcal{HS}p \\ (u^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{H}}(\sigma). \end{array}\right] \leq \mathrm{negl}(\lambda).$$

**Zero-Knowledge.** $\Sigma$ *satisfies* zero-knowledge *if there exists a PPT simulator $\mathcal{S}$ such that any PPT adversary $\mathcal{A}$ wins the following game with probability $1/2 + \mathrm{negl}(\lambda)$:*

1. $b \xleftarrow{\$} \{0, 1\}$.
2. *If* $b = 0$: $\sigma \leftarrow \mathsf{K}(1^\lambda)$, $(u, r, \mathrm{st}') \leftarrow \mathcal{A}^{\mathcal{H}}(\mathrm{st}, \sigma)$, $\pi \leftarrow \mathsf{P}^{\mathcal{H}}(\sigma, u, r)$, $b' \leftarrow \mathcal{A}^{\mathcal{H}}(\mathrm{st}', \pi)$.
3. *If* $b = 1$: $(\sigma, \mathrm{st}_S) \leftarrow \mathcal{S}(0, 1^\lambda)$, $(u, r, \mathrm{st}') \leftarrow \mathcal{A}^{\mathcal{S}(1, \mathrm{st}_S, \cdot)}(\mathrm{st}, \sigma)$, $\pi \leftarrow \mathcal{S}(2, \mathrm{st}_S, u)$, $b' \leftarrow \mathcal{A}^{\mathcal{S}(1, \mathrm{st}_S, \cdot)}(\mathrm{st}', \pi)$.
4. *$\mathcal{A}$ wins the game if $b' = b$.*

Our VDPC construction in this paper is based on a NIZK obtained using the Fiat-Shamir transform [15] applied to an interactive Sigma Zero-Knowledge proof protocol satisfying the properties in Definition 7. We use the same notations for relations as for Definition 6.

**Definition 7 ( [5, Definition 2.5]).** *For a relation $\mathsf{R}_\sigma$ and its relaxed counterpart $\mathsf{R}'_\sigma$ with $\mathsf{R}_\sigma \subseteq \mathsf{R}'_\sigma$ (parameterised by a common reference string $\sigma$), an interactive Sigma protocol $\Sigma_{\mathsf{I}} = (\mathsf{K}_{\mathsf{I}}, \mathsf{P}_{\mathsf{I}}, \mathsf{V}_{\mathsf{I}})$ with a challenge space $\mathcal{C}$ and public-private inputs $(u, r)$, consists of a common reference string generation algorithm*

$\mathsf{K}_\mathsf{I}$ *and a pair of interactive algorithms* $(\mathsf{P}_\mathsf{I}, \mathsf{V}_\mathsf{I})$ *such that the following properties are satisfied.*

- ***Completeness:*** *For* $\sigma \leftarrow \mathsf{K}_\mathsf{I}(1^\lambda)$ *and all* $(u, r) \in \mathsf{R}_\sigma$*, an interaction between an honest prover* $\mathsf{P}_\mathsf{I}(\sigma, u, r)$ *and an honest verifier* $\mathsf{V}_\mathsf{I}(\sigma, u)$ *is accepted with probability* $1 - \mathrm{negl}(\lambda)$*. Protocol transcripts have the form* $\mathsf{tr} = (u, w, x, z)$*, where* $w$ *is the first* commitment *message sent from* $\mathsf{P}_\mathsf{I}$ *to* $\mathsf{V}_\mathsf{I}$*,* $x \leftarrow \mathcal{C}$ *is the second random* challenge *message sent from* $\mathsf{V}_\mathsf{I}$ *to* $\mathsf{P}_\mathsf{I}$*, and* $z$ *is the third* response *message sent from* $\mathsf{P}_\mathsf{I}$ *to* $\mathsf{V}_\mathsf{I}$*.*
- ***Special soundness:*** *There exists an efficient PPT* extractor $\mathcal{E}$ *such that, given* $\sigma \leftarrow \mathsf{K}_\mathsf{I}(1^\lambda)$ *and two accepting protocol transcripts* $\mathsf{tr} = (u, w, x, z)$ *and* $\mathsf{tr}' = (u, w, x', z')$ *with* $x \neq x'$*, computes an extracted witness* $\bar{r}$ *satisfying* $(u, \bar{r}) \in \mathsf{R}'_\sigma$*, with probability* $1 - \mathrm{negl}(\lambda)$ *(over the choice of* $\sigma$*).*
- ***Honest-verifier zero-knowledge (HVZK):*** *There exists an efficient PPT* simulator $\mathcal{S}$ *such that any PPT adversary* $\mathcal{A}$ *wins the following game with probability* $1/2 + \mathrm{negl}(\lambda)$*:*
  1. $b \xleftarrow{\$} \{0, 1\}$.
  2. *If* $b = 0$*:* $\sigma \leftarrow \mathsf{K}_\mathsf{I}(1^\lambda)$*,* $(u, r, \mathrm{st}) \leftarrow \mathcal{A}(\sigma)$*,* $\pi \leftarrow (\mathsf{P}_\mathsf{I}(\sigma, u, r), \mathsf{V}_\mathsf{I}(\sigma, u))$*,* $b' \leftarrow \mathcal{A}(\mathrm{st}, \pi)$.
  3. *If* $b = 1$*:* $(\sigma, \mathrm{st}_S) \leftarrow \mathcal{S}(0, 1^\lambda)$*,* $(u, r, \mathrm{st}') \leftarrow \mathcal{A}(\mathrm{st}, \sigma)$*,* $\pi \leftarrow \mathcal{S}(1, \mathrm{st}_S, u)$*,* $b' \leftarrow \mathcal{A}(\mathrm{st}', \pi)$.
  4. $\mathcal{A}$ *wins the game if* $b' = b$*.*

The Fiat-Shamir transformation [15] takes an interactive Sigma protocol $\Sigma_\mathsf{I}$ for relations $\mathsf{R}, \mathsf{R}'$ and transforms it into a NIZK using a random oracle $\mathcal{H}$ by letting the prove algorithm compute the verifier's challenge from the common reference string $\sigma$, public input $u$, and commitment message $w$, setting $x = \mathcal{H}(\sigma, u, w)$.

### A.2 Commitment Schemes

We now recall the standard definitions of correctness, hiding and binding properties of a commitment scheme.

**Definition 8.** *Given a commitment scheme* $\mathsf{C} = (\mathsf{CKeygen}, \mathsf{Commit}, \mathsf{COpen})$*, we define the following properties:*

**Correctness.** $\mathsf{C}$ *satisfies* correctness *if, for* $pp = (ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}(1^\lambda)$ *and all* $\mathsf{m} \in \mathcal{M}$*, if* $(C, \mathsf{o}) \leftarrow \mathsf{Commit}(\mathsf{m})$*, then* $\mathsf{COpen}(C, \mathsf{o}) = 1$.

**Hiding.** $\mathsf{C}$ *is* computationally hiding *if no PPT adversary* $\mathcal{A}$ *can win the following game with non-negligible probability:*

1. $pp = (ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}(1^\lambda)$,
2. $(\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(pp)$,
3. $(C, \mathsf{o}) \leftarrow \mathsf{Commit}(\mathsf{m}_b)$ *for* $b \xleftarrow{\$} \{0, 1\}$,
4. $b' \leftarrow \mathcal{A}(C)$.

$\mathcal{A}$ *wins the game if* $b' = b$*.*

**Binding.** $\mathsf{C}$ *is* computationally binding *if, for* $pp = (ck, \mathcal{M}, \mathcal{R}) \leftarrow \mathsf{CKeygen}(1^\lambda)$*, the following probability (over the randomness of PPT* $\mathcal{A}$ *and* $\mathsf{CKeygen}$*) is negligible:*

$$\Pr[(C, \mathsf{o}, \mathsf{o}') \leftarrow \mathcal{A}(pp) : \mathsf{m}(\mathsf{o}) \neq \mathsf{m}(\mathsf{o}') \wedge \mathsf{COpen}(C, \mathsf{o}) = \mathsf{COpen}(C, \mathsf{o}') = 1].$$

## B Proof of Lemma 1

**Correctness.** Follows from $\|\boldsymbol{r}\|^2 \leq \mathcal{B}^2 md$, $\|y\boldsymbol{m}\|^2 \leq (\gamma_y \alpha d)^2 v_1$ and $\|\boldsymbol{u}\|^2 \leq \beta^2 v_2 d$.

**Hiding.** It's easy to see that if M-LWE$_{n-1,m+v_1+v_2,q,\mathcal{B}}$ is hard, then the commitment key $ck^{\mathsf{td}}$ output by $\mathsf{CAddTd}(ck)$ is computationally indistinguishable from a commitment key output by $\mathsf{CKeygen}$ given in Section 5.1, i.e. the *key indistinguishability* property is satisfied. From here, the hiding property of the 'trapdoored' HMC follows by Prop. 1 and the hiding property of the standard HMC (see [14, Lemma 2.3] and [8, Lemma 3.4]).

**Trapdoor Binding.** The intuition behind the binding proof is simply to ignore the trapdoor rows in the commitment key and the last coordinate in the commitment.

More formally, assume that M-SIS$_{n-1,m+v_1+v_2,q,2\gamma_{\text{com}}}$ is hard. Then, we have $q > 2\gamma_{\text{com}}$ as otherwise $(q, 0, \ldots, 0)$ yields a trivial M-SIS solution. Now let $(y, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{r})$ and $(y, \boldsymbol{m}', \boldsymbol{u}', \boldsymbol{r}')$ be a binding collision pair w.r.t. the same relaxation factor $y \neq 0$ such that $yC = \boldsymbol{G}^{\mathsf{td}} \cdot (\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})^\top = \boldsymbol{G}^{\mathsf{td}} \cdot (\boldsymbol{r}', y\boldsymbol{m}', \boldsymbol{u}')^\top$ and $\|(\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})\|, \|(\boldsymbol{r}', y\boldsymbol{m}', \boldsymbol{u}')\| \leq \gamma_{\text{com}}$ with $\boldsymbol{G}^{\mathsf{td}} \leftarrow \mathsf{CAddTd}(\boldsymbol{G})$ and $\boldsymbol{G} \leftarrow \mathsf{CKeygen}(1^\lambda)$. Then, we have $\boldsymbol{G'}^{\mathsf{td}} \cdot (\boldsymbol{r} - \boldsymbol{r}', y(\boldsymbol{m} - \boldsymbol{m}'), \boldsymbol{u} - \boldsymbol{u}')^\top = \boldsymbol{0}$ where $\boldsymbol{G'}^{\mathsf{td}}$ is the submatrix of $\boldsymbol{G}^{\mathsf{td}}$ with the last row removed. Note in fact that $\boldsymbol{G'}^{\mathsf{td}}$ is proper M-SIS matrix independent of the trapdoor. By the M-SIS$_{n-1,m+v_1+v_2,q,2\gamma_{\text{com}}}$ assumption, we get $(\boldsymbol{r} - \boldsymbol{r}', y(\boldsymbol{m} - \boldsymbol{m}'), \boldsymbol{u} - \boldsymbol{u}') = \boldsymbol{0}$ over $R_q$, which implies

$$y(\boldsymbol{m} - \boldsymbol{m}') = \boldsymbol{0} \text{ over } R_q. \tag{18}$$

Now, by the assumption on the norm of the openings, it is clear that $\|y(\boldsymbol{m} - \boldsymbol{m}')\|_\infty \leq 2\gamma_{\text{com}} < q$. Hence, (18) holds over $R = \mathbb{Z}[X]/(X^d + 1)$ (i.e., without mod $q$). Since $y \neq 0$ and $R$ has no zero divisors, it must be the case that $\boldsymbol{m} = \boldsymbol{m}'$, which concludes the binding proof w.r.t. same relaxation factor.

In the general trapdoor-binding case with *different* relaxation factors $y' \neq y$, we cross-multiply the two opening equations $yC = \boldsymbol{G}^{\mathsf{td}} \cdot (\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})^\top, y'C = \boldsymbol{G}^{\mathsf{td}} \cdot (\boldsymbol{r}', y'\boldsymbol{m}', \boldsymbol{u}')^\top$ by $y'$ and $y$ respectively before substracting them to get $\boldsymbol{G'}^{\mathsf{td}} \cdot (y'\boldsymbol{r} - y\boldsymbol{r}', y'y(\boldsymbol{m} - \boldsymbol{m}'), y'\boldsymbol{u} - y\boldsymbol{u}')^\top = \boldsymbol{0}$ and proceed similarly to above, using the bound $\|y'(\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})\| \leq \sqrt{d}\|y'\|\|(\boldsymbol{r}, y\boldsymbol{m}, \boldsymbol{u})\| \leq \sqrt{d}\gamma_Y \cdot \gamma_{\text{com}}$. □

## C Decryptable Unbounded-Message Commitment (UMC)

In this section, we describe that the UMC commitment [3] can be easily made *decryptable* when the committed message is *short*. In particular, the verifiable decryptable UMC commitment can be seen as the verifiable Regev encryption studied in [22].

