SMILE: Set Membership from Ideal Lattices with Applications to Ring Signatures and Confidential Transactions*

Vadim Lyubashevsky[1], Ngoc Khanh Nguyen[1,2], and Gregor Seiler[1,2]

[1] IBM Research Europe – Zurich, Switzerland
[2] ETH Zurich, Switzerland

Abstract. In a set membership proof, the public information consists of a set of elements and a commitment. The prover then produces a zero-knowledge proof showing that the commitment is indeed to some element from the set. This primitive is closely related to concepts like ring signatures and "one-out-of-many" proofs that underlie many anonymity and privacy protocols. The main result of this work is a new succinct lattice-based set membership proof whose size is logarithmic in the size of the set.

We also give a transformation of our set membership proof to a ring signature scheme. The ring signature size is also logarithmic in the size of the public key set and has size 16 KB for a set of 2^5 elements, and 22 KB for a set of size 2^{25}. At an approximately 128-bit security level, these outputs are between 1.5X and 7X smaller than the current state of the art succinct ring signatures of Beullens et al. (Asiacrypt 2020) and Esgin et al. (CCS 2019).

We then show that our ring signature, combined with a few other techniques and optimizations, can be turned into a fairly efficient Monero-like confidential transaction system based on the MatRiCT framework of Esgin et al. (CCS 2019). With our new techniques, we are able to reduce the transaction proof size by factors of about 4X - 10X over the aforementioned work. For example, a transaction with two inputs and two outputs, where each input is hidden among 2^{15} other accounts, requires approximately 30KB in our protocol.

1 Introduction

Privacy-based transaction systems are steadily gaining in popularity to the point that central banks of the US and the EU are exploring an eventual shift to digital currency. Transaction systems can be equipped with various degrees of privacy, possibilities for auditability, and permission types for joining the transaction network. The common element at the heart of most of these schemes is a zero-knowledge proof which can be adapted to endow the scheme with the desired features. The most efficient zero-knowledge proofs which allow for proving a rich

* This is the full version of [27] presented at CRYPTO 2021.

set of statements are generally based on the hardness of the discrete logarithm problem over elliptic curves. This poses a problem for the eventual use of digital currency because the timeline for widescale deployment of these transaction systems could very well coincide with the advent of a quantum computer that is able to break them. It is therefore important to begin considering schemes which are based on assumptions that are believed to be resistant to quantum attacks.

The currently most efficient, in terms of size and speed, quantum-safe basic primitives are based on the hardness of lattice problems with algebraic structure. Lattice-based constructions are therefore natural candidates for more advanced cryptographic tools like zero-knowledge proofs. Over the last few years, there has indeed been rapid progress in the field of lattice-based zero knowledge (e.g. [4, 11, 17, 32, 8, 16, 18, 1, 15, 25]). There now exist fairly practical protocols for proving knowledge of pre-images of lattice-based 1-way functions, arithmetic sums and products of committed values, as well as various primitives such as ring signatures and group signatures. In virtually all of these cases, the lattice-based solutions result in the most efficient (potentially) quantum-safe option.

As far as a relatively complete quantum-safe transaction system, the recent work of Esgin et al. [18], also based on the hardness of lattice problems, appears to be the most efficient solution. Their work adapts the RingCT protocol [29], which serves as the foundation of the digital currency Monero, and provides formal definitions upon which they construct their MatRiCT protocol. While certainly not as efficient as discrete logarithm based schemes, this work showed that a lattice-based confidential transaction system is something that may eventually be a very reasonable solution.

Our Results and Related Work. At the core of many privacy-based protocols (including the one from [18]) is a set membership proof in which the prover shows, in zero-knowledge, that a commitment is to a value from a public set. This concept is very closely related to "one-out-of-many" proofs [20] and ring signatures [31]. The main result of this work is a new set membership proof which is logarithmic in the size of the set and leads to a ring signature scheme with outputs noticeably smaller than the currently shortest schemes from [18, 6].[3] We point out that "one-out-of many" proofs [20], in which the prover shows that one of the commitments in a set is a commitment to 0, are actually equivalent to the ring signatures that we construct. This is because lattice-based public keys can be thought of as commitments to 0. We then show how to use our ring signature scheme / "one-out-of-many" proof, together with a few other optimizations of prior work, to create a more efficient confidential transaction system based upon the MatRiCT definitions.

We now give a brief overview of where the efficiency advantage comes from. The shorter proofs in our scheme are partly a result of the fact that the modulus

[3] One can also obtain ring signatures which are linear (rather than logarithmic) in the size of the public key set by plugging in a lattice-based signature scheme based on a trapdoor function, such as [30], into the generic framework of [31]. Even though for small set sizes (around a dozen), this may be smaller than our solution, it quickly becomes much larger (see Table 2).

in our underlying polynomial ring stays the same for all practical set sizes. On the other hand, if the size of the set is $n = 32^m$, then the exponent of the modulus in the ring used in [18] increases linearly in m. The reason for this difference is that [18] use "Ajtai-type" commitments which compress the input, but only allow for commitments of "short" messages. In our construction, however, we use BDLOP commitments [5] which allow commitments to arbitrary-size elements, at the expense of a slightly larger commitment size. But because the number of commitments we need is logarithmic in the size of the set, this does not pose a problem with the commitment size becoming too big.

An additional advantage of BDLOP commitments that we extensively use is that if one plans ahead by choosing a long-enough randomness vector in the beginning of the protocol, then one can adjoin a new commitment at any time and the size of the commitment only increases by the size of the committed message. In particular, the increase in size does not depend on the security parameter, which is what one would need if creating a new commitment. We use this property when combining our new techniques along with the framework for proving various relations committed to in BDLOP commitments from [1, 15, 25]. Thus our constructions essentially have just one BDLOP commitment for the entire protocol. We further reduce the transaction size by employing an amortization technique so that the proof contains just two elements whose size depends on the security parameter.

In the rest of the introduction, we give rather detailed high-level descriptions of our constructions. The reason for this level of detail is that the protocols in the body of the paper use optimizations that combine the new ideas together with prior work in a non-black box manner, which tends to somewhat obfuscate the high level picture. In the introduction, we instead give slightly less efficient constructions that try to highlight the separate parts making up the complete protocols. We would then hope that with the high-level intuition in hand, the interested reader can better follow the complete protocols in the body.

1.1 The Polynomial Ring and BDLOP Commitments

Throughout this paper, we will be working over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^{128}+1)$ where q is set such that $X^{128}+1 = \prod_{i=1}^{32}(X^4 - r_i)$ and $X^4 - r_i$ are irreducible modulo q (c.f. [28] for how to set q to obtain such a factorization). We will be exclusively using BDLOP commitments [5], where a commitment to a polynomial vector $\vec{\boldsymbol{m}} \in \mathcal{R}_q^k$ is of the form

$$\begin{bmatrix} \boldsymbol{B}_0 \\ \boldsymbol{B}_1 \end{bmatrix} \vec{\boldsymbol{r}} + \begin{bmatrix} \vec{\boldsymbol{0}} \\ \vec{\boldsymbol{m}} \end{bmatrix} = \begin{bmatrix} \vec{\boldsymbol{t}}_0 \\ \vec{\boldsymbol{t}}_1 \end{bmatrix}, \tag{1}$$

where $\boldsymbol{B}_i$ are uniform[4] public random matrices and $\vec{\boldsymbol{r}}$ is a random low-norm vector which serves as the commitment randomness. To open the commitment without revealing it, one would ideally want to give a zero-knowledge proof of a low-norm $\vec{\boldsymbol{r}}$ satisfying $\boldsymbol{B}_0\vec{\boldsymbol{r}} = \vec{\boldsymbol{t}}_0$. Unfortunately, there is no particularly efficient zero-knowledge proof for this statement, and so a relaxed opening is defined which consists of a vector $\vec{\boldsymbol{v}}$ and a polynomial $\boldsymbol{c}$ satisfying $\boldsymbol{B}_0\vec{\boldsymbol{v}} = \vec{\boldsymbol{t}}_0$ such that $\|\boldsymbol{c}\|$ and $\|\boldsymbol{c}\vec{\boldsymbol{v}}\|$ are small (but $\vec{\boldsymbol{v}}$ is not necessarily small itself). The committed message is then implicitly

$$\vec{\boldsymbol{m}} = \vec{\boldsymbol{t}}_1 - \boldsymbol{B}_1\vec{\boldsymbol{v}}. \tag{2}$$

An efficient zero-knowledge proof for the above opening was given in [5]. That work also showed how to prove linear (over $\mathcal{R}_q$) relations of $\vec{\boldsymbol{m}}$ without increasing the proof size. For this, it's in fact enough to just be able to prove that the commitment is to $\vec{\boldsymbol{0}}$. The reason is that a commitment of $\vec{\boldsymbol{m}}$ can be easily converted to a commitment of $\vec{\boldsymbol{m}} + \vec{\boldsymbol{m}}'$ by adding $\vec{\boldsymbol{m}}'$ to $\vec{\boldsymbol{t}}_1$. Similarly, for any matrix $\boldsymbol{L}$ over $\mathcal{R}_q$, one can convert a commitment of $\vec{\boldsymbol{m}}$ to one of $\boldsymbol{L}\vec{\boldsymbol{m}}$ by multiplying the bottom part by $\boldsymbol{L}$ to obtain $\begin{bmatrix}\boldsymbol{B}_0\\ \boldsymbol{L}\boldsymbol{B}_1\end{bmatrix}\cdot\vec{\boldsymbol{r}} + \begin{bmatrix}\vec{\boldsymbol{0}}\\ \boldsymbol{L}\vec{\boldsymbol{m}}\end{bmatrix} = \begin{bmatrix}\vec{\boldsymbol{t}}_0\\ \boldsymbol{L}\vec{\boldsymbol{t}}_1\end{bmatrix}$. Thus proving that the message $\vec{\boldsymbol{m}}$ in (1) satisfies $\boldsymbol{L}\vec{\boldsymbol{m}} = \vec{\boldsymbol{u}}$, involves proving that the commitment $\begin{bmatrix}\vec{\boldsymbol{t}}_0\\ \boldsymbol{L}\vec{\boldsymbol{t}}_1 - \vec{\boldsymbol{u}}\end{bmatrix}$ with public key $\begin{bmatrix}\boldsymbol{B}_0\\ \boldsymbol{L}\boldsymbol{B}_1\end{bmatrix}$ is a commitment to $\vec{\boldsymbol{0}}$.

Later works (e.g. [1, 15, 25]) showed how to prove more complicated relations between the committed messages in BDLOP commitments. These include proving multiplicative relations among the polynomials comprising $\vec{\boldsymbol{m}}$ and proving linear relations over $\mathbb{Z}_q$ (rather than $\mathcal{R}_q$) of the integer coefficients comprising $\vec{\boldsymbol{m}}$. An important feature of these aforementioned proofs is that the proof size does not grow with the number of relations that one needs to prove about one commitment. So the cost, in terms of proof size, of proving multiple relations about one commitment is the cost of proving the most expensive one.

1.2 The New Set Membership Proof

In this work we extend the toolbox of what can be proved about $\vec{\boldsymbol{m}}$ in BDLOP commitments by showing how to do set membership proofs. Given a collection of polynomial vectors $\vec{\boldsymbol{p}}_i$, and a commitment to one on them, we would like to prove that the committed $\vec{\boldsymbol{w}}$ is indeed one of the $\vec{\boldsymbol{p}}_i$.

More specifically, the public information consists of $\boldsymbol{P} = [\vec{\boldsymbol{p}}_1 \mid \ldots \mid \vec{\boldsymbol{p}}_n]$, where $n = l^m = 32^m$, and a commitment ω. The prover gives a zero knowledge proof that a commitment ω opens to $(\vec{v}_1, \ldots, \vec{v}_m, \vec{\boldsymbol{w}})$ where

$$\boldsymbol{P}\cdot(\vec{v}_1 \otimes \ldots \otimes \vec{v}_m) = \vec{\boldsymbol{w}} \tag{3}$$

$$\forall i, \vec{v}_i \in \{0,1\}^l \text{ and } \|\vec{v}_i\|_1 = 1. \tag{4}$$

[4] For efficiency, a large portion of $\boldsymbol{B}_i$ can be the identity matrix (c.f. [5]), but we ignore the form of the public randomness in this paper, as it does not affect any output sizes.

Notice that by definition of the $\vec{v}_i$, their tensor product will be a vector of length n consisting of all zeros and one 1 (this decomposition observation was originally used in [20]). If each vector $\vec{v}_i$ will be committed as a polynomial $\boldsymbol{m}_i$ in the BDLOP commitment,[5] then (4) can already be proved using the aforementioned techniques from [1, 15]. Our main result in this work is an efficient proof of (3) whose size is linear in m, and thus logarithmic in the number of elements in $\boldsymbol{P}$. We also prove a more generic k-dimensional version of this problem. In this version, there are k public lists

$$\boldsymbol{P}^{(1)} = \left[\vec{\boldsymbol{p}}_1^{(1)} \mid \ldots \mid \vec{\boldsymbol{p}}_n^{(1)}\right], \ldots, \boldsymbol{P}^{(k)} = \left[\vec{\boldsymbol{p}}_1^{(k)} \mid \ldots \mid \vec{\boldsymbol{p}}_n^{(k)}\right]$$

and $\vec{w}$ is a sum of k elements, one taken from each set. The prover gives a zero knowledge proof that the commitment ω opens to

$$(\vec{v}_1^{(1)}, \ldots, \vec{v}_m^{(1)}, \ldots, \vec{v}_1^{(k)}, \ldots, \vec{v}_m^{(k)}, \vec{\boldsymbol{w}})$$

where

$$\sum_{j=1}^{k} \boldsymbol{P}^{(j)} \cdot (\vec{v}_1^{(j)} \otimes \ldots \otimes \vec{v}_m^{(j)}) = \vec{\boldsymbol{w}} \tag{5}$$

$$\forall i, j,\ \vec{v}_i^{(j)} \in \{0,1\}^l \text{ and } \|\vec{v}_i^{(j)}\|_1 = 1 \tag{6}$$

This proof is of size $O(mk)$, so there is no amortization happening. But being able to prove the above will allow us to amortize away many of the other parts of the anonymous transaction protocol.

1.3 Set Membership Proof Sketch

We now give a sketch of how to prove (3) and (4). Let us first define the set $\mathcal{M}_q = \mathbb{Z}_q + \mathbb{Z}_q X + \mathbb{Z}_q X^2 + \mathbb{Z}_q X^3$. Because of the way we defined $\mathcal{R}_q$, the NTT and inverse NTT functions are bijective functions $\mathsf{NTT}(\boldsymbol{w}) : \mathcal{R}_q \to \mathcal{M}_q^{32}$ and $\mathsf{NTT}^{-1}(\vec{w}) : \mathcal{M}_q^{32} \to \mathcal{R}_q$ where

$$\mathsf{NTT}(\boldsymbol{w}) = (\boldsymbol{w} \bmod X^4 - r_1, \ldots, \boldsymbol{w} \bmod X^4 - r_{32}).$$

These functions extend to polynomial vectors in the natural way by being applied to each polynomial separately.

We will also need to overload the inner product operator. For a polynomial $\boldsymbol{w}$ such that $\mathsf{NTT}(\boldsymbol{w}) = \vec{w} = (w_1, \ldots, w_{32}) \in \mathcal{M}_q^{32}$, define the function $g(\boldsymbol{w}) = \sum_{i=1}^{32} w_i$. In other words, it's just the sum of the NTT coefficients as polynomials in $\mathcal{M}_q$. For two vectors $\vec{w}, \vec{w}' \in \mathcal{M}_q^{32}$, we define $\langle \vec{w}, \vec{w}' \rangle =$

[5] Actually the inverse NTT of the vector $\vec{v}_i$, which is an element of $\mathcal{R}_q$, will be committed – see Section 1.3.

$g(\mathsf{NTT}^{-1}(w)\,\mathsf{NTT}^{-1}(w'))$. It resembles an inner product because we can equivalently write it as

$$\langle \vec{w}, \vec{w}' \rangle = \sum_{i=1}^{32} w_i w_i' \bmod (X^4 - r_i).$$

The multiplication is performed modulo different polynomials, amd so this function is not an inner product. But it is commutative and satisfies $\langle \vec{w} + \vec{w}', \vec{w}'' \rangle = \langle \vec{w}, \vec{w}'' \rangle + \langle \vec{w}', \vec{w}'' \rangle$. Similarly, for $\vec{w} = (\vec{w}_1, \ldots, \vec{w}_k), \vec{w}' = (\vec{w}_1', \ldots, \vec{w}_k')$, where each $\vec{w}_i, \vec{w}_i' \in \mathcal{M}_q^{32}$, one defines $\langle \vec{w}, \vec{w}' \rangle = \sum_{i=1}^{k} \langle \vec{w}_i, \vec{w}_i' \rangle$.

For convenience, we will now rewrite the set membership problem to be over $\mathcal{M}_q$. In particular, the public information consists of vectors $P = [\vec{p}_1 \mid \ldots \mid \vec{p}_n]$ where each $\vec{p}_i \in \mathcal{M}_q^{32k}$, for some arbitrary k. And we also have a commitment to a vector $\vec{w} \in \mathcal{M}_q^{32k}$ such that $\vec{w} = \vec{p}_i$ for some i. Notice that the $\vec{p}_i$ and $\vec{w}$ are the NTT of the $\vec{\boldsymbol{p}}_i, \vec{\boldsymbol{w}}$ from (3). To commit to the vector $\vec{w}$, we define the polynomial vector $\vec{\boldsymbol{w}} = \mathsf{NTT}^{-1}(\vec{w}) \in \mathcal{R}_q^k$ and then use the BDLOP commitment from (1) to commit to $\vec{\boldsymbol{w}}$. Later rows of this BDLOP commitment will also include commitments to the vectors $\vec{v}_1, \ldots, \vec{v}_m \in \mathcal{M}_q^{32}$ (defined as in (4)). We will define the polynomials $\boldsymbol{v}_j = \mathsf{NTT}^{-1}(\vec{v}_j)$ and commit to them in the BDLOP commitment. Note that we can already prove (4) using the techniques from [1, 15] by proving that $\vec{\boldsymbol{v}} \cdot (\vec{\boldsymbol{1}} - \vec{\boldsymbol{v}}) = \vec{\boldsymbol{0}}$ and that the NTT coefficients of each polynomial in $\vec{\boldsymbol{v}}$ sum to 1.

We now describe how to prove (3) – in other words, that $P \cdot (\vec{v}_1 \otimes \ldots \otimes \vec{v}_m) - \vec{w} = \vec{0}$. We will prove this by showing that for a random challenge $\vec{\gamma} \in \mathcal{M}_q^{32k}$, the "inner product" $\langle P \cdot (\vec{v}_1 \otimes \ldots \otimes \vec{v}_m) - \vec{w}, \vec{\gamma} \rangle = 0$. Because $\mathbb{Z}_q[X]/(X^4 - r_i)$ are fields and of size q^4, it's not hard to see that if the left term in the inner product is not $\vec{0}$, then the probability of the inner product being 0 is exactly q^{-4}. Because we will be working with a $q \approx 2^{32}$, this probability is approximately 2^{-128}, so no repetitions are required.

We now get to the main technical part of the protocol. Let's break up P into 32 parts as $P = [\ P_1 \mid \ldots, \mid P_{32}\ ]$ and define $P' := \begin{bmatrix} \gamma^T P_1 \\ \vdots \\ \gamma^T P_{32} \end{bmatrix} \in \mathcal{M}_q^{32 \times 32^{m-1}}$.

Then using the property that $\vec{v}_i$ are vectors over $\mathcal{M}_q$ with just constant coefficients,[6] with some algebraic manipulation (see (18)), it can be shown that

$$\langle P(\vec{v}_1 \otimes \ldots \otimes \vec{v}_m) - \vec{w}, \vec{\gamma} \rangle = \langle \vec{v}_1, P'(\vec{v}_2 \otimes \ldots \otimes \vec{v}_m) \rangle - \langle \vec{w}, \vec{\gamma} \rangle. \tag{7}$$

To prove that the left-hand side is 0, it is therefore equivalent to prove that the right-hand side is 0. The crucial part is that the right-hand side contains

[6] Intuitively, if the coefficients of $\vec{v}_i$ were polynomials of degree > 0, then the term $\langle \vec{v}_1, P'(\vec{v}_2 \otimes \ldots \otimes \vec{v}_m) \rangle$ in (7) would make very little algebraic sense because there is a multiplication on one side of P' which involves reduction modulo $X^4 - r_j$, and then there would be a multiplication on the other side which would get reduced modulo different $X^4 - r_{j'}$. But since vectors $\vec{v}_i$ only have constant terms, the "inner product" with $\vec{v}_i$ does not involve any modular reduction.

an expression which selects one element from a set P' – but this set is 32 times smaller than P. If we define $\vec{x} = P'(\vec{v}_2 \otimes \ldots \otimes \vec{v}_m)$ and send a commitment to $\vec{x}$, then proving the original set membership involves proving a new set membership proof in which the set is 32 times smaller, as well as the equation $\langle \vec{v}_1, \vec{x} \rangle = \langle \vec{w}, \vec{\gamma} \rangle$. If this latter equation can be proved with a constant number of commitments (in our case, it will essentially be one), then continuing the proof recursively would mean that the whole proof requires approximately $2m$ commitments for sets P containing $n = 32^m$ elements.

Both $\vec{w}$ and $\vec{\gamma}$ are vectors in $\mathcal{M}_q^{32k}$, so let us write them as $\vec{w} = (\vec{w}_1, \ldots, \vec{w}_k)$ and $\vec{\gamma} = (\vec{\gamma}_1, \ldots, \vec{\gamma}_k)$ where $\vec{w}_i, \vec{\gamma}_i \in \mathcal{M}_q^{32}$. Then

$$\langle \vec{v}_1, \vec{x} \rangle = \langle \vec{w}, \vec{\gamma} \rangle \Leftrightarrow g(\boldsymbol{v}_1 \boldsymbol{x}) = g\left(\sum_{i=1}^{k} \boldsymbol{w}_i \boldsymbol{\gamma}_i \right),$$

where the bold letters correspond to the inverse NTTs and the function g is the sum of the NTT's of the polynomial. Because we have BDLOP commitments to $\boldsymbol{x}$ and $\boldsymbol{w}_i$, we can compute a commitment to $\boldsymbol{y} = \boldsymbol{v}_1 \boldsymbol{x} - \sum_{i=1}^{k} \boldsymbol{w}_i \boldsymbol{\gamma}_i$, and then we just have to prove that the sum of the NTT coefficients of this polynomial is 0. For this, we employ a lemma used in [15], which states that for the ring $\mathcal{R}_q$ as defined in this section and a polynomial $\boldsymbol{y} \in \mathcal{R}_q = \sum_{i=0}^{127} y_i X^i$, we have $g(\boldsymbol{y}) = 32(y_0 + y_1 X + y_2 X^2 + y_3 X^3)$. In other words, the sum of the NTT coefficients is 0 if and only if the first four coefficients of the polynomial representation are 0. To prove this in zero knowledge, we can first commit to a masking polynomial $\boldsymbol{z}$ whose first 4 coefficients are 0 and the rest uniform in $\mathbb{Z}_q$, and then output $\boldsymbol{y} + \boldsymbol{z}$ and prove that this is indeed the right sum. The verifier can then check that the first four coefficients are 0. We don't need to multiply $\boldsymbol{y}$ by a challenge because in our case, it already contains a challenge $\vec{\gamma}$. In the body of the paper, we present an efficient way to do this proof which does not require committing to $\boldsymbol{y}$ and so we just need an extra commitment to $\vec{x} \in \mathcal{M}_q^{32}$ at each level of the recursion.

1.4 From Set Membership to Ring Signatures

A ring signature scheme allows a signer to sign in a way that hides the public key that he is using. More specifically, the signer creates a set comprised of his public key and other public keys for which he may not know the secret key. He then creates a signature with the property that the verifier can check that the message was signed by an entity who knows the secret key to one of the public keys in the list. We now sketch how one can convert a "Schnorr-like" lattice-based signature scheme into a ring signature by using a set membership proof.

The basic signature scheme underlying the ring signature follows the usual "Fiat-Shamir with Aborts" approach for constructing lattice-based digital signatures (e.g. [23, 24, 14]). In particular, the secret key is a low-norm vector $\vec{\boldsymbol{s}}$,

while the public key consists of a random matrix $\boldsymbol{A}$ and a vector $\vec{\boldsymbol{t}} = \boldsymbol{A}\vec{\boldsymbol{s}}$. The signature is then a "relaxed" zero-knowledge proof of knowledge (made non-interactive using the Fiat-Shamir transform) of a vector $\vec{\boldsymbol{s}}'$ and a polynomial $\boldsymbol{c}'$, both with small norms, satisfying $\boldsymbol{c}'\vec{\boldsymbol{t}} = \boldsymbol{A}\vec{\boldsymbol{s}}'$.

The ring signature public information consists of the matrix $\boldsymbol{A}$ and vectors $\vec{\boldsymbol{t}}_1, \ldots, \vec{\boldsymbol{t}}_n$. A signer who knows an $\vec{\boldsymbol{s}}_i$ satisfying $\boldsymbol{A}\vec{\boldsymbol{s}}_i = \vec{\boldsymbol{t}}_i$ will want to give a zero-knowledge proof knowledge of $\vec{\boldsymbol{s}}', \boldsymbol{c}'$, and $i \in [0, n)$ satisfying $\boldsymbol{c}'\vec{\boldsymbol{t}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}'$. An interactive version of this proof is presented in Figure 1 and it is then made non-interactive using the Fiat-Shamir transform and inserting the message to be signed into the random oracle which is used to produce the challenge.

Private information: $\vec{v}_1, \ldots, \vec{v}_m \in \{0,1\}^l$ as in (4), and $\vec{\boldsymbol{s}}$ with a small norm
Public information: $\boldsymbol{A}$, $\boldsymbol{T} = [\ \vec{\boldsymbol{t}}_1 \mid \ldots \mid \vec{\boldsymbol{t}}_n]$, where $n = l^m$ s.t. $\boldsymbol{T} \cdot (\vec{v}_1 \otimes \ldots \otimes \vec{v}_m) = \boldsymbol{A}\vec{\boldsymbol{s}}$

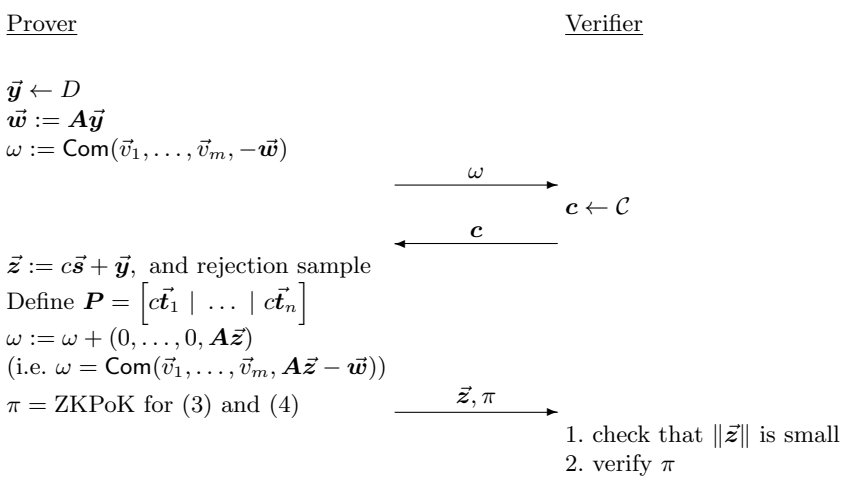

Fig. 1. A lattice-based ring signature using the set membership proof. Com is a BDLOP commitment, while D is a distribution that outputs polynomial vectors with small coefficients. As in Section 1.3, a BDLOP commitment to $\vec{v}_i$ is a commitment to the polynomial $\mathsf{NTT}^{-1}(\vec{v}_i) \in \mathcal{R}_q$.

To see that this proof is complete (assuming that all the norm-checks pass), notice that $\boldsymbol{A}\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_i = \boldsymbol{A}\vec{\boldsymbol{y}} = \vec{\boldsymbol{w}}$. And this is exactly what π proves. The zero-knowledge property follows from the fact that π is a zero-knowledge proof and that $\vec{\boldsymbol{z}}$ is independent of $\vec{\boldsymbol{s}}$ and $\boldsymbol{c}$ due to the employed rejection sampling. To see that the protocol is a proof of knowledge, note that verifying π implies that $\boldsymbol{A}\vec{\boldsymbol{z}} - c\vec{\boldsymbol{t}}_i = \vec{\boldsymbol{w}}$. Because the $\vec{v}_i$ and $\vec{\boldsymbol{w}}$ in the commitment are fixed, if we rewind the prover with a different challenge $\boldsymbol{c}'$, we will obtain $\boldsymbol{A}\vec{\boldsymbol{z}}' - \boldsymbol{c}'\vec{\boldsymbol{t}}_i = \vec{\boldsymbol{w}}$. Eliminating $\vec{\boldsymbol{w}}$ by subtracting the two equations results in the statement that we would like to extract.

Ring Size	2^3	2^5	2^6	2^{10}	2^{12}	2^{15}	2^{21}	2^{25}
Falafl [6]	30		32		35		39	
Esgin et al. [18]	19		31		59		148	
Raptor [22] / [31]+[30]	10		81		5161			
This Work		**16**		**18**		**19**		**22**

Fig. 2. Sizes, in KB, of the different lattice-based ring signature schemes with approximately 128 bits of security. The sizes for [6, 18, 22] are taken from [6, Table 1].

1.5 Bimodal Gaussians (almost) for Free

The goal of the rejection sampling in the signing algorithm is to remove the dependence of the secret key $\vec{\boldsymbol{s}}$ from the output $\vec{\boldsymbol{z}}$. If the distribution D in Figure 1 is a zero-centered discrete Gaussian, then the distribution of $\vec{\boldsymbol{z}} = \boldsymbol{c}\vec{\boldsymbol{s}} + \vec{\boldsymbol{y}}$ before rejection sampling is performed is a discrete Gaussian centered at $\boldsymbol{c}\vec{\boldsymbol{s}}$. In order for the rejection probability to not be too large (e.g. $< 1 - 1/e$), one needs the standard deviation of the $\vec{\boldsymbol{z}}$ after the rejection sampling to be around $12 \cdot \|\boldsymbol{c}\vec{\boldsymbol{s}}\|$ [24]. In [13], it was shown that if one can get the distribution of $\vec{\boldsymbol{z}}$ before rejection sampling to follow a *bimodal* Gaussian distribution with the two centers being $\pm\boldsymbol{c}\vec{\boldsymbol{s}}$, then one only needs the standard deviation of the $\vec{\boldsymbol{z}}$ after rejection sampling to be $\|\boldsymbol{c}\vec{\boldsymbol{s}}\|/\sqrt{2}$ for the same repetition rate. Such a reduction has a direct consequence on reducing the output length and increasing the SIS-hardness of the underlying problem.

The way to create a bimodal gaussian with the two centers being $\pm\boldsymbol{c}\vec{\boldsymbol{s}}$ is for the prover to choose a $y \leftarrow D$ and also a $b \leftarrow \{-1, 1\}$ and then create $\vec{\boldsymbol{z}} = b\boldsymbol{c}\vec{\boldsymbol{s}} + \vec{\boldsymbol{y}}$. It is crucial for security that b remains hidden and so the verifier is not allowed to know b or use it during verification. This could be an issue in regular signature schemes because the verifier would need to directly check that

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \boldsymbol{c}\vec{\boldsymbol{t}} + \vec{\boldsymbol{w}}. \tag{8}$$

Since $\boldsymbol{A}\vec{\boldsymbol{z}} = \boldsymbol{A}(b\boldsymbol{c}\vec{\boldsymbol{s}} + \vec{\boldsymbol{y}})$, we would need $\boldsymbol{A}\vec{\boldsymbol{s}} = -\boldsymbol{A}\vec{\boldsymbol{s}}$ to always hold. In our case, this does not hold, but it will not pose a problem because the verifier does not directly verify (8) because, for privacy, the prover cannot send $\vec{\boldsymbol{w}}$ in the clear anyway. Instead, the verifier gets $\mathsf{Com}(\vec{\boldsymbol{w}})$ and a ZK proof that this commitment opens to a $\vec{\boldsymbol{w}}$ satisfying (8). Since the prover already sends a commitment to $\vec{\boldsymbol{w}}$ along with the ones for $\vec{v}_i$ (and eventually all the "garbage terms" required in π), he can just increase the commitment size by one (128-degree) polynomial and also commit to b. Then the proof π would need to be modified to prove that

$$[b\boldsymbol{c}\vec{\boldsymbol{t}}_1 \mid \ldots \mid b\boldsymbol{c}\vec{\boldsymbol{t}}_n] \cdot (\vec{v}_1 \otimes \ldots \otimes \vec{v}_m) = \vec{\boldsymbol{w}} - \boldsymbol{A}\vec{\boldsymbol{z}}.$$

Notice that because $b \in \{-1, 1\}$ and all the $\vec{v}_i$ consist of all 0's and one 1, this can be rewritten as

$$[\boldsymbol{c}\vec{\boldsymbol{t}}_1 \mid \ldots \mid \boldsymbol{c}\vec{\boldsymbol{t}}_n] \cdot (b\vec{v}_1 \otimes \vec{v}_2 \otimes \ldots \otimes \vec{v}_m) = \vec{\boldsymbol{w}} - \boldsymbol{A}\vec{\boldsymbol{z}},$$

and so the only thing that changes is that instead of committing to $\vec{v}_1$, the prover commits to $b\vec{v}_1$. He then just has to show that the coefficients of $b\vec{v}_1$ are in $\{0, b\}$ rather than $\{0, 1\}$ – but this proof is exactly the same if we already have a commitment to b (which we proved to be in $\{-1, 1\}$).

1.6 Application to Confidential Transactions

We now show how to construct a confidential transaction system in the model of [18]. The setup is the following: at any given moment, the state (which is managed by the blockchain, and is outside the scope of this work) consists of a set of accounts $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$, each of which contains a public key and a coin. The state also contains a set of serial numbers which implicitly correspond to the accounts that were already spent (to prevent double-spending). The secret account key associated to each account is $\mathsf{ask} = (\mathsf{sk}, \mathsf{ck}, \mathsf{amt})$, which consists of the secret key corresponding to pk and the commitment key ck, which is the randomness used to create the BDLOP commitment cn to the amount amt in the account. As in [18], we will assume that amt takes values between 0 and $2^{64} - 1$. Since we are working over rings with 32 NTT slots, we will represent the values in base 4. The basic operation has the sender choosing M input accounts for which he knows the secret keys associated to $\mathsf{pk}^{(1)}, \ldots, \mathsf{pk}^{(M)}$, and then creating S new output accounts with given public keys for which he does not need to know the associated secret keys. There are three correctness constraints. The first is that the spender knows the associated secret keys for the M input accounts. The second is that the sum of the values of the input coins (i.e. the sum of the amt) equals to the sum of the values of the output coins. And the third is that none of the M input accounts were used as inputs in any previous transaction.

In addition to correctness, there are also secrecy and anonymity requirements. The secrecy requirement states that nothing about the amounts amt is known except that the sum of the input and output coins is equal. The spender's anonymity is defined by hiding the spenders account among N other accounts. In particular, rather than stating which M accounts the spender is using, he will instead choose M sets of N accounts each, and then choose one account from each set in a way that hides which of the N accounts has been chosen. How the spender chooses the $N - 1$ other accounts is a policy issue that is outside the scope of this work.

The public information for the system consists of a polynomial matrix $\boldsymbol{B}$ which forms the "top part" of the BDLOP commitment. The polynomial vectors $\vec{\boldsymbol{b}}_c$ (which will be used to commit to amt) and $\vec{\boldsymbol{b}}_s$ (which will be used to "commit" to zero, with the commitment being the serial number) form the "bottom part" of the commitments. In particular, sk is a low-norm vector $\vec{\boldsymbol{s}}$ where

$$\begin{bmatrix} \boldsymbol{B} \\ \vec{\boldsymbol{b}}_s \end{bmatrix} \vec{\boldsymbol{s}} = \begin{bmatrix} \mathsf{pk} \\ \mathsf{sn} \end{bmatrix}. \tag{9}$$

And ck is another low-norm vector $\vec{\boldsymbol{r}}$ such that

$$\begin{bmatrix} \boldsymbol{B} \\ \vec{\boldsymbol{b}}_c \end{bmatrix} \vec{\boldsymbol{r}} + \begin{bmatrix} 0 \\ \mathsf{amt} \end{bmatrix} = \mathsf{cn}. \tag{10}$$

Correctness. Let's ignore anonymity for a moment, and just briefly discuss how the correctness of the protocol could be handled. If the spender wants to spend accounts $\mathsf{act}^{(1)}, \ldots, \mathsf{act}^{(M)}$, then he outputs the values $\mathsf{sn}^{(j)}, \vec{\boldsymbol{s}}^{(j)}, \vec{\boldsymbol{r}}^{(j)}, \mathsf{amt}^{(j)}$ for the input accounts, and the verifier can check that (9) and (10) are satisfied. Furthermore, the verifier checks that none of the $\mathsf{sn}^{(j)}$ are in the set of used serial numbers, and adds these $\mathsf{sn}^{(j)}$ to the set. Note that because the value of $\vec{\boldsymbol{s}}^{(j)}$ is uniquely determined by $\boldsymbol{B}$ and pk (unless SIS is easy), the value of sn is uniquely tied to pk; and so it is not possible to spend a coin more than once. The spender then creates valid output tokens with the values of pk that he is given and creates the output coins with by picking small vectors $\vec{\boldsymbol{r}}$ and using them to create BDLOP commitments to amt as in (10). He then outputs these $\vec{\boldsymbol{r}}$ and amt so that everyone can check that the sum of the input amounts is equal to the sum of the output amounts.

Anonymity and Secrecy. We now sketch how anonymity and secrecy is achieved in our confidential transactions protocol. The spender chooses the M accounts $\mathsf{act}^{(j)} = (\mathsf{pk}^{(j)}, \mathsf{cn}^{(j)})$ that he wants to spend. He puts each of the right hand sides of (10) (i.e. the coin commitments) from these accounts into M lists $\boldsymbol{T}^{(j)}$, one coin per list. The rest of the lists are filled with N coins from accounts among which the spender wants to hide his. He then creates S output accounts $\mathsf{act}^{(j)} = (\mathsf{pk}^{(j)}, \mathsf{cn}^{(j)})$ using the given public keys. He does not need to hide these accounts and so he just creates S lists of size 1 for the output coins. He then wants to create one BDLOP commitment that includes all the coin values (i.e. the amt) from the input and output tokens. This protocol is described in Figure 6. Once the spender has one BDLOP commitment, he can prove that the sum of the input and output tokens matches, which can be done using techniques similar to those in [18, 25].

The prover also needs to show that he knows $\vec{\boldsymbol{s}}$ that satisfy (9) for the input accounts. He does this by creating M lists $\boldsymbol{U}^{(j)}$ that are derived from $\boldsymbol{T}^{(j)}$. If the spender's coin is in position i in the list $\boldsymbol{T}^{(j)}$, then he puts $\begin{bmatrix} \mathsf{pk}_i^{(j)} \\ \mathsf{sn}^{(j)} \end{bmatrix}$ into position i. He then fills the list with the public keys from the accounts corresponding to the coins in $\boldsymbol{T}^{(j)}$. For the serial numbers, he attaches the same one (i.e. the one corresponding to his public key) to all the public keys. In particular, if the spender wants to hide the j^{th} account that he will be using in position i among $N-1$ other accounts $\mathsf{act}_1, \ldots, \mathsf{act}_{i-1}, \mathsf{act}_{i+1}, \ldots, \mathsf{act}_N$, then the lists $\boldsymbol{T}^{(j)}$ and $\boldsymbol{U}^{(j)}$ are

$$\boldsymbol{T}^{(j)} = \left[\mathsf{cn}_1^{(j)}, \ldots, \mathsf{cn}_N^{(j)}\right]$$

$$\boldsymbol{U}^{(j)} = \left[\begin{bmatrix} \mathsf{pk}_1^{(j)} \\ \mathsf{sn}^{(j)} \end{bmatrix}, \cdots, \begin{bmatrix} \mathsf{pk}_{i-1}^{(j)} \\ \mathsf{sn}^{(j)} \end{bmatrix}, \begin{bmatrix} \mathsf{pk}_i^{(j)} \\ \mathsf{sn}^{(j)} \end{bmatrix}, \begin{bmatrix} \mathsf{pk}_{i+1}^{(j)} \\ \mathsf{sn}^{(j)} \end{bmatrix}, \ldots, \begin{bmatrix} \mathsf{pk}_N^{(j)} \\ \mathsf{sn}^{(j)} \end{bmatrix}\right]$$

(M, S)	ring size N 2^5	2^{10}	2^{15}	2^{20}	2^{25}
$(1, 2)$ This Work	22 KB	24 KB	25 KB	27 KB	28 KB
$(1, 2)$ Esgin et al. [18]	100 KB	160 KB	250 KB	375 KB	520 KB
$(2, 2)$ This Work	24 KB	27 KB	30 KB	33 KB	36 KB
$(2, 2)$ Esgin et al. [18]	110 KB	190 KB	300 KB	440 KB	660 KB

Fig. 3. Transaction proof sizes depending on ring size (anonymity set size) N, number M of input accounts, and number S of output accounts. The sizes for [18] are taken from [18, Figure 1].

M	25	50	75	100
size (This Work $N = 1024$)	100 KB	180 KB	262 KB	345 KB
size (Esgin et al. [18] $N = 100$)	370 KB	610 KB	900 KB	1170 KB

Fig. 4. Transaction proof sizes with M input accounts and $S = 2$ output accounts. The anonymity set N is 100 in [18] and $32^2 = 1024$ in our work. The sizes for [18] are taken from [18, Figure 2].

For the lists $\boldsymbol{U}^{(j)}$, the spender simply wants to prove that he knows the secret keys $\vec{\boldsymbol{s}}^{(j)}$ for the elements in the same position as those in $\boldsymbol{T}^{(j)}$. Since the positions are already committed to, the proof of knowledge of the $\vec{\boldsymbol{s}}^{(j)}$ does not require any extra BDLOP commitments and the proof of knowledge of the $\vec{\boldsymbol{s}}^{(j)}$ can be amortized into the output vector $\vec{\boldsymbol{z}}$ in Figure 6. The verifier will need to check that the serial numbers $\mathsf{sn}^{(j)}$ have never been used (i.e. don't appear in the "used" pile) and that the lists $\boldsymbol{T}^{(j)}, \boldsymbol{m}^{(j)}$ are valid (i.e. the positions $\mathsf{pk}_i^{(j)}$ in list $\boldsymbol{T}^{(j)}$ and $\mathsf{cn}_i^{(j)}$ in list $\boldsymbol{U}^{(j)}$ correspond to some account $\mathsf{act} = \left(\mathsf{pk}_i^{(j)}, \mathsf{cn}_i^{(j)}\right)$). The verifier also has to verify the proof from Figure 6 and the addition proof confirming that the amounts in the input and output accounts match.

The protocol in Figure 6, which is at the center of the confidential transaction protocol, creates a new BDLOP commitment and proves that it is committing to the same values as the M input and S output accounts. It additionally proves that the spender knows the secret keys of the M input accounts. This involves using the protocol for the k-dimensional version of the set membership problem as well as an amortization technique which will allow us to only send one "masked value" for all the randomness used in the $M+S$ accounts.

Aggregating BDLOP Commitments. Before describing the protocol in Figure 6, we ignore the part where each of the M input accounts are hidden among N others, and give a simpler protocol in Figure 5 that takes k BDLOP commitments with distinct randomnesses, and creates one BDLOP commitment to the same messages. The improvement in this protocol over the trivial one is in the fact that only one output $\vec{\boldsymbol{z}}$ is enough to prove knowledge that all k commitments

are valid. The norm of this vector $\vec{z}$ is larger by a factor of k (or $\sqrt{k}$ in the asymptote), so its representation grows only logarithmically in k.

The protocol in Figure 5 takes as input k BDLOP commitments under randomness $\vec{s}_i$ and produces one BDLOP commitment ω under randomness $\vec{r}$. The commitment includes all the $\boldsymbol{m}_i$ and one additional "garbage polynomial" $\tilde{\boldsymbol{w}}$. When the prover computes and outputs $\vec{z}$, he proves that all the k commitments under $\vec{s}_i$ are valid. The rest of the steps are needed to show that the commitment under $\vec{r}$ is to the same $\boldsymbol{m}_i$. We discuss this in more detail below.

The proof that the k commitments are valid follows from the ideas in [4] where one does rewinding by keeping most of the challenge fixed. As long as the new challenge still has κ bits of entropy conditioned on the prior challenge, the soundness error will still be $\approx 2^{-\kappa}$. Without loss of generality, suppose that we would like to prove that the new commitment is a commitment to $\boldsymbol{m}_1$ (in the row that contains $\boldsymbol{g}_1$). Let $(\vec{\boldsymbol{w}}, \omega, \boldsymbol{c}_1, \boldsymbol{c}_2 \ldots, \boldsymbol{c}_k, \vec{\boldsymbol{z}}, \pi)$ be the transcript of one run and $(\vec{\boldsymbol{w}}, \omega, \boldsymbol{c}_1', \boldsymbol{c}_2 \ldots, \boldsymbol{c}_k, \vec{\boldsymbol{z}}', \pi')$ be the view of the second run when we rewind while keeping all the challenges, except for $\boldsymbol{c}_1$ fixed.

Rewinding on the second verification equation, we obtain $(\boldsymbol{c}_1 - \boldsymbol{c}_1')\vec{\boldsymbol{t}}_1 = \boldsymbol{A}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}')$. By (2), this implies that the message $\boldsymbol{m}_i$ committed to by $\begin{bmatrix} \vec{\boldsymbol{t}}_1 \\ \boldsymbol{u}_1 \end{bmatrix}$ satisfies

$$(\boldsymbol{c}_1 - \boldsymbol{c}_1')\boldsymbol{m}_1 = (\boldsymbol{c} - \boldsymbol{c}')\boldsymbol{u}_1 - \vec{\boldsymbol{b}} \cdot (\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}'). \tag{11}$$

Notice that repeating this for all i, we can prove that all the commitments $\begin{bmatrix} \vec{\boldsymbol{t}}_i \\ \boldsymbol{u}_i \end{bmatrix}$ are valid. The intuition for proving that ω is a commitment to the same messages is to prove that the messages in the commitment of ω (call them $\bar{\boldsymbol{m}}_i$ and $\bar{\boldsymbol{w}}$) satisfy the linear equation

$$\sum_i \boldsymbol{c}_i \bar{\boldsymbol{m}}_i = \sum_i \boldsymbol{c}_i \boldsymbol{u}_i + \bar{\boldsymbol{w}} - \vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{z}}. \tag{12}$$

Rewinding in the same way as above, we would obtain

$$(\boldsymbol{c}_1 - \boldsymbol{c}_1')\bar{\boldsymbol{m}}_1 = (\boldsymbol{c}_1 - \boldsymbol{c}_1')\boldsymbol{u}_1 - \vec{\boldsymbol{b}} \cdot (\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}').$$

Substituting $\vec{\boldsymbol{b}} \cdot (\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}')$ from (11), we get $(\boldsymbol{c}_1 - \boldsymbol{c}_1')\bar{\boldsymbol{m}}_1 = (\boldsymbol{c}_1 - \boldsymbol{c}_1')\boldsymbol{m}_1$. And since $\boldsymbol{c}_1 - \boldsymbol{c}_1'$ is invertible, we have $\boldsymbol{m}_1 = \bar{\boldsymbol{m}}_1$ as desired.

We now observe that we exactly prove (12). The proof π proves that ω is a valid commitment and therefore there is a unique $\vec{\boldsymbol{v}}$ (and a short polynomial d s.t. $d\vec{\boldsymbol{v}}$ has small norm) satisfying $\boldsymbol{g}_i - \vec{\boldsymbol{a}}_i \cdot \vec{\boldsymbol{v}} = \bar{\boldsymbol{m}}_i$ and $\boldsymbol{g}_w - \vec{\boldsymbol{a}}_w \cdot \vec{\boldsymbol{v}} = \bar{w}$. Because we also prove that $\sum \boldsymbol{c}_i \boldsymbol{u}_i$ is a valid commitment, it implies that $\langle \vec{\boldsymbol{a}}^*, \vec{\boldsymbol{v}} \rangle + \sum \boldsymbol{c}_i \boldsymbol{v}_i = \boldsymbol{g}^*$. If we expand out the definitions of $\vec{\boldsymbol{a}}^*$ and $\boldsymbol{g}^*$, and then plug it in, along with the expressions for $(\boldsymbol{c}_i - \boldsymbol{c}_i')\boldsymbol{g}_i$ and $(\boldsymbol{c}_i - \boldsymbol{c}_i')\boldsymbol{g}_w$, into the previous equation, we will exactly end up with (12).

We now sketch the zero-knowledge proof. By assumption, π can be simulated and $\vec{\boldsymbol{z}}$ is independent of $\vec{\boldsymbol{s}}_i$ and $\boldsymbol{c}_i$ by rejection sampling. The BDLOP commitment ω is indistinguishable from uniform by the LWE assumption, and

Private information: For $1 \leq i \leq k$, polynomials $\boldsymbol{m}_i$, low-norm vectors $\vec{\boldsymbol{s}}_i$
Public information: Uniformly random $\boldsymbol{B}, \vec{\boldsymbol{b}}, \boldsymbol{A}, \vec{\boldsymbol{a}}_w, \vec{\boldsymbol{a}}_i, \begin{bmatrix} \vec{\boldsymbol{t}}_i \\ \boldsymbol{u}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{B} \\ \vec{\boldsymbol{b}} \end{bmatrix} \vec{\boldsymbol{s}}_i + \begin{bmatrix} \vec{\boldsymbol{0}} \\ \boldsymbol{m}_i \end{bmatrix}$

Prover		Verifier
$(\vec{\boldsymbol{y}}, \vec{\boldsymbol{r}}) \leftarrow D_y \times D_r$ $\vec{\boldsymbol{w}} := \boldsymbol{B}\vec{\boldsymbol{y}};\ \tilde{\boldsymbol{w}} := \vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{y}}$ $\begin{bmatrix} \boldsymbol{A} \\ \vec{\boldsymbol{a}}_1 \\ \dots \\ \vec{\boldsymbol{a}}_k \\ \vec{\boldsymbol{a}}_w \end{bmatrix} \vec{\boldsymbol{r}} + \begin{bmatrix} \vec{\boldsymbol{0}} \\ \boldsymbol{m}_1 \\ \dots \\ \boldsymbol{m}_k \\ \tilde{\boldsymbol{w}} \end{bmatrix} = \begin{bmatrix} \vec{\boldsymbol{f}} \\ \boldsymbol{g}_1 \\ \dots \\ \boldsymbol{g}_k \\ \tilde{\boldsymbol{g}}_w \end{bmatrix} = \omega$	$\xrightarrow{\vec{\boldsymbol{w}}, \omega}$	
		$\boldsymbol{c}_1, \dots, \boldsymbol{c}_k \leftarrow \mathcal{C}$
	$\xleftarrow{\boldsymbol{c}_1, \dots, \boldsymbol{c}_k}$	
$\vec{\boldsymbol{z}} := \vec{\boldsymbol{y}} + \sum \boldsymbol{c}_i \vec{\boldsymbol{s}}_i$, and rejection sample $\vec{\boldsymbol{a}}^* := \sum_{i=1}^{k} \boldsymbol{c}_i \vec{\boldsymbol{a}}_i - \vec{\boldsymbol{a}}_w$ $\boldsymbol{g}^* := \sum_{i=1}^{k} \boldsymbol{c}_i \boldsymbol{g}_i - \tilde{\boldsymbol{g}}_w + \vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{z}}$ $\pi =$ ZKPoK that $\begin{bmatrix} \vec{\boldsymbol{f}} \\ \boldsymbol{g}^* \end{bmatrix}$ under public key $\begin{bmatrix} \boldsymbol{A} \\ \vec{\boldsymbol{a}}^* \end{bmatrix}$ is a commitment to $\sum_{i=1}^{k} \boldsymbol{c}_i \boldsymbol{u}_i$	$\xrightarrow{\vec{\boldsymbol{z}}, \pi}$	
		1. check that $\|\vec{\boldsymbol{z}}\|$ is small 2. check that $\sum_{i=1}^{k} \boldsymbol{c}_i \vec{\boldsymbol{t}}_i = \boldsymbol{B}\vec{\boldsymbol{z}} - \vec{\boldsymbol{w}}$ 3. Compute $\vec{\boldsymbol{a}}^*, \boldsymbol{g}^*$ and verify π

Fig. 5. A protocol which takes commitments $\begin{bmatrix} \vec{\boldsymbol{t}}_i \\ \boldsymbol{u}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{B} \\ \vec{\boldsymbol{b}} \end{bmatrix} \vec{\boldsymbol{s}}_i + \begin{bmatrix} \vec{\boldsymbol{0}} \\ \boldsymbol{m}_i \end{bmatrix}$ to $\boldsymbol{m}_i$ under distinct randomnesses $\vec{\boldsymbol{s}}_i$, and outputs one BDLOP commitment ω to all the $\boldsymbol{m}_i$ (and some auxiliary garage term(s)) under one common randomness $\vec{\boldsymbol{r}}$. Along with outputting the commitment, the protocol also proves that $\begin{bmatrix} \vec{\boldsymbol{t}}_i \\ \boldsymbol{u}_i \end{bmatrix}$ are valid commitments and that the new commitment is to the same $\boldsymbol{m}_i$.

Private information: For $1 \leq j \leq k$, $V^{(j)} = (\vec{v}_1^{(j)}, \ldots, \vec{v}_m^{(j)}) \in \{0,1\}^{l \times m}$ s.t. $\|\vec{v}_i^{(j)}\|_1 = 1$, $\vec{\boldsymbol{s}}^{(j)}$ with a small norm, and message polynomials $\boldsymbol{m}^{(j)}$

Public information: $\boldsymbol{B}$, $\vec{\boldsymbol{b}}$, $\boldsymbol{T}^{(j)} = \left[\begin{bmatrix} \vec{\boldsymbol{t}}_1^{(j)} \\ \boldsymbol{u}_1^{(j)} \end{bmatrix} \mid \ldots \mid \begin{bmatrix} \vec{\boldsymbol{t}}_n^{(j)} \\ \boldsymbol{u}_n^{(j)} \end{bmatrix} \right]$, where $n = l^m$, s.t

$\boldsymbol{T}^{(j)} \cdot (\vec{v}_1^{(j)} \otimes \ldots \otimes \vec{v}_m^{(j)}) = \begin{bmatrix} \boldsymbol{B} \\ \vec{\boldsymbol{b}} \end{bmatrix} \vec{\boldsymbol{s}}^{(j)} + \begin{bmatrix} \vec{\boldsymbol{0}} \\ \boldsymbol{m}^{(j)} \end{bmatrix}$

Prover		Verifier
$\vec{\boldsymbol{y}} \leftarrow D$ $\vec{\boldsymbol{w}} := \boldsymbol{B}\vec{\boldsymbol{y}}$ $\tilde{\boldsymbol{w}} := \vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{y}}$ $\omega = \mathsf{Com}(\boldsymbol{m}^{(1)}, \ldots, \boldsymbol{m}^{(k)}, V^{(1)}, \ldots, V^{(k)}, \tilde{\boldsymbol{w}}, -\vec{\boldsymbol{w}})$	$\xrightarrow{\omega}$	
		$\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(k)} \leftarrow \mathcal{C}$
	$\xleftarrow{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(k)}}$	
$\vec{\boldsymbol{z}} := \vec{\boldsymbol{y}} + \sum \boldsymbol{c}^{(j)} \vec{\boldsymbol{s}}^{(j)}$, and rejection sample Define $\boldsymbol{P}^{(i)} = \boldsymbol{c}^{(i)} \boldsymbol{T}^{(i)}$ Define $\boldsymbol{g}^* := \sum_{j=1}^{k} \boldsymbol{c}^{(j)} \boldsymbol{g}^{(j)} - \tilde{\boldsymbol{g}}^{(w)} + \vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{z}}$ $\pi =$ ZKPoK that $\begin{bmatrix} \boldsymbol{B}\vec{\boldsymbol{z}} - \vec{\boldsymbol{w}} \\ \boldsymbol{g}^* \end{bmatrix}$ is a commitment to $\sum_{j=1}^{k} \boldsymbol{P}^{(j)} \cdot (\vec{v}_1^{(j)} \otimes \cdots \otimes \vec{v}_m^{(j)})$	$\xrightarrow{\vec{\boldsymbol{z}}, \pi}$	
		1. check that $\|\vec{\boldsymbol{z}}\|$ is small 2. verify π

Fig. 6. Given $\boldsymbol{T}^{(j)} \cdot (\vec{v}_1^{(j)} \otimes \ldots \otimes \vec{v}_m^{(j)}) = \begin{bmatrix} \boldsymbol{B} \\ \vec{\boldsymbol{b}} \end{bmatrix} \vec{\boldsymbol{s}}^{(j)} + \begin{bmatrix} \vec{\boldsymbol{0}} \\ \boldsymbol{m}^{(j)} \end{bmatrix}$, the prover creates a BDLOP commitment to all the k $\boldsymbol{m}^{(j)}$ and proves its correctness. The new commitment Com uses public matrices (e.g. $\boldsymbol{A}$, etc. as in Figure 5) which we do not explicitly state in this sketch. The terms comprising $\boldsymbol{g}^*$ are parts of ω, and are described in detail in the protocol in Figure 5 (except with subscripts instead of superscripts).