For ease of presentation, we assume that a single ring element $m \in R$ is committed. The result easily extends to the case when a message *vector* is committed. Let $(\boldsymbol{A}, \boldsymbol{b})$ be a random matrix-vector pair over $R_q$ (with appropriate

dimensions to ensure hiding and binding properties) and $\boldsymbol{r}$ be a randomness sampled appropriately. Then, a UMC commitment $(\boldsymbol{t}, v)$ to $m$ is computed as follows.

$$\begin{aligned} \boldsymbol{t} &= \boldsymbol{A}\boldsymbol{r} \pmod{q}, \\ v &= \langle \boldsymbol{b}, \boldsymbol{r} \rangle + m \pmod{q}. \end{aligned}$$

Here, we take the commitment matrices to be uniformly random as in [9], which does not affect the security properties and the commitment size (in [3], they are more structured). Now, assume that $\|m\|_\infty < p$ and $\boldsymbol{b} = \boldsymbol{s}^\top \boldsymbol{A} + \boldsymbol{e}$ for some secret vector $\boldsymbol{s}$ and *short* error $\boldsymbol{e}$. Now, modify the commitment computation as follows.

$$\begin{aligned} \boldsymbol{t} &= p\boldsymbol{A}\boldsymbol{r} \pmod{q}, \\ v &= p\langle \boldsymbol{b}, \boldsymbol{r} \rangle + m \pmod{q}. \end{aligned}$$

With this setting, it is easy to observe that

$$\begin{aligned} v - \boldsymbol{s}^\top \boldsymbol{t} &= (p\langle \boldsymbol{b}, \boldsymbol{r} \rangle + m) - \boldsymbol{s}^\top p\boldsymbol{A}\boldsymbol{r} = (p\boldsymbol{s}^\top \boldsymbol{A}\boldsymbol{r} + p\langle \boldsymbol{e}, \boldsymbol{r} \rangle + m) - \boldsymbol{s}^\top p\boldsymbol{A}\boldsymbol{r} \\ &= p\langle \boldsymbol{e}, \boldsymbol{r} \rangle + m. \end{aligned}$$

Hence, assuming $\|p\langle \boldsymbol{e}, \boldsymbol{r} \rangle + m\|_\infty < q/2$, we get

$$(v - \boldsymbol{s}^\top \boldsymbol{t} \bmod q) \bmod p = m. \tag{19}$$

Of course, the above is true for honestly-generated commitments. When the commitment is accompanied by a relaxed opening proof as in (1), we have for some $(m', \boldsymbol{r}', y)$ that

$$\begin{aligned} y\boldsymbol{t} &= p\boldsymbol{A}\boldsymbol{r}' \pmod{q}, \\ yv &= p\langle \boldsymbol{b}, \boldsymbol{r}' \rangle + ym' \pmod{q}. \end{aligned}$$

If the opening proof additionally proves that $m' \in \mathcal{M}$ such that $\mathcal{M} = \{x \in R : \|x\|_\infty < p/w\}$, where $\|y\|_1 \leq w$ for any relaxation factor $y$, then the decryption can proceed analogous to (19) as long as a "good" relaxation factor $y'$ is hit in decryption algorithm (and $q$ is sufficiently large). This procedure is essentially the same as Algorithm 5 in [22]. We also note that two moduli can be used in the "top" (i.e., $\boldsymbol{t}$) and "bottom" (i.e., $v$) parts of the commitment for improved efficiency.

## D Formal Definitions for RingCT-like Protocols

A RingCT protocol [26] allows a *spender* to conduct a transaction on blockchain while hiding sensitive information such as transaction amount and spender identity. As such important information is hidden from the outside world, the spender has to do additional work by proving in zero-knowledge that her transaction satisfies certain validity requirements. For example, she must prove that she is not

**Table 2.** 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 |
| $\mathsf{R}_{\text{in}}$ | 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 $\mathsf{ask} = (\mathsf{sk}, \mathsf{cnk}, \mathsf{amt})$ with a secret key, coin key & amount |
| $\mathsf{A}_{\text{in}}$ | set of all input accounts arranged as a $M \times N$ matrix where the $i$-th row contains $\mathsf{R}_{\text{in}}[i]$ |
| $\mathsf{PK}_{\text{out}}$ | set of output public keys with $\lvert\mathsf{PK}_{\text{out}}\rvert = S$ |
| $\mathsf{CN}_{\text{out}}$ | set of output coins with $\lvert\mathsf{PK}_{\text{out}}\rvert = S$ |
| $\mathsf{Amt}_{\text{out}}$ | set of output amounts with $\lvert\mathsf{Amt}_{\text{out}}\rvert = S$ |
| $\mathsf{CK}_{\text{out}}$ | set of output coin keys with $\lvert\mathsf{CK}_{\text{out}}\rvert = S$ |
| $\mathsf{A}_{\text{out}}$ | set of output accounts with $\lvert\mathsf{A}_{\text{out}}\rvert = S$ |
| $\Pi$ | the proof output |
| $\mathsf{SN}$ | set of serial numbers |
| $\mathsf{tx}$ | a transaction $\mathsf{tx} = (\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \Pi, \mathsf{SN})$ |
| $\mathsf{V}$ | set of all valid amounts |
| $\mathsf{TD}$ | set of all trapdoors in the system |
| $\mathcal{N}$ | number of auditors in the system |

double-spending or trying to spend a negative amount. The protocol makes use of various cryptographic tools such as a (linkable) ring signature, zero-knowledge proofs and commitment schemes.

In this section, we recall the formal model for RingCT-like protocols from [14] and also introduce a formal auditability feature. We use $S[i]$ to denote either the $i$-th element of an ordered set $S$ or the $i$-th bit of an integer $S$. For an algorithm $F$, we use $O \leftarrow F(S)$ to indicate that $F$ is run on all the elements of a set $S$ such that $o_i \leftarrow F(s_i)$ for all $s_i \in S$ and $O = \{o_0, o_1, \ldots\}$. As in [14], we fix the notations in Table 2, where we additionally introduce the last two notations $\mathsf{TD}$ and $\mathcal{N}$. The blockchain state $\mathbb{B}$ contains the lists of all registered accounts $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$ and all verified transactions. The syntax of the algorithms defining a RingCT protocol is as follows.

- $pp \leftarrow \mathbf{Setup}(1^\lambda)$ : given the security parameter $\lambda$, output the system parameters $pp$. W assume $pp$ is an implicit input to all the remaining functions.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{KeyGen}()$ : output a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{sn} \leftarrow \mathbf{SerialGen}(\mathsf{sk})$ : on input a secret key $\mathsf{sk}$, output a serial number $\mathsf{sn}$ associated to $\mathsf{sk}$.
- $(\mathsf{cn}, \mathsf{cnk})/\ \perp \leftarrow \mathbf{Mint}(\mathsf{amt})$ : on input an amount $\mathsf{amt}$, if $\mathsf{amt} \in \mathsf{V}$, output a coin $\mathsf{cn}$ and its coin key $\mathsf{cnk}$. Otherwise, output $\perp$. If $\mathsf{cnk}$ is given as an input, then **Mint** computes a deterministic function such that $\mathsf{cn} = \mathbf{Mint}(\mathsf{amt}, \mathsf{cnk})$.
- $(\mathsf{act}, \mathbb{B}) \leftarrow \mathbf{AccountGen}(\mathsf{pk}, \mathsf{cn}, \mathbb{B})$ : on input a public key $\mathsf{pk}$ and a coin $\mathsf{cn}$, register an account $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$ to the blockchain state $\mathbb{B}$. Output $\mathsf{act}$ and updated state $\mathbb{B}$.

**Table 3.** Structure of the list $\mathcal{L}$ used in the security model.

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

- $0/1 \leftarrow \mathbf{CheckAct}(\mathsf{pk}, \mathsf{cn}, \mathbb{B})$ : 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.
- $(\mathsf{tx}, \mathsf{CK}_{\text{out}}) \leftarrow \mathbf{Spend}(\mathsf{A}_{\text{in}}, \mathsf{R}_{\text{in}}, \mathsf{K}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$ : 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 2, mint output coins as $(\mathsf{CN}_{\text{out}}, \mathsf{CK}_{\text{out}}) \leftarrow \mathbf{Mint}(\mathsf{Amt}_{\text{out}})$. Generate the serial numbers by running $\mathsf{SN} \leftarrow \mathbf{SerialGen}(\mathsf{SK}_{\text{in}})$ and a proof $\Pi$. Output $(\mathsf{tx}, \mathsf{CK}_{\text{out}}) = ((\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \Pi, \mathsf{SN}), \mathsf{CK}_{\text{out}})$.[10]
- $0/1 \leftarrow \mathbf{IsSpent}(\mathsf{SN}, \mathbb{B})$ : 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.
- $\emptyset/(\mathsf{A}_{\text{out}}, \mathbb{B}) \leftarrow \mathbf{Verify}(\mathsf{tx}, \mathbb{B})$ : on input a transaction tx as in Table 2, if $\mathbf{IsSpent}(\mathsf{SN}, \mathbb{B}) = 1$ or $\mathbf{CheckAct}(\mathsf{A}_{\text{in}}, \mathbb{B}) = 0$, output $\emptyset$. Check the proof $\Pi$ and output $\emptyset$ if not valid. Otherwise, add tx to $\mathbb{B}$ and run $(\mathsf{A}_{\text{out}}, \mathbb{B}) \leftarrow \mathbf{AccountGen}(\mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \mathbb{B})$. Output $\mathsf{A}_{\text{out}} \neq \emptyset$ and updated $\mathbb{B}$.
- $-1/\perp/\mathsf{R}_{\text{in}} \leftarrow \mathbf{Audit}(\mathsf{tx}, \mathsf{td}, \mathbb{B})$ : on input a transaction tx as in Table 2 and a trapdoor td, if $\mathbf{Verify}(\mathsf{tx}, \mathbb{B}) = \emptyset$ or td is not the correct trapdoor for tx, return $-1$. Otherwise, try to recover the set $\mathsf{R}_{\text{in}}$ of real spent accounts for tx using td. If successful, return $\mathsf{R}_{\text{in}}$. Otherwise, return $\perp$.

### D.1 Security Definitions

The list $\mathcal{L}$ in Table 3 represents a database, where any of the following can act as a unique identifier of a row: a public key, a secret key, a serial number, a coin or a coin key. Retrieval of a row in $\mathcal{L}$ is denoted, for example, by $\mathcal{L}[\mathsf{pk}]$ for some public key pk. Then, $\mathcal{L}[\mathsf{pk}].\mathsf{amt}$ denotes the amount associated with the public key pk. IsCrpt denotes the "is corrupted" tag.

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

- $\textsc{PkGen}(i)$ : on the $i$-th query, run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathbf{KeyGen}()$, $\mathsf{sn} \leftarrow \mathbf{SerialGen}(\mathsf{sk}_i)$ and output $\mathsf{pk}_i$. Add $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{sn})$ to $\mathcal{L}$ where IsCrpt tag is set to zero and the remaining fields are left empty.
- $\textsc{Mint}(\mathsf{amt})$ : run $(\mathsf{cn}, \mathsf{cnk}) \leftarrow \mathbf{Mint}(\mathsf{amt})$, and output cn.
- $\textsc{ActGen}(\mathsf{pk}, \mathsf{amt}, \mathbb{B})$ : run $(\mathsf{act}, \mathbb{B}) \leftarrow \mathbf{AccountGen}(\mathsf{pk}, \mathsf{cn}, \mathbb{B})$ for $(\mathsf{cn}, \mathsf{cnk}) \leftarrow \mathbf{Mint}(\mathsf{amt})$. Insert $(\mathsf{cn}, \mathsf{cnk}, \mathsf{amt})$ to $\mathcal{L}[\mathsf{pk}]$ and output $(\mathsf{act}, \mathbb{B})$.
- $\textsc{Corrupt}(\mathsf{act})$ : 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 $\mathcal{L}[\mathsf{pk}].\mathsf{sk}$, $\mathcal{L}[\mathsf{pk}].\mathsf{cnk}$ 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, $\mathcal{L}[\mathsf{cn}].\mathsf{cnk}$ and $\mathcal{L}[\mathsf{cn}].\mathsf{amt}$ are returned.

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

– $\textsc{Spend}(\mathsf{A}_{\text{in}}, \mathsf{R}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$ : Retrieve from $\mathcal{L}$ all account secret keys $\mathsf{K}_{\text{in}}$ associated to $\mathsf{R}_{\text{in}}$. Run $(\mathsf{tx}, \mathsf{CK}_{\text{out}}) \leftarrow \mathbf{Spend}(\mathsf{A}_{\text{in}}, \mathsf{R}_{\text{in}}, \mathsf{K}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$ and $B \leftarrow \mathbf{Verify}(\mathsf{tx}, \mathbb{B})$. If $B = \emptyset$ (i.e., the verification fails), return $\perp$. Otherwise, return $\mathsf{tx}$ 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.

We denote the set of all oracles defined above together with the random oracle by $\textsc{Orc}$. There are two flavours of properties for RingCT with respect to the positioning of the accounts in $\mathsf{R}_{\text{in}}$ inside $\mathsf{A}_{\text{in}}$: 1) *with shuffling* and 2) *without shuffling*. In the latter case, all the accounts in $\mathsf{R}_{\text{in}}$ are restricted to be in the same column. The definitions are given for the former case.

Note that we do not introduce a tracing oracle since, in the security definitions, we will simply give the adversary access to all the trapdoors when possible.

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

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

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

where $\mathsf{cn}_i = \mathbf{Mint}(\mathsf{amt}_i, \mathsf{cnk}_i)$ for some $\mathsf{cnk}_i$'s in the domain of coin keys, $\mathsf{act}_i \leftarrow \mathbf{AccountGen}(\mathsf{pk}_i, \mathsf{cn}_i)$ for $i = 0, \ldots, M-1$, $\mathsf{R}_{\text{in}} = \{\mathsf{act}_0, \ldots, \mathsf{act}_{M-1}\}$, $\mathsf{Amt}_{\text{out}} = \{\mathsf{amt}_{\text{out},0}, \ldots, \mathsf{amt}_{\text{out},S-1}\}$, $\mathsf{A}_{\text{in}}$ and $\mathsf{tx}$ are as in Table 2, and $\mathsf{K}_{\text{in}} = \{(\mathsf{sk}_0, \mathsf{cnk}_0, \mathsf{amt}_0), \ldots, (\mathsf{sk}_{M-1}, \mathsf{cnk}_{M-1}, \mathsf{amt}_{M-1})\}$. If $\epsilon = 0$, then the protocol is said to be *perfectly correct*. If $\epsilon = \text{negl}(\lambda)$, then it is said to be *statistically correct*.

**Anonymity.** Informally, anonymity means that the real spender's accounts are hidden among the uncorrupted accounts as long as there are two or more sets of uncorrupted input accounts that can be successfully spent.

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

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

where `Exp:Anonymity` is defined as follows.

1. Generate $\mathcal{N}$ auditor trapdoor/key pairs: $(\mathsf{TD}^{\mathcal{J}}, (\boldsymbol{t}_0^{\mathcal{J}}, \boldsymbol{t}_1^{\mathcal{J}}, \boldsymbol{t}_2^{\mathcal{J}})) \leftarrow \mathbf{TdRowGen}(pp)$ for $\mathcal{J} = 1, \ldots, \mathcal{N}$. Let $\boldsymbol{t} := ((\boldsymbol{t}_0^1, \boldsymbol{t}_1^1, \boldsymbol{t}_2^1), \ldots, (\boldsymbol{t}_0^{\mathcal{N}}, \boldsymbol{t}_1^{\mathcal{N}}, \boldsymbol{t}_2^{\mathcal{N}}))$ denote the sequence of all auditor public keys.

2. $(\mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}}, \mathsf{R}^0_{\text{in}}, \mathsf{R}^1_{\text{in}}, \mathsf{st}) \leftarrow \mathcal{A}^{\text{ORC}}(pp, \boldsymbol{t})$ : $\mathcal{A}$ 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
   - $\mathsf{st}$ 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 2,
   - $\mathsf{R}^0_{\text{in}}, \mathsf{R}^1_{\text{in}} \subset \mathsf{A}_{\text{in}}$ such that both $\mathsf{R}^0_{\text{in}}$ and $\mathsf{R}^1_{\text{in}}$ contain exactly one account from each row of $\mathsf{A}_{\text{in}}$.
3. $(\mathsf{tx}_i, \mathsf{CK}^i_{\text{out}}) \leftarrow \mathbf{Spend}(\mathsf{A}_{\text{in}}, \mathsf{R}^i_{\text{in}}, \mathsf{K}^i_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{Amt}_{\text{out}})$ for $i = 0, 1$ : Both sets $\mathsf{R}^0_{\text{in}}$ and $\mathsf{R}^1_{\text{in}}$ of real input accounts are spent with the arguments specified by $\mathcal{A}$ where
   - $\mathsf{K}^i_{\text{in}}$ is the set of account secret keys of the accounts in $\mathsf{R}^i_{\text{in}}$ retrieved from $\mathcal{L}$ for $i = 0, 1$.

   If $\mathbf{Verify}(\mathsf{tx}_i, \mathbb{B}) = \emptyset$ for some $i \in \{0, 1\}$, then set $\mathsf{tx}_0 = \mathsf{tx}_1 = \perp$.
4. $b \leftarrow \{0, 1\}$.
5. $b' \leftarrow \mathcal{A}^{\text{ORC}}(\mathsf{tx}_b, \mathsf{CK}^b_{\text{out}}, \mathsf{Amt}^0_{\text{in}}, \mathsf{Amt}^1_{\text{in}}, \mathsf{st})$ : $\mathcal{A}$ is given access to all the oracles, the state $\mathsf{st}$, one of the **Spend** outputs, and the input amounts in $\mathsf{K}^0_{\text{in}}$ and $\mathsf{K}^1_{\text{in}}$. 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 $\mathsf{R}^0_{\text{in}}$ and $\mathsf{R}^1_{\text{in}}$ are generated by PkGen and Mint, respectively, and all accounts in $\mathsf{R}^0_{\text{in}}$ and $\mathsf{R}^1_{\text{in}}$ are generated by ActGen,
- all public keys in $\mathsf{PK}_{\text{out}}$ are generated by PkGen,
- $\mathsf{tx}_0 \neq \perp$ and $\mathsf{tx}_1 \neq \perp$,
- no account (including its public key and coin) in $\mathsf{R}^0_{\text{in}}$ or $\mathsf{R}^1_{\text{in}}$ has been corrupted (i.e., queried to Corrupt),
- $(\cdot, \mathsf{R}^i_{\text{in}} \cdot, \cdot)$ has never been queried to Spend for $i = 0, 1$,
- $b' = b$.

In the anonymity definition, the adversary is not allowed to get access to the trapdoors or a tracing oracle. Clearly, there is no anonymity guarantee against the parties (e.g., the auditors) who have access to the correct trapdoor(s). Moreover, to have a stronger anonymity property where the adversary can make queries to a tracing oracle, one requires a CCA-secure encryption, which we avoid here to keep the structure of MatRiCT-Au similar to that of MatRiCT.

**Balance.** Informally, balance means that no adversary is able to spend a set $A$ of accounts under his control such that the sum of output amounts is more than the sum of the amounts in $A$.

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

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

where `Exp:Balance` is defined as follows.

1. Generate $\mathcal{N}$ auditor trapdoor/key pairs: $(\mathsf{TD}^{\mathcal{J}}, (\boldsymbol{t}_0^{\mathcal{J}}, \boldsymbol{t}_1^{\mathcal{J}}, \boldsymbol{t}_2^{\mathcal{J}})) \leftarrow \mathbf{TdRowGen}(pp)$ for $\mathcal{J} = 1, \ldots, \mathcal{N}$. Let $\mathsf{TD} := (\mathsf{TD}^1, \ldots, \mathsf{TD}^{\mathcal{N}})$ denote the sequence of all auditor trapdoors and $\boldsymbol{t} := ((\boldsymbol{t}_0^1, \boldsymbol{t}_1^1, \boldsymbol{t}_2^1), \ldots, (\boldsymbol{t}_0^{\mathcal{N}}, \boldsymbol{t}_1^{\mathcal{N}}, \boldsymbol{t}_2^{\mathcal{N}}))$ denote the sequence of all auditor public keys.
2. $(\mathsf{tx}_1, \mathsf{Amt}_{\text{out}}^1, \mathsf{CK}_{\text{out}}^1), \cdots, (\mathsf{tx}_t, \mathsf{Amt}_{\text{out}}^t, \mathsf{CK}_{\text{out}}^t) \leftarrow \mathcal{A}^{\text{ORC}}(pp, \mathsf{TD}, \boldsymbol{t})$ : The adversary $\mathcal{A}$ is given access to all the oracles ORC together with $pp$, $\mathsf{TD}$ and $\boldsymbol{t}$, and outputs a set of $t$ transactions where
   - $\mathsf{tx}_i = (\mathsf{A}_{\text{in}}^i, \mathsf{PK}_{\text{out}}^i, \mathsf{CN}_{\text{out}}^i, \Pi_i, \mathsf{SN}_i)$ for $i = 1, \ldots, t$,
   - $\mathsf{Amt}_{\text{out}}^i$'s and $\mathsf{CK}_{\text{out}}^i$'s are sets of output amounts and coin keys, respectively, for *uncorrupted* output public keys with $|\mathsf{CK}_{\text{out}}^i| = |\mathsf{Amt}_{\text{out}}^i| \leq |\mathsf{PK}_{\text{out}}^i| = |\mathsf{CN}_{\text{out}}^i|$ for all $i \in \{1, \ldots, t\}$.
3. $B_i \leftarrow \mathbf{Verify}(\mathsf{tx}_i, \mathbb{B})$ for $i = 1, \ldots, t$.

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

- for all $i \in \{1, \ldots, t\}$, all public keys and coins in $\mathsf{A}_{\text{in}}^i$ are generated by PKGEN and MINT, respectively, and all accounts in $\mathsf{A}_{\text{in}}^i$ are generated by ACTGEN,
- $\bigcap_{i=1}^t \mathsf{SN}_i = \emptyset$,
- $B_i \neq \emptyset$ for all $i = 1, \ldots, t$,
- there exists a $j^* \in [1, t]$ such that $\sum_{i=0}^{S'-1} \mathsf{Amt}_{\text{out}}^{j^*}[i] > \sum_{i=0}^{M-1} \mathsf{amt}_{\text{in},i}$ where $S' = |\mathsf{Amt}_{\text{out}}^{j^*}|$, $M = |\mathsf{SN}_{j^*}|$, $\mathsf{amt}_{\text{in},i} = \mathcal{L}[\mathsf{sn}_i].\mathsf{amt}$ for all $\mathsf{sn}_i \in \mathsf{SN}_{j^*}$ 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 $i \in [1, t]$ and $0 \leq j < |\mathsf{PK}_{\text{out}}^i|$, if $\mathcal{L}[\mathsf{pk}_{i,j}].\mathsf{IsCrpt} = 0$ for $\mathsf{pk}_{i,j} = \mathsf{PK}_{\text{out}}^i[j]$, then $\mathsf{CK}_{\text{out}}^i[j] = \mathcal{L}[\mathsf{pk}_{i,j}].\mathsf{cnk}$, $\mathsf{Amt}_{\text{out}}^i[j] = \mathcal{L}[\mathsf{pk}_{i,j}].\mathsf{amt}$ and $\mathsf{CN}_{\text{out}}^i[j] = \mathbf{Mint}(\mathsf{Amt}_{\text{out}}^i[j], \mathsf{CK}_{\text{out}}^i[j])$.[11] That is, for all *uncorrupted* output public keys, the corresponding output coin key, output amount and output coin provided by the adversary are correct.

Contrary to MatRiCT, our balance definition here is stronger in the sense that the adversary is additionally given access to all the trapdoors. Even with this change, we show that the balance proof of MatRiCT can be easily adapted to handle the stronger case thanks to the way trapdoors are used in MatRiCT-Au.

**Auditability.** Our auditability definition is similar to the traceability of group signatures [4]. Informally, auditability means that no adversary, who has access to all the trapdoors and can corrupt arbitrary number of accounts in a transaction, can create a *valid auditable* transaction that fails to trace or traces back to a set of real input accounts where there is an uncorrupted account.