$\vec{\boldsymbol{w}}$ is unique once $\vec{\boldsymbol{z}}$ and $\boldsymbol{c}_i$ are chosen. Something worth noting is that while $\vec{\boldsymbol{w}} = \boldsymbol{B}\vec{\boldsymbol{y}}$ can be sent in the clear, the value $\tilde{\boldsymbol{w}} = \vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{y}}$ needs to be sent as part of a commitment because revealing it in the clear would end up revealing some function of the $\boldsymbol{m}_i$.

Aggregation and Set Membership. Converting the protocol from Figure 5 into the one in Figure 6 uses very similar intuition as when converting a signature scheme into a ring signature scheme in Figure 1.

We will now proceed to briefly explain the transition from the protocol in Figure 5 to the one in Figure 6. First, the second verifier check in Figure 5 cannot

be done in the clear – that is the verifier cannot know $\vec{\boldsymbol{w}}$. If he knows $\vec{\boldsymbol{w}}$, then he can compute the weighted sum of the committed values $\sum \boldsymbol{c}_i \vec{\boldsymbol{t}}_i$, which would leak information about which commitments were chosen. The prover therefore must commit to $\vec{\boldsymbol{w}}$. So the commitment ω in Figure 6 creates commitments to $\boldsymbol{m}^{(j)}, \tilde{\boldsymbol{w}}$ exactly like to $\boldsymbol{m}_i, \tilde{\boldsymbol{w}}$ in Figure 5, and also commits to $\vec{\boldsymbol{w}}$ and to $V^{(j)}$, which are needed for the set membership proof.

The prover then sets up the $\vec{\boldsymbol{a}}^*$ and $\boldsymbol{g}^*$ exactly as in Figure 5. Therefore $\boldsymbol{g}^*$ is a commitment to the bottom part of $\sum_{j=1}^{k} \boldsymbol{c}^{(j)} \boldsymbol{T}^{(j)} \cdot (\vec{v}_1^{(j)} \otimes \cdots \otimes \vec{v}_m^{(j)})$. From the second verification equation in Figure 5, we know that the top part of the preceding is $\boldsymbol{B}\vec{\boldsymbol{z}} - \vec{\boldsymbol{w}}$, and we can create a commitment to this value by adding $\boldsymbol{B}\vec{\boldsymbol{z}}$ to the commitment of $-\vec{\boldsymbol{w}}$ that we already have. We therefore have a commitment to $\sum_{j=1}^{k} \boldsymbol{c}^{(j)} \boldsymbol{T}^{(j)} \cdot (\vec{v}_1^{(j)} \otimes \cdots \otimes \vec{v}_m^{(j)})$ and creating the proof π is therefore equivalent to creating a proof for (5) and (6). Showing that this protocol is sound is done the same way as the one in Figure 5 because $\vec{\boldsymbol{z}}$ and π in Figure 6 satisfy the three verification parts in Figure 5.

2 Preliminaries

2.1 Notation

Let $\mathsf{N} \in \mathbb{N}$ be a security parameter and q be an odd prime. We write $x \leftarrow S$ when $x \in S$ is sampled uniformly at random from the finite set S and similarly $x \leftarrow D$ when x is sampled according to the distribution D. For $a < b$ and $n \in \mathbb{N}$, we define $[a, b] := \{a, a+1 \ldots, b\}$ and $[n] := [1, n]$. Given two functions $f, g : \mathbb{N} \to [0, 1]$, we write $f(\mu) \approx g(\mu)$ if $|f(\mu) - g(\mu)| < \mu^{-\omega(1)}$. A function f is negligible if $f \approx 0$. We write $\mathsf{negl}(n)$ to denote an unspecified negligible function in n.

For a power of two d, denote $\mathcal{R}$ and $\mathcal{R}_q$ respectively to be the rings $\mathbb{Z}[X]/(X^d+1)$ and $\mathbb{Z}_q[X]/(X^d+1)$. Bold lower-case letters denote elements in $\mathcal{R}$ or $\mathcal{R}_q$ and bold lower-case letters with arrows represent column vectors with coefficients in $\mathcal{R}$ or $\mathcal{R}_q$. We also write bold upper-case letters for matrices in $\mathcal{R}$ or $\mathcal{R}_q$. By default, for a polynomial denoted as a bold letter, we write its i-th coefficient as its corresponding regular font letter subscript i, e.g. $f_0 \in \mathbb{Z}_q$ is a constant coefficient of $\boldsymbol{f} \in \mathcal{R}_q$.

2.2 Cyclotomic Rings

Suppose q splits into l prime ideals of degree d/l in $\mathcal{R}$. This means $X^d + 1 \equiv \boldsymbol{\varphi}_1 \ldots \boldsymbol{\varphi}_l \pmod q$ with irreducible polynomials $\boldsymbol{\varphi}_j$ of degree d/l modulo q. We assume that $\mathbb{Z}_q$ contains a primitive $2l$-th root of unity $\zeta \in \mathbb{Z}_q$ but no elements whose order is a higher power of two, i.e. $q - 1 \equiv 2l \pmod{4l}$. Therefore, we have

$$X^d + 1 \equiv \prod_{j \in \mathbb{Z}_l} \left(X^{\frac{d}{l}} - \zeta^{2j+1} \right) \pmod q. \tag{13}$$

Let $\mathcal{M}_q := \{\boldsymbol{p} \in \mathbb{Z}_q[X] : \deg(\boldsymbol{p}) < d/l\}$ be the $\mathbb{Z}_q$-module of polynomials of degree less than d/l. We define the Number Theoretic Transform (NTT) of a polynomial $\boldsymbol{p} \in \mathcal{R}_q$ as follows:

$$\mathsf{NTT}(\boldsymbol{p}) := \begin{bmatrix} \hat{\boldsymbol{p}}_0 \\ \vdots \\ \hat{\boldsymbol{p}}_{l-1} \end{bmatrix} \in \mathcal{M}_q^l \text{ where } \mathsf{NTT}(\boldsymbol{p})_j = \hat{\boldsymbol{p}}_j = \boldsymbol{p} \bmod (X^{\frac{d}{l}} - \zeta^{2j+1}).$$

Furthermore, we expand the definition of NTT to vectors of polynomials $\vec{\boldsymbol{p}} \in \mathcal{R}_q^k$, where the NTT operation is applied to each coefficient of $\vec{\boldsymbol{p}}$, resulting in a vector in $\mathcal{M}_q^{kl}$.

We also define the inverse NTT operation. Namely, for a vector $\vec{v} \in \mathcal{M}_q^l$, $\mathsf{NTT}^{-1}(\vec{v})$ is the polynomial $\boldsymbol{p} \in \mathcal{R}_q$ such that $\mathsf{NTT}(\boldsymbol{p}) = \vec{v}$.

Let $\vec{v} = (v_0, \ldots, v_{l-1}), \vec{w} = (w_0, \ldots, w_{l-1}) \in \mathcal{M}_q^l$. Then, we define the component-wise product $\vec{v} \circ \vec{w}$ to be the vector $\vec{u} = (u_0, \ldots, u_{l-1}) \in \mathcal{M}_q^l$ such that

$$u_j = v_j w_j \bmod (X^{\frac{d}{l}} - \zeta^{2j+1})$$

for $j \in \mathbb{Z}_l$. By definition, we have the following property of the inverse NTT operation:

$$\mathsf{NTT}^{-1}(\vec{v}) \cdot \mathsf{NTT}^{-1}(\vec{w}) = \mathsf{NTT}^{-1}(\vec{v} \circ \vec{w}).$$

Similarly, we define the *inner product*:

$$\langle \vec{v}, \vec{w} \rangle = \sum_{j=0}^{l-1} \left(v_j w_j \bmod (X^{\frac{d}{l}} - \zeta^{2j+1}) \right).$$

We remark that this operation is not an inner product in the strictly mathematical sense (e.g. it is not linear). However, it has a few properties which are characteristic for an inner product. For instance, given arbitrary vectors $\vec{x}, \vec{y}, \vec{z} \in \mathcal{M}_q^l$ and scalar $c \in \mathbb{Z}_q$ we have: $\langle \vec{x}, \vec{y} \rangle = \langle \vec{y}, \vec{x} \rangle$ (symmetry), $\langle \vec{x} + \vec{y}, \vec{z} \rangle = \langle \vec{x}, \vec{z} \rangle + \langle \vec{y}, \vec{z} \rangle$ (distributive law) and $\langle c\vec{x}, \vec{y} \rangle = c\langle \vec{x}, \vec{z} \rangle$. We also highlight that the definition of $\langle \cdot, \cdot \rangle$ depends on the factors of $X^d + 1$ modulo q.

We generalise the newly introduced operations to work for vectors $\vec{v} = (\vec{v}_1, \ldots, \vec{v}_k)$ and $\vec{w} = (\vec{w}_1, \ldots, \vec{w}_k) \in \mathcal{M}_q^{kl}$ of length being a multiple of l in the usual way. In particular $\langle \vec{v}, \vec{w} \rangle = \sum_{i=1}^{k} \langle \vec{v}_i, \vec{w}_i \rangle$.

Eventually, for a matrix $A \in \mathcal{M}_q^{n \times kl}$ with rows $\vec{a}_1, \ldots, \vec{a}_n \in \mathcal{M}_q^{kl}$ and a vector $\vec{v} \in \mathcal{M}_q^{kl}$, we define the matrix-vector operation:

$$A\vec{v} = \begin{pmatrix} \langle \vec{a}_1, \vec{v} \rangle \\ \vdots \\ \langle \vec{a}_n, \vec{v} \rangle \end{pmatrix} \in \mathcal{M}_q^n.$$

In proving linear relations, we will need the following simple lemma.

Lemma 2.1. *Let $n, k \in \mathbb{N}$. Then, for any $A \in \mathcal{M}_q^{nl \times kl}, \vec{v} \in \mathcal{M}_q^{nl}$ and $\vec{s} \in \mathbb{Z}_q^{kl}$ we have*

$$\langle A\vec{s}, \vec{v} \rangle = \langle \vec{s}, A^T \vec{v} \rangle.$$

Proof. We prove the statement for $k = n = 1$. The proof can then be easily using the definition of an inner product. Let $\vec{a}_i$ be the $(i+1)$-th row of A and $a_{i,j} \in \mathcal{M}_q$ be its $(j+1)$-th coefficient. Similarly, we define s_i and v_i to be the $(i+1)$-th coefficient of $\vec{s}$ and $\vec{v}$ respectively. Then, by definition we have:

$$\begin{aligned}
\langle A\vec{s}, \vec{v} \rangle &= \sum_{i=0}^{l-1} \langle \vec{a}_i, \vec{s} \rangle v_i \bmod (X^{\frac{d}{l}} - \zeta^{2i+1}) \\
&= \sum_{i=0}^{l-1} \left(\sum_{j=0}^{l-1} a_{i,j} s_j \bmod (X^{\frac{d}{l}} - \zeta^{2j+1}) \right) v_i \bmod (X^{\frac{d}{l}} - \zeta^{2i+1}) \\
&= \sum_{i=0}^{l-1} \sum_{j=0}^{l-1} a_{i,j} s_j v_i \bmod (X^{\frac{d}{l}} - \zeta^{2i+1}) \\
&= \sum_{j=0}^{l-1} s_j \left(\sum_{i=0}^{l-1} a_{i,j} v_i \bmod (X^{\frac{d}{l}} - \zeta^{2i+1}) \right) \\
&= \langle \vec{s}, A^T \vec{v} \rangle.
\end{aligned} \tag{14}$$

Here, the crucial step was the observation that for $\vec{s} \in \mathbb{Z}_q^l$ and any $i, j \in \mathbb{Z}_l$ we have:

$$a_{i,j} s_j \bmod (X^{\frac{d}{l}} - \zeta^{2j+1}) = a_{i,j} s_j,$$

i.e. there is no reduction modulo the polynomial when multiplying by a scalar. □

Last but not least, we recall the following lemma from [15].

Lemma 2.2. *Let $\boldsymbol{p} = p_0 + p_1 X + \ldots + p_{d-1} X^{d-1} \in \mathcal{R}_q$. Then,*

$$\frac{1}{l} \sum_{i=0}^{l} \mathsf{NTT}(\boldsymbol{p})_i = \sum_{i=0}^{d/l-1} p_i X^i.$$

For our constructions in this work, the practical hardness of either of the problems against known attacks is not affected by the parameter m. Therefore, we sometimes simply write M-SIS$_{\kappa,B}$ or M-LWE$_{\lambda,\chi}$. The parameters κ and λ denote the *module ranks* for M-SIS and M-LWE, respectively. Also, when χ is a uniform distribution for the set $[-\mu, \mu]$, we simply denote M-LWE$_{\lambda,\mu}$.

2.3 Probability Distributions

In this paper we sample the coefficients of the random polynomials in the commitment scheme using the distribution χ on $\{-1, 0, 1\}$ where ± 1 both have probability 5/16 and 0 has probability 6/16 identically as in [8, 1, 15].

Discrete Gaussian distribution. We now define the discrete Gaussian distribution used for the rejection sampling.

Definition 2.3. *The* discrete Gaussian distribution *on $\mathcal{R}^\ell$ centered around $\vec{\boldsymbol{v}} \in \mathcal{R}^\ell$ with standard deviation $\mathfrak{s} > 0$ is given by*

$$D^{\ell d}_{\boldsymbol{v},\mathfrak{s}}(\vec{\boldsymbol{z}}) = \frac{e^{-\|\vec{\boldsymbol{z}}-\vec{\boldsymbol{v}}\|^2/2\mathfrak{s}^2}}{\sum_{\vec{\boldsymbol{z}}'\in\mathcal{R}^\ell} e^{-\|\vec{\boldsymbol{z}}'\|^2/2\mathfrak{s}^2}}.$$

When it is centered around $\vec{\mathbf{0}} \in \mathcal{R}^\ell$ we write $D^{\ell d}_{\mathfrak{s}} = D^{\ell d}_{\vec{\mathbf{0}},\mathfrak{s}}$

2.4 BDLOP Commitment Scheme

We recall the BDLOP commitment scheme from [5] used recently in [1, 10, 15, 26]. Suppose that we want to commit to a message vector $\vec{\boldsymbol{m}} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) \in \mathcal{R}_q^n$ for $n \geq 1$ and that module ranks of κ and λ are required for M-SIS and M-LWE security, respectively. Then, in the key generation, a matrix $\boldsymbol{B}_0 \leftarrow \mathcal{R}_q^{\kappa\times(\kappa+\lambda+n)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_n \leftarrow \mathcal{R}_q^{\kappa+\lambda+n}$ are generated and output as public parameters. Note that one could choose to generate $\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_n$ in a more structured way as in [5] since it saves some computation. However, for readability, we write the commitment matrices in the "Knapsack" form as above. In our case, the hiding property of the commitment scheme is established via the duality between the Knapsack and M-LWE problems. We refer to [18, Appendix C] for a more detailed discussion.

To commit to the message $\vec{\boldsymbol{m}}$, we first sample $\vec{\boldsymbol{r}} \leftarrow \chi^{d\cdot(\kappa+\lambda+n)}$. Now, there are two parts of the commitment scheme: the binding part and the message encoding part. In particular, we compute

$$\begin{aligned}\vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0\vec{\boldsymbol{r}} \bmod q,\\ \boldsymbol{t}_i &= \langle\vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}\rangle + \boldsymbol{m}_i \bmod q,\end{aligned}$$

for $i \in [n]$, where $\vec{\boldsymbol{t}}_0$ forms the binding part and each $\boldsymbol{t}_i$ encodes a message polynomial $\boldsymbol{m}_i$. In this paper, when we write that we compute a BDLOP commitment to a vector $\vec{m} = (\vec{m}_1, \ldots, \vec{m}_n) \in \mathcal{M}_q^{nl}$, we mean that we commit to the vector of polynomials $\vec{\boldsymbol{m}} = (\mathsf{NTT}^{-1}(\vec{m}_1), \ldots, \mathsf{NTT}^{-1}(\vec{m}_n)) \in \mathcal{R}_q^n$ as above.

Next, we define the notion of a weak opening of the commitment [1].

Definition 2.4. *A* weak opening *for the commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1 \parallel \cdots \parallel \boldsymbol{t}_n$ consists of a polynomial $\bar{\boldsymbol{c}} \in \mathcal{R}_q$, a randomness vector $\vec{\boldsymbol{r}}^*$ over $\mathcal{R}_q$ and messages $\boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_n^* \in \mathcal{R}_q$ such that*

$$\begin{aligned}&\|\bar{\boldsymbol{c}}\|_1 \leq 2d \text{ and } \bar{\boldsymbol{c}} \text{ is invertible over } \mathcal{R}_q\\ &\|\bar{\boldsymbol{c}}\vec{\boldsymbol{r}}^*\|_2 \leq 2\beta,\\ &\boldsymbol{B}_0\vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0,\\ &\langle\vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^*\rangle + \boldsymbol{m}_i^* = \boldsymbol{t}_i \text{ for } i \in [n].\end{aligned}$$

Attema et al. [1] show that the commitment scheme is still binding with respect to weak openings if M-SIS$_{\kappa,8d\beta}$ is hard.

3 Efficient Lattice-Based Set Membership Proof

In this section we construct an efficient logarithmic-size ring signature protocol using recent results [1, 15, 25, 26] as the building blocks. Security analysis of the interactive protocol is described in Appendix B. Then, in Appendix C we instantiate our protocol as a ring signature using the Fiat-Shamir transform. Eventually, Appendix D focuses on how to amortize the ring signature generation using similar techniques as in [4].

3.1 Overview

In order to showcase our main techniques, let us consider the following set membership problem. Namely, suppose we would like to prove knowledge of a secret element $\vec{w}_i \in \mathcal{M}_q^{kl}$, for some $k \in \mathbb{N}$, such that $\vec{w} \in S$, where S is a public set $S = \{\vec{p}_1, \ldots, \vec{p}_n\} \subseteq \mathcal{M}_q^{kl}$ of size $n = l^m$ which is a power of l.

We now use the observation from [17, 20, 7] that $\vec{w} \in S$ if and only if there exists a binary vector $\vec{v} \in \{0,1\}^n$ with exactly one 1 such that $P\vec{v} = \vec{w}$ where $P \in \mathcal{M}_q^{kl \times n}$ is the matrix with i-th column being $\vec{p}_i$. One could then directly prove knowledge of $\vec{w}$ and $\vec{v}$ which satisfy conditions above using e.g. the protocol from [15, 25]. However, the proof size grows significantly when n gets bigger. In order to overcome this limitation, [20, 7] observe that vector $\vec{v}$ can be uniquely decomposed into smaller vectors $\vec{v}_1, \ldots, \vec{v}_m \in \{0,1\}^l$ which have exactly one 1 each and

$$\vec{v} = \vec{v}_1 \otimes \vec{v}_2 \otimes \cdots \otimes \vec{v}_m. \tag{15}$$

In the end, we want to commit to $\vec{w}$ and smaller vectors $\vec{v}_1, \ldots, \vec{v}_m$ and prove

$$P(\vec{v}_1 \otimes \cdots \otimes \vec{v}_m) = \vec{w} \tag{16}$$

along with

$$\vec{v}_i \circ (\vec{v}_i - \vec{1}) = \vec{0} \text{ and } \langle \vec{1}, \vec{v}_i \rangle = 1 \text{ for } i \in [m] \tag{17}$$

where for an integer $a \in \mathbb{Z}_q$, $\vec{a} := (a, \ldots, a) \in \mathbb{Z}_q^l$. We highlight that Equation 16 is over the $\mathbb{Z}_q$-module $\mathcal{M}_q$ (see Section 2.2).

We now present a new recursive approach to prove (16) and (17) efficiently. For readability, we first introduce the following notation:

$$\begin{aligned} &\vec{u}_j := \vec{v}_j \otimes \cdots \otimes \vec{v}_m \text{ for } j \in [m], \\ &P_1 := P \text{ and } \vec{x}_1 = (\vec{x}_{1,1}, \ldots, \vec{x}_{1,k}) := \vec{w}. \end{aligned}$$

We start by sending the BDLOP commitments (as described in Sections 1.1 and 2.4) to $\vec{v}_1, \ldots, \vec{v}_m, \vec{w}_1, \ldots, \vec{w}_k$ to the verifier:

$$\begin{aligned} \vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0 \vec{\boldsymbol{r}} \bmod q, \\ \boldsymbol{t}_i &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{v}_i) \bmod q \text{ for } i \in [m] \\ \boldsymbol{t}_{m+i} &= \langle \vec{\boldsymbol{b}}_{m+i}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{x}_i) \bmod q \text{ for } i \in [k]. \end{aligned}$$

Then, a verifier $\mathcal{V}$ sends a challenge $\vec{\gamma}_1 = (\vec{\gamma}_{1,1}, \ldots, \vec{\gamma}_{1,k}) \leftarrow \mathcal{M}_q^{kl}$. Clearly, if (16) holds then we have

$$\langle P_1(\vec{v}_1 \otimes \vec{u}_2) - \vec{x}_1, \vec{\gamma}_1 \rangle = 0.$$

Otherwise, the probability that the inner product above is equal to zero is exactly $q^{-d/l}$ which is negligible.

Now, by Lemma 2.1 and using the fact that each $\vec{v}_i \in \mathbb{Z}_q^l$, we have:

$$\begin{aligned}
\langle P_1(\vec{v}_1 \otimes \vec{u}_2) - \vec{x}_1, \vec{\gamma}_1 \rangle &= \langle \vec{v}_1 \otimes \vec{u}_2, P_1^T \vec{\gamma}_1 \rangle - \langle \vec{x}_1, \vec{\gamma}_1 \rangle \\
&= \sum_{i=1}^{l} v_{1,i} \langle \vec{u}_2, P_{1,i}^T \vec{\gamma}_1 \rangle - \langle \vec{x}_1, \vec{\gamma}_1 \rangle \\
&= \sum_{i=1}^{l} v_{1,i} \gamma_1^T P_{1,i} \vec{u}_2 - \langle \vec{x}_1, \vec{\gamma}_1 \rangle \\
&= \vec{v}_1^T P_2 \vec{u}_2 - \langle \vec{x}_1, \vec{\gamma}_1 \rangle = \langle \vec{v}_1, P_2 \vec{u}_2 \rangle - \langle \vec{x}_1, \vec{\gamma}_1 \rangle
\end{aligned} \tag{18}$$

where we denote

$$P_1 = \begin{pmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,l} \end{pmatrix} \in \mathcal{M}_q^{l \times l^m}$$

and the matrix P_2 is defined as

$$P_2 := \begin{pmatrix} \gamma_1^T P_{1,1} \\ \vdots \\ \gamma_1^T P_{1,l} \end{pmatrix} \in \mathcal{M}_q^{l \times l^{m-1}}. \tag{19}$$

Let us define the following vectors:

$$\vec{x}_2 := P_2 \vec{u}_2 \in \mathcal{M}_q^l \text{ and } \vec{y}_1 := \vec{v}_1 \circ \vec{x}_2 - \sum_{i=1}^{k} \vec{x}_{1,i} \circ \vec{\gamma}_{1,i}. \tag{20}$$

First, we prove that $\vec{x}_2$ is constructed correctly. Note that by definition of $\vec{u}_2$ we have

$$\vec{x}_2 = P_2(\vec{v}_2 \otimes \cdots \otimes \vec{v}_m)$$

which is of the form (16) but with one less tensor. Hence, in order to prove this equation, we recursively follow the argument above. Then, assuming one can prove (20) for $\vec{x}_2$, by Lemma 2.2 we know that $\langle P_1(\vec{v}_1 \otimes \cdots \otimes \vec{v}_m) - \vec{x}_1, \vec{\gamma}_1 \rangle = 0$ if and only if $\boldsymbol{y}_1 := \mathsf{NTT}^{-1}(\vec{y}_1)$ has the first d/l coefficients equal to zero. We present how to prove this property for $\boldsymbol{y}_1$ below.

Let us fix $j = 2$. Suppose that $j < m$. Then, in order to show that $\vec{x}_2$ from (20) is well-formed, we apply the exact strategy as before. Namely, we send a commitment to $\vec{x}_j$:

$$\boldsymbol{t}_{m+k+j-1} = \langle \vec{\boldsymbol{b}}_{m+k+j-1}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{x}_j).$$

Then, given a challenge $\vec{\gamma}_j \leftarrow \mathcal{M}_q^l$, we deduce as in Equation 18 that

$$\langle P_j(\vec{v}_j \otimes \vec{u}_{j+1}) - \vec{x}_j, \vec{\gamma} \rangle = \langle \vec{v}_j, P_{j+1} \vec{u}_{j+1} \rangle - \langle \vec{x}_j, \vec{\gamma}_j \rangle$$

where

$$P_j = \left(P_{j,1}\ P_{j,2}\ \cdots\ P_{j,l}\right) \in \mathcal{M}_q^{l \times l^{m-j+1}}$$

and the matrix P_{j+1} is defined as

$$P_{j+1} := \begin{pmatrix} \gamma_j^T P_{j,1} \\ \vdots \\ \gamma_j^T P_{j,l} \end{pmatrix} \in \mathcal{M}_q^{l \times l^{m-j}}. \tag{21}$$

Next, we define vectors $\vec{x}_{j+1}, \vec{y}_j \in \mathcal{M}_q^l$:

$$\vec{x}_{j+1} := P_{j+1}\vec{u}_{j+1} \text{ and } \vec{y}_j := \vec{v}_j \circ \vec{x}_{j+1} - \vec{x}_j \circ \vec{\gamma}_j. \tag{22}$$

Now, in order to prove well-formedness of $\vec{x}_{j+1}$ we simply run the argument from this paragraph for $j := j+1$. Assuming that $\vec{x}_{j+1}$ is constructed correctly, we also need to prove that the coefficients of $\vec{y}_j$ sum up to 0, i.e. the first d/l coefficients of $\boldsymbol{y}_j = \mathsf{NTT}^{-1}(\vec{y}_j)$ are all zeroes. Below we describe how it can be done for all the $\boldsymbol{y}_j$'s simultaneously.