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

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

where `Exp:Auditability` is defined as follows.

[11] Without loss of generality, the indices for corrupted public keys are assumed to be the last ones so that the indexing matches.

1. Generate $\mathcal{N}$ auditor trapdoor/key pairs: $(\mathsf{TD}^{\mathcal{J}}, (\boldsymbol{t}_0^{\mathcal{J}}, \boldsymbol{t}_1^{\mathcal{J}}, \boldsymbol{t}_2^{\mathcal{J}})) \leftarrow \mathbf{TdRowGen}(pp)$ for $\mathcal{J} = 1, \ldots, \mathcal{N}$. Let $\mathsf{TD} := (\mathsf{TD}^1, \ldots, \mathsf{TD}^{\mathcal{N}})$ denote the sequence of all auditor trapdoors and $\boldsymbol{t} := ((\boldsymbol{t}_0^1, \boldsymbol{t}_1^1, \boldsymbol{t}_2^1), \ldots, (\boldsymbol{t}_0^{\mathcal{N}}, \boldsymbol{t}_1^{\mathcal{N}}, \boldsymbol{t}_2^{\mathcal{N}}))$ denote the sequence of all auditor public keys.
2. $(\mathsf{tx}_1, \mathsf{Amt}_{\text{out}}^1, \mathsf{CK}_{\text{out}}^1), \cdots, (\mathsf{tx}_t, \mathsf{Amt}_{\text{out}}^t, \mathsf{CK}_{\text{out}}^t), i^* \leftarrow \mathcal{A}^{\text{ORC}}(pp, \mathsf{TD}, \boldsymbol{t})$ : The adversary $\mathcal{A}$ is given access to all auditor trapdoors $\mathsf{TD}$ and public keys $\boldsymbol{t}$ and all oracles ORC together with $pp$, and outputs an index $1 \le i^* \le t$ and a set of $t$ transactions where
   - $\mathsf{tx}_i = (\mathsf{A}_{\text{in}}^i, \mathsf{PK}_{\text{out}}^i, \mathsf{CN}_{\text{out}}^i, \Pi_i, \mathsf{SN}_i)$ for $i = 1, \ldots, t$,
   - $\mathsf{Amt}_{\text{out}}^i$'s and $\mathsf{CK}_{\text{out}}^i$'s are sets of output amounts and coin keys, respectively, for *uncorrupted* output public keys with $|\mathsf{CK}_{\text{out}}^i| = |\mathsf{Amt}_{\text{out}}^i| \le |\mathsf{PK}_{\text{out}}^i| = |\mathsf{CN}_{\text{out}}^i|$ for all $i \in \{1, \ldots, t\}$.
3. $B_i \leftarrow \mathbf{Verify}(\mathsf{tx}_i, \mathbb{B})$ for $i = 1, \ldots, t$.
4. $R_{i^*} \leftarrow \mathbf{Audit}(\mathsf{tx}_{i^*}, \mathsf{td}, \mathbb{B})$.[12]

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

- for all $i \in \{1, \ldots, t\}$, all public keys and coins in $\mathsf{A}_{\text{in}}^i$ are generated by PKGEN and MINT, respectively, and all accounts in $\mathsf{A}_{\text{in}}^i$ are generated by ACTGEN,
- $\bigcap_{i=1}^t \mathsf{SN}_i = \emptyset$,
- $B_i \neq \emptyset$ for all $i = 1, \ldots, t$,
- $R_{i^*} = \perp$ or $R_{i^*}$ contains an *uncorrupted* account $\mathsf{act}$ such that $\text{SPEND}(\cdot, \mathsf{R}_{\text{in}}, \cdot, \cdot)$ for $\mathsf{act} \in \mathsf{R}_{\text{in}}$ has never been queried.

## E MatRiCT-Au: Lattice-Based Auditable RingCT

We now describe MatRiCT-Au. For completeness, we take most of the algorithms verbatim from MatRiCT and show that minor changes (marked in blue color) suffice to add auditability thanks to our new tools from prior sections. As in [14], we let $r$ be the integer such that any sum of the amounts fits into $r$ bits. The notations in MatRiCT are recalled in Table 4. We discuss the algorithms briefly here and refer the reader to [14] for further details.

**Table 4.** Identifiers for MatRiCT-Au.

| | |
|---|---|
| $R_q$, $R_{\hat{q}}$ | Cyclotomic rings of degree $d$ : $R_q = \mathbb{Z}_q[X]/(X^d+1)$, $R_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^d+1)$ |
| $n, \hat{n}, n_s$ | heights of $\boldsymbol{G}$, $\hat{\boldsymbol{G}}$ and $\boldsymbol{H}$, respectively |
| $m, \hat{m}$ | the dimensions of randomness vectors over $R_q$ and $R_{\hat{q}}$, respectively |
| $N = \beta^k$ | ring size of ring signature (and anonymity set size) |
| $\ell$ | the real spender's column index with $0 \le \ell < N$ |
| $r$ | the bit-length of an amount, i.e., $\mathsf{amt} \in \mathsf{V} = [0, 2^r - 1]$ |
| $\mathcal{B}$ | the maximum absolute coefficient of fresh randomness |

[12] Assume w.l.o.g. that the auditing option in $\mathsf{tx}_{i^*}$ is on and $\mathsf{td}$ is the trapdoor for $\mathsf{tx}_{i^*}$.

In general, 3 commitment keys $\boldsymbol{G}, \hat{\boldsymbol{G}}$ and $\boldsymbol{H}$ are used in the system where $\boldsymbol{H}$ and $\boldsymbol{G}$ are defined over $R_q$ and $\hat{\boldsymbol{G}}$ is defined over $R_{\hat{q}}$ for positive integers $q, \hat{q}$. These matrices are generated using Algorithm 3, which runs an extendable output function $\mathsf{Sam}$. $\mathsf{Sam}$ is modelled as a random oracle in the security analysis. For ease of presentation, we assume that $\mathsf{SamMat}$ works as in the following example. If we run $\boldsymbol{G}_0 \leftarrow \mathsf{SamMat}(\rho, 2, q, n, m)$ and $\boldsymbol{G}_1 \leftarrow \mathsf{SamMat}(\rho, 4, q, n, m)$, then we get $\boldsymbol{G}_0 = [\,\boldsymbol{A} \,\|\, \boldsymbol{b}_0 \,\|\, \boldsymbol{c}_0\,]$ and $\boldsymbol{G}_1 = [\,\boldsymbol{A} \,\|\, \boldsymbol{b}_0 \| \boldsymbol{b}_1 \,\|\, \boldsymbol{c}_0 \| \boldsymbol{c}_1\,]$. This is to simplify the presentation of how the auditors generate the trapdoor rows and how they are used by the spenders. Moreover, to make sure that the auditors create a trapdoor for the largest possible matrix, we introduce $\mathsf{MAX}$, which denotes the maximum number of input/output accounts possible.

**Algorithm 3** $\mathsf{SamMat}(\rho', v, q', n', m', \mathrm{str})$

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

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

To enable optional auditing feature, we assume that all auditors publish their 'trapdoor' rows by running $\mathbf{TdRowGen}(pp)$, which calls our VPDC trapdoor key generation algorithm $\mathsf{CAddTd}$ from Section 5. Note that when writing $\hat{\boldsymbol{G}} = [\,\boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C}\,]$ with $\boldsymbol{A} \in R_{\hat{q}}^{\hat{n} \times \hat{m}}$, $\boldsymbol{B} \in R_{\hat{q}}^{\hat{n} \times v_1}$ and $\boldsymbol{C} \in R_{\hat{q}}^{\hat{n} \times v_2}$, the auditors sample $\boldsymbol{s}' \leftarrow R_{\hat{q}}^{\hat{n}-1}$, $\boldsymbol{e}_0 \leftarrow \mathbb{S}_{\mathcal{B}_e}^{\hat{m}}$, $\boldsymbol{e}_1 \leftarrow \mathbb{S}_{\mathcal{B}_e}^{v_1}$, and $\boldsymbol{e}_2 \leftarrow \mathbb{S}_{\mathcal{B}_e}^{v_2}$. The output of $\mathbf{TdRowGen}(pp)$ provides the longest possible trapdoor row and the users simply truncate $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ returned by the algorithm depending on the number of input/output accounts of their transaction as given in Alg. 9. In particular, a spender requires the first $v_1$ (resp. $v_2$) coordinates of $\boldsymbol{t}_1$ (resp. $\boldsymbol{t}_2$) for $v_1 = v_2 = k\beta + r \cdot S + \lceil \log(M + S - 1) \rceil (r-1)$ for her particular choice of $(M, S)$.

The system parameters are set using Algorithm 4. Algorithms 5, 6 and 7 describe public-secret key pair generation, serial number generation and minting a coin, respectively.

The **Spend** function is described in Algorithms 9, 10, 11 and 12. Algorithm 11 first sets some parameters and computes "corrector values" $c_i$'s, which are

**Algorithm 4** $\mathbf{Setup}(1^\lambda)$ $\lambda$ is the security parameter

1: Choose int. params $k, \beta, r, n, m, \hat{n}, \hat{m}, d, q, \hat{q}$ such that $N = \beta^k$
2: Set $w, p$ such that $|\mathcal{C}_{w,p}^d| > 2^{256}$
3: $\rho \leftarrow \{0,1\}^{256}$
4: Pick a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{C}_{w,p}^d$
5: Set a bound, $\mathsf{MAX}$, on the maximum number input/output accounts
6: **return** $pp = (\rho, \mathcal{H}, k, \beta, r, n, m, \hat{n}, \hat{m}, n_s, d, w, p, q, \hat{q}, \mathsf{MAX})$

**Algorithm 5 KeyGen**($pp$)

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

**Algorithm 6 SerialGen**($\mathsf{sk}$) for a secret key $\mathsf{sk} \in R_q^m$

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

**Algorithm 7 Mint**($\mathsf{amt}$) for $\mathsf{amt} \in [0, 2^r - 1]$

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

**Algorithm 8 TdRowGen**($pp$) ▷ New to MatRiCT-Au

1: $v = k\beta + r \cdot \mathsf{MAX} + \lceil \log(2 \cdot \mathsf{MAX} - 1) \rceil (r - 1)$
2: $\hat{\boldsymbol{G}} \leftarrow \mathsf{SamMat}(\rho, 2v, \hat{q}, \hat{n}, \hat{m}, \text{“Gbig”})$
3: $(\hat{\boldsymbol{G}}', \mathsf{td}) \leftarrow \mathsf{CAddTd}(\hat{\boldsymbol{G}})$
4: Parse the last row $\boldsymbol{t}$ of $\hat{\boldsymbol{G}}'$ as $(\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2) \in R_{\hat{q}}^{\hat{m}} \times R_{\hat{q}}^{v} \times R_{\hat{q}}^{v}$
5: **return** $(\mathsf{td}, \boldsymbol{t} = (\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2))$

used to prove balance. Then, the output coins are minted and the spender uses Algorithm 9 to run an aggregated binary proof on the bits of the output amounts and the unary representation of the real spender's index $\ell$.

The only significant difference in spending of MatRiCT-Au over MatRiCT comes in Algorithm 9, where we replace the last row of $\hat{\boldsymbol{G}}$ with the trapdoor vectors when the spender chooses to be audited by Authority $\mathcal{J}$ ($\mathcal{J} > 0$). If $\mathcal{J} = 0$, then MatRiCT-Au runs exactly as MatRiCT, i.e., auditability option is off.

The commitment $C$ at Step 26 is called the "corrector commitment" and is used in the balance proof. One can observe that it has the form of a sequence of carry values. The prover must prove that $C$ is of the form as given at Step 26. This is guaranteed by the check at Step 18 of Algorithm 13.

After this, $M$ ring signatures are run to prove ownership of an account from each row of $\mathsf{A}_{\text{in}}$. To prevent double-spending, a serial number for each account spent is also computed. A final ring signature is executed to prove that the balance is preserved, where $\sum_{i=0}^{S-1} \mathsf{cn}_{\text{out},i} - \sum_{i=0}^{M-1} \mathsf{cn}_{i,\ell} + C$ is shown to be a commitment to zero for the same index $\ell \in [0, N-1]$.