Eventually, for $j = m$ we want to prove that $\vec{x}_m = P_m\vec{u}_m = P_m\vec{v}_m$ which is a simple linear proof from [15]. We also want to show $\langle \vec{1}, \vec{v}_i \rangle = 1$ for $i \in [m]$. All these relations can be combined into one linear equation:

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & P_m \\ B & 0 & \cdots & 0 & 0 \\ 0 & B & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & B & 0 \end{pmatrix} \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix} = \begin{pmatrix} \vec{x}_m \\ \vec{e}_1 \\ \vdots \\ \vec{e}_1 \end{pmatrix} \tag{23}$$

where

$$B = \begin{pmatrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \in \mathbb{Z}_q^{l \times l} \text{ and } \vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{Z}_q^l.$$

Let us denote $P_m \in \mathcal{M}_q^{(m+1)l \times ml}$ to be the matrix on the left-hand side of Equation 23.

We proceed to proving (23). First, we get a challenge vector

$$\vec{\gamma}_m = (\vec{\gamma}_{m,1}, \ldots, \vec{\gamma}_{m,m+1}) \leftarrow \mathcal{M}_q^{(m+1)l}$$

from $\mathcal{V}$ and deduce that:

$$\left\langle \tilde{P}_m \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix} - \begin{pmatrix} \vec{x}_m \\ \vec{e}_1 \\ \vdots \\ \vec{e}_1 \end{pmatrix}, \vec{\gamma}_m \right\rangle = \left\langle \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix}, \tilde{P}_m^T \vec{\gamma}_m \right\rangle - \langle \vec{x}_m, \vec{\gamma}_{m,1} \rangle - \sum_{i=1}^{m} \langle \vec{e}_1, \vec{\gamma}_{m,i+1} \rangle.$$

Let $\vec{x}_{m+1} = (\vec{x}_{m+1,1}, \ldots, \vec{x}_{m+1,m}) := \tilde{P}_m^T \vec{\gamma}_m \in \mathcal{M}_q^{ml}$ and

$$\vec{y}_m := \left(\sum_{i=1}^{m} \vec{v}_i \circ \vec{x}_{m+1,i} \right) - \vec{x}_m \circ \vec{\gamma}_{m,1} - \vec{e}_1 \circ \sum_{i=1}^{m} \vec{\gamma}_{m,i}. \tag{24}$$

Note that in this case $\vec{x}_{m+1}$ is public (as opposed to $\vec{x}_1, \ldots, \vec{x}_m$). Then, as before we get that $\boldsymbol{y}_m = y_{m,0} + y_{m,1}X + \ldots + y_{m,d-1}X^{d-1} = \mathsf{NTT}^{-1}(\vec{y}_m)$ satisfies:

$$y_{m,0} + \ldots + y_{m,d/l-1}X^{d/l-1} = \frac{1}{l} \left\langle \tilde{P}_m \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix} - \begin{pmatrix} \vec{x}_m \\ \vec{e}_1 \\ \vdots \\ \vec{e}_1 \end{pmatrix}, \vec{\gamma}_m \right\rangle.$$

Therefore, we need to argue that $\boldsymbol{y}_m$ has the first d/l polynomial coefficients equal to 0.

Finally, what have left to prove is that (i) polynomials $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ have the first d/l coefficients equal to zero and (ii) vectors $\vec{v}_i$ are binary. We first focus on (i) and adapt the strategy shown in [15]. At the beginning, we will commit to a uniformly random polynomial $\boldsymbol{g}$ which has the first d/l coefficients equal to zero:

$$\boldsymbol{t}_{k+2m} = \langle \vec{\boldsymbol{b}}_{k+2m}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}.$$

Then, we will reveal the polynomial

$$\boldsymbol{h} = \boldsymbol{g} + \boldsymbol{y}_1 + \ldots + \boldsymbol{y}_m. \tag{25}$$

Hence, the verifier manually checks the the first d/l coefficients of $\boldsymbol{h}$ are indeed zeroes. On the other hand, to prove (25) we follow the approach for proving multiplicative relations from [1].

Let $\vec{\boldsymbol{y}} \leftarrow D^{(\kappa+\lambda+k+2m)}$ be the masking vector. That is, given a challenge polynomial $\boldsymbol{c} \leftarrow C$ from a challenge distribution C (defined in Section 3.2), the prover will output a masked opening $\vec{\boldsymbol{z}}$ of the randomness $\vec{\boldsymbol{r}}$ defined as: $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$. Then, define polynomials $\boldsymbol{f}_\eta$ as:

$$\boldsymbol{f}_\eta = \begin{cases} \langle \vec{\boldsymbol{b}}_\eta, \vec{\boldsymbol{y}} \rangle - \boldsymbol{c}\boldsymbol{v}_\eta \text{ if } \eta \in [m] \\ \langle \vec{\boldsymbol{b}}_{m+i}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{c}\boldsymbol{x}_{1,i} \text{ for } \eta = m+i; i \in [k] \\ \langle \vec{\boldsymbol{b}}_{m+k+j}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{c}\boldsymbol{x}_{j+1} \text{ for } \eta = m+k+j; j \in [m-1] \\ \langle \vec{\boldsymbol{b}}_{k+2m}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{c}\boldsymbol{g} \text{ if } \eta = k+2m \end{cases}$$

where $\boldsymbol{x}_j = \mathsf{NTT}^{-1}(\vec{x}_j)$ and similarly for $\boldsymbol{v}_i$ and $\boldsymbol{\gamma}_j$. Note that $\boldsymbol{f}_\eta = \langle \vec{\boldsymbol{b}}_\eta, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\vec{\boldsymbol{t}}_\eta$ for all η and thus can be calculated by the verifier.

First, let us focus on $\boldsymbol{y}_1$. By definition we have (see (20)):

$$\boldsymbol{F}_1 := \boldsymbol{f}_1 \boldsymbol{f}_{m+k+1} + \boldsymbol{c} \sum_{i=1}^{k} \boldsymbol{\gamma}_{1,i} \boldsymbol{f}_{m+i} = \boldsymbol{\omega}_1 + \boldsymbol{\psi}_1 \boldsymbol{c} + \boldsymbol{y}_1 \boldsymbol{c}^2$$

where polynomials $\boldsymbol{\omega}_1, \boldsymbol{\psi}_1$ are defined as follows

$$\begin{aligned}\boldsymbol{\omega}_1 &:= \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}} \rangle \langle \vec{\boldsymbol{b}}_{m+k+1}, \vec{\boldsymbol{y}} \rangle \\ \boldsymbol{\psi}_1 &:= \sum_{i=1}^{k} \boldsymbol{\gamma}_{1,i} \langle \vec{\boldsymbol{b}}_{m+i}, \vec{\boldsymbol{y}} \rangle - \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}} \rangle \boldsymbol{x}_2 - \langle \vec{\boldsymbol{b}}_{m+k+1}, \vec{\boldsymbol{y}} \rangle \boldsymbol{v}_1\end{aligned}$$

Now, by Definition of $\boldsymbol{y}_j$ (see (22)), for fixed $j \in [2, m-1]$ we have:

$$\boldsymbol{F}_j := \boldsymbol{f}_j \boldsymbol{f}_{m+k+j} + \boldsymbol{c}\boldsymbol{\gamma}_j \boldsymbol{f}_{m+k+j-1} = \boldsymbol{\omega}_j + \boldsymbol{\psi}_j \boldsymbol{c} + \boldsymbol{y}_j \boldsymbol{c}^2$$

where

$$\begin{aligned}\boldsymbol{\omega}_j &:= \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}} \rangle \langle \vec{\boldsymbol{b}}_{m+k+j}, \vec{\boldsymbol{y}} \rangle \\ \boldsymbol{\psi}_j &:= \boldsymbol{\gamma}_j \langle \vec{\boldsymbol{b}}_{m+k+j-1}, \vec{\boldsymbol{y}} \rangle - \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}} \rangle \boldsymbol{x}_{j+1} - \langle \vec{\boldsymbol{b}}_{m+k+j}, \vec{\boldsymbol{y}} \rangle \boldsymbol{v}_j.\end{aligned} \tag{26}$$

In case of $j = m$, we transform Equation 24 into:

$$\boldsymbol{F}_m := \boldsymbol{c} \left(- \sum_{i=1}^{m} \boldsymbol{x}_{m+1,i} \boldsymbol{f}_i + \boldsymbol{\gamma}_{m,1} \boldsymbol{f}_{k+2m-1} - \boldsymbol{e}_1 \sum_{i=1}^{m} \boldsymbol{\gamma}_{m,i} \right) = \boldsymbol{\psi}_m \boldsymbol{c} + \boldsymbol{y}_m \boldsymbol{c}^2$$

where

$$\boldsymbol{\psi}_m := - \sum_{i=1}^{m} \boldsymbol{x}_{m+1,i} \langle \vec{\boldsymbol{b}}_i ., \vec{\boldsymbol{y}} \rangle + \boldsymbol{\gamma}_{m,1} \langle \vec{\boldsymbol{b}}_{k+2m-1}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{e}_1 \sum_{i=1}^{m} \boldsymbol{\gamma}_{m,i}. \tag{27}$$

Clearly, all $\boldsymbol{F}_j$ can be computed by the verifier. Therefore, if we denote

$$\boldsymbol{\omega}_{\mathsf{sm}} := \sum_{i=1}^{m-1} \boldsymbol{\omega}_i \text{ and } \boldsymbol{\psi}_{\mathsf{sm}} := \sum_{i=1}^{m} \boldsymbol{\psi}_i - \langle \vec{\boldsymbol{b}}_{k+2m}, \vec{\boldsymbol{y}} \rangle \tag{28}$$

then we obtain:

$$\sum_{j=1}^{m} \boldsymbol{F}_j - \boldsymbol{c}\boldsymbol{f}_{k+2m} - \boldsymbol{c}^2 \boldsymbol{h} = \boldsymbol{\omega}_{\mathsf{sm}} + \boldsymbol{\psi}_{\mathsf{sm}} \boldsymbol{c} + (\boldsymbol{y}_1 + \ldots + \boldsymbol{y}_m + \boldsymbol{g} - \boldsymbol{h}) \boldsymbol{c}^2.$$

Hence, we want to prove that the coefficient corresponding to the quadratic term of $\sum_{j=1}^{m} \boldsymbol{F}_j - \boldsymbol{c}\boldsymbol{f}_{k+2m} - \boldsymbol{c}^2 \boldsymbol{h}$ vanishes.

Recall that we still need to prove (ii), i.e. all $\vec{v}_i$'s are binary. We first get challenges $\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m \leftarrow \mathcal{R}_q$ from the verifier. Then, we observe that

$$\sum_{i=1}^{m} \boldsymbol{\alpha}_i (\boldsymbol{f}_i^2 + \boldsymbol{c}\boldsymbol{f}_i) = \boldsymbol{\omega}_{\mathsf{bin}} + \boldsymbol{\psi}_{\mathsf{bin}} \boldsymbol{c} + \left(\sum_{i=1}^{m} \boldsymbol{\alpha}_i \boldsymbol{v}_i (\boldsymbol{v}_i - \mathbf{1}) \right) \boldsymbol{c}^2$$

where

$$\boldsymbol{\omega}_{\mathsf{bin}} := \sum_{i=1}^{m} \boldsymbol{\alpha}_i \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle^2 \text{ and } \boldsymbol{\psi}_{\mathsf{bin}} := \sum_{i=1}^{m} \boldsymbol{\alpha}_i \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle (\mathbf{1} - 2\boldsymbol{v}_i). \tag{29}$$

Therefore, we combine (i) and (ii) by proving that the quadratic term in

$$\boldsymbol{\alpha}_0 \left(\sum_{j=1}^{m} \boldsymbol{F}_j - \boldsymbol{c}\boldsymbol{f}_{k+2m} - \boldsymbol{c}^2\boldsymbol{h} \right) + \sum_{i=1}^{m} \boldsymbol{\alpha}_i (\boldsymbol{f}_i^2 + \boldsymbol{c}\boldsymbol{f}_i) \tag{30}$$

is equal to zero. In order to do so, we commit to the garbage polynomial

$$\boldsymbol{t}_{k+2m+1} = \langle \vec{\boldsymbol{b}}_{k+2m+1}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\psi}_{\mathsf{bin}} + \boldsymbol{\alpha}_0 \boldsymbol{\psi}_{\mathsf{sm}}$$

and additionally send $\boldsymbol{\omega} := \langle \vec{\boldsymbol{b}}_{k+2m+1}, \vec{\boldsymbol{y}} \rangle + \boldsymbol{\omega}_{\mathsf{bin}} + \boldsymbol{\alpha}_0 \boldsymbol{\omega}_{\mathsf{sm}}$. Then, the verifier computes $\boldsymbol{f}_{k+2m+1} = \langle \vec{\boldsymbol{b}}_{k+2m+1}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_{k+2m+1}$ and checks whether:

$$\boldsymbol{\alpha}_0 \left(\sum_{j=1}^{m} \boldsymbol{F}_j - \boldsymbol{c}\boldsymbol{f}_{k+2m} - \boldsymbol{c}^2\boldsymbol{h} \right) + \sum_{i=1}^{m} \boldsymbol{\alpha}_i (\boldsymbol{f}_i^2 + \boldsymbol{c}\boldsymbol{f}_i) + \boldsymbol{f}_{k+2m+1} \stackrel{?}{=} \boldsymbol{\omega}.$$

3.2 Main Protocol

We present our main lattice-based one-out-of-many proof using the techniques from Section 3.1 and show how it can be turned into an efficient, logarithmic-sized ring signature.

Similarly as in the previous works [17, 18], the secret key of a user is a vector $\vec{\boldsymbol{s}} \leftarrow [-\mu, \mu]^{\ell d}$ of short polynomials over $\mathcal{R}_q$ and the corresponding public key $\vec{\boldsymbol{pk}} \in \mathcal{R}_q^k$ is defined as $\vec{\boldsymbol{pk}} := \boldsymbol{A}\vec{\boldsymbol{s}}$ for a public matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times \ell}$. Suppose there are $n = l^m$ users in the ring [7] and for $\iota \in [n]$, let $\vec{\boldsymbol{pk}}_\iota$ be the public key corresponding to the ι-th user. Then, during the signing process, user ι wants to prove knowledge of a short vector $\vec{\boldsymbol{s}}$ such that

$$\boldsymbol{A}\vec{\boldsymbol{s}} \in \{\vec{\boldsymbol{pk}}_1, \ldots, \vec{\boldsymbol{pk}}_n\}$$

without revealing any information about its index ι.

We present the main protocol in Fig. 7 with verification equations in Fig. 9. User $\iota \in [n]$, which acts as a prover $\mathcal{P}$, starts by decomposing the index vector $\vec{v} = (0, \ldots, 0, 1, 0, \ldots, 0) \in \{0,1\}^n$, where the ι-th coefficient is equal to 1, into m smaller vectors of length l as in (15). Note that each $\vec{v}_i \in \mathbb{Z}_q^l$ satisfies (17). At the same time, $\mathcal{P}$ samples a masking $\vec{\boldsymbol{y}}' \leftarrow D_{\boldsymbol{s}'}^{\ell d}$ and computes $\vec{\boldsymbol{w}}' = (\boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_k) = \boldsymbol{A}\vec{\boldsymbol{y}}' \in \mathcal{R}_q^k$. Furthermore, for the linear proof $\mathcal{P}$ generates a random $\boldsymbol{g} \in \mathcal{R}_q$ such that $g_0 = \ldots = g_{d/l-1} = 0$. Now, the prover sends

[7] If there are less than l^m users then we simply add the zero vectors as public keys so that the ring has exactly l^m elements. Then the proof that the prover knows a short preimage to one of the columns implies that they must know a preimage to one of the actual public keys because knowing a preimage for one of the zero columns would constitute a SIS solution.

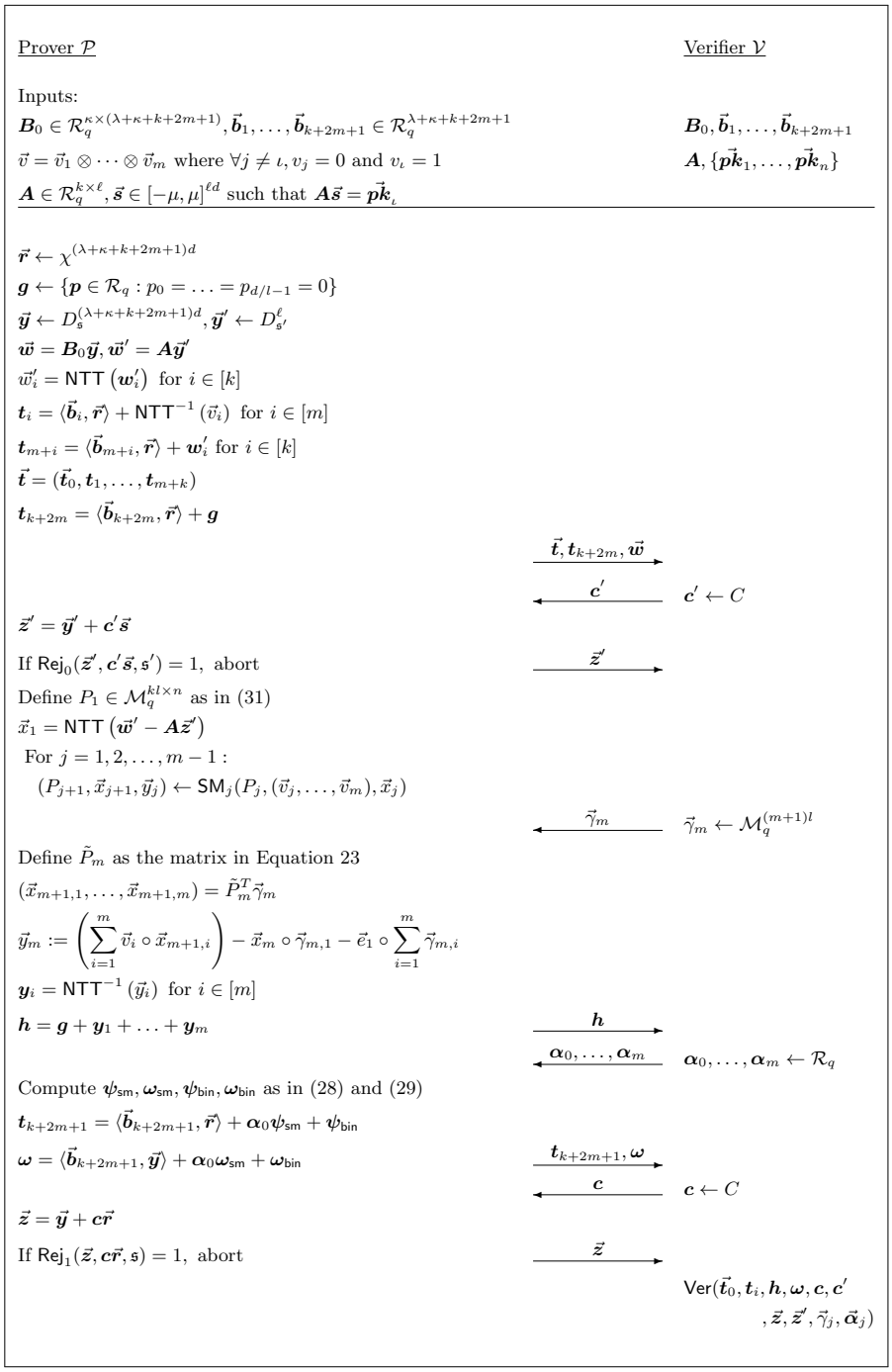

Fig. 7. Interactive protocol for our ring signature construction. Verifications equations Ver and the sub-protocol $\mathsf{SM}_j(P_j, (\vec{v}_j, \ldots, \vec{v}_m), \vec{x}_j)$ are defined in Fig. 9 and 8 respectively. The rejection sampling algorithms Rej_i are defined in Fig. 10.

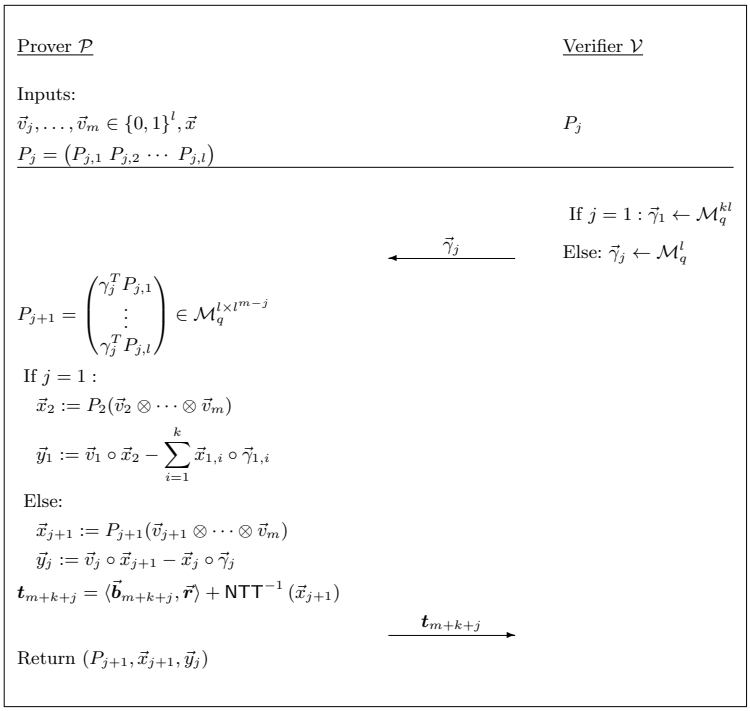

Fig. 8. The sub-protocol $\mathsf{SM}_j(P_j, (\vec{v}_j, \ldots, \vec{v}_m), \vec{x}_j)$ used in Fig. 7.

the BDLOP commitments to $\vec{v}_i$ as well as to $\vec{\boldsymbol{w}}'$ and $\boldsymbol{g}$. Namely, it generates a randomness vector $\vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+2m+1)d}$ and sends:

$$\begin{aligned} \vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0 \vec{\boldsymbol{r}} \bmod q, \\ \boldsymbol{t}_i &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{v}_i) \ \text{ for } i \in [m] \\ \boldsymbol{t}_{m+i} &= \langle \vec{\boldsymbol{b}}_{m+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{w}'_i \text{ for } i \in [k]. \\ \boldsymbol{t}_{k+2m} &= \langle \vec{\boldsymbol{b}}_{k+2m}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g} \end{aligned}$$

Additionally, $\mathcal{P}$ computes $\vec{\boldsymbol{w}} = \boldsymbol{B}_0 \vec{\boldsymbol{y}}$ for $\vec{\boldsymbol{y}}$ sampled from $D_{\mathfrak{s}}^{(\kappa+k+2m+1)d}$. Then, $\mathcal{P}$ sends

$$(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \boldsymbol{t}_{k+2m}, \vec{\boldsymbol{w}})$$

to the verifier.

The verifier $\mathcal{V}$ outputs a challenge polynomial $\boldsymbol{c}' \leftarrow C$. Next, $\mathcal{P}$ computes $\vec{\boldsymbol{z}}' = \vec{\boldsymbol{y}}' + \boldsymbol{c}'\vec{\boldsymbol{s}}$ and applies the rejection sampling algorithm. If it does not abort, $\mathcal{P}$ returns $\vec{\boldsymbol{z}}'$.

Let $P \in \mathcal{M}_q^{kl \times n}$ be the matrix defined as

$$P = \left(\mathsf{NTT}\left(-\boldsymbol{c}' \cdot \vec{\boldsymbol{pk}}_1\right) \middle| \cdots \middle| \mathsf{NTT}\left(-\boldsymbol{c}' \cdot \vec{\boldsymbol{pk}}_n\right) \right), \tag{31}$$

$\underline{\mathsf{Ver}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k+2m+1}, \boldsymbol{h}, \boldsymbol{\omega}, \boldsymbol{c}, \boldsymbol{c}', \vec{\boldsymbol{z}}, \vec{\boldsymbol{z}}', \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m)}$

$\|\vec{\boldsymbol{z}}'\|_2 \stackrel{?}{<} \beta' = \mathfrak{s}'\sqrt{2\ell d}$

$\|\vec{\boldsymbol{z}}\|_2 \stackrel{?}{<} \beta = \mathfrak{s}\sqrt{2(\lambda+\kappa+k+2m+1)d}$

$\boldsymbol{B}_0\vec{\boldsymbol{z}} \stackrel{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$

$(\boldsymbol{t}_{m+1}, \ldots, \boldsymbol{t}_{m+k}) = (\boldsymbol{t}_{m+1}, \ldots, \boldsymbol{t}_{m+k}) - \boldsymbol{A}\vec{\boldsymbol{z}}' \in \mathcal{R}_q^k$

$\forall j \in [k+2m+1], \boldsymbol{f}_j = \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_j$

$\forall i \in [m+1], \boldsymbol{\gamma}_{m,i} := \mathsf{NTT}^{-1}(\vec{\gamma}_{1,i})\,; \forall j \in [1,k], \boldsymbol{\gamma}_{1,j} := \mathsf{NTT}^{-1}(\vec{\gamma}_{1,j})$

$\forall j \in [2, m-1], \boldsymbol{\gamma}_j = \mathsf{NTT}^{-1}(\vec{\gamma}_j)$

$(\boldsymbol{x}_{m+1,1}, \ldots, \boldsymbol{x}_{m+1,m}) := \mathsf{NTT}^{-1}\left(\tilde{P}_m^T \vec{\gamma}_m\right)$ where $\tilde{P}_m$ is the matrix in (23)

$\boldsymbol{e}_1 := \mathsf{NTT}^{-1}((1, 0, \ldots, 0))$

$\boldsymbol{F}_1 := \boldsymbol{f}_1 \boldsymbol{f}_{m+k+1} + \boldsymbol{c}\sum_{i=1}^{k} \boldsymbol{\gamma}_{1,i}\boldsymbol{f}_{m+i}$

$\forall j \in [2, m-1], \boldsymbol{F}_j := \boldsymbol{f}_j \boldsymbol{f}_{m+k+j} + \boldsymbol{c}\boldsymbol{\gamma}_j \boldsymbol{f}_{m+k+j-1}$

$\boldsymbol{F}_m := \boldsymbol{c}\left(-\sum_{i=1}^{m} \boldsymbol{x}_{m+1,i}\boldsymbol{f}_i + \boldsymbol{\gamma}_{m,1}\boldsymbol{f}_{k+2m-1} - \boldsymbol{e}_1 \sum_{i=1}^{m} \boldsymbol{\gamma}_{m,i}\right)$

$\boldsymbol{\alpha}_0\left(\sum_{j=1}^{m} \boldsymbol{F}_j - \boldsymbol{c}\boldsymbol{f}_{k+2m} - \boldsymbol{c}^2\boldsymbol{h}\right) + \sum_{i=1}^{m} \boldsymbol{\alpha}_i(\boldsymbol{f}_i^2 + \boldsymbol{c}\boldsymbol{f}_i) + \boldsymbol{f}_{k+2m+1} \stackrel{?}{=} \boldsymbol{\omega}$

For $i = 0, \ldots, d/l - 1$:

  $h_i \stackrel{?}{=} 0$

Fig. 9. Verification equations for the protocol in Fig. 7.

i.e. the i-th column of P is equal to $\mathsf{NTT}\left(-\boldsymbol{c}' \cdot \vec{\boldsymbol{pk}}_i\right) \in \mathcal{M}_q^{kl}$. Clearly, it can be computed by the verifier. Also, define

$$\vec{w} = \mathsf{NTT}\left(\boldsymbol{w}' - \boldsymbol{A}\vec{\boldsymbol{z}}'\right) \in \mathcal{M}_q^{kl}.$$

Then, user ι wants to prove that $P(\vec{v}_1 \otimes \cdots \otimes \vec{v}_m) = \vec{w}$. Obviously, the verifier can manually construct a commitment to $\vec{w}$ by subtracting $(\boldsymbol{t}_{m+1}, \ldots, \boldsymbol{t}_{m+k})$ by $\boldsymbol{A}\vec{\boldsymbol{z}}'$. One observes that this is the equation of type (16) and it is where we apply the strategy described in Section 3.1. Namely, for $j = 1, 2, 3, \ldots, m-1$, we run a two-round sub-protocol $\mathsf{SM}_j(P_j, (\vec{v}_j, \ldots, \vec{v}_m), \vec{x}_j)$ defined in Fig. 8 which does the following. The verifier $\mathcal{V}$ starts by sending a challenge vector $\vec{\gamma}_j$. Then, $\mathcal{P}$ computes the matrix P_{j+1} and vectors $\vec{x}_{j+1}, \vec{y}_j \in \mathcal{M}_q^l$ as defined in the previous section. Eventually, it outputs the commitment to $\vec{x}_{j+1}$:

$$\boldsymbol{t}_{m+k+j} = \langle \vec{\boldsymbol{b}}_{m+k+j}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{x}_{j+1}).$$

In the end, the sub-protocol returns

$$(P_{j+1}, \vec{x}_{j+1}, \vec{y}_j) \leftarrow \mathsf{SM}_j(P_j, (\vec{v}_j, \ldots, \vec{v}_m), \vec{x}_j).$$

After executing the SM sub-protocol $m-1$ times, the verifier sends $\vec{\gamma}_m \leftarrow \mathcal{M}_q^{(m+1)l}$. Then, in order to prove Equation 23, $\mathcal{P}$ first computes $\vec{y}_m$ as in Equation 24 and outputs the polynomial $\boldsymbol{h} = \boldsymbol{g} + \boldsymbol{y}_1 + \ldots + \boldsymbol{y}_m$, where $\boldsymbol{y}_i = \mathsf{NTT}^{-1}(\vec{y}_i)$ for $i \in [m]$.