**Algorithm 9** BinaryCommit ▷ Commitment Step of Binary Proof

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

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

$ck = \hat{\boldsymbol{G}} \leftarrow \mathsf{SamMat}(\rho, v, \hat{q}, \hat{n}, \hat{m}, \text{“Gbig”})$ for $v = 2 \cdot \left(\sum_{j=0}^{t-1} s_j\right)$
**if** $\mathcal{J} > 0$ **then** ▷ Auditing by Authority $\mathcal{J}$ active
  Retrieve Authority $\mathcal{J}$'s trapdoor row vectors as $\boldsymbol{t}_0 \in R_{\hat{q}}^{\hat{m}}$, $\boldsymbol{t}_1 \in R_{\hat{q}}^{v'}$, $\boldsymbol{t}_2 \in R_{\hat{q}}^{v'}$
  Remove the last $v' - v/2$ coordinates of $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$
  Let $\hat{\boldsymbol{G}} = [\,\boldsymbol{A} \,\|\, \boldsymbol{B} \,\|\, \boldsymbol{C}\,]$ for $\boldsymbol{A} \in R_{\hat{q}}^{\hat{n} \times \hat{m}}, \boldsymbol{B} \in R_{\hat{q}}^{\hat{n} \times (v/2)}, \boldsymbol{C} \in R_{\hat{q}}^{\hat{n} \times (v/2)}$
  Replace the last rows of $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ with $\boldsymbol{t}_0, \boldsymbol{t}_1$ and $\boldsymbol{t}_2$, respectively
**end if**
$\boldsymbol{r}_b \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot \hat{m}}$ and $\boldsymbol{r}_a \leftarrow \{-\hat{\mathcal{B}}_{\text{big}}, \ldots, \hat{\mathcal{B}}_{\text{big}}\}^{d \cdot \hat{m}}$
**for** $j = 0, \ldots, t-1$ **do** ▷ Iterate over each bit sequence
  **if** $\text{Bool}_j = \text{True}$ **then** ▷ The case of Hamming weight equal to 1.
    **if** $b_0^{(j)} = 0$, **then** $i^* = -1$ ▷ Out of $[1, s_j - 1]$
    **else** $i^* \leftarrow \{1, \ldots, s_j - 1\}$ & $a_{i^*}^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
    **for** $i = 1, \ldots, s_j - 1$ and $i \neq i^*$ **do** ▷ $i$ starts from 1.
      **if** $b_i^{(j)} = 0$, **then** $a_i^{(j)} \leftarrow \{-(\mathcal{B}_j - p), \ldots, \mathcal{B}_j - p\}^d$
      **else** $a_i^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
    **end for**
    $a_0^{(j)} = -\sum_{i=1}^{s_j-1} a_i^{(j)}$
  **else**
    **for** $i = 0, \ldots, s_j - 1$ **do** ▷ $i$ starts from 0.
      $a_i^{(j)} \leftarrow \{-\mathcal{B}_j, \ldots, \mathcal{B}_j\}^d$
    **end for**
  **end if**
**end for**
$\boldsymbol{b} = \left(b_0^{(0)}, \ldots, b_{s_{t-1}-1}^{(t-1)}\right)$, $\boldsymbol{a} = \left(a_0^{(0)}, \ldots, a_{s_{t-1}-1}^{(t-1)}\right)$
$\boldsymbol{c} = \left(a_0^{(0)}(1 - 2b_0^{(0)}), \ldots, a_{s_{t-1}-1}^{(t-1)}(1 - 2b_{s_{t-1}-1}^{(t-1)})\right)$
$\boldsymbol{d} = \left(-(a_0^{(0)})^2, \ldots, -(a_{s_{t-1}-1}^{(t-1)})^2\right)$
$B = \mathsf{Com}_{ck}(\boldsymbol{b}, \boldsymbol{c}; \boldsymbol{r}_b)$, $A = \mathsf{Com}_{ck}(\boldsymbol{a}, \boldsymbol{d}; \boldsymbol{r}_a)$ in $R_{\hat{q}}^{\hat{n}}$ with $ck = \hat{\boldsymbol{G}}$
**return** $(\boldsymbol{r}_a, \boldsymbol{r}_b), (A, B), \{(a_0^{(j)}, \ldots, a_{s_j-1}^{(j)})\}_{j=0}^{t-1}$

The ring signature algorithm is essentially a one-out-of-many proof, whose commitment step is described in Algorithm 10. This proves that the prover knows an opening of one of the $N$ inputs (for example, $\mathsf{pk}_{0,0}, \ldots, \mathsf{pk}_{0,N-1}$).

In Algorithm 12, the spender computes the masked responses of the underlying ZKPs. Each bit given to the binary proof is masked by some polynomial $a \in R_{\hat{q}}$. Similarly, masked randomnesses are computed as in the one-out-of-many proof of [11].

**Algorithm 10** RingCommit ▷ Commitment Step of Ring Sign.

**INPUT:** GenSerial $\in$ {True, False}; $(P_0, \ldots, P_{N-1})$ where $P_i \in R_q^n$; $(p_{0,0}, \ldots, p_{N-1,k-1})$ where $p_{i,j} \in R_q$; $\mathcal{B}, \mathcal{B}_{\text{big,k}} \in \mathbb{Z}^+$.
**OUTPUT:** $(\boldsymbol{\rho}_0, \ldots, \boldsymbol{\rho}_{k-1}), (E_0, F_0, \ldots, E_{k-1}, F_{k-1})$ where $\boldsymbol{\rho}_j \in R_q^m$, $E_j \in R_q^n$ and $F_j \in R_q^{n_s}$. $F_j$'s are omitted when GenSerial = False.

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

Algorithm 13 summarizes the verification, where the same norm checks as in Algorithm 12 are performed, the “missing” components are computed and then the hash output is checked.

**Algorithm 15** Compute $p_{i,j}$ for $k \in \{1, 2\}$

**INPUT:** $\ell, a_{0,0}, \ldots, a_{k-1,\beta-1}$
**OUTPUT:** $p_{0,0}, \ldots, p_{N-1,k-1} \in R_q$

**if** $k = 1$ **then**
    **for** $i = 0, \ldots, N-1$ **do**
        $p_{i,0} = a_{0,i_0}$ ▷ $i = (i_0, \ldots, i_{k-1})$ in base $\beta$
    **end for**
    **return** $(p_{0,0}, \ldots, p_{N-1,0})$
**else if** $k = 2$ **then**
    **for** $i = 0, \ldots, N-1$ **do**
        $p_{i,0} = a_{0,i_0} \cdot a_{1,i_1}$
        $p_{i,1} = \delta_{\ell_0,i_0} \cdot a_{1,i_1} + \delta_{\ell_1,i_1} \cdot a_{0,i_0}$
    **end for**
    **return** $(p_{0,0}, p_{0,1}, \ldots, p_{N-1,0}, p_{N-1,1})$
**end if**

The auditing function (Algorithm 14), which is unique to MatRiCT-Au, simply uses the decryption algorithm from Section 5 to recover the spender index $\ell$ and the output amounts $\mathsf{amt}_{\text{out},j}$'s from the commitment $B$. Note that $\mathsf{CDecGR}$ is run over $R_{\hat{q}}$, and $\boldsymbol{s} = (-\boldsymbol{s}', 1)$ and $B$ are in $R_{\hat{q}}^{\hat{n}}$. The decryptable message

**Algorithm 11 Spend-I**

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

$\mathcal{B}_a = \lceil 20 \cdot pkd \rceil$, $\mathcal{B}_r = \lceil p(S+1)rd \rceil$
$T_g = d^3 \left( \mathcal{B}_a^4 k\beta(\beta+1) + \mathcal{B}_r^4 r(S+1) \right) / (4d)$
$\mathcal{B}_{\text{big}} = \lceil 1.2(M+S+1)\mathcal{B}pwmd \rceil$, $\hat{\mathcal{B}}_{\text{big}} = \lceil 8(M+S+1)\mathcal{B}pw\hat{m}d \rceil$
$\mathcal{B}_{\text{big,k}} = \lceil 1.2 \cdot (M+S+1)\mathcal{B}(pw)^k md \rceil$
$\mathcal{B}'_{\text{big,k}} = \lceil 2.4 \cdot (M+S+1)\mathcal{B}(pw)^k md \rceil$
**for** $i = 0, \ldots, r-2$ **do** ▷ $c_0 = c_r = 0$
    $c_{i+1} = \left( c_i + \sum_{j=0}^{S-1} \mathsf{amt}_{\text{out},j}[i] - \sum_{j=0}^{M-1} \mathsf{amt}_{\text{in},j}[i] \right) / 2$
**end for**
**for** $i = 0, \ldots, S-1$ **do**
    $(\mathsf{cn}_{\text{out},i}, \mathsf{cnk}_{\text{out},i}) \leftarrow \textbf{Mint}(\mathsf{amt}_{\text{out},i})$
**end for**
$\mathsf{CN}_{\text{out}} = (\mathsf{cn}_{\text{out},0}, \ldots, \mathsf{cn}_{\text{out},S-1})$
$\mathsf{CK}_{\text{out}} = (\mathsf{cnk}_{\text{out},0}, \ldots, \mathsf{cnk}_{\text{out},S-1})$
$\boldsymbol{b} = \{(\beta, \text{True}, (\delta_{\ell_j,0}, \ldots, \delta_{\ell_j,\beta-1}), \mathcal{B}_a)\}_{j=0}^{k-1}$
$\boldsymbol{b} = \boldsymbol{b} \cup (r-1, \text{False}, (c_1, \ldots, c_{r-1}), \mathcal{B}_r)$
**for** $j = 0, \ldots, S-1$ **do**
    $\boldsymbol{b} = \boldsymbol{b} \cup (r, \text{False}, \mathsf{Bits}(\mathsf{amt}_{\text{out},j}), \mathcal{B}_r)$
**end for**
$ck = \boldsymbol{G} \leftarrow \mathsf{SamMat}(\rho, r, q, n, m, \text{“G”})$
$\boldsymbol{r}_c \leftarrow \{-\mathcal{B}, \ldots, \mathcal{B}\}^{d \cdot m}$, $\boldsymbol{r}_d \leftarrow \{-\mathcal{B}_{\text{big}}, \ldots, \mathcal{B}_{\text{big}}\}^{d \cdot m}$
$(\boldsymbol{r}_a, \boldsymbol{r}_b), (A, B), (a_{0,0}, \ldots, a_{k-1,\beta-1}, a_{c,1}, \ldots, a_{c,r-1}, a_{\text{out},0}^{(0)}, \ldots, a_{\text{out},r-1}^{(S-1)})$
$\leftarrow \mathsf{BinaryCommit}(k+1+S, \boldsymbol{b}, \mathcal{B}, \hat{\mathcal{B}}_{\text{big}}, \mathcal{J})$ ▷ $a_{c,0} = a_{c,r} = 0$
**for** $i = 0, \ldots, S-1$ **do**
    $\boldsymbol{r}_g^{(i)} \leftarrow \{-\mathcal{B}_{\text{big}}, \ldots, \mathcal{B}_{\text{big}}\}^{d \cdot m}$
    $G_i = \mathsf{Com}_{ck}(a_{\text{out},0}^{(i)}, \ldots, a_{\text{out},r-1}^{(i)}; \boldsymbol{r}_g^{(i)})$ in $R_q^n$
**end for**
$C = \mathsf{Com}_{ck}(c_0 - 2c_1, \ldots, c_{r-1} - 2c_r; \boldsymbol{r}_c)$ in $R_q^n$
$D = \mathsf{Com}_{ck}(a_{c,0} - 2a_{c,1}, \ldots, a_{c,r-1} - 2a_{c,r}; \boldsymbol{r}_d)$ in $R_q^n$
Compute $\boldsymbol{p} = (p_{0,0}, \ldots, p_{N-1,k-1})$ using Alg. 15 with $(\ell, a_{0,0}, \ldots, a_{k-1,\beta-1})$
**for** $i = 0, \ldots, M-1$ **do**
    $\mathsf{sn}_i = \textbf{SerialGen}(\boldsymbol{r}_{i,\ell})$ ▷ Not recomputed if restarted
    $(\boldsymbol{\rho}_0^{(i)}, \ldots, \boldsymbol{\rho}_{k-1}^{(i)}), (E_0^{(i)}, F_0^{(i)}, \ldots, E_{k-1}^{(i)}, F_{k-1}^{(i)}) \leftarrow$
    $\mathsf{RingCommit}(\text{True}, (\mathsf{pk}_{i,0}, \ldots, \mathsf{pk}_{i,N-1}), \boldsymbol{p}, \mathcal{B}, \mathcal{B}_{\text{big,k}})$
**end for**
**for** $j = 0, \ldots, N-1$ **do**
    $P_j = \sum_{i=0}^{S-1} \mathsf{cn}_{\text{out},i} - \sum_{i=0}^{M-1} \mathsf{cn}_{i,j} + C$ in $R_q^n$
**end for**
$(\boldsymbol{\rho}_0^{(M)}, \ldots, \boldsymbol{\rho}_{k-1}^{(M)}), (E_0^{(M)}, \ldots, E_{k-1}^{(M)}) \leftarrow$
$\mathsf{RingCommit}(\text{False}, (P_0, \ldots, P_{N-1}), \boldsymbol{p}, \mathcal{B}, \mathcal{B}'_{\text{big,k}})$
$x = \mathcal{H}(A, B, C, D, E_0^{(0)}, \ldots, E_{k-1}^{(M)}, F_0^{(0)}, \ldots, F_{k-1}^{(M)}, G_0, \ldots, G_{S-1}, \mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}})$