Next, $\mathcal{V}$ sends uniform polynomials $\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m \leftarrow \mathcal{R}_q$. Then, $\mathcal{P}$ returns a commitment

$$\boldsymbol{t}_{k+2m+1} = \langle \vec{\boldsymbol{b}}_{k+2m+1}, \vec{\boldsymbol{y}} \rangle + \boldsymbol{\psi}$$

to the garbage polynomial $\boldsymbol{\psi} = \boldsymbol{\psi}_{\mathsf{bin}} + \boldsymbol{\alpha}_0 \boldsymbol{\psi}_{\mathsf{sm}}$ along with $\boldsymbol{\omega} := \langle \vec{\boldsymbol{b}}_{k+2m+1}, \vec{\boldsymbol{y}} \rangle + \boldsymbol{\omega}_{\mathsf{bin}} + \boldsymbol{\alpha}_0 \boldsymbol{\omega}_{\mathsf{sm}}$ (where their components are defined in (28) and (29)).

Finally, the verifier picks a challenge $\boldsymbol{c} \leftarrow C$ and outputs $\boldsymbol{c}$. Here, the coefficients of a challenge $\boldsymbol{c} \leftarrow C$ are independently identically distributed with $P(0) = 1/2$ and $\Pr(1) = \Pr(-1) = 1/4$ [8] Then, prover $\mathcal{P}$ computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and applies rejection sampling. If it does not abort, $\mathcal{P}$ returns $\vec{\boldsymbol{z}}$.

Acknowledgements

We would like to thank anonymous reviews for useful feedback. This work was supported by the SNSF ERC Transfer Grant CRETP2-166734 FELICITY.

References

1. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 470–499. Springer, 2020.
2. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.
3. Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, Dec 1993.
4. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *CRYPTO*, pages 669–699, 2018.
5. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, pages 368–385, 2018.
6. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492. Springer, 2020.
7. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *ESORICS (1)*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265. Springer, 2015.
8. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 176–202. Springer, 2019.
9. Ivan Damgård. On Σ-Protocols. Lecture on Cryptologic Protocol Theory; Faculty of Science, University of Aarhus, 2010.
10. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *Public Key Cryptography (1)*, volume 12710 of *Lecture Notes in Computer Science*, pages 99–130. Springer, 2021.

[8] We will make use of the properties of C described in [1]. We refer to Appendix **??** for more details.

11. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM Conference on Computer and Communications Security*, pages 574–591. ACM, 2018.
12. Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. *CoRR*, abs/2003.05207, 2020.
13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO (1)*, pages 40–56, 2013.
14. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.
15. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 259–288. Springer, 2020.
16. Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 115–146. Springer, 2019.
17. Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS*, volume 11464 of *Lecture Notes in Computer Science*, pages 67–88. Springer, 2019.
18. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *CCS*, pages 567–584. ACM, 2019.
19. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
20. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*, pages 253–280, 2015.
21. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.
22. Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In *ACNS*, volume 11464 of *Lecture Notes in Computer Science*, pages 110–130. Springer, 2019.
23. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.
24. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.
25. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *CCS*, pages 1051–1070. ACM, 2020.
26. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *Public Key Cryptography (1)*, volume 12710 of *Lecture Notes in Computer Science*, pages 215–241. Springer, 2021.
27. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Smile: Set membership from ideal lattices with applications to ring signatures and confidential transactions. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 611–640, Cham, 2021. Springer International Publishing.

28. Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT (1)*, volume 10820 of *Lecture Notes in Computer Science*, pages 204–224. Springer, 2018.
29. Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptol. ePrint Arch.*, 2015:1098, 2015.
30. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2017. https://csrc.nist.gov/projects/post-quantum-cryptography/ round-1-submissions.
31. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.
32. Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 147–175. Springer, 2019.

A Additional Background

A.1 More on the Challenge Space

Let $\mathcal{C} := \{-1, 0, 1\}^d \subset \mathcal{R}_q$ be the challenge set of ternary polynomials with coefficients $-1, 0, 1$. We define the following probability distribution $C : \mathcal{C} \to [0, 1]$. The coefficients of a challenge $\boldsymbol{c} \leftarrow C$ are independently identically distributed with $P(0) = 1/2$ and $\Pr(1) = \Pr(-1) = 1/4$.

Consider the coefficients of the polynomial $\boldsymbol{c} \bmod (X^{d/l} - \zeta^{2j+1})$ for $\boldsymbol{c} \leftarrow C$. Then, all coefficients follow the same distribution over $\mathbb{Z}_q$. Let us write Y for the random variable over $\mathbb{Z}_q$ that follows this distribution. Attema et al. [1] give an upper bound on the maximum probability of Y.

Lemma A.1. *Let the random variable Y over $\mathbb{Z}_q$ be defined as above. Then for all $x \in \mathbb{Z}_q$,*

$$\Pr(Y = x) \leq \frac{1}{q} + \frac{2l}{q} \sum_{j=0}^{l-1} \prod_{i=0}^{l-1} \left| \frac{1}{2} + \frac{1}{2} \cos\left(2\pi(2j+1)y\zeta^i/q\right) \right|. \tag{32}$$

In particular, [1, 15] computed that for $q \approx 2^{32}$, the maximum probability for each coefficient of $c \bmod X^{d/l} - \zeta^{2j+1}$ is around $2^{-31.4}$. In general, we will call this probability p.

An immediate consequence of Lemma A.1 is that polynomial $\boldsymbol{c} \leftarrow C$ is invertible in $\mathcal{R}_q$ with overwhelming probability as long as parameters q, d, l are selected so that $q^{-d/l}$ is negligible.

A.2 Module-SIS and Module-LWE Problems

Security of the [5] commitment scheme used in our protocols relies on the well-known computational lattice problems, namely Module-LWE (M-LWE) and Module-SIS (M-SIS) [21]. Both problems are defined over $\mathcal{R}_q$.

Definition A.2 (M-SIS$_{\kappa,m,B}$). *Given $\boldsymbol{A} \leftarrow \mathcal{R}_q^{\kappa \times m}$, the* Module-SIS *problem with parameters $\kappa, m > 0$ and $0 < B < q$ asks to find $\vec{\boldsymbol{z}} \in \mathcal{R}_q^m$ such that $\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{0}}$ over $\mathcal{R}_q$ and $0 < \|\vec{\boldsymbol{z}}\| \leq B$. An algorithm $\mathcal{A}$ is said to have advantage ϵ in solving M-SIS$_{\kappa,m,B}$ if*

$$\Pr\left[0 < \|\vec{\boldsymbol{z}}\| \leq B \wedge \boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{0}} \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{\kappa \times m};\ \vec{\boldsymbol{z}} \leftarrow \mathcal{A}(\boldsymbol{A})\right] \geq \epsilon.$$

Definition A.3 (M-LWE$_{m,\lambda,\chi}$). *The* Module-LWE *problem with parameters $m, \lambda > 0$ and an error distribution χ over $\mathcal{R}$ asks the adversary $\mathcal{A}$ to distinguish between the following two cases: 1) $(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})$ for $\boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \lambda}$, a secret vector $\vec{\boldsymbol{s}} \leftarrow \chi^\lambda$ and error vector $\vec{\boldsymbol{e}} \leftarrow \chi^m$, and 2) $(\boldsymbol{A}, \vec{\boldsymbol{b}}) \leftarrow \mathcal{R}_q^{m \times \lambda} \times \mathcal{R}_q^m$. Then, $\mathcal{A}$ is said to have advantage ϵ in solving M-LWE$_{m,\lambda,\chi}$ if*

$$\begin{aligned}\Big|\Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \lambda};\ \vec{\boldsymbol{s}} \leftarrow \chi^\lambda;\ \vec{\boldsymbol{e}} \leftarrow \chi^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})\right] \\ - \Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \lambda};\ \vec{\boldsymbol{b}} \leftarrow \mathcal{R}_q^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \vec{\boldsymbol{b}})\right]\Big| \geq \epsilon.\end{aligned} \tag{33}$$

We also recall the (simplified) Extended-MLWE problem [26].

Definition A.4 (Extended M-LWE$_{m,\lambda,\mathfrak{s}}$). *The* Extended Module-LWE *problem with parameters $m, \lambda > 0$ and the standard deviation $\mathfrak{s}$ asks the adversary $\mathcal{A}$ to distinguish between the following two cases:*

1. $(\boldsymbol{B}, \boldsymbol{B}\vec{\boldsymbol{r}}, \boldsymbol{c}, \vec{\boldsymbol{z}}, \mathsf{sign}\left(\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}} \rangle\right))$ *for $\boldsymbol{B} \leftarrow \mathcal{R}_q^{m \times (m+\lambda)}$, a secret vector $\vec{\boldsymbol{r}} \leftarrow \chi^{m+\lambda}$ and $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(m+\lambda)d}, \boldsymbol{c} \leftarrow C$*
2. $(\boldsymbol{B}, \vec{\boldsymbol{u}}, \boldsymbol{c}, \vec{\boldsymbol{z}}, \mathsf{sign}\left(\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}} \rangle\right))$ *for $\boldsymbol{B} \leftarrow \mathcal{R}_q^{m \times (m+\lambda)}, \vec{\boldsymbol{u}} \leftarrow \mathcal{R}_q^m$ and $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(m+\lambda)d}, \boldsymbol{c} \leftarrow C$.*

where $\mathsf{sign}(a) = 1$ if $a \geq 0$ and 0 otherwise. Then, $\mathcal{A}$ is said to have advantage ϵ in solving Extended M-LWE$_{m,\lambda,\mathfrak{s}}$ if

$$\begin{aligned}\Big|\Pr\left[b = 1 \,\middle|\, \boldsymbol{B} \leftarrow \mathcal{R}_q^{m \times (m+\lambda)};\ \vec{\boldsymbol{r}} \leftarrow \chi^{m+\lambda};\ \vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(m+\lambda)d};\ \boldsymbol{c} \leftarrow C;\ b \leftarrow \mathcal{A}(\boldsymbol{B}, \boldsymbol{B}\vec{\boldsymbol{r}}, \vec{\boldsymbol{z}}, \boldsymbol{c}, s)\right] \\ - \Pr\left[b = 1 \,\middle|\, \boldsymbol{B} \leftarrow \mathcal{R}_q^{m \times \lambda};\ \vec{\boldsymbol{u}} \leftarrow \mathcal{R}_q^m;\ \vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(m+\lambda)d}; \boldsymbol{c} \leftarrow C\ ;\ b \leftarrow \mathcal{A}(\boldsymbol{B}, \vec{\boldsymbol{u}}, \vec{\boldsymbol{z}}, \boldsymbol{c}, s)\right]\Big| \geq \epsilon.\end{aligned}$$

where $s = \mathsf{sign}\left(\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}} \rangle\right)$, and probability distributions χ and C are defined in Sections 2.3 and A.1 respectively.

A.3 Rejection Sampling

In lattice-based zero-knowledge proofs, the prover will want to output a vector $\vec{\boldsymbol{z}}$ whose distribution should be independent of a secret randomness vector $\vec{\boldsymbol{r}}$, so

that $\vec{z}$ cannot be used to gain any information on the prover's secret. During the protocol, the prover computes $\vec{z} = \vec{y} + c\vec{r}$ where $\vec{r}$ is the randomness used to commit to the prover's secret, $c \leftarrow C$ is a challenge polynomial, and $\vec{y}$ is a "masking" vector. In order to remove the dependency of $\vec{z}$ on $\vec{r}$, one applies *rejection sampling* [24].

Lemma A.5 (Rejection Sampling). *Let $V \subseteq \mathcal{R}^{\ell}$ be a set of polynomials with norm at most T and $\rho\colon V \to [0,1]$ be a probability distribution. Now, sample $\vec{v} \leftarrow \rho$ and $\vec{y} \leftarrow D_{\mathfrak{s}}^{\ell d}$, set $\vec{z} = \vec{y} + \vec{v}$, and run $b \leftarrow \mathsf{Rej}_0(\vec{z}, \vec{v}, \mathfrak{s})$ as defined in Fig. 10. Then, the probability that $b = 0$ is at least $(1 - 2^{-100})/M$ and the distribution of $(\vec{v}, \vec{z})$, conditioned on $b = 0$, is within statistical distance of $2^{-100}/M$ of the product distribution $\rho \times D_{\mathfrak{s}}^{\ell d}$.*

$\underline{\mathrm{Rej}_0(\vec{z}, \vec{v}, \mathfrak{s})}$	$\underline{\mathrm{Rej}_1(\vec{z}, \vec{v}, \mathfrak{s})}$
01 $u \leftarrow [0,1)$	01 If $\langle \vec{z}, \vec{v} \rangle < 0$
02 If $u > \frac{1}{M} \cdot \exp\left(\frac{-2\langle \vec{z}, \vec{v} \rangle + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right)$	02 return 1
03 return 1	03 $u \leftarrow [0,1)$
04 Else	04 If $u > \frac{1}{M} \cdot \exp\left(\frac{-2\langle \vec{z}, \vec{v} \rangle + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right)$
05 return 0	05 return 1
	06 Else
	07 return 0

Fig. 10. Two rejection sampling algorithms: the one used generally in previous works [24] (left) and the one proposed recently in [26] (right).

We recall how parameters $\mathfrak{s}$ and M in Lemma A.5 are selected. Concretely, the repetition rate M is chosen to be an upper-bound on:

$$\frac{D_{\mathfrak{s}}^{\ell d}(\vec{z})}{D_{\vec{v},\mathfrak{s}}^{\ell d}(\vec{z})} = \exp\left(\frac{-2\langle \vec{z}, \vec{v} \rangle + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right) \leq \exp\left(\frac{24\mathfrak{s}\|\vec{v}\| + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right) = M^9. \tag{34}$$

For the inequality we used the tail bound which says that with probability at least $1 - 2^{100}$ we have $|\langle \vec{z}, \vec{v} \rangle| < 12\mathfrak{s}\|\vec{v}\|$ for $\vec{z} \leftarrow D_{\mathfrak{s}}^{\ell d}$ [3, 24]. Hence, by setting $\mathfrak{s} = 11\|\vec{v}\|$ we obtain $M \approx 3$.

Recently, Lyubashevsky et al. [26] proposed a modified rejection sampling algorithm (see $\mathsf{Rej}_1(\vec{z}, \vec{v}, \mathfrak{s})$ in Fig. 10) where it forces $\vec{z}$ to satisfy $\langle \vec{z}, \vec{v} \rangle \geq 0$, otherwise it aborts. With this additional assumption, we can set M in the following way:

$$\exp\left(\frac{-2\langle \vec{z}, \vec{v} \rangle + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right) \leq \exp\left(\frac{\|\vec{v}\|^2}{2\mathfrak{s}^2}\right) = M. \tag{35}$$

Hence, for $M \approx 3$ one would select $\mathfrak{s} = 0.675 \cdot \|\vec{v}\|$. Note that the probability for $\vec{z} \leftarrow D_{\sigma}^{\ell d}$ that $\langle \vec{z}, \vec{v} \rangle \geq 0$ is at least $1/2$. Hence, the expected number of

[9] Here, the inner product is over $\mathbb{Z}$, i.e. $\langle \vec{z}, \vec{v} \rangle = \langle \vec{z}, \vec{v} \rangle$ where vectors $\vec{z}, \vec{v}$ are polynomial coefficients of $\vec{z}$ and $\vec{v}$ respectively.

rejections would be at most $2M = 6$. On the other hand, if one aims for $M = 6$ repetitions using (34), then $\mathfrak{s} = 6.74 \cdot \|\vec{\boldsymbol{v}}\|$. Thus, [26] manages to reduce the standard deviation by around a factor of 10.

Lemma A.6 ([26]). *Let $V \subseteq \mathcal{R}^\ell$ be a set of polynomials with norm at most T and $\rho\colon V \to [0,1]$ be a probability distribution. Now, sample $\vec{\boldsymbol{v}} \leftarrow \rho$ and $\vec{\boldsymbol{y}} \leftarrow D_{\mathfrak{s}}^{\ell d}$, set $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \vec{\boldsymbol{v}}$, and run $b \leftarrow \mathsf{Rej}_1(\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \mathfrak{s})$ as defined in Fig. 10. Then, the probability that $b = 0$ is at least $1/(2M)$ and the distribution of $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{z}})$, conditioned on $b = 0$, is identical to the distribution $\mathcal{F}$ where $\mathcal{F}$ is defined as follows: sample $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{z}}) \leftarrow \rho \times D_{\mathfrak{s}}^{ld}$ and if $\langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{z}} \rangle \geq 0$, output $b = 0$ with probability $1/M$ and $b = 1$ otherwise.*

Finally, we highlight that this procedure reveals the sign of $\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{v}} \rangle$. This is still fine when working with "one-time commitments" [26] since we only leak one bit of information. However, secure signature schemes cannot be produced using this method because each generation of a signature reveals some information about the secret key.

By using this technique, zero-knowledge property of the protocol relies on the (simplified) Extended-MLWE problem [26] where the adversary is given the additional one bit of information about the secret. We describe this problem in Section A.2.

B Security Analysis of the One-out-of-Many Proof

We summarise the security properties of the interactive protocol described in Fig. 7 with the following theorem.

Theorem B.1. *The protocol in Fig. 7 is complete, computational honest verifier zero-knowledge under the Extended-MLWE assumption and computational special sound under the Module-SIS assumption. More precisely, let p be the maximum probability over $\mathbb{Z}_q$ of the coefficients of $\boldsymbol{c} \bmod X^{d/l} - \zeta^{2j+1}$ as in Lemma A.1.*

Then, for completeness, the honest prover convinces the honest verifier with probability $\varepsilon \approx 1/(2M^2)$.

For honest-verifier zero-knowledge, there exists a simulator $\mathcal{S}$, that, without access to secret information, outputs a simulation of a non-aborting transcript of the protocol between $\mathcal{P}$ and $\mathcal{V}$. Then for every algorithm $\mathcal{A}$ that has advantage ε in distinguishing the simulated transcript from an actual transcript, there is an algorithm $\mathcal{A}'$ with the same running time that has advantage $\varepsilon - 2^{-100}/M$ in distinguishing Extended-MLWE$_{\kappa+k+2m+1,\lambda,\mathfrak{s}}$.

For soundness, there is an extractor $\mathcal{E}$ with the following properties. When given rewindable black-box access to a prover $\mathcal{P}^*$ that convinces $\mathcal{V}$ with probability $\varepsilon \geq 32\mathsf{p}^{d/l}$ over the random tape $\xi \in \{0,1\}^x$, challenges $\boldsymbol{c}, \boldsymbol{c}' \leftarrow C, (\vec{\gamma}_1, \ldots, \vec{\gamma}_m) \leftarrow \mathcal{M}_q^{(k+2m-1)l}, \vec{\boldsymbol{\alpha}} \leftarrow \mathcal{R}_q^{m+1}$, $\mathcal{E}$ either outputs an index $\iota^* \in [n]$, short polynomial $\boldsymbol{c}^* \in \mathcal{R}_q$ and a short vector $\vec{\boldsymbol{s}}^* \in \mathcal{R}_q^\ell$ such that $\|\boldsymbol{c}^*\|_\infty \leq 2, \|\vec{\boldsymbol{s}}^*\| \leq 2\beta'$ and $\boldsymbol{A}\vec{\boldsymbol{s}}^* = \boldsymbol{c}^* \vec{\boldsymbol{pk}}_{\iota^*}$, or a M-SIS$_{\kappa,8d\beta}$ solution for $\boldsymbol{B}_0$ in expected time $\mathsf{poly}(\mathsf{N})/\varepsilon$.

We prove the statement in the subsections below.

B.1 Correctness

It follows directly from Lemmas A.5 and A.6 that an honest prover does not abort with probability at least $1/(2M^2) - 2^{-100}/M \approx 1/(2M^2)$. Furthermore, the distribution of $\vec{\boldsymbol{z}}'$ has statistical distance at most 2^{-100} from $D_{\mathfrak{s}'}^{\ell d}$ and by the standard tail-bound inequality [3][Lemma 1.5(i)] we have $\|\vec{\boldsymbol{z}}'\| \leq \beta' = \mathfrak{s}'\sqrt{2\ell d}$ with probability $1 - 2^{-\log(e/2)\ell d/4}$ [10]. We argue similarly that $\vec{\boldsymbol{z}}$ satisfies $\|\vec{\boldsymbol{z}}\| \leq \beta = \mathfrak{s}\sqrt{2(k+\kappa+\lambda+k+2m+1)d}$ with an overwhelming probability.

Finally, Lines 13 and 15 follow directly from the discussion in Section 3.1.

B.2 Honest-Verifier Zero-Knowledge

For zero-knowledge we construct an argument similar to [26] and define the hybrid simulator $\mathcal{S}_h$ which outputs a transcript as follows. First, $\mathcal{S}_h$ has the index $\iota \in [n]$ and $\vec{\boldsymbol{s}} \in [-\mu, \mu]^{\ell d}$ such that $\boldsymbol{A}\vec{\boldsymbol{s}} = \boldsymbol{p}\vec{\boldsymbol{k}}_\iota$. Next, it generates the challenges $\boldsymbol{c}, \boldsymbol{c}', \vec{\gamma}_i, \vec{\alpha}_i$ as in Fig. 7. Then, it samples $\vec{\boldsymbol{z}}' \leftarrow D_{\mathfrak{s}'}^{\ell d}$ and continues with probability $1/M$. Thus, $\mathcal{S}_h$ computes $\vec{\boldsymbol{w}}' = \boldsymbol{A}\boldsymbol{z} - \boldsymbol{c}'\boldsymbol{p}\vec{\boldsymbol{k}}_\iota$.

Now, the simulator can generate the randomness $\vec{\boldsymbol{r}} \leftarrow \chi^{(\kappa+\lambda+k+2m+1)d}$ and compute the BDLOP commitments $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k+2m}$ as the prover $\mathcal{P}$ in Fig. 7. Furthermore, it generates $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(\kappa+\lambda+k+2m+1)d}$ and if $\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}} \rangle \geq 0$ over $\mathbb{Z}_q$ then it continues with probability $1/M$. Now, it can also calculate the commitment $\boldsymbol{t}_{k+2m+1}$ by computing $\boldsymbol{\psi}_{\mathsf{sm}}, \boldsymbol{\psi}_{\mathsf{bin}}$ as in (28) and (29) but substituting any inner product $\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle$ with $\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle$.

Finally, $\mathcal{S}_h$ generates a uniformly random polynomial $\boldsymbol{h}$ with the first d/l coefficient equal to zero. Then, variables $\boldsymbol{\omega}$ and $\vec{\boldsymbol{w}}$ are uniquely determined from the verification equations. By Lemmas A.5 and A.6, distribution of the constructed transcript is within statistical distance approximately $2^{-100}/M$ of the distribution of the honest transcript.

Now, we define the actual simulator $\mathcal{S}$. It first samples challenges $\boldsymbol{c}, \boldsymbol{c}'\vec{\gamma}_i, \boldsymbol{\alpha}_i$ identically as in Fig. 7. Then, it generates $\vec{\boldsymbol{t}} \leftarrow \mathcal{R}_q^{\kappa+k+2m+1}$ and a uniformly random polynomial $\boldsymbol{h}$ such that $h_0 = \ldots = h_{d/l-1} = 0$. Next, $\mathcal{S}$ samples $\vec{\boldsymbol{z}}' \leftarrow D_{\mathfrak{s}'}^{\ell d}$ and continues with probability $1/M$. Similarly, it samples $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(\kappa+\lambda+k+2m+1)d}$, generates a new randomness vector $\vec{\boldsymbol{r}}^* \leftarrow \chi^{(\kappa+\lambda+k+2m+1)d}$ and checks whether $\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}}^* \rangle \geq 0$ over $\mathbb{Z}_q$. If so, it outputs the transcript with probability $1/M$.

Suppose that an algorithm $\mathcal{A}$ has advantage at least $\varepsilon - 2^{-100}$ in distinguishing the simulated transcript from $\mathcal{S}_h$ and from $\mathcal{S}$. We construct an adversary $\mathcal{A}'$ which distinguishes Extended-MLWE$_{\kappa+k+2m+1,\lambda,\mathfrak{s}}$ with the same probability. At the beginning, $\mathcal{A}'$ is given a challenge tuple $(\boldsymbol{B}, \vec{\boldsymbol{u}}, \boldsymbol{c}, \vec{\boldsymbol{z}}, s)$ where we denote

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_0 \\ \vec{\boldsymbol{b}}_1 \\ \vdots \\ \vec{\boldsymbol{b}}_{k+2m+1} \end{pmatrix} \text{ and } \vec{\boldsymbol{u}} = \begin{pmatrix} \vec{\boldsymbol{u}}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{k+2m+1} \end{pmatrix}.$$

[10] We choose parameters ℓ and d in Fig. 11 so that the probability is overwhelming.

First, $\mathcal{A}'$ runs $\mathcal{A}$ which outputs the correct statement

$$\left((\boldsymbol{A}, p\vec{\boldsymbol{k}}_1, \ldots, p\vec{\boldsymbol{k}}_n), (\vec{\boldsymbol{s}}, \iota)\right)$$

such that $\boldsymbol{A}\vec{\boldsymbol{s}} = p\vec{\boldsymbol{k}}_\iota$ and $\|\vec{\boldsymbol{s}}\|_\infty \leq 1$. Then, $\mathcal{A}'$ generates challenges $\boldsymbol{c}', \vec{\gamma}_i, \boldsymbol{\alpha}_j$ as before and samples $\boldsymbol{z}' \leftarrow D^{\ell d}_{\mathsf{s}'}$ and continues with probability $1/M$. Then, it computes $\vec{\boldsymbol{w}}' = \boldsymbol{A}\vec{\boldsymbol{z}}' - \boldsymbol{c}' p\vec{\boldsymbol{k}}_\iota$. Now, $\mathcal{A}'$ calculates $\vec{x}_i, \vec{y}_i, \boldsymbol{g}$ as in Fig. 7. Lastly, it computes $\boldsymbol{\psi} = \boldsymbol{\psi}_{\mathsf{sm}} + \boldsymbol{\psi}_{\mathsf{bin}}$ as in (28) and (29) but substituting each $\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle$ by $\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{u}_i$. Then, $\mathcal{A}$ creates a message vector

$$\vec{\boldsymbol{m}} = \left(\vec{\boldsymbol{0}}, \boldsymbol{v}_1, \ldots, \vec{\boldsymbol{v}}_m, \vec{\boldsymbol{w}}', \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m, \boldsymbol{g}, \boldsymbol{\psi}\right) \in \mathcal{R}_q^{\kappa+k+2m+1}$$

where $\boldsymbol{v}_i = \mathsf{NTT}^{-1}(\vec{v}_i)$ and similarly for $\boldsymbol{x}_i$. Next, $\mathcal{A}'$ computes the commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{u}} + \vec{\boldsymbol{m}}$ as well as $\boldsymbol{\omega}, \mathbf{w}, \vec{\boldsymbol{w}}$ from the verification equations. Then, if $s = 0$ then $\mathcal{A}'$ aborts. Otherwise, it outputs the constructed transcript $\mathcal{T}$ to $\mathcal{A}$ with probability $1/M$. Eventually, when $\mathcal{A}$ sends a bit b, $\mathcal{A}'$ also returns b.