**Algorithm 12 Spend-II** ▷ No mod $q$ or $\hat{q}$ in this function!

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

**for** $j = 0, \ldots, k-1$ and $i = 0, \ldots, \beta - 1$ **do**
  $f_{j,i} = x\delta_{\ell_j,i} + a_{j,i}$ ▷ masked spender index repr.
**end for**
**for** $i = 1, \ldots, r-1$ **do**
  $f_{c,i} = xc_i + a_{c,i}$ ▷ masked corrector values
**end for**
**for** $j = 0, \ldots, S-1$ and $i = 0, \ldots, r-1$ **do**
  $f_{\text{out},i}^{(j)} = x \cdot \mathsf{amt}_{\text{out},j}[i] + a_{\text{out},i}^{(j)}$ ▷ masked out. amt. bits
**end for**
$\boldsymbol{f}_1 = (f_{0,1}, \ldots, f_{k-1,\beta-1})$ ▷ $f_{j,0}$'s are excluded
**if** $\|\boldsymbol{f}_1\|_\infty > \mathcal{B}_a - p$, **then** Go to Step 20 of Alg. 11
$\boldsymbol{f}_r = \left(f_{c,1}, \ldots, f_{c,r-1}, f_{\text{out},0}^{(0)}, \ldots, f_{\text{out},r-1}^{(S-1)}\right)$
**if** $\|\boldsymbol{f}_r\|_\infty > \mathcal{B}_r - p$, **then** Go to Step 20 of Alg. 11
**for** $j = 0, \ldots, k-1$ **do**
  **if** $\|f_{j,0}\| > \mathcal{B}_a\sqrt{d(\beta-1)}$, **then** Go to Step 20 of Alg. 11
**end for**
$\boldsymbol{g} = (f_{0,0}(x - f_{0,0}), \ldots, f_{k-1,\beta-1}(x - f_{k-1,\beta-1}))$
$\boldsymbol{g} = \boldsymbol{g} \cup (f_{c,1}(x - f_{c,1}), \ldots, f_{c,r-1}(x - f_{c,r-1}))$
$\boldsymbol{g} = \boldsymbol{g} \cup \left(f_{\text{out},0}^{(0)}(x - f_{\text{out},0}^{(0)}), \ldots, f_{\text{out},r-1}^{(S-1)}(x - f_{\text{out},r-1}^{(S-1)})\right)$
**if** $\|\boldsymbol{g}\| > \sqrt{T_g}$, **then** Go to Step 20 of Alg. 11
$\boldsymbol{z}_b = x\boldsymbol{r}_b + \boldsymbol{r}_a$ ▷ $\hat{m}$-dimensional
**if** $\|\boldsymbol{z}_b\|_\infty > \hat{\mathcal{B}}_{\text{big}} - \mathcal{B}pw$, **then** Go to Step 20 of Alg. 11
$\boldsymbol{z}_c = x\boldsymbol{r}_c + \boldsymbol{r}_d$
**for** $i = 0, \ldots, S-1$ **do**
  $\boldsymbol{z}_{\text{out},i} = x\boldsymbol{r}_{\text{out},i} + \boldsymbol{r}_g^{(i)}$ where $\boldsymbol{r}_{\text{out},i} = \mathsf{cnk}_{\text{out},i}$
**end for**
**if** $\|(\boldsymbol{z}_c, \boldsymbol{z}_{\text{out},0}, \ldots, \boldsymbol{z}_{\text{out},S-1})\|_\infty > \mathcal{B}_{\text{big}} - \mathcal{B}pw$, **then** Go to Step 20 of Alg. 11
**for** $i = 0, \ldots, M-1$ **do**
  $\boldsymbol{z}^{(i)} = x^k\boldsymbol{r}_{i,\ell} - \sum_{j=0}^{k-1} x^j\boldsymbol{\rho}_j^{(i)}$
  **if** $\|\boldsymbol{z}^{(i)}\|_\infty > \mathcal{B}_{\text{big,k}} - \mathcal{B}(pw)^k$, **then** Go to Step 20 of Alg. 11
**end for**
$\boldsymbol{z}^{(M)} = x^k\boldsymbol{r}_{M,\ell} - \sum_{j=0}^{k-1} x^j\boldsymbol{\rho}_j^{(M)}$ where $\boldsymbol{r}_{M,\ell} = \sum_{i=0}^{S-1}\boldsymbol{r}_{\text{out},i} - \sum_{i=0}^{M-1}\mathsf{cnk}_{i,\ell} + \boldsymbol{r}_c$
**if** $\|\boldsymbol{z}^{(M)}\|_\infty > \mathcal{B}'_{\text{big,k}} - (M+S+1)\mathcal{B}(pw)^k$, **then** Go to Step 20 of Alg. 11
**return** $\mathsf{CK}_{\text{out}}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}}, \mathsf{SN} = (\mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1})$, $\mathcal{J}$ and $\Pi = (B, C, E_1^{(0)}, \ldots, E_{k-1}^{(M)}, F_1^{(0)}, \ldots, F_{k-1}^{(M)}, x, \boldsymbol{f}_1, \boldsymbol{f}_r, \boldsymbol{z}_b, \boldsymbol{z}_c, \boldsymbol{z}^{(0)}, \ldots, \boldsymbol{z}^{(M)}, \boldsymbol{z}_{\text{out},0}, \ldots, \boldsymbol{z}_{\text{out},S-1})$

**Algorithm 13 Verify**

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

**if** $\|\boldsymbol{f}_1\|_\infty > \mathcal{B}_a - p$, **then return** False
**if** $\|\boldsymbol{f}_r\|_\infty > \mathcal{B}_r - p$, **then return** False
Parse $\boldsymbol{f}_1 = (f_{0,1}, \ldots, f_{k-1,\beta-1})$ as in Alg. 12
Parse $\boldsymbol{f}_r = (f_{c,1}, \ldots, f_{c,r-1}, f_{\text{out},0}^{(0)}, \ldots, f_{\text{out},r-1}^{(S-1)})$ as in Alg. 12
**for** $j = 0, \ldots, k-1$ **do**
    $f_{j,0} = x - \sum_{i=1}^{\beta-1} f_{j,i}$
    **if** $\|f_{j,0}\| > \mathcal{B}_a\sqrt{d(\beta-1)}$, **then return** False
**end for**
Compute $\boldsymbol{g}$ as in Alg. 12
**if** $\|\boldsymbol{g}\| > \sqrt{T_g}$, **then return** False
**if** $\|\boldsymbol{z}_b\|_\infty > \hat{\mathcal{B}}_{\text{big}} - \mathcal{B}pw$, **then return** False
**if** $\|(\boldsymbol{z}_c, \boldsymbol{z}_{\text{out},0}, \ldots, \boldsymbol{z}_{\text{out},S-1})\|_\infty > \mathcal{B}_{\text{big}} - \mathcal{B}pw$, **then return** False
**if** $\|(\boldsymbol{z}^{(0)}, \ldots, \boldsymbol{z}^{(M-1)})\|_\infty > \mathcal{B}_{\text{big,k}} - \mathcal{B}(pw)^k$, **then return** False
**if** $\|\boldsymbol{z}^{(M)}\|_\infty > \mathcal{B}'_{\text{big,k}} - (M+S+1)\mathcal{B}(pw)^k$, **then return** False
$\boldsymbol{f} = (f_{0,0}, \ldots, f_{k-1,\beta-1}) \cup \boldsymbol{f}_r$ ▷ $f_{j,0}$'s are included.
Update $\hat{\boldsymbol{G}}$ as in Alg. 9 if $\mathcal{J} > 0$
$A = \mathsf{Com}_{ck}(\boldsymbol{f}, \boldsymbol{g}; \boldsymbol{z}_b) - xB$ in $R_{\hat{q}}^{\hat{n}}$ for $ck = \hat{\boldsymbol{G}}$
$D = \mathsf{Com}_{ck}(f_{c,0} - 2f_{c,1}, \ldots, f_{c,r-1} - 2f_{c,r}; \boldsymbol{z}_c) - xC$ in $R_q^n$ where $f_{c,0} = f_{c,r} = 0$ and $ck = \boldsymbol{G}$ here and in the rest
**for** $i = 0, \ldots, S-1$ **do**
    $G_i = \mathsf{Com}_{ck}(f_{\text{out},0}^{(i)}, \ldots, f_{\text{out},r-1}^{(i)}; \boldsymbol{z}_{\text{out},i}) - x\mathsf{cn}_{\text{out},i}$ in $R_q^n$
**end for**
**for** $l = 0, \ldots, M-1$ **do**
    $E_0^{(l)} = \left[\sum_{i=0}^{N-1}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)\mathsf{pk}_{l,i}\right] - \mathsf{Com}_{ck}(\boldsymbol{0}; \boldsymbol{z}^{(l)}) - \sum_{j=1}^{k-1} E_j^{(l)} x^j$ in $R_q^n$ where $i = (i_0, \ldots, i_{k-1})$ in base $\beta$
    $F_0^{(l)} = x^k\mathsf{sn}_l - \boldsymbol{H}\cdot\boldsymbol{z}^{(l)} - \sum_{j=1}^{k-1} F_j^{(l)} x^j$ in $R_q^{n_s}$
**end for**
**for** $j = 0, \ldots, N-1$ **do**
    $P_j = \sum_{i=0}^{S-1}\mathsf{cn}_{\text{out},i} - \sum_{i=0}^{M-1}\mathsf{cn}_{i,j} + C$ in $R_q^n$
**end for**
$E_0^{(M)} = \left[\sum_{i=0}^{N-1}\left(\prod_{j=0}^{k-1} f_{j,i_j}\right)P_i\right] - \mathsf{Com}_{ck}(\boldsymbol{0}; \boldsymbol{z}^{(M)}) - \sum_{j=1}^{k-1} E_j^{(M)} x^j$ in $R_q^n$
**if** $x \neq \mathcal{H}(A, B, C, D, E_0^{(0)}, \ldots, E_{k-1}^{(M)}, F_0^{(0)}, \ldots, F_{k-1}^{(M)}, G_0, \ldots, G_{S-1}, \mathsf{sn}_0, \ldots, \mathsf{sn}_{M-1}, \mathsf{A}_{\text{in}}, \mathsf{PK}_{\text{out}}, \mathsf{CN}_{\text{out}})$, **then return** False
**return** True

**Algorithm 14 Audit**$(\mathsf{tx}, \mathsf{td} = \boldsymbol{s}, M, S)$ ▷ New to MatRiCT-Au

Parse $\mathsf{tx} = (\cdot, \cdot, \cdot, \mathit{\Pi}, \cdot)$
Parse $\mathit{\Pi} = (B, C, \ldots, F_{k-1}^{(M)}, x, \boldsymbol{f}_1, \ldots, \boldsymbol{z}_{\mathrm{out},S-1})$ as in Alg. 13
$v_1 = k\beta + Sr + \lceil \log(M + S - 1) \rceil (r - 1)$
$(\boldsymbol{m}', x') \leftarrow \mathsf{CDecGR}(B, x, \mathsf{td}, v_1)$ (Alg. 2) ▷ over $R_{\hat{q}}$ with $\boldsymbol{s}, B \in R_{\hat{q}}^{\hat{n}}$
Parse $\boldsymbol{m}' = (b_{0,0}, \ldots, b_{k-1,\beta-1}, *, \ldots, *, b_{\mathrm{out},0}^{(0)}, \ldots, b_{\mathrm{out},r-1}^{(0)}, \ldots, b_{\mathrm{out},0}^{(S-1)}, \ldots, b_{\mathrm{out},r-1}^{(S-1)})$
▷ The middle $\lceil \log(M + S - 1) \rceil (r - 1)$ bits are discarded
$\ell' = \sum_{i=0}^{k-1} \beta^i \sum_{j=0}^{\beta-1} j b_{i,j}$
$\mathsf{amt}_{\mathrm{out},j} = \sum_{i=0}^{r-1} 2^i b_{\mathrm{out},i}^{(j)}$ for $j = 0, \ldots, S - 1$
**return** $(\ell', \mathsf{amt}_{\mathrm{out},0}, \ldots, \mathsf{amt}_{\mathrm{out},S-1})$ ▷ should be equal to the values in Alg. 11

space $\mathcal{D}$ here is equal to the set of $v_1$-dimensional binary vectors over $R_{\hat{q}}$ with the first $k\beta$ bits having exactly $k$ '1's for $v_1 = k\beta + Sr + \lceil \log(M + S - 1) \rceil (r - 1)$.