One observes that if $\vec{\boldsymbol{u}} = \boldsymbol{B}\vec{\boldsymbol{r}}$ for some $\vec{\boldsymbol{r}}$ then $\mathcal{T}$ is a perfectly simulated transcript from $\mathcal{S}_h$. However, when $\vec{\boldsymbol{u}} \leftarrow \mathcal{R}_q^{\kappa+k+2m+1}$ then $\vec{\boldsymbol{t}}$ is also uniformly random. Hence, $\mathcal{T}$ is a perfectly simulated transcript from $\mathcal{S}$.

B.3 Knowledge Soundness

We first recall the following heavy-rows lemma which can be easily generalised from [9].

Lemma B.2. *Let $K \in \mathbb{N}$, $H \in \{0,1\}^{n \times m}$, for some $n, m > 1$, be a matrix such that a fraction ε of the inputs are 1. We say that a row is "heavy" if it contains a fraction at least ε/K of ones. Then, more than $(1 - 1/K)$ of ones in H are located in heavy rows.*

We set $K = 4$. Note that by assumption, the success probability of the prover $\mathcal{P}^*$ is at least $\varepsilon > 2K^2\mathsf{p}^{d/l}$.

In order to use the heavy-rows lemma, we slightly modify the protocol in Fig. 7 so that, instead of generating $\boldsymbol{c} \leftarrow C$, the verifier sends two uniformly random binary polynomials $\boldsymbol{c}_0, \boldsymbol{c}_1 \leftarrow \{0,1\}^d$ and both parties set the actual challenge $\boldsymbol{c} = \boldsymbol{c}_0 - \boldsymbol{c}_1$. Clearly, the distribution of $\boldsymbol{c}$ is the same as for $\boldsymbol{c} \leftarrow C$. Similarly, we modify for $\boldsymbol{c}' \leftarrow C$ and pick uniformly random binary polynomials $\boldsymbol{c}'_0$ and $\boldsymbol{c}'_1$.

Let us define the matrix H_0 as follows. The rows of H_0 are indexed by all possible random tapes $\xi \in \{0,1\}^x$ and columns of H_0 are indexed by all possible values for $(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}}, \boldsymbol{c}_0, \boldsymbol{c}_1) \in \{0,1\}^d \times \{0,1\}^d \times \mathcal{M}_q^{(k+2m-1)l} \times \mathcal{R}_q^{m+1} \times \{0,1\}^d \times \{0,1\}^d$. Then we set

$$H_0[\xi][(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}}, \boldsymbol{c}_0, \boldsymbol{c}_1)] = 1$$

if $\mathcal{P}^*$ convinces the verifier for given random tape and challenges and 0 otherwise. It is easy to see that H_0 contains ε fraction of ones, where ε is the success probability.

Moreover, for $\xi \in \{0,1\}^x$, we define additional matrix H_ξ as follows. Namely, the rows of H_ξ are indexed by all possible random challenges $(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}}) \in \{0,1\}^d \times \{0,1\}^d \times \mathcal{M}_q^{(k+2m-1)l} \times \mathcal{R}_q^{m+1}$ and columns of H_0 are indexed by all possible values for $(\boldsymbol{c}_0, \boldsymbol{c}_1) \in \{0,1\}^d \times \{0,1\}^d$. Then, we set

$$H_\xi[(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}})][(\boldsymbol{c}_0, \boldsymbol{c}_1)] := H_0[\xi][(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}}, \boldsymbol{c}_0, \boldsymbol{c}_1)].$$

Extractor $\mathcal{E}$ runs the following algorithm $\mathcal{E}'$ $O(\mathsf{N})$ times:

1. Run $\mathcal{P}^*$ on random tape $\xi \leftarrow \{0,1\}^x$, challenges $(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}}, \boldsymbol{c}_0, \boldsymbol{c}_1)$ until it succeeds. This takes expected $1/\varepsilon$ time.
2. Run $\mathcal{P}^*$ on the random tape ξ and fresh challenges $(\boldsymbol{c}^*_0, \boldsymbol{c}^*_1, \vec{\gamma}^\ddagger_1, \ldots, \vec{\gamma}^\ddagger_m, \vec{\boldsymbol{\alpha}}^\ddagger, \boldsymbol{c}^\ddagger_0, \boldsymbol{c}^\ddagger_1)$ until it succeeds and $\boldsymbol{c}^*_0 - \boldsymbol{c}^*_1 \neq \boldsymbol{c}'_0 - \boldsymbol{c}'_1$. If after N/ε attempts $\mathcal{P}^*$ has not returned a correct transcript, abort.
3. Run $\mathcal{P}^*$ on the random tape ξ and challenges $(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}}, \tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1)$ where $\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1 \leftarrow \{0,1\}^d$ are freshly sampled. If $(\boldsymbol{c}_0 - \boldsymbol{c}_1) - (\tilde{\boldsymbol{c}}_0 - \tilde{\boldsymbol{c}}_1)$ is not invertible over $\mathcal{R}_q$ then restart. If after N/ε attempts $\mathcal{P}^*$ has not returned a valid response $\vec{\boldsymbol{z}}$, abort.
4. Run $\mathcal{P}^*$ on the random tape ξ and challenges $(\boldsymbol{c}^*_0, \boldsymbol{c}^*_1, \vec{\gamma}^\ddagger_1, \ldots, \vec{\gamma}^\ddagger_m, \vec{\boldsymbol{\alpha}}^\ddagger, \tilde{\boldsymbol{c}}^\ddagger_0, \tilde{\boldsymbol{c}}^\ddagger_1)$ where $\tilde{\boldsymbol{c}}^\dagger_0, \tilde{\boldsymbol{c}}^\ddagger_1 \leftarrow \{0,1\}^d$ are freshly sampled. If $(\boldsymbol{c}^\ddagger_0 - \boldsymbol{c}^\ddagger_1) - (\tilde{\boldsymbol{c}}^\ddagger_0 - \tilde{\boldsymbol{c}}^\ddagger_1)$ is not invertible over $\mathcal{R}_q$ then restart. If after N/ε attempts $\mathcal{P}^*$ has not returned a valid response $\vec{\boldsymbol{z}}$, abort.

Clearly, the algorithm runs in expected time $\mathsf{poly}(\mathsf{N})/\varepsilon$.

We analyse the abort probability of $\mathcal{E}'$. Let abort be the event that $\mathcal{E}'$ aborts and abort_i be the event that it aborts in Step i. Also, denote heavy_0 to be the event that ξ is a heavy row of H_0. Then, by Lemma B.2:

$$\Pr[\mathsf{abort}] \leq \Pr[\mathsf{abort}|\mathsf{heavy}_0] + \Pr[\neg\mathsf{heavy}_0] < \Pr[\mathsf{abort}|\mathsf{heavy}_0] + \frac{1}{K}.$$

and by the union bound: $\Pr[\mathsf{abort}|\mathsf{heavy}_0] \leq \sum_{i=2}^{4} \Pr[\mathsf{abort}_i|\mathsf{heavy}_0]$.

First, we focus on bounding $\Pr[\mathsf{abort}_2|\mathsf{heavy}_0]$. Assuming that the row ξ is heavy, with probability at least $\varepsilon/K - 2^{-128} > \varepsilon/(2K)$ the challenges $(\boldsymbol{c}^*_0, \boldsymbol{c}^*_1, \vec{\gamma}^\ddagger_1, \ldots, \vec{\gamma}^\ddagger_m, \vec{\boldsymbol{\alpha}}^\ddagger, \boldsymbol{c}^\ddagger_0, \boldsymbol{c}^\ddagger_1)$ give a successful transcript and $\boldsymbol{c}^\ddagger_0 - \boldsymbol{c}^\ddagger_1 \neq \boldsymbol{c}'_0 - \boldsymbol{c}'_1$. Hence, the probability that all N/ε trials fail can be bounded by:

$$\Pr[\mathsf{abort}_2|\mathsf{heavy}_0] < (1 - \varepsilon/(2K))^{\lambda/\varepsilon} < \exp(-2K\mathsf{N}).$$

Next, we concentrate on bounding $\Pr[\mathsf{abort}_3|\mathsf{heavy}_0]$. First, note that heavy_0 implies that at least $\varepsilon^* := \varepsilon/K$ of all inputs of the matrix H_ξ are ones. Let us define the event heavy_1 where $(\boldsymbol{c}'_0, \boldsymbol{c}'_1, \vec{\gamma}_1, \ldots, \vec{\gamma}_m, \vec{\boldsymbol{\alpha}})$ (used in Step 1) is a heavy row in H_ξ, i.e. it contains a fraction at least ε^*/K of ones. Then, again by Lemma B.2 we get:

$$\Pr[\mathsf{abort}_3|\mathsf{heavy}_0] < \Pr[\mathsf{abort}_3|\mathsf{heavy}_0 \wedge \mathsf{heavy}_1] + 1/K.$$

Now, if the row is heavy then for a new random sample $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1)$ the prover $\mathcal{P}^*$ has probability at least $\varepsilon^*/K - \mathsf{p}^{d/l} > \varepsilon^*/(2K)$ of obtaining a valid response

(recall that we need $(\boldsymbol{c}_0 - \boldsymbol{c}_1) - (\tilde{\boldsymbol{c}}_0 - \tilde{\boldsymbol{c}}_1)$ to be invertible). Thus, the probability that all N/ε trials fail can be bounded by:

$$\Pr[\mathsf{abort}_3 | \mathsf{heavy}_0 \wedge \mathsf{heavy}_1] < (1 - \varepsilon^*/(2K))^{\mathsf{N}/\varepsilon} < \exp(-2K^2\mathsf{N})$$

We can analogously upper-bound $\Pr[\mathsf{abort}_4 | \mathsf{heavy}_0]$ and get:

$$\Pr[\mathsf{abort}] \leq \frac{3}{K} + \exp(-2K\mathsf{N}) + 2\exp(-2K^2\mathsf{N}) < \frac{3}{4} + 2^{-\mathsf{N}}.$$

Hence, by running $\mathcal{E}'$ $O(\mathsf{N})$ times, $\mathcal{E}$ manages to get the following four accepting transcripts:

$$\mathcal{T}_{0,0} = \left(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \vec{\boldsymbol{w}}, \boldsymbol{t}_{m+k+1}, \ldots, \boldsymbol{t}_{k+2m-1}, \boldsymbol{t}_{k+2m}, \boldsymbol{t}_{k+2m+1}, \boldsymbol{c}', \vec{\boldsymbol{z}}', \vec{\gamma}_i, \boldsymbol{h}, \boldsymbol{\alpha}_i, \boldsymbol{\omega}, \boldsymbol{c}, \vec{\boldsymbol{z}}\right)$$
$$\mathcal{T}_{0,1} = \left(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \vec{\boldsymbol{w}}, \boldsymbol{t}_{m+k+1}, \ldots, \boldsymbol{t}_{k+2m-1}, \boldsymbol{t}_{k+2m}, \boldsymbol{t}_{k+2m+1}, \boldsymbol{c}', \vec{\boldsymbol{z}}', \vec{\gamma}_i, \boldsymbol{h}, \boldsymbol{\alpha}_i, \boldsymbol{\omega}, \tilde{\boldsymbol{c}}, \tilde{\vec{\boldsymbol{z}}}\right)$$
$$\mathcal{T}_{1,0} = \left(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \vec{\boldsymbol{w}}, \boldsymbol{t}^{\ddagger}_{m+k+1}, \ldots, \boldsymbol{t}^{\ddagger}_{k+2m-1}, \boldsymbol{t}_{k+2m}, \boldsymbol{t}^{\ddagger}_{k+2m+1}, \boldsymbol{c}^*, \tilde{\vec{\boldsymbol{z}}}', \vec{\gamma}^{\ddagger}_i, \boldsymbol{h}^{\ddagger}, \boldsymbol{\alpha}^{\ddagger}_i, \boldsymbol{\omega}^{\ddagger}, \boldsymbol{c}^{\ddagger}, \vec{\boldsymbol{z}}^{\ddagger}\right)$$
$$\mathcal{T}_{1,1} = \left(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \vec{\boldsymbol{w}}, \boldsymbol{t}^{\ddagger}_{m+k+1}, \ldots, \boldsymbol{t}^{\ddagger}_{k+2m-1}, \boldsymbol{t}_{k+2m}, \boldsymbol{t}^{\ddagger}_{k+2m+1}, \boldsymbol{c}^*, \tilde{\vec{\boldsymbol{z}}}', \vec{\gamma}^{\ddagger}_i, \boldsymbol{h}^{\ddagger}, \boldsymbol{\alpha}^{\ddagger}_i, \boldsymbol{\omega}^{\ddagger}, \tilde{\boldsymbol{c}}^{\ddagger}, \tilde{\vec{\boldsymbol{z}}}^{\ddagger}\right)$$

where $\boldsymbol{c} - \tilde{\boldsymbol{c}}, \boldsymbol{c}^{\ddagger} - \tilde{\boldsymbol{c}}^{\ddagger}$ are invertible over $\mathcal{R}_q$ and $\boldsymbol{c} \neq \boldsymbol{c}^{\ddagger}$.

Let us consider transcripts $\mathcal{T}_{0,0}$ and $\mathcal{T}_{0,1}$. From the verification equations we obtain:

$$\boldsymbol{B}_0(\vec{\boldsymbol{z}} - \tilde{\vec{\boldsymbol{z}}}) = (\boldsymbol{c} - \tilde{\boldsymbol{c}})\vec{\boldsymbol{t}}_0 = \bar{\boldsymbol{c}}\vec{\boldsymbol{t}}_0 \tag{36}$$

where $\bar{\boldsymbol{c}} = \boldsymbol{c} - \tilde{\boldsymbol{c}}$. Define $\vec{\boldsymbol{r}}^* = \bar{\boldsymbol{c}}^{-1}(\vec{\boldsymbol{z}} - \tilde{\vec{\boldsymbol{z}}})$ and $\vec{\boldsymbol{y}}^* = \vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{r}}^*$. Then, $\boldsymbol{B}_0\vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0$ and $\boldsymbol{B}_0\vec{\boldsymbol{y}}^* = \boldsymbol{B}\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0 = \vec{\boldsymbol{w}}$.

Consider an arbitrary accepting transcript with different last challenge $\boldsymbol{c}^{\dagger}$ and final response $\vec{\boldsymbol{z}}^{\dagger}$. Let us define $\bar{\boldsymbol{c}}^{\dagger} = \boldsymbol{c} - \boldsymbol{c}^{\dagger}$ and $\vec{\boldsymbol{y}}^{\dagger} := \vec{\boldsymbol{z}}^{\dagger} - \boldsymbol{c}^{\dagger}\vec{\boldsymbol{r}}^*$. We claim that $\vec{\boldsymbol{y}}^{\dagger} = \vec{\boldsymbol{y}}^*$. Indeed, from the verification equations we have:

$$\boldsymbol{B}_0(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}^{\dagger}) = \bar{\boldsymbol{c}}^{\dagger}\vec{\boldsymbol{t}}_0.$$

Combining this with (36) we obtain:

$$\boldsymbol{B}_0\left(\bar{\boldsymbol{c}}^{\dagger}(\vec{\boldsymbol{z}} - \tilde{\vec{\boldsymbol{z}}}) - \bar{\boldsymbol{c}}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}^{\dagger})\right) = \vec{\boldsymbol{0}}.$$

Hence, unless we find a M-SIS solution for $\boldsymbol{B}_0$, we have

$$\bar{\boldsymbol{c}}^{\dagger}(\vec{\boldsymbol{z}} - \tilde{\vec{\boldsymbol{z}}}) = \bar{\boldsymbol{c}}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}^{\dagger})$$

which implies that $\vec{\boldsymbol{y}}^* - \vec{\boldsymbol{y}}^{\dagger} = (\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}^{\dagger}) - \bar{\boldsymbol{c}}^{\dagger}\vec{\boldsymbol{r}}^* = \vec{\boldsymbol{0}}$.

Now, we extract the messages $\boldsymbol{m}^*_1, \ldots, \boldsymbol{m}^*_{k+2m+1} \in \mathcal{R}_q$:

$$\boldsymbol{m}^*_i := \boldsymbol{t}_i - \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^* \rangle \tag{37}$$

and conclude that we obtained the weak opening $(\bar{\boldsymbol{c}}, \vec{\boldsymbol{r}}^*, \boldsymbol{m}^*_1, \ldots, \boldsymbol{m}^*_{k+2m+1})$ of the commitment $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k+2m+1})$. In particular, we have $\|\bar{\boldsymbol{c}}\vec{\boldsymbol{r}}^*\| \leq 2\beta$.

Let us consider Lines 10-13 of the verification equations in Fig. 9. Note that we can write

$$\boldsymbol{f}_j = \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_j = \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}^* \rangle - \boldsymbol{c}\boldsymbol{m}^*_j \text{ for } j \in [k+2m+1].$$

Denote $\boldsymbol{v}_i = \boldsymbol{m}^*_i$ for $i \in [m]$, $\boldsymbol{x}_{1,i} = \boldsymbol{w}'_i = \boldsymbol{m}^*_{m+i}$ for $i \in [k]$ as well as $\boldsymbol{x}_{j+1} = \boldsymbol{m}^*_{k+m+j}$ for $j \in [m-1]$. Then, by evaluating the equations in Lines 10-13 with $\boldsymbol{f}_j$ above, we can compute polynomials $\boldsymbol{\omega}^*_j, \boldsymbol{\psi}^*_j, \boldsymbol{y}^*_j$ defined as:

$$\begin{aligned}\boldsymbol{F}_j &= \boldsymbol{\omega}^*_j + \boldsymbol{\psi}^*_j \boldsymbol{c} + \boldsymbol{y}^*_j \boldsymbol{c}^2 \text{ for } j \in [m-1],\\ \boldsymbol{F}_m &= \boldsymbol{\psi}^*_m \boldsymbol{c} + \boldsymbol{y}^*_m \boldsymbol{c}^2.\end{aligned}$$

Thus, the equation in Line 13 can be written equivalently as a quadratic equation in $\boldsymbol{c}$ of degree two:

$$f(\boldsymbol{c}) = \boldsymbol{\mu}_2 \boldsymbol{c}^2 + \boldsymbol{\mu}_1 \boldsymbol{c} + \boldsymbol{\mu}_0$$

where the coefficients of f are independent of $\boldsymbol{c}$ and

$$\boldsymbol{\mu}_2 := \sum_{i=1}^{m} \boldsymbol{\alpha}_i \boldsymbol{v}_i(\boldsymbol{v}_i - \boldsymbol{1}) + \boldsymbol{\alpha}_0(\boldsymbol{y}^*_1 + \ldots + \boldsymbol{y}^*_m + \boldsymbol{g} - \boldsymbol{h}). \tag{38}$$

We also highlight that $\boldsymbol{v}_i, \boldsymbol{y}^*_i, \boldsymbol{g}$ and $\boldsymbol{h}$ are independent of the challenge polynomials $\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m \leftarrow \mathcal{R}_q$.

First, suppose that for some $i \in [m]$ we have $\boldsymbol{v}_i(\boldsymbol{v}_i - \boldsymbol{1}) \neq \boldsymbol{0}$. This implies that $\boldsymbol{\mu}_2 = \boldsymbol{0}$ with probability $q^{-d/l}$. Hence, assume that $\boldsymbol{\mu}_2 \neq \boldsymbol{0}$. Consequently, there exists a $j \in \mathbb{Z}_l$ such that $\boldsymbol{\mu}_2 \neq \boldsymbol{0} \pmod{X^{d/l} - \zeta^{2j+1}}$. Note that there are at most two distinct $\boldsymbol{a}_1, \boldsymbol{a}_2 \in \mathcal{M}_q$ such that $f(\boldsymbol{a}_1) = f(\boldsymbol{a}_2) = \boldsymbol{0}$ modulo $X^{d/l} - \zeta^{2j+1}$. Hence, by the union bound, the probability that $\boldsymbol{c} = \boldsymbol{a}_b \pmod{X^{d/l} - \zeta^{2j+1}}$ for some $b \in \{0,1\}$ is at most $2\mathsf{p}^{d/l}$. Therefore, the success probability of the prover can be bounded by $2\mathsf{p}^{d/l} + q^{-d/l}$ which is by our assumption less than ε. Hence, we must have $\boldsymbol{v}_i(\boldsymbol{v}_i - \boldsymbol{1}) = \boldsymbol{0}$, i.e. vectors $\vec{v}_i = \mathsf{NTT}(\boldsymbol{v}_i)$ are binary. Similarly, we define $\vec{x}_i = \mathsf{NTT}(\vec{\boldsymbol{x}}_i)$ for $i \in [m-1]$.

Let $\vec{x}_{m+1} = (\vec{x}_{m+1,1}, \ldots, \vec{x}_{m+1,m}) := \tilde{P}_m^T \vec{\gamma}_m \in \mathcal{M}_q^{ml}$ where $\tilde{P}_m$ is the matrix in (23) Then, from discussion in Section 3.1 and Line 12 of Fig. 9 we know that the first d/l coefficients of $\boldsymbol{y}^*_m$ satisfy:

$$y_{m,0} + \ldots + y_{m,d/l-1} X^{d/l-1} = \frac{1}{l} \left\langle \tilde{P}_m \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix} - \begin{pmatrix} \vec{x}_m \\ \vec{e}_1 \\ \vdots \\ \vec{e}_1 \end{pmatrix}, \vec{\gamma}_m \right\rangle.$$

Note that we can use Lemma 2.1 since we already proved $\vec{v}_1, \ldots, \vec{v}_m \in \mathbb{Z}_q^l$.

Suppose that

$$\tilde{P}_m \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix} \neq \begin{pmatrix} \vec{x}_m \\ \vec{e}_1 \\ \vdots \\ \vec{e}_1 \end{pmatrix}.$$

Firstly, using an identical argument as above, if $\boldsymbol{h} \neq \boldsymbol{g}+\boldsymbol{y}_1^*+\ldots+\boldsymbol{y}_m^*$ then $\boldsymbol{\mu}_2 = \boldsymbol{0}$ with probability $q^{-d/l}$. Hence, assume that $\boldsymbol{\mu} \neq \boldsymbol{0}$. Then, with probability at most $2\mathsf{p}^{d/l}$ we have $f(\boldsymbol{c}) = \boldsymbol{0}$ for $\boldsymbol{c} \leftarrow C$. Otherwise, assume

$$\boldsymbol{h} = \boldsymbol{g} + \boldsymbol{y}_1^* + \ldots + \boldsymbol{y}_m^*.$$

Then, $y_{m,0}+\ldots+y_{m,d/l-1}X^{d/l-1}$ is a uniformly random polynomial in $\mathcal{M}_q$ since $\vec{\gamma}_m \leftarrow \mathcal{M}_q^{(m+1)l}$. Since $\boldsymbol{g}, \boldsymbol{y}_1^*, \ldots, \boldsymbol{y}_{m-1}^*$ are independent of $\vec{\gamma}_m$ and the first d/l coefficients of $\boldsymbol{h}$ are all zeroes, the probability that

$$y_{m,0} + \ldots + y_{m,d/l-1}X^{d/l-1} = -(\boldsymbol{g} + \boldsymbol{y}_1^* + \ldots + \boldsymbol{y}_{m-1}^*) \bmod X^{d/l}$$

is $q^{-d/l}$. Since the success probability ε of a prover is more than $2\mathsf{p}^{d/l}+q^{-d/l}$, we obtain Equation 23 which says that each $\vec{v}_i$ contains only one 1 and $\hat{P}_m\vec{v}_m = \vec{x}_m$.

Now, we conduct an inductive argument and assume that for some $j \in [m-1]$ we have

$$\hat{P}_{j+1}(\vec{v}_{j+1} \otimes \ldots \otimes \vec{v}_m) = \vec{x}_{j+1}.$$

This automatically implies that the first d/l coefficients of $\boldsymbol{y}_{j+1}^*, \ldots, \boldsymbol{y}_m^*$ are all zeroes. Since $\vec{v}_j \in \mathbb{Z}_q^l$, we deduce from our argument in Section 3.1 that the first d/l coefficients of $\boldsymbol{y}_j$ satisfy:

$$y_{j,0} + \ldots + y_{j,d/l-1}X^{d/l-1} = \frac{1}{l}\langle \hat{P}_j(\vec{v}_j \otimes \ldots \otimes \vec{v}_m) - \vec{x}_j, \vec{\gamma}_j\rangle.$$

Suppose $\hat{P}_j(\vec{v}_j \otimes \ldots \otimes \vec{v}_m) \neq \vec{x}_j$. This implies that $y_{j,0} + \ldots + y_{j,d/l-1}X^{d/l-1}$ is a uniformly random polynomial in $\mathcal{M}_q$. Arguing similarly as above, we get that the success probability when $\boldsymbol{h} \neq \boldsymbol{g} + \boldsymbol{y}_1^* + \ldots + \boldsymbol{y}_m^*$ is at most $q^{-d/l} + 2\mathsf{p}^{d/l}$. Otherwise, since polynomials $\boldsymbol{g}, \boldsymbol{y}_1^*, \ldots, \boldsymbol{y}_{j-1}^*$ are independent of $\vec{\gamma}_j$, we get that:

$$y_{j+1,0} + \ldots + y_{j+1,d/l-1}X^{d/l-1} = -(\boldsymbol{g} + \boldsymbol{y}_1^* + \ldots + \boldsymbol{y}_{j-1}^*) \bmod X^{d/l}$$

with negligible probability $q^{-d/l}$. Since $\varepsilon > q^{-d/l} + 2\mathsf{p}^{d/l}$, we must have $\hat{P}_j(\vec{v}_j \otimes \ldots \otimes \vec{v}_m) = \vec{x}_j$. In particular, for $j = 1$ we get

$$\hat{P}_1(\vec{v}_1 \otimes \cdots \otimes \vec{v}_m) = \vec{x}_1.$$

In summary, we obtain a binary vector $\vec{v} = \vec{v}_1 \otimes \cdots \otimes \vec{v}_m$ with one 1 in some ι-th position and:

$$\boldsymbol{A}\vec{\boldsymbol{z}}' = \vec{\boldsymbol{w}}' + \boldsymbol{c}'\vec{\boldsymbol{p}\boldsymbol{k}}_\iota$$

where $\vec{\boldsymbol{w}}' = \vec{\boldsymbol{x}}_1 + \boldsymbol{A}\vec{\boldsymbol{z}}'$.