Note that **Audit** also recovers the "corrector values" and just discards them because they are not really useful for auditors. We discuss an optimization around this in the next section.

A nice feature of MatRiCT-Au is that multiple auditors can audit the *same* transaction *without* revealing their trapdoors to each other. This can be easily realized by having the spender use multiple trapdoor rows in the commitment key $\hat{\boldsymbol{G}}$, where each trapdoor row belongs to a different auditor. It is, of course, also possible that different users opt to be audited by different auditors and even some users may opt to be not audited at all (by choosing $\mathcal{J} = 0$).

*Remark 3.* Using our new results on auditability, it is easy to extend the group signature in [14] so that adversarially-generated signatures can also be opened. In particular, the opening of the group signature runs as in **Audit** (Algorithm 14) and hence in time polylog$(N)$, unlike the $O(N)$ time search required in [14].

### E.1 Optimized Gadget for MatRiCT-Au

As discussed, recovery of "corrector values" in **Audit** is not really needed for MatRiCT-Au application and therefore, we can treat them to be part of the auxiliary message that is not recovered in decryption. The easiest way to do this (without modifying the original MatRiCT protocol structure) is to modify the gadget vector $\boldsymbol{g}$. We can further exploit the fact that the first $k\beta$ bits in the decrypted message has only $k$ '1's. That is, the true entropy of the message to be recovered is $\log N + Sr = k \log \beta + Sr$ bits instead of $v_1 = k\beta + Sr + \lceil \log(M + S - 1) \rceil (r - 1)$.

For our concrete parameter setting of MatRiCT-Au, we always have $k = 1$ and $\beta = N \ll 2^d$, meaning the spender index $\ell$ is represented by the $\ell$-th unit vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ in $R^N$. Since $N \ll 2^d$, it is more than sufficient to encode the user index $\ell$ using the least significant bits (LSBs) of the coefficients of the product $\langle \boldsymbol{g}, \boldsymbol{m} \rangle$ that is recovered in decryption (namely, $m''$ in Alg. 2). Therefore, we will set the first part $\boldsymbol{g}_0$ of $\boldsymbol{g}$ to recover the spender's index $\ell$ below.

Then, since we do not need to recover the corrector values, we can just set the gadget vector coordinates that multiply the corrector values as zero. Therefore, we will set the second part of $\boldsymbol{g}$ as the $\gamma$-dimensional zero vector for $\gamma := \lceil \log(M+S-1) \rceil (r-1)$.

Finally, in our concrete parameter setting, we have $r = d = 64$ as in MatRiCT. Therefore, it is sufficient to have $S$ more bits in the coefficients of the product $\langle \boldsymbol{g}, \boldsymbol{m} \rangle$ to represent the bits of the output amounts. Overall, it is sufficient to have $(S+1)$-bit entropy in the coefficients of the product $\langle \boldsymbol{g}, \boldsymbol{m} \rangle$, and hence $t \geq 2^{S+1}$.

Now, let $N-1 = (i_0, \ldots, i_{\kappa-1})_2$ in binary representation for $\kappa = \lceil \log_2(N) \rceil$. Using these and the above notations, the optimized gadget vector for MatRiCT-Au is $\boldsymbol{g} = (\boldsymbol{g}_0, \boldsymbol{0}^\gamma, \boldsymbol{g}_1)$ with $\boldsymbol{g}_0 \in R^N$ and $\boldsymbol{g}_1 \in R^{Sr} = R^{Sd}$, where

$$\boldsymbol{g}_0^\top = (0, 1, X, X+1, \ldots, i_0 + i_1 X + \cdots + i_{\kappa-1} X^{\kappa-1}), \tag{20}$$

$$\boldsymbol{g}_1^\top = 2 \cdot (\boldsymbol{X}^\top, 2\boldsymbol{X}^\top, \ldots, 2^{S-1}\boldsymbol{X}^\top), \text{ and} \tag{21}$$

$$\boldsymbol{X}^\top := (1, X, X^2, \ldots, X^{d-1}). \tag{22}$$

In the decryption of the optimized **Audit** function, the auditor will recover $m''$ as in Alg. 2. Then, it will use the LSBs of the coefficients of $m''$ to recover the user index $\ell$ (note that these LSBs represent the bits of $\ell$). Finally, it will use the $S$ most significant bits of the coefficients of $m''$ to recover the output amounts and hence the transaction amount.

The overall advantage of the optimized gadget vector is that the condition $t \geq 2^\tau = 2^{\lceil v_1/d \rceil}$ is relaxed to $t \geq 2^{S+1}$. For our concrete parameters, this means that we only require $t \geq 8$ instead of $t \geq 64$, which translates to about $8\times$ saving in $\hat{q}$.

### E.2 Parameter Setting

Our parameter setting of MatRiCT-Au is similar to MatRiCT, and we target an anonymity level $1/N$ for $N \leq 100$, set the number of bits of an amount $r = 64$ and the number of transaction input-output accounts $M, S \leq 2$. The following parameters remain the same: $\mathcal{B} = 1$, $(d, w, p) = (64, 56, 8)$, $k = 1$, $n_s = 1$. One of the system moduli $q$ is also not affected. Therefore, we can use $(n, m) = (18, 38)$ and $q = 2^{31} - 2^{18} + 2^3 + 1$ as in MatRiCT. We also fix $\mathcal{B}_\mathrm{e} = 1$. The main change we need to do is to increase the other system modulus $\hat{q}$ to accommodate for the requirements of the decryption algorithm.

First, as required in Theorem 2, we need to satisfy the condition that challenge differences are invertible in $R_t$ for some $t \geq 2$. Here, we use the results of [10, 13] and employ the *Partition-and-Sample (PaS)* technique. In particular, we set the number of factors of $R_t$ to be 2, which means that PaS method will have $\delta = 64/2 = 32$ coefficient index sets $S_i$ with $|S_i| = 2$. It is easy to see that the non-invertibility probability for the standard way of sampling $w = 56$ non-zero coefficients out of $d = 64$ coefficients is upper bounded by the non-

invertibility probability for the challenge distribution in which we fix[13] exactly $2k$ index sets with Hamming weight $\tilde{w} = 1$ and $(28 - k)$ index sets with Hamming weight $\tilde{w} = 2$, for some $k \in \{0, 1, 2, 3, 4\}$ in PaS technique (the remaining index sets, if any, simply have Hamming weight zero). We upper bound the non-invertibility probability for the latter distribution for each possible $k \in \{0, 1, 2, 3, 4\}$. Note that the total Hamming weight is always $w = 56$. Therefore, we can stick to the standard sampling of challenge coefficients while relying on the results of [10, 13]. Particularly, we used the scripts of [10] to compute a bound on the non-invertibility probability for each subcase of $k \in \{0, 1, 2, 3, 4\}$ above and found that this probability for $t = 13$ is at most $2^{-101}$ in any subcase. Overall, for $d = 64$, we see that we can set $t = 13$ such that $R_t$ splits into two factors and $t$ is a prime with $t \equiv 5 \bmod 8$ while having challenge differences invertible, except with probability at most $2^{-101}$.

Now, we look at the requirements due to the decryption of $B$ commitment via **Audit**, which is defined over $R_{\hat{q}}$. To satisfy the requirement in Theorem 2 for the commitment $B$ partially-decrypted in **Audit** (Alg. 14), we see that $\hat{q} > 2^{55.28}$ is sufficient for the given parameters. This is only about a 2-bit increase in comparison to the $\hat{q}$ modulus $\approx 2^{53}$ in MatRiCT. With the slightly increased $\hat{q}$, we also need to also increase $\hat{n}, \hat{m}$ to $(\hat{n}, \hat{m}) = (35, 69)$ (from $(32, 65)$ in MatRiCT). Observe that $\hat{m} - \hat{n} = \hat{n} - 1$ and thus the two M-LWE assumptions w.r.t $\hat{q}$ in Lemma 3 have matching dimension parameters. This parameter setting yields a "root Hermite factor" of $\delta \approx 1.0045$ (or smaller) for all M-SIS and M-LWE security estimations that are done as in MatRiCT (see [8, Section 3.2.4] for more details on M-SIS/LWE practical security estimations). As a result, the proof length increase in MatRiCT-Au as shown in Table 1 is only $\sim 3\%$, and the public key (4.36 KB) and serial number (248 bytes) lengths remain the same as in MatRiCT (since they are defined over $R_q$).

### E.3 Implementation

Our implementation is done in C/C++ on a standard desktop machine with Intel i7-7700K CPU and we try to keep most of the things same between MatRiCT and MatRiCT-Au implementations. We use the AES-256 in counter mode with AES-NI hardware instructions [18] to implement the pseudorandom generator and use the SHAKE-256 [25] to implement the hash function $\mathcal{H}$. During the benchmarks we use gcc 11.2.0 to compile the source code with compiler options `-O3 -march=native` enabled. As mentioned before, we use $q = 2^{31} - 2^{18} + 2^3 + 1$ in MatRiCT-Au (as the original MatRiCT). We choose $\hat{q} = (2^{27} - 2^{11} + 1)(2^{29} - 2^8 + 1)$ where both prime factors are NTT-friendly as in MatRiCT, and we use $t = 13$.

Compared with MatRiCT, in addition to the existing constant-time NTT and uniform sampler implementations, we make the **Spend** function constant-time. Therefore, we remove the early-evaluation rejection technique [27] and the

[13] Note that the choice of the index sets does not change the non-invertibility probability due to the independence of the coefficients in any two index sets.

dependency of the GMP library [16] during the norm checks of Alg. 12. In **Audit**, the decryption algorithm $\mathsf{CDecGR}$ is run with honestly-generated commitments and proofs, while also using the optimized gadget vector in App. E.1.

The running time comparison between MatRiCT and MatRiCT-Au is shown in Table 5 and the proof lengths are compared in Table 1. For the runtimes, we take the average number of cycles and divide it by $3 \cdot 10^6$. Compared to MatRiCT, MatRiCT-Au only adds slight overhead (5%–10%) in the runtime of transaction generation. Note that the runtimes of MatRiCT from [14] in Table 5 were retrieved by using the older gcc version 8.3.0, which is the likely cause of the transaction generation of MatRiCT-Au being faster than MatRiCT for $N = 10$ and $(M, S) = (1, 2)$. We also note that the given implementation does not exploit architecture-specific optimizations such as AVX2.

**Table 5.** Runtimes (in ms) of MatRiCT and MatRiCT-Au at 3 GHz.

| | Anonymity level | 1/10 | | 1/50 | | 1/100 | |
|---|---|---|---|---|---|---|---|
| | #inputs → #outputs | 1 → 2 | 2 → 2 | 1 → 2 | 2 → 2 | 1 → 2 | 2 → 2 |
| MatRiCT [14] | Key Gen. | 2 | 2 | 2 | 2 | 2 | 2 |
| | Transaction Gen. | 242 | 375 | 301 | 471 | 360 | 620 |
| | Verification | 20 | 23 | 25 | 31 | 31 | 40 |
| MatRiCT-Au | Key Gen. | 2 | 2 | 2 | 2 | 2 | 2 |
| | Transaction Gen. | 233 | 414 | 318 | 508 | 402 | 654 |
| | Verification | 21 | 25 | 26 | 32 | 33 | 42 |
| | Trapdoor Gen. | 4 | 5 | 5 | 6 | 6 | 7 |
| | Audit | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |

## F Security of MatRiCT-Au

In this section, we study the security properties of MatRiCT-Au. The correctness of MatRiCT remains unaffected by our minor changes and MatRiCT-Au satisfies the same correctness property. Throughout the section, we fix $v_1 = v_2 = k\beta + r \cdot \mathsf{MAX} + \lceil \log(2 \cdot \mathsf{MAX} - 1) \rceil (r - 1)$.