Now, extractor $\mathcal{E}$ applies the exact approach as above for transcripts $\mathcal{T}_{1,0}$ and $\mathcal{T}_{1,1}$. Concretely, $\mathcal{E}$ manages to obtain a weak opening $(\vec{\boldsymbol{c}}^\ddagger, \vec{\boldsymbol{r}}^\ddagger, \boldsymbol{m}_i^\ddagger)$ for $\vec{\boldsymbol{t}}^\ddagger = (\vec{\boldsymbol{t}}_0, \ldots, \boldsymbol{t}_{k+m}, \boldsymbol{t}_{k+m+1}^\ddagger, \ldots, \boldsymbol{t}_{k+2m-1}^\ddagger, \boldsymbol{t}_{k+2m}, \boldsymbol{t}_{k+2m+1}^\ddagger)$. Let $\vec{v}_i^\ddagger = \mathsf{NTT}\left(\boldsymbol{m}_i^\ddagger\right)$ and $\boldsymbol{w}_j^* = \boldsymbol{m}_{m+j}^\ddagger$ for $i \in [m], j \in [k]$. Denote $\vec{v}^\ddagger = \vec{v}_1^\ddagger \otimes \cdots \otimes \vec{v}_m^\ddagger$ and $\vec{\boldsymbol{w}}^* =$

$(\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_k^*)$. Then, $\vec{v}^\ddagger$ is a binary vector with exactly one 1 in the ι^*-th position, for some $\iota^* \in [n]$ and

$$\boldsymbol{A}\tilde{\vec{\boldsymbol{z}}}' = \vec{\boldsymbol{w}}^* + \boldsymbol{c}^* \vec{\boldsymbol{pk}}_{\iota^*}.$$

We claim that $\boldsymbol{m}_i^* = \boldsymbol{m}_i^\dagger$ for all $i \in [k+m]$ unless we find a M-SIS solution. This would imply that $\vec{\boldsymbol{w}}' = \vec{\boldsymbol{w}}^*, \vec{v} = \vec{v}^\ddagger$ and $\iota = \iota^*$. Suppose that for some $i \in [k+m]$ we have $\boldsymbol{m}_i^* \neq \boldsymbol{m}_i^\dagger$. This implies that $\vec{\boldsymbol{r}}^* \neq \vec{\boldsymbol{r}}^\ddagger$ and thus:

$$\vec{\boldsymbol{B}}_0(\bar{\boldsymbol{c}}^\ddagger \bar{\boldsymbol{c}} \vec{\boldsymbol{r}} - \bar{\boldsymbol{c}} \bar{\boldsymbol{c}}^\ddagger \vec{\boldsymbol{r}}) = \vec{\boldsymbol{0}}.$$

Since $\bar{\boldsymbol{c}}, \bar{\boldsymbol{c}}^\ddagger$ are invertible, we obtain a non-trivial M-SIS solution with the Euclidean norm at most $8d\beta$.

Finally, we conclude that $\mathcal{E}$ finds $\vec{\boldsymbol{z}}'', \tilde{\vec{\boldsymbol{z}}}' \in \mathcal{R}_q^{\ell d}$ with Euclidean norm at most β', polynomials $\boldsymbol{c}', \tilde{\boldsymbol{c}} \in \mathcal{C}$ and an index $\iota \in [n]$ such that:

$$\boldsymbol{A}(\vec{\boldsymbol{z}}'' - \tilde{\vec{\boldsymbol{z}}}') = (\boldsymbol{c}' - \boldsymbol{c}^*)\vec{\boldsymbol{pk}}_\iota.$$

C Ring Signature Construction

In this section we apply the multi-round Fiat-Shamir transformation on the protocol in Fig. 7 to obtain an efficient lattice-based ring signature. Finally, we include small optimisations which slightly reduce the signature sizes.

C.1 Definitions

We start by recalling the standard definition of a ring signature. Namely, it consists of four algorithms $(\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Ver})$ which are defined below:

- $\mathsf{pp} \leftarrow \mathsf{RS.Setup}(1^{\mathsf{N}})$: on input a security parameter N, it generates public paremeters pp used by the scheme.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp})$: on input pp it outputs the public/secret key pair.
- $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m}, L)$: on input pp, secret key sk, message m and a ring $L = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, it outputs a signature σ.
- $b \leftarrow \mathsf{RS.Ver}(\mathsf{pp}, \sigma, \mathsf{m}, L)$: on input pp, signature σ, message m and a ring L, it deterministically outputs a bit b.

We consider the following properties of ring signatures.

Definition C.1 (Correctness). *A ring signature* $\mathsf{RS} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Ver})$ *provides statistical correctness if for every* $\mathsf{pp} \leftarrow \mathsf{RS.Setup}$*, every pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RS.KeyGen}$*, every ring* R*, such that* $\mathsf{pk} \in L$ *and* $|L| = \mathsf{poly}(\mathsf{N})$*, and every message* m *we have:*

$$\Pr\left[\mathsf{RS.Ver}(\mathsf{pp}, \mathsf{RS.Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m}, R), \mathsf{m}, L) = 1\right] = 1 - \mathsf{negl}(\mathsf{N}).$$

Definition C.2 (Anonymity). *A ring signature* $\mathsf{RS} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Ver})$ *is anonymous if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{matrix}\mathsf{pp} \leftarrow \mathsf{RS.Setup}(1^{\mathsf{N}}); (\mathsf{m}, i_0, i_1, L) \leftarrow \mathcal{A}^{\mathsf{RS.KeyGen}(\mathsf{pp})} \\ b \leftarrow \{0,1\}; \sigma \leftarrow \mathsf{RS.Sign}(\mathsf{pp}, \mathsf{sk}_{i_b}, \mathsf{m}, L)\end{matrix} : b' \leftarrow \mathcal{A}(\sigma)\right] = \frac{1}{2} + \mathsf{negl}(\mathsf{N})$$

where $L = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$*, each* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *is generated from* $\mathsf{RS.KeyGen}(\mathsf{pp})$ *and* $i_0, i_1 \in [n]$*.*

Definition C.3 (Unforgeability). *A ring signature* $\mathsf{RS} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Ver})$ *is unforgeable (w.r.t. insider collusion) if for all PPT adversaries* $\mathcal{A}$*:*

$$\Pr\left[\begin{matrix}\mathsf{pp} \leftarrow \mathsf{RS.Setup}(1^{\mathsf{N}}); \\ (\sigma, \mathsf{m}, L) \leftarrow \mathcal{A}^{\mathsf{PKGen},\mathsf{Sign},\mathsf{Corrupt}}(\mathsf{pp})\end{matrix} : \mathsf{RS.Ver}(\mathsf{pp}, \sigma, \mathsf{m}, L) = 1\right] = \mathsf{negl}(\mathsf{N})$$

where

- PKGen*: on the* i*-th query picks a randomness* ρ_i*, runs* $(\mathsf{pk}_i, \mathsf{sk}_i) = \mathsf{RS.KeyGen}(\mathsf{pp}; \rho_i)$ *and outputs* pk_i*,*
- $\mathsf{Sign}(i, \mathsf{m}, L)$ *: returns* $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{m}, L)$ *provided* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *has been generated by* PKGen*,*
- $\mathsf{Corrupt}(i)$ *: returns the randomness* ρ_i *provided* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *has been generated by* PKGen*,*
- $\mathcal{A}$ *outputs* (σ, m, L) *so that* $\mathsf{Sign}(\cdot, \mathsf{m}, L)$ *has not been queried and* L *contains only public keys* pk_i *generated by* PKGen *where* $\mathsf{Corrupt}(i)$ *has not been queried.*

C.2 Multi-round Fiat-Shamir Transformation

We construct a ring signature for (at most) $n = l^m$ users from an interactive protocol in Fig. 7 by applying the multi-round Fiat-Shamir transform (see [12, Definition 11]). In order to do so, we need four random oracles H_1, H_k, H_{m+1}, where each $H_i : \{0,1\}^* \to \mathcal{M}_q^{il}$, and $G : \{0,1\}^* \to \mathcal{C}$ where G follows the distribution C (see Section A.1).

We define the ring signature $\mathsf{RS} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Ver})$ as follows:

- $\mathsf{RS.Setup}(1^{\mathsf{N}})$: given a security parameter, it first generates appropriate parameters $q, l, d, k, \ell, \kappa, \lambda$ so that the corresponding M-LWE and M-SIS problems are hard (e.g. Fig. 11). Then, it samples $\boldsymbol{A} \leftarrow \mathcal{R}_q^{k \times \ell}, \boldsymbol{B}_0 \leftarrow \mathcal{R}_q^{\kappa \times (\kappa+\lambda+k+2m+1)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{k+2m+1} \leftarrow \mathcal{R}_q^{\kappa+\lambda+k+2m+1}$. It outputs

$$\mathsf{pp} = (q, l, d, k, \ell, \kappa, \lambda, \boldsymbol{A}, \boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{k+2m+1}).$$

- $\mathsf{RS.KeyGen}(\mathsf{pp})$: it samples $\vec{\boldsymbol{s}} \leftarrow [-\mu, \mu]^{\ell d}$ and outputs $(\vec{\boldsymbol{s}}, \boldsymbol{A}\vec{\boldsymbol{s}})$.

– $\mathsf{RS.Sign}(\mathsf{pp}, \vec{\boldsymbol{s}}, \mathsf{m}, L)$: if $\boldsymbol{A}\vec{\boldsymbol{s}} \notin L$ then it aborts. Otherwise, it identifies the index $\iota \in [n]$ such that $\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{pk}}_\iota$ where $L = \{\vec{\boldsymbol{pk}}_1, \ldots, \vec{\boldsymbol{pk}}_n\}$. Then, it conducts the prover's algorithm in the protocol in Fig. 7 and obtains challenges by computing the following:

$$\begin{aligned}
&\boldsymbol{c}' = G(1, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \boldsymbol{t}_{k+2m}, \vec{\boldsymbol{w}}, \mathsf{m}),\\
&\vec{\phi}_1 = H_k(2, \boldsymbol{c}', \vec{\boldsymbol{z}}'),\\
&\vec{\phi}_{j+1} = H_1\left(j+2, \vec{\phi}_j, \boldsymbol{t}_{m+k+j}\right) \text{ for } j \in [m-2],\\
&\vec{\phi}_m = H_{m+1}\left(m+1, \vec{\phi}_{m-1}, \boldsymbol{t}_{k+2m-1}\right),\\
&(\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m) = \mathsf{NTT}^{-1}\left(H_{m+1}\left(m+2, \vec{\phi}_m, \boldsymbol{h}\right)\right)\\
&\boldsymbol{c} = G(m+3, \boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m, \boldsymbol{t}_{k+2m+1}, \boldsymbol{\omega}).
\end{aligned} \tag{39}$$

Also, if any rejection step fails, the algorithm starts the non-interactive protocol from Fig. 7 again.
Finally, it outputs the signature:

$$\sigma = \left(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k+2m+1}, \boldsymbol{h}, \boldsymbol{c}, \boldsymbol{c}', \vec{\boldsymbol{z}}', \vec{\boldsymbol{z}}\right).$$

– $\mathsf{RS.Ver}(\mathsf{pp}, \sigma, \mathsf{m}, L)$: it checks the verification equations from Fig. 9 as follows. Firstly, it makes sure that vectors $\vec{\boldsymbol{z}}, \vec{\boldsymbol{z}}'$ are small as in Lines 01-02. Then, Line 03 is equivalent to checking:

$$\boldsymbol{c}' \stackrel{?}{=} G(1, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \boldsymbol{t}_{k+2m}, \boldsymbol{B}_0\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0, \mathsf{m}).$$

Next, it checks that the first d/l coefficients of $\boldsymbol{h}$ are all equal to zero. Finally, we observe that challenges $\vec{\phi}_i$ and $\boldsymbol{\alpha}_j$ can be deterministically computed from σ as in Equation 39. Hence, the algorithm computes $\boldsymbol{\omega}$ as in Line 11 and checks whether:

$$\boldsymbol{c} \stackrel{?}{=} G(m+3, \boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m, \boldsymbol{t}_{k+2m+1}, \boldsymbol{\omega}).$$

If all the equations hold, it outputs 1 and 0 otherwise.

C.3 Security Analysis

Theorem C.4. *A ring signature* RS *defined above provides statistical correctness and is anonymous under the Extended-MLWE*$_{\kappa+k+2m+1,\lambda,\mathfrak{s}}$ *assumption. Moreover, it is unforgeable in the random oracle model if M-LWE*$_{\ell,\mu}$*, Extended-MLWE*$_{\kappa+k+2m+1,\lambda,\mathfrak{s}}$*, M-SIS*$_{\kappa,8d\beta}$ *and M-SIS*$_{k,2\sqrt{\beta'^2+d}}$ *are hard.*

Proof. Correctness and anonymity follow directly from Theorem B.1. Hence, we focus on the unforgeability property.

We apply the standard proof strategy used in [17, 20]. Namely, consider a polynomial time adversary $\mathcal{A}$ that makes at most Q_k, Q_s and Q_h queries to

PKGen, Sign and to the random oracles respectively and wins the unforgeability game with non-negligible probability ε.

Given public parameters, we first pick a random $j \in [Q_k]$ and then, in the t-th query to PKGen, we set $\mathsf{pk}_j := \vec{\boldsymbol{p}}$ for some random $\vec{\boldsymbol{p}} \leftarrow \mathcal{R}_q^k$. Our goal is to run $\mathcal{A}$ using this key for user j and hoping to use exact rewinding as in Section B.3 to obtain forgeries with a ring L which contains j. The extractor will then find small vector $\bar{\boldsymbol{s}}^*$ and a small polynomial $\bar{\boldsymbol{c}}$ such that $\boldsymbol{A}\bar{\boldsymbol{s}}^* = \bar{\boldsymbol{c}}\mathsf{pk}_i$ for some $i \in L$ (or a MSIS solution). Then, with probability $1/Q_k$ we get $i = j$ and thus, we obtain a short solution to the MSIS problem:

$$\left(\boldsymbol{A} \middle| \vec{\boldsymbol{p}}\vec{\boldsymbol{k}}_\iota\right) \begin{pmatrix} \vec{\boldsymbol{s}}^* \\ -\bar{\boldsymbol{c}} \end{pmatrix} = \vec{\boldsymbol{0}}$$

where $\|(\vec{\boldsymbol{s}}^*, -\bar{\boldsymbol{c}})\| \leq 2\sqrt{\beta'^2 + d}$. We now provide more details on the attack using the hybrid argument.

Hybrid 1. Consider the game where the adversary is given to oracles PKGen and $\mathsf{Corrupt}$ as in the real-life experiment, but for signing queries $\mathsf{Sign}(i, \mathsf{m}, L)$ we additionally program the random oracle queries. Namely, when running $\mathsf{RS.Sign}(\mathsf{sk}_i, \mathsf{m}, L)$ we choose $\boldsymbol{c}, \boldsymbol{c}', \vec{\phi}_i, \boldsymbol{\alpha}_i$ as the verifier in Fig. 7 and program

$$\begin{aligned}
&\boldsymbol{c}' := G(1, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \boldsymbol{t}_{k+2m}, \vec{\boldsymbol{w}}, \mathsf{m}),\\
&\vec{\phi}_1 := H_k(2, \boldsymbol{c}', \vec{\boldsymbol{z}}'),\\
&\vec{\phi}_{j+1} := H_1\left(j+2, \vec{\phi}_j, \boldsymbol{t}_{m+k+j}\right) \text{ for } j \in [m-2],\\
&\vec{\phi}_m := H_{m+1}\left(m+1, \vec{\phi}_{m-1}, \boldsymbol{t}_{k+2m-1}\right),\\
&\mathsf{NTT}\left((\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m)\right) := H_{m+1}\left(m+2, \vec{\phi}_m, \boldsymbol{h}\right)\\
&\boldsymbol{c} := G(m+3, \boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m, \boldsymbol{t}_{k+2m+1}, \boldsymbol{\omega}).
\end{aligned}$$

If G, H_1, H_k, H_{m+1} were already queried on the given input, we abort.

Now we analyse the probability of an adversary which tries to distinguish between the real-life experiment and Hybrid 1.

Lemma C.5. *The statistical distance between the views in the unforgeability game and Hybrid 1 is at most*

$$\varepsilon - \left((m+3)Q_s(Q_s+Q_h) - O(1)\right) 2^{-\mathsf{N}}.$$

Proof. We apply the proof technique from [24, Lemma 5.3] and bound the probability of collision when programming the random oracles. Let us concentrate on a single round of the modified signing oracle.

First, we show that each time the signing oracle is called, the probability of generating $\vec{\boldsymbol{y}} \leftarrow D^{d(\kappa+\lambda+k+2m+1)d}$, such that $\vec{\boldsymbol{w}} := \boldsymbol{B}_0\vec{\boldsymbol{y}}$ was a part of any previous queries to G is at most $(Q_s + Q_h)2^{-\kappa d}$. With an overwhelming probability,

the matrix $\boldsymbol{B}_0$ can be written in the Hermite Normal Form as $\boldsymbol{B}_0 = \left[\bar{\boldsymbol{B}}_0 || \boldsymbol{I}\right]$ (see [18, Appendix C]). Hence, by [24, Lemma 4.4] we have that for fixed $\boldsymbol{t} \in \mathcal{R}_q^\kappa$:

$$\Pr[\boldsymbol{B}_0\vec{\boldsymbol{y}} = \vec{\boldsymbol{t}}] \leq \Pr[\vec{\boldsymbol{y}}_1 = \vec{\boldsymbol{t}} - \bar{\boldsymbol{B}}_0\vec{\boldsymbol{y}}_0] \leq \max_{\boldsymbol{t}' \in \mathcal{R}_q^\kappa} \Pr[\vec{\boldsymbol{y}}_1 = \vec{\boldsymbol{t}}' : \vec{\boldsymbol{y}}_1 \leftarrow D_{\mathfrak{s}}^{\kappa d}] \leq 2^{-\kappa d}.$$

Now, assuming that $\boldsymbol{c}' \leftarrow C$ was chosen at random and programmed successfully, the probability that $(2, \boldsymbol{c}', \vec{\boldsymbol{z}}')$ was queried to H_k is at most $(Q_s + Q_h)/2^{\mathsf{N}}$ [11]. Otherwise $\vec{\phi}_1 \leftarrow \mathcal{M}_q^{kl}$ is successfully programmed. Similarly, we argue then with challenges $\vec{\phi}_2, \ldots, \vec{\phi}_m, (\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_m)$ and $\boldsymbol{c}$. Finally, we conclude that the probability of abort in one signing query is at most $(m+3)(Q_s + Q_h)/2^{\mathsf{N}}$.

Since there are Q_s queries to the signing oracle, one can distinguish between the real-life experiment and Hybrid 1 with probability at most $(m+3)Q_s(Q_s + Q_h)/2^{\mathsf{N}}$. □

Hybrid 2. It is the same experiment as in Hybrid 1 but the challenger additionally picks a uniformly random index $j \in [Q_k]$ and when the adversary outputs a forgery (σ, m, L), it also checks whether $\mathsf{pk}_j \in L$. If not, the adversary loses the game.

Clearly, since $j \leftarrow [Q_k]$, we have the following simple observation.

Lemma C.6. *If there exists an adversary $\mathcal{A}$ which wins the Hybrid 1 game with probability ε, then it also wins the Hybrid 2 game with probability ε/Q_k.*

Hybrid 3. It is the same experiment as in Hybrid 2 but if an adversary calls $\mathsf{Corrupt}$ on input j, then it automatically loses the game.

Lemma C.7. *If there exists an adversary $\mathcal{A}$ which wins the Hybrid 2 game with probability ε, then it also wins the Hybrid 3 game with probability ε.*

Proof. This automatically follows from an observation that if $\mathcal{A}$ outputs a valid forgery (σ, m, L) in Hybrid 2 and $\mathsf{pk}_j \in L$ then it could not have called $\mathsf{Corrupt}$ on input j. □

Hybrid 4. It is the same experiment as in Hybrid 3 but now when $\mathsf{Sign}(j, \cdot, \cdot)$ is called, we output the simulated transcripts as in Section B.2.

Lemma C.8. *If there exists a PPT adversary $\mathcal{A}$ which can distinguish Hybrid 3 from Hybrid 4 with probability ε, then there exists a PPT adversary $\mathcal{A}'$ which solves Extended-MLWE$_{\kappa+k+2m+1,\lambda,\mathfrak{s}}$ with probability at least $(\varepsilon - O(2^{\mathsf{N}}))/Q_s$.*

Proof. It is a direct implication of the honest-verifier zero-knowledge property in Theorem B.1 and the fact that $\mathcal{A}$ makes at most Q_s queries to the signing oracle. □

[11] Indeed, note that there are at most $Q_s + Q_h$ queries to H_k of the form $(2, \cdot, \cdot)$.

Hybrid 5. It is the same experiment as in Hybrid 4 but now, when PKGen is called on the j-th query, we sample $\vec{\boldsymbol{s}} \leftarrow [-\mu, \mu]^{\ell d}, \vec{\boldsymbol{p}} \leftarrow \mathcal{R}_q^k$ and set $(\mathsf{sk}_j \mathsf{pk}_j) = (\vec{\boldsymbol{s}}, \vec{\boldsymbol{p}})$ instead.

Lemma C.9. *If there exists a PPT adversary $\mathcal{A}$ which distinguishes between Hybrid 4 and Hybrid 5 with probability ε, then there is a PPT adversary $\mathcal{A}'$ which wins the M-LWE$_{\ell,\mu}$ game with probability at least ε.*

Proof. Given a distinguisher $\mathcal{A}$, we can clearly simulate both hybrids when given the M-LWE challenge $(\boldsymbol{A}, \vec{\boldsymbol{t}})$, i.e. either $\vec{\boldsymbol{t}} = \boldsymbol{A}\vec{\boldsymbol{s}}$ (Hybrid 4) or $\vec{\boldsymbol{t}} \leftarrow \mathcal{R}_q^\kappa$ (Hybrid 5). □

Finally, we focus on Hybrid 5 and run the same strategy as the extractor $\mathcal{E}$ for knowledge soundness in Theorem B.1. Namely, we construct the non-interactive version of $\mathcal{E}'$ (see Section B.3). The first step would be to run the adversary $\mathcal{A}$ until $\mathcal{A}$ tries to create a forged ring signature with uncorrupted users in the ring L and where the signature does not come from the signing oracle (Step 1 of $\mathcal{E}'$). Then, we obtain a successful forged ring signature $\sigma = \left(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k+2m+1}, \boldsymbol{h}, \boldsymbol{c}, \boldsymbol{c}', \vec{\boldsymbol{z}}', \vec{\boldsymbol{z}}\right)$ on message m in a ring L. Next, we rewind the adversary to the point when it queries G on input $(1, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m+k}, \boldsymbol{t}_{k+2m}, \boldsymbol{B}_0\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0, \mathsf{m})$ (where it obtained a challenge $\boldsymbol{c}'$) and we return a fresh polynomial $\boldsymbol{c}^*$. Obviously, if the tuple was not queried, then the success probability of $\mathcal{A}$ is at most $1/2^{\mathsf{N}}$. If $\boldsymbol{c}^* = \boldsymbol{c}'$ or $\mathcal{A}$ does not produce a valid forgery then we restart the procedure. If it fails after N/ε trials then abort (Step 2 of $\mathcal{E}'$). In such a way, we follow the further steps of $\mathcal{E}'$.

In the end, as done by $\mathcal{E}$, we run $\mathcal{E}'$ $O(\mathsf{N})$ times to either obtain a M-SIS$_{\kappa,8d\beta}$ solution or get $\bar{\boldsymbol{s}}^* \in \mathcal{R}_q^\ell$ and $\bar{\boldsymbol{c}} \in \mathcal{R}_q$ such that $\|\bar{\boldsymbol{s}}^*\| \leq 2\beta', \|\bar{\boldsymbol{c}}\|_\infty \leq 2$ and $\boldsymbol{A}\bar{\boldsymbol{s}}^* = \bar{\boldsymbol{c}}\vec{\boldsymbol{p}}\vec{\boldsymbol{k}}_\iota$ for some ι. With probability at most $1/Q_k$, $\vec{\boldsymbol{p}}\vec{\boldsymbol{k}}_\iota = \vec{\boldsymbol{p}}$ and thus $(\bar{\boldsymbol{s}}^*, -\bar{\boldsymbol{c}})$ is a M-SIS$_{k,2\sqrt{\beta'^2+d}}$ solution. Moreover, since $\mathcal{A}$ is a PPT adversary which wins with non-negligible probability ε, the extractor $\mathcal{E}$ runs in expected $\mathsf{poly}(\mathsf{N})$ time.

□

C.4 Various Optimisations

Firstly, we can directly apply the Dilithium compression techniques [14] in our BDLOP commitment scheme as described in [26]. That is, we drop low-order bits of the top part $\vec{\boldsymbol{t}}_0$ and reduce the length of the randomness vector $\vec{\boldsymbol{r}}$ by κ. We refer to [26, Appendix B] for more details.

On the other hand, it seems non-trivial to use similar techniques to compress the actual signature part, e.g. $\vec{\boldsymbol{p}}\vec{\boldsymbol{k}}_i$ or $\vec{\boldsymbol{z}}'$, due to the structure of our set membership proof in Section 3.1. However, we observe that the Bimodal Gaussian technique [13] can be applied on $\vec{\boldsymbol{z}}'$ to further reduce the proof size.

Concretely, the prover starts the protocol as in Fig. 7 with the following two changes. Firstly, it sends a commitment to a vector $\vec{b} = (b, b, \ldots, b) \in \mathcal{M}_q^l$:

$$\boldsymbol{t}_{k+2m+2} = \langle \vec{\boldsymbol{b}}_{k+2m+2}, \vec{\boldsymbol{r}} \rangle + b = \langle \vec{\boldsymbol{b}}_{k+2m+2}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}\left(\vec{b}\right)$$

where $b \leftarrow \{-1, 1\}$. Secondly, $\mathcal{P}$ decomposes the index vector $\vec{v} = \vec{v}_1 \otimes \cdots \otimes \vec{v}_m$ and it sets $\vec{v}_1 := b\vec{v}_1 = \vec{b} \circ \vec{v}_1$. Then, it commits to $\vec{v}_1, \ldots, \vec{v}_m$ as before.

Given a challenge $\boldsymbol{c}' \leftarrow C$, $\mathcal{P}$ computes

$$\vec{\boldsymbol{z}}' := \vec{\boldsymbol{y}}' + b\boldsymbol{c}'\vec{\boldsymbol{s}}$$

and applies the bimodal rejection sampling similarly as in [13]. Then, we want to prove that for P defined in (31):

$$P(\vec{v}_1 \otimes \cdots \otimes \vec{v}_m) = \vec{w}$$

where $\vec{w} = \mathsf{NTT}\,(\vec{\boldsymbol{w}}' - \boldsymbol{A}\vec{\boldsymbol{z}}')$. This can be proven by applying the techniques in Section 3.1. What is new to show is the following: (i) we need to prove $\vec{b}$ has coefficients in $\{-1, 1\}$ and they are all the same, (ii) $\vec{v}_1$ has coefficients in $\{0, b\}$ and it has exactly one b.

Note that proving (i) is equivalent to showing that $(\vec{b} + \vec{1}) \circ (\vec{b} - \vec{1}) = \vec{0}$ and

$$\begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ -1 & 0 & 0 & \cdots & 1 \end{pmatrix} \vec{b} = \vec{0}.$$

Next, (ii) can be done by proving $\vec{v}_1 \circ (\vec{v}_1 - \vec{b}) = \vec{0}$ and $\langle 1, \vec{v} \rangle = \langle \vec{e}_1, \vec{b} \rangle$. We recall that we defined $\vec{e}_1 = (1, 0, \ldots, 0)$. In summary, all the new changes can be described as additional linear and multiplicative relations which can be simply combined with (23) and (30) respectively.

With this technique, we significantly reduce the standard deviations $\mathfrak{s}'$ at the cost of committing to one more vector $\vec{b} \in \mathcal{M}_q^l$. Concretely, for a repetition rate M we would set $\mathfrak{s}'$ which satisfies $M = \exp(T'^2/2\mathfrak{s}'^2)$ where T' is the upper-bound on $\|\boldsymbol{c}'\vec{\boldsymbol{s}}\|$ (see [13] for more details).

C.5 Parameter Selection

Similarly as in [17, 18, 6], we apply the Fiat-Shamir transformation [19] on the interactive protocol in Fig. 7 to obtain a ring signature. We believe the construction as well as the proofs are folklore, thus we provide a concrete instantiation of a ring signature in Appendix C. Nevertheless, we summarise our parameter selection in Fig. 11.

First, we set $(q, d, l) = (\approx 2^{32}, 128, 32)$ so that $q^{-d/l} \approx \mathsf{p}^{d/l} \approx 2^{-128}$. Next, we aim for the repetition rate of our protocol to be 3. Hence, we set M such that $2M^2 = 3$ [12], i.e. $M = \sqrt{3/2}$. To compute the bounds T' and T on $\|\boldsymbol{c}'\vec{\boldsymbol{s}}\|$ and $\|\boldsymbol{c}\vec{\boldsymbol{r}}\|$ respectively, we apply the exact method as in [26, Appendix C]. Namely, we use the observation that

$$\|\boldsymbol{c}'\vec{\boldsymbol{s}}\|^2 \leq d \left\| \sum_{i=1}^{\ell} \sigma_{-1}(\boldsymbol{s}_i)\boldsymbol{s}_i \right\|_1$$

[12] Recall that in Fig. 7 we run two rejection algorithms.

n	m	μ	k	ℓ	κ	λ	proof size
2^5	1	5	4	13	10	10	15.96 KB
2^{10}	2	5	4	13	10	10	17.27 KB
2^{15}	3	5	4	13	10	10	18.73 KB
2^{20}	4	5	4	13	10	10	20.15 KB
2^{25}	5	5	4	13	10	10	21.53 KB

Fig. 11. Ring signature sizes for $n = l^m$ users. For all parameter sets, we choose $(q, d, l) = (\approx 2^{32}, 128, 32)$.

where $\vec{\boldsymbol{s}} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_\ell)$ and σ_{-1} is the Galois automorphism $\sigma_{-1} : X \longmapsto X^{-1}$. Then, we heuristically choose T' so that the expression on the right-hand side is less than T'^2 with probability at least 99%. Similarly we compute T.

Next, we set standard deviations $\mathfrak{s}$ and $\mathfrak{s}'$ such that:

$$M = \exp\left(\frac{T^2}{2\mathfrak{s}^2}\right) = \exp\left(\frac{T'^2}{2\mathfrak{s}'^2}\right).$$

Recall that we use Bimodal Gaussian sampling for $\vec{\boldsymbol{z}}'$.

In order to compute appropriate parameters for the length ℓ and height k of the public matrix $\boldsymbol{A} \in \mathcal{R}_q^{k\times\ell}$ we use the analysis from Dilithium [14]. Namely, the extractor in Theorem B.1 can find an index $\iota \in [n]$, short polynomial $\boldsymbol{c}^*$ and a vector $\vec{\boldsymbol{s}}^* \in \mathcal{R}_q^\ell$ such that $\|\boldsymbol{c}^*\|_\infty \leq 2$ and $\|\vec{\boldsymbol{s}}^*\| \leq 2\beta'$ and $\boldsymbol{A}\vec{\boldsymbol{s}}^* = \boldsymbol{c}^*\vec{\boldsymbol{pk}}_\iota$. If we assume that for uniformly $\boldsymbol{A} \leftarrow \mathcal{R}_q^{k\times\ell}$ and $\vec{\boldsymbol{s}} \leftarrow [-\mu, \mu]^{\ell d}$, $\vec{\boldsymbol{pk}}_\iota = \boldsymbol{A}\vec{\boldsymbol{s}}$ is indistinguishable from a random vector (MLWE assumption), the extractor ends up with a M-SIS solution:

$$\left(\boldsymbol{A}|\vec{\boldsymbol{pk}}_\iota\right)\begin{pmatrix}\vec{\boldsymbol{s}}^* \\ -\boldsymbol{c}^*\end{pmatrix} = \vec{\boldsymbol{0}}$$

where $\|(\vec{\boldsymbol{s}}^*, -\boldsymbol{c}^*)\| \leq 2\sqrt{\beta'^2 + d}$ (see Theorem C.4). Recall that in Fig. 9 we defined $\beta' = \mathfrak{s}'\sqrt{2\ell d}$. Hence, we adjust the parameters k, ℓ so that (i) M-LWE$_{\ell,\mu}$ is hard and (ii) finding a M-SIS solution with Euclidean norm $2\sqrt{\beta'^2 + d}$ is hard. Next, we assume that the Extended-MLWE is almost as hard as M-LWE. Then, to set κ and λ we make sure that Extended-MLWE$_{\kappa+k+2m+2,\lambda,\mathfrak{s}}$ and M-SIS$_{\kappa,8d\beta}$ are hard where $\beta = \mathfrak{s}\sqrt{2(\lambda + \kappa + k + 2m + 2)d}$. We measure the hardness with the root Hermite factor δ and aim for $\delta \approx 1.0042$, similarly as it was done in previous works [8, 18, 1, 15].

We now turn to computing the signature size. As "full-sized" elements of $\mathcal{R}_q$ we have $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k+2m+2}$ and $\boldsymbol{h}$ (it is missing d/l coefficients but this has negligible impact on the sizes). Therefore, we have in total $\kappa + k + 2m + 2$ full elements of $\mathcal{R}_q$ which give us

$$(\kappa + k + 2m + 2)d \log q \text{ bits.}$$

What we have left are vectors of short polynomials $\vec{\boldsymbol{z}}$ and $\vec{\boldsymbol{z}}'$. Since they come from a Gaussian distribution with standard deviation $\mathfrak{s}$ and $\mathfrak{s}'$ respectively, with high probability we can upper-bound their coefficients by $6\mathfrak{s}$ and $6\mathfrak{s}'$ [24]. Thus, they require at most:

$$\ell d \log(12\mathfrak{s}') + (\kappa + \lambda + k + 2m + 2) d \log(12\mathfrak{s}) \text{ bits.}$$

However, by encoding $\vec{\boldsymbol{z}}, \vec{\boldsymbol{z}}'$ using a Huffman code as in [8], we obtain slightly smaller proof size. Finally, the challenges $\boldsymbol{c}, \boldsymbol{c}'$ cost at most $4 \cdot d = 512$ bits.

In our signature size computation we additionally compress the commitment size by not sending low-order bits of $\vec{\boldsymbol{t}}_0$ and reducing the length of the randomness vector $\vec{\boldsymbol{r}}$ identically as in [26, Appendix B] and apply the Bimodal Gaussian technique described above.

D Amortizing Ring Signatures

Set membership sum proof. In order to construct amortized ring signatures, we consider proving the following type of equations. Namely, suppose we have multiple index vectors $\vec{v}_1, \ldots, \vec{v}_r \in \{0,1\}^n$, where each $\vec{v}_j$ corresponds to an index ι_j for $j \in [r]$, and we want to prove knowledge of $\vec{v}_1, \ldots, \vec{v}_r$ along with a vector $\vec{w} \in \mathcal{M}_q^{kl}$ such that

$$\sum_{i=1}^{r} P_1^i \vec{v}_i = \vec{w}.$$

for some public matrices $P_1^1, \ldots, P_1^r \in \mathcal{M}_q^{kl \times n}$.

The naive solution would be to compute each $\vec{w}_j := P_1^j \vec{v}_j$, apply the proof from Section 3.1 and show that $\vec{w}_1 + \ldots + \vec{w}_r = \vec{w}$. However, this implies committing to all $\vec{w}_1, \ldots, \vec{w}_r$ which is rather costly.

We take a recursive approach as in Section 3.1. Concretely, for each $\vec{v}_j$, let us tensor-decompose it into m smaller vectors: $\vec{v}_j = \vec{v}_{j,1} \otimes \cdots \vec{v}_{j,m}$ for $j \in [r]$. Then, we want to prove:

$$\sum_{i=1}^{r} P_1^i (\vec{v}_{i,1} \otimes \cdots \otimes \vec{v}_{i,m}) = \vec{x}_1. \tag{40}$$

Similarly as in (18), Equation 40 implies that for a random challenge $\vec{\gamma}_1 \leftarrow \mathcal{M}_q^{kl}$:

$$\left\langle \sum_{i=1}^{r} P_1^i (\vec{v}_{i,1} \otimes \cdots \otimes \vec{v}_{i,m}) - \vec{x}_j, \vec{\gamma}_j \right\rangle = \sum_{i=1}^{r} \langle \vec{v}_{i,1}, \tilde{P}_2^i \vec{u}_{i,2} \rangle - \langle \vec{x}_1, \vec{\gamma}_1 \rangle \tag{41}$$

is equal to zero where $\vec{u}_{i,2} := \vec{v}_{i,2} \otimes \cdots \otimes \vec{v}_{i,m}$, the matrix P_1^i can be written as

$$P_1^i = \left(P_{1,1}^i \; P_{1,2}^i \cdots \; P_{1,l}^i \right) \in \mathcal{M}_q^{kl \times l^m}$$

and the matrix $\tilde{P}_2^i$ is defined as

$$\tilde{P}_2^i = \begin{pmatrix} \gamma_1^T P_{1,1}^i \\ \vdots \\ \gamma_1^T P_{1,l}^i \end{pmatrix} \in \mathcal{M}_q^{l \times l^{m-1}}. \tag{42}$$

As before, we construct $\vec{x}_{i,2}$ and $\vec{y}_1$ defined by:

$$\vec{x}_{i,2} = \tilde{P}_2^i(\vec{v}_{i,2} \otimes \cdots \otimes \vec{v}_{i,m}) \text{ and } \vec{y}_1 = \sum_{i=1}^{r} \vec{v}_{i,1} \circ \vec{x}_{i,2} - \sum_{j=1}^{k} \vec{x}_{1,j} \circ \vec{\gamma}_{1,j}$$

and commit to $\vec{x}_{i,2}$. After proving that $\vec{x}_{i,2}$ is well-formed, we use Lemma 2.2 and argue that the polynomial $\boldsymbol{y}_1 = \mathsf{NTT}^{-1}(\vec{y}_1)$ has the first d/l coefficients equal to zero.

Now, note that equations for all $\vec{x}_{i,2}$, where $i \in [r]$, can be combined into one:

$$P_2^1(\vec{v}_{1,2} \otimes \cdots \otimes \vec{v}_{1,m}) + \ldots + P_2^r(\vec{v}_{r,2} \otimes \cdots \otimes \vec{v}_{r,m}) = \begin{pmatrix} \vec{x}_{1,2} \\ \vec{x}_{2,2} \\ \vdots \\ \vec{x}_{r,2} \end{pmatrix}$$

where each matrix $P_2^i \in \mathcal{M}_q^{rl \times l^{m-1}}$ is defined as follows:

$$P_2^1 = \begin{pmatrix} \tilde{P}_2^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, P_2^2 = \begin{pmatrix} 0 \\ \tilde{P}_2^2 \\ \vdots \\ 0 \end{pmatrix}, \cdots, P_2^r = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \tilde{P}_2^r \end{pmatrix}.$$

Clearly, this is an equation of type (40) with $m-1$ tensors. Thus, we recursively continue until we end up with a standard linear equation. In the end, proving linear and multiplicative relations along with showing that polynomials $\boldsymbol{y}_j = \mathsf{NTT}^{-1}(\vec{y}_j)$ have first d/l coefficients equal to zero follows identically as in Section 3.1.

Proving knowledge of multiple secret keys. Suppose we want to prove knowledge of $r \geq 1$ secret keys $\vec{\boldsymbol{s}}_1, \ldots, \vec{\boldsymbol{s}}_r \in [-\mu, \mu]^{\ell d}$ such that $\boldsymbol{A}\vec{\boldsymbol{s}}_j = \vec{\boldsymbol{pk}}_{\iota_j}$ for $\iota_1, \ldots, \iota_r \in [n]$.

We sketch out the amortized protocol which stems from [4]. Similarly as in Fig. 7 prover $\mathcal{P}$ starts by sending the BDLOP commitments e.g. to the tensor decompositions of vectors $\vec{v}_1, \ldots, \vec{v}_r$, where each $\vec{v}_j \in \{0,1\}^n$ has 1 in exactly the ι_j-th position, and to $\vec{\boldsymbol{w}}' = \boldsymbol{A}\vec{\boldsymbol{y}}'$ for $\vec{\boldsymbol{y}}' \leftarrow D_{\mathfrak{s}'}^{\ell d}$.

The verifier generates r challenges $\boldsymbol{c}_1', \ldots, \boldsymbol{c}_r' \leftarrow C$ and sends them to $\mathcal{P}$. Then, the prover computes

$$\vec{\boldsymbol{z}}' = \vec{\boldsymbol{y}}' + \boldsymbol{c}_1'\vec{\boldsymbol{s}}_r + \ldots + \boldsymbol{c}_r'\vec{\boldsymbol{s}}_r$$

and applies rejection sampling. After outputting $\vec{\boldsymbol{z}}'$, $\mathcal{P}$ wants to prove the following equation:

$$\sum_{i=1}^{r} P_1^i \vec{v}_i = \vec{w}' - \mathsf{NTT}\left(\boldsymbol{A}\vec{\boldsymbol{z}}'\right).$$

where $\vec{w}' = \mathsf{NTT}\left(\vec{\boldsymbol{w}}'\right)$ and each $P_1^i \in \mathcal{M}_q^{k \times n}$ is defined as

$$P_1^i = \left(\mathsf{NTT}\left(-\boldsymbol{c}_i'\vec{\boldsymbol{pk}}_1\right)\middle|\cdots\middle|\mathsf{NTT}\left(-\boldsymbol{c}_i'\vec{\boldsymbol{pk}}_n\right)\right).$$

Now, note that this equation is of type (40) for $\vec{x}_1 = \vec{w}' - \mathsf{NTT}\left(\boldsymbol{A}\vec{\boldsymbol{z}}'\right)$. Hence, prover $\mathcal{P}$ simply follows the strategy described above.

E Payment System

In this section we use our one-out-of-many proof to construct a payment system with confidential transactions. In such a system there is a (distributed) database or blockchain that records *accounts*. An account is a pair $\mathsf{act} = (\mathsf{pk}, \mathsf{cn})$ of a *public key* pk and a *coin* cn. A coin contains some *amount* amt of money. Now users of the system can transfer money between each other by computing *transactions*. In a transaction a sender spends the coins in several input accounts and puts their amounts into new output accounts with fresh public keys and newly minted coins for one or more recipients. The system only accepts transactions where the sender is the legitimate owner of the spent coins by knowing the secret keys corresponding to the public keys in the accounts. Also each coin is only allowed to be spent once (no double spending) and the sum of the amounts from the spent coins must be equal to the sum of the amounts in the minted coins (balance) so that transactions do not change the total amount of money in the system.

We follow the model and construction from [18], which is a slight modification of the RingCT model [18]. The anonymity property of this model can directly be achieved with a one-out-of-many proof. The amounts in the accounts are hidden in that coins are commitments to amounts, $\mathsf{cn} = \mathsf{Com}(\mathsf{amt}; \mathsf{cnk})$, where we call the commitment randomness cnk a *coin key*. The balance property of a transaction is proven in zero-knowledge. Then, the sender hides his identity by also proving knowledge of his secret keys in zero-knowledge, and the public keys of the output accounts are not the long-term public keys of the recipients but rerandomizations or completely fresh keys that can not be linked to the recipients. Finally, the system hides the graph of transactions by hiding each input account among many more untouched accounts. This is where the one-out-of-many proof is used. Double spending is prevented with the help of so-called serial numbers sn. For each public key one can compute a unique serial number that can only be linked to the public key when knowing the secret key. Now transactions reveal the serial numbers of the spent accounts and include a proof that the serial numbers were correctly computed. The database also records all spent serial numbers and transaction are only accepted when the serial numbers are not yet contained among the spent ones.

In our protocol a public key is an Module-LWE vector $\mathsf{pk} = \vec{\boldsymbol{u}} = \boldsymbol{B}_0\vec{\boldsymbol{s}}$. More precisely, $\vec{\boldsymbol{u}}$ is the top part of a BDLOP commitment to a single polynomial. So $\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa\times(\lambda+\kappa+1)}$ is the public commitment matrix, and the randomness vector $\mathsf{sk} = \vec{\boldsymbol{s}} \in \mathcal{R}_q^{\lambda+\kappa+1}$ is the secret key. We need the additional error polynomial in $\vec{\boldsymbol{s}}$ for the serial number corresponding to $\vec{\boldsymbol{u}}$. In fact, we set sn to be the commitment $\mathsf{sn} = \boldsymbol{n} = \langle\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{s}}\rangle$ to zero.

A coin is again a BDLOP commitment to a single polynomial $\check{\boldsymbol{a}} \in \mathcal{R}_q$ that encodes a vector with coefficients in $\{0, 1, 2, 3\}$ and represents an amount amt in base 4 in the range $[0, 2^{64}-1]$. So, the coin is of the form $\mathsf{cn} = \vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1 \in \mathcal{R}_q^{\kappa+1}$ where

$$\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}}$$
$$\boldsymbol{t}_1 = \langle\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle + \check{\boldsymbol{a}}$$

with the ternary randomness vector $\mathsf{cnk} = \vec{\boldsymbol{r}} \in \mathcal{R}_q^{\kappa+\lambda+1}$.

Now, in a transaction with m input accounts $(\vec{\boldsymbol{u}}_i^{(\mathrm{in})}, \vec{\boldsymbol{t}}_i^{(\mathrm{in})})$, $i = 1, \ldots, m$, and n output accounts $(\vec{\boldsymbol{u}}_i^{(\mathrm{out})}, \vec{\boldsymbol{t}}_i^{(\mathrm{out})})$, $i = 1, \ldots, n$, the sender needs to prove knowledge of all of the coin keys $\vec{\boldsymbol{r}}_i^{(\mathrm{in})}$, $\vec{\boldsymbol{r}}_i^{(\mathrm{out})}$, and the secret keys $\vec{\boldsymbol{s}}_i^{(\mathrm{in})}$ for all of the input public keys. In addition, we will need commitments to auxiliary data with top part $\vec{\boldsymbol{t}}_0' = \boldsymbol{B}_0'\vec{\boldsymbol{r}}'$ in the protocol and have the sender prove knowledge of an opening for this as well. Since the auxiliary commitment contains more than one message polynomial, the commitment matrix $\boldsymbol{B}_0'$ is wider than $\boldsymbol{B}_0'$ and we assume it is an extension of $\boldsymbol{B}_0$. We use standard approximate proofs, but amortize over all of them so the sender only needs to send one masked opening

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \sum_{i=1}^{m} \boldsymbol{\alpha}_i\vec{\boldsymbol{s}}_i^{(\mathrm{in})} + \sum_{i=1}^{m} \boldsymbol{\beta}_i\vec{\boldsymbol{r}}_i^{(\mathrm{in})} + \sum_{i=1}^{n} \boldsymbol{\gamma}_i\vec{\boldsymbol{r}}_i^{(\mathrm{out})} + \boldsymbol{c}\vec{\boldsymbol{r}}'$$

with independent challenge polynomials $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$, and $\boldsymbol{c}$. In this equation the secret keys and coin keys are shorter vectors than the auxiliary commitment randomness vector $\vec{\boldsymbol{r}}'$ and we understand the former as implicitly zero-padded to the same length as $\vec{\boldsymbol{r}}'$.

Then the sender wants to prove the equation

$$\boldsymbol{B}_0'\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}} + \sum_{i=1}^{m} \boldsymbol{\alpha}_i\boldsymbol{u}_i^{(\mathrm{in})} + \sum_{i=1}^{m} \boldsymbol{\beta}_i\vec{\boldsymbol{t}}_{i,0}^{(\mathrm{in})} + \sum_{i=1}^{n} \boldsymbol{\gamma}_i\vec{\boldsymbol{t}}_{i,0}^{(\mathrm{out})} + \boldsymbol{c}\vec{\boldsymbol{t}}_0'.$$

Doing this directly in the clear would reveal the input accounts. So, instead, we use our one-out-of-many proof. In a nutshell, for each of the input accounts there are matrices $\boldsymbol{U}_i^{(\mathrm{in})} \in \mathcal{R}_q^{\kappa\times l^r}$ and $\boldsymbol{T}_i^{(\mathrm{in})} = \begin{pmatrix}\boldsymbol{T}_{i,0}^{(\mathrm{in})}\\ \boldsymbol{T}_{i,1}^{(\mathrm{in})}\end{pmatrix} \in \mathcal{R}_q^{(\kappa+1)\times l^r}$ that contain the actual public keys and coins in one of their columns, together with $l^r - 1$ other public keys / coins. Then, the sender's goal is to prove the equation

$$\boldsymbol{B}_0'\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}} + \sum_{i=1}^{m} \boldsymbol{P}_{1,i}\vec{\boldsymbol{v}}_i + \sum_{i=1}^{n} \boldsymbol{\gamma}_i\vec{\boldsymbol{t}}_{i,0}^{(\mathrm{out})} + \boldsymbol{c}\vec{\boldsymbol{t}}_0'$$

in committed form with matrices $\boldsymbol{P}_{1,i} = \alpha_i \boldsymbol{U}_i^{(\text{in})} + \beta_i \boldsymbol{T}_{i,0}^{(\text{in})}$ and selector vectors $\vec{\boldsymbol{v}}_i$. In particular, he cannot send $\vec{\boldsymbol{w}}$ in the clear but must send a commitment to it, which we choose to be part of the auxiliary commitment $\vec{\boldsymbol{t}}'$. See Section 3 for our Technique of proving this equation with a framework proof and a tensor decomposition of the selector vectors.

For this proof and the other proofs to come we need a non-amortized opening proof for the auxiliary commitment, but we want to avoid the cost of sending a separate masked opening for it. Interestingly, this is not necessary. After setting up the one-out-of-many proof in a first stage of the protocol where the verifier sends the challenges $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$, the prover knows the first part $\vec{\boldsymbol{y}}' = \vec{\boldsymbol{y}} + \sum_{i=1}^{m} \boldsymbol{\alpha}_i \vec{\boldsymbol{s}}_i^{(\text{in})} + \sum_{i=1}^{m} \boldsymbol{\beta}_i \vec{\boldsymbol{r}}_i^{(\text{in})} + \sum_{i=1}^{n} \boldsymbol{\gamma}_i \vec{\boldsymbol{r}}_i^{(\text{out})}$ of the amortized masked opening from Equation (E). So he can use this as the masking vector for the auxiliary opening proof and not sample a new masking vector. In the protocol this means he sends the vector $\vec{\boldsymbol{w}}' = \boldsymbol{B}_0' \vec{\boldsymbol{y}}'$.

Next, for the balance proof, the sender recommits to the amounts from the coins in the auxiliary commitment $\vec{\boldsymbol{t}}'$. So, he sends the commitment polynomials

$$\begin{aligned}
\boldsymbol{t}'_{k+1} &= \langle \vec{\boldsymbol{b}}'_{k+1}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{a}}_1^{(\text{in})} \\
&\vdots \\
\boldsymbol{t}'_{k+m} &= \langle \vec{\boldsymbol{b}}'_{k+m}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{a}}_m^{(\text{in})} \\
\boldsymbol{t}'_{k+m+1} &= \langle \vec{\boldsymbol{b}}'_{k+m+1}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{a}}_1^{(\text{out})} \\
&\vdots \\
\boldsymbol{t}'_{k+m+n} &= \langle \vec{\boldsymbol{b}}'_{k+m+n}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{a}}_n^{(\text{out})}
\end{aligned}$$

where $k \geq 1$ is some offset; concretely $k = 7$ in the protocol. Then, the sender shows that these commitments are to the same amounts as in the accounts. At the same time he reveals the serial numbers $\boldsymbol{n}_i^{(\text{in})} = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{s}}_i^{(\text{in})} \rangle$ of the spent accounts and proves their correctness. More precisely, let $\boldsymbol{t}'_1 = \langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{r}}' \rangle$ be a garbage commitment to zero, independent of all challenges, and notice that

$$\begin{aligned}
&\langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{z}} \rangle - \sum_{i=1}^{m} \boldsymbol{\alpha}_i \boldsymbol{n}_i^{(\text{in})} - \sum_{i=1}^{m} \boldsymbol{\beta}_i \boldsymbol{t}_{i,1}^{(\text{in})} - \sum_{i=1}^{n} \boldsymbol{\gamma}_i \boldsymbol{t}_{i,1}^{(\text{out})} - \boldsymbol{c}\boldsymbol{t}'_1 \\
&= \langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{y}} \rangle - \sum_{i=1}^{m} \boldsymbol{\beta}_i \check{\boldsymbol{a}}_i^{(\text{in})} - \sum_{i=1}^{n} \boldsymbol{\gamma}_i \check{\boldsymbol{a}}_i^{(\text{out})}
\end{aligned}$$

is an amortized masked opening of the amounts that doesn't depend on the challenge $\boldsymbol{c}$ and can be computed by the verifier. On the other hand,

$$\langle \vec{\boldsymbol{b}}'_{k+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}'_{k+i} = \langle \vec{\boldsymbol{b}}'_{k+i}, \vec{\boldsymbol{y}}' \rangle - \boldsymbol{c}\check{\boldsymbol{a}}_i^{(\text{in})}$$

are openings of the auxiliary recommitments with challenge $\boldsymbol{c}$ where the masking polynomials depend on the first-stage challenges. Therefore, by multiplying the

amortized opening by the challenge $\boldsymbol{c}$, the auxiliary openings by the challenges $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$, and subtracting, we obtain an opening of the differences of the amounts with quadratic challenges of the form $\boldsymbol{c}\boldsymbol{\beta}_i$. Hence, when the recommitments are correct, this masked opening has total degree one. Finally, with a second garbage commitment $\boldsymbol{t}'_2 = \langle \vec{\boldsymbol{b}}'_2, \vec{\boldsymbol{r}}'' \rangle + \langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{y}} \rangle$ we can remove the dependency from the challenge $\boldsymbol{c}$. This strategy can then be combined with the one-out-of-many proof to again hide the input accounts. In summary, the sender proves the equation

$$\begin{aligned}
&\boldsymbol{c}\langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{z}} \rangle - \sum_{i=1}^{m} \boldsymbol{\beta}_i \langle \vec{\boldsymbol{b}}'_{k+i}, \vec{\boldsymbol{z}} \rangle - \sum_{i=1}^{n} \boldsymbol{\gamma}_i \langle \vec{\boldsymbol{b}}'_{k+m+i}, \vec{\boldsymbol{z}} \rangle + \langle \vec{\boldsymbol{b}}'_2, \vec{\boldsymbol{z}} \rangle \\
&= \boldsymbol{w}'' + \sum_{i=1}^{m} \boldsymbol{c}\boldsymbol{\alpha}_i \boldsymbol{n}_i^{(\text{in})} + \sum_{i=1}^{m} \boldsymbol{c}\boldsymbol{\beta}_i (\boldsymbol{T}_{i,1}^{(\text{in})} \vec{\boldsymbol{v}}_i - \boldsymbol{t}'_{k+i}) + \sum_{i=1}^{n} \boldsymbol{c}\boldsymbol{\gamma}_i (\boldsymbol{t}_{i,1}^{(\text{out})} - \boldsymbol{t}'_{k+m+i}) \\
&\quad + \boldsymbol{c}\boldsymbol{t}'_2 + \boldsymbol{c}^2 \boldsymbol{t}'_1,
\end{aligned}$$

where

$$\boldsymbol{w}'' = -\sum_{i=1}^{m} \boldsymbol{\beta}_i \langle \vec{\boldsymbol{b}}'_{k+i}, \vec{\boldsymbol{y}}' \rangle - \sum_{i=1}^{n} \boldsymbol{\gamma}_i \langle \vec{\boldsymbol{b}}'_{k+m+i}, \vec{\boldsymbol{y}}' \rangle + \langle \vec{\boldsymbol{b}}'_2, \vec{\boldsymbol{y}}' \rangle.$$

The equation is proved in committed form as before, because sending $\boldsymbol{w}''$ in the clear would reveal the input accounts.

With the recommitments to the amounts we can use the addition proof from [25] to prove the balance property. In fact, in [25] the authors concentrated on proving an additive relation among three integers represented in binary, but the techniques can easily be extended to more integers in the base-4 representation that we are using. The only thing that is not completely