### F.1 Anonymity

The anonymity of MatRiCT-Au is mostly similar to that of MatRiCT. The only difference is that we replace the commitment key $\hat{\boldsymbol{G}}$ with a uniformly random one (as in MatRiCT) in the first hybrid below.

**Lemma 3.** *(Anonymity) Let $\mathcal{A}$ be a PPT adversary, and $\mathrm{Adv}_{\mathcal{A}}^{LWE}$, $\mathrm{Adv}_{\mathcal{A}}^{LWE_2}$ and $\mathrm{Adv}_{\mathcal{A}}^{LWE_3}$ be the advantages of $\mathcal{A}$ over solving M-LWE$_{m-n-n_s,m,q,\mathcal{B}}$, M-LWE$_{\hat{m}-\hat{n},\hat{m},\hat{q},\mathcal{B}}$ and M-LWE$_{\hat{n}-1,\hat{m}+v_1+v_2,\hat{q},\mathcal{B}_e}$, respectively. Then, the advantage of $\mathcal{A}$ against* `Exp:Anonymity` *without shuffling is at most*

$$\mathrm{Adv}_{\mathcal{A}}^{Ano} \leq \mathrm{Adv}_{\mathcal{A}}^{LWE_2} + k(M+1) \cdot \mathrm{Adv}_{\mathcal{A}}^{LWE} + \mathrm{Adv}_{\mathcal{A}}^{LWE_3}.$$

*Proof (Lemma 3).* From the original experiment, we switch to a game that is identical to the initial anonymity game in MatRiCT as follows.
$\mathbf{Game}_0$ : This is identical to `Exp:Anonymity` without shuffling.
$\mathbf{Game}_1$ : The challenger replaces the trapdoor row vectors $\boldsymbol{t}_0$, $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ in the commitment key $\hat{\boldsymbol{G}}$ with uniformly random vectors in $R_{\hat{q}}^{\hat{m}}$, $R_{\hat{q}}^{v_1}$ and $R_{\hat{q}}^{v_2}$, respectively. This game is indistinguishable from the previous one by M-LWE$_{\hat{n}-1,\hat{m}+v_1+v_2,\hat{q},\mathcal{B}_e}$.

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_0} - \mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_1}\right| \leq \mathrm{Adv}_{\mathcal{A}}^{\mathrm{LWE}_3}.$$

The result of $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ in MatRiCT. Therefore, the rest follows by the anonymity of MatRiCT [14, Lemma 5.1]. □

### F.2 Balance

We first recall the relation proved by the binary ZKP in MatRiCT, which is also proved by MatRiCT-Au.

**Lemma 4 ( [14, Lemma 5.5]).** *Assume that the following holds*

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

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

- $y \in \Delta\mathcal{C}_{w,p}^d$; $\boldsymbol{b} \in R_{\hat{q}}^{v_1}$; $\hat{\boldsymbol{c}} \in R_{\hat{q}}^{v_2}$; $\hat{\boldsymbol{r}} \in R_{\hat{q}}^{\hat{m}}$
- $yB = \mathsf{Com}_{ck}(y\boldsymbol{b}, \hat{\boldsymbol{c}}; \hat{\boldsymbol{r}})$,
- *All coordinates* $b_i$ *of* $\boldsymbol{b}$ *are in* $\{0,1\}$,
- $\hat{\boldsymbol{c}}$ *is uniquely determined by* $y, \boldsymbol{f}_1, \boldsymbol{f}_r, x$ *and* $\boldsymbol{b}$,
- *For the first* $k\beta$ *coordinates* $b_{0,0}, \ldots, b_{k-1,\beta-1}$ *of* $\boldsymbol{b}$, $\sum_{i=0}^{\beta-1} b_{j,i} = 1$, *i.e., there is only a single 1 in* $\{b_{i,0}, \ldots, b_{i,\beta-1}\}$ *for all* $0 \leq i \leq k-1$,
- $\|(y\boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})\| \leq \gamma_{\mathrm{B}}$ *for*
  $\gamma_{\mathrm{B}} = \left((2pw)^2 d\left(k\mathcal{B}_a^2\beta(\beta+1) + \mathcal{B}_r^2 r(S + \lceil \log(M+S-1)\rceil)\right) + 4\hat{\mathcal{B}}_{\mathrm{big}}^2 \hat{m} d\right)^{1/2}$.

Balance proof of MatRiCT-Au follows analogous to that of MatRiCT due to the following fact (as also discussed in Lemma 1). The additional trapdoor information is only relevant to the last coordinate of the commitments over $R_{\hat{q}}$ (i.e., commitments generated with a 'trapdoored' commitment key). Therefore, by simply ignoring the last coordinate of such commitments, the balance proof follows. This means that we require M-SIS$_{\hat{n}-1,*,\hat{q},2\gamma_{\mathrm{bin}}}$ to be hard, which implies that the requirement of MatRiCT is satisfied, i.e., M-SIS$_{\hat{n},*,\hat{q},2\gamma_{\mathrm{bin}}}$ is also hard.

**Lemma 5.** *(Balance) Assume that* $q > \left(2p\sqrt{K}\right)^K$ *and* $q \equiv 2K+1 \bmod 4K$ *for some* $1 < K \leq d$ *where* $K$ *is a power of 2. Let* $\kappa = k(k+1)/2$ *and* $\theta$ *be a positive real number such that the Euclidean norm of any product of* $\kappa - 1$

*elements in* $\Delta\mathcal{C}^d_{w,p}$ *is at most* $\theta$*. If M-LWE*$_{m-n-n_s,m,q,\mathcal{B}}$*, M-SIS*$_{n,m+r,q,2\gamma}$ *and M-SIS*$_{\hat{n}-1,\hat{m}+v_1+v_2,\hat{q},2\gamma_{\text{bin}}}$ *are hard, where*
$\gamma_{\text{bin}} = 2p\sqrt{dw}\left(T_g + \hat{\mathcal{B}}^2_{\text{big}}\hat{m}d\right)^{1/2}$ *and*

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

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

### F.3 Auditability

The auditability follows essentially from the facts that MatRiCT-Au is balanced (i.e., ring signatures created with respect to the rows of $\mathsf{A}_{\text{in}}$ are unforgeable) and that the decryption algorithm returns the same message used to create the zero-knowledge proof. Therefore, the adversary cannot create a zero-knowledge proof, where an honest user is traced by an adversarially-generated transaction. We provide further details in the following two lemmas, where the setting of $\tau$ is w.r.t. the optimized gadget vector in App. E.1.

**Lemma 6.** *Let* $v_1 = v_2 = k\beta + Sr + \lceil\log(M+S-1)\rceil(r-1)$, $\tau = \lceil(\log N + Sr)/d\rceil$, *and* $\|e\|_{bnd,\infty} := \sqrt{(\hat{m}+v_1+v_2)d}\mathcal{B}_e\gamma_{\text{B}} + 2pw(2^\tau-1) + t/2$ *for* $\gamma_{\text{B}}$ *defined in Lemma 4. Assume that the following holds (see also Remark 4)*

- $t \geq 2^\tau$,
- *Any challenge difference in* $\Delta\mathcal{C}^d_{w,p}$ *is invertible in* $R_t$,
- $\lfloor\hat{q}/t\rfloor > 4pw\|e\|_{bnd,\infty} + t(1/2+2pw)$,
- $\mathsf{tx}$ *is a valid transaction generated with at most* $q_H$ *random oracle queries,*
- *the size of the set of "good" challenges* $\mathcal{G}_{\mathsf{tr}}$ *for* $\mathsf{tx}$ *is at least 3,*
- *assumptions in Lemma 4 are satisfied.*

*Then, except with a negligible probability,* $\ell' = \ell$ *for* $(\ell', \ldots) \leftarrow \textbf{\textit{Audit}}(\mathsf{tx}, \mathsf{td})$ *and the input* $\ell$ *in Algorithm 11. Further,* ***Audit*** *(Algorithm 14) runs in expected* $q_H$ *iterations.*

*Proof.* From Lemma 4, we have an extracted opening $(y, \boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})$ of $B$ such that $yB = \mathsf{Com}_{ck}(y\boldsymbol{b}, \hat{\boldsymbol{c}}; \hat{\boldsymbol{r}})$, with $\|(y\boldsymbol{b}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{r}})\| \leq \gamma_{\text{B}}$ (and the other additional properties stated in Lemma 4). From here, the claim follows using the results in Theorem 2 and the assumptions in this lemma statement. □

*Remark 4.* As 3 accepting protocol transcripts are required to "complete" the special soundness proof of our binary ZKP, we require the additional assumption in Lemma 6 that $|\mathcal{G}_{\mathsf{tr}}| \geq 3$. The extracted witness is still determined by 2 accepting protocol transcripts, and the third one is needed to conclude with probability 1 that the extracted message is binary. If the assumption does not hold, a cheating prover has a negligible chance of producing a valid transcript.

**Lemma 7 (Auditability).** *If all the assumptions in Lemmas 5 and 6 hold, then no PPT adversary can win* `Exp:Auditability` *without shuffling with non-negligible probability.*

*Proof.* Lemma 6 shows that for any auditing-enabled transaction, **Audit** (Algorithm 14) will output an index $\ell' \in [0, N-1]$ (expect for a negligible probability). Therefore, the adversary's chance of winning `Exp:Auditability` while **Audit** failing to output an index is negligible.

Let $\ell' \leftarrow \mathbf{Audit}(\mathsf{tx}_{i^*}, \mathsf{td}, \mathbb{B})$. The other case when there is an *uncorrupted* account in the $\ell'$-th column of $\mathsf{A}_{\text{in}}$ is ruled out by the soundness of the one-out-of-many proof, where knowledge of the secret key of $\ell'$-th public key is proven (with relaxation). That is, such an adversary succeeding in winning `Exp:Auditability` with non-negligible probability leads to an algorithm for solving M-SIS as in **Case 1 (forgery)** in the balance proof of [14, Lemma 5.7]. □

## G Application of Our VPDC to MatRiCT$^+$

The real spender's index $\ell$ is encoded the same way in both MatRiCT [14] and MatRiCT$^+$ [13]. In fact, as noted in [13], MatRiCT$^+$ uses a special case of the ring signature in MatRiCT. Therefore, we can similarly apply our techniques to MatRiCT$^+$ to extend it into a formally auditable construction, where even adversarially-created protocol outputs can be audited. The main difference in MatRiCT$^+$ that relates to our decryption approach is that the commitment $B$ for the binary proof encodes only the bits for the user index representation and a couple more bits used for the so-called "patching" technique. Therefore, the decryption of $B$ allows the auditor to recover the user index. What we need to do is to use the norm-bound $\gamma_{\text{B}}$ of the opening of the $B$ commitment from [13] to see the effect on the parameters and the proof size.

To satisfy the challenge difference invertibility in $R_t$, we can use [23, Corollary 1.2]. In MatRiCT$^+$, challenges have infinity norm at most $p = 1$, and thus any challenge difference has infinity norm at most $2p = 2$. Hence, applying [23, Corollary 1.2], we get that we need $t > 8$ for a prime $t \equiv 5 \bmod 8$ (with $R_t$ splitting into two factors) to guarantee invertibility of challenge differences. Therefore, we can set $t = 13$. We also set $\mathcal{B}_{\text{e}} = 1$ as before.

We now need to consider the size of $\hat{q}$ as before to satisfy the requirement in Theorem 2. Using the norm-bounds from [13], we computed that $\hat{q} > 2^{45.55}$ is sufficient (instead of $\hat{q} \approx 2^{32}$ in MatRiCT$^+$). From here, we also need to change the parameters $\hat{n}$ and $\hat{\kappa}$ in [13] ($\hat{m}$ in MatRiCT is equivalent to $\hat{n} + \hat{\kappa}$ in MatRiCT$^+$). Concretely, we set $(\hat{n}, \hat{\kappa}) = (8, 7)$. Observe that $\hat{n} - 1 = \hat{\kappa}$ to have matching M-LWE dimension parameters as before. The other parameters remain the same as in MatRiCT$^+$ since they are not related to $\hat{q}$. From here, we computed the proof sizes of an auditable version of MatRiCT$^+$ and saw that there is less than 9 KB increase in the proof sizes over the original (uncompressed) MatRiCT$^+$. The original (uncompressed) MatRiCT$^+$ has proof sizes between 47 and 61 KB for $N = 11$, $S = 2$ and $M \leq 100$. We can further benefit from the

compression techniques as in MatRiCT$^+$ to decrease the additional overhead to at most about 6 KB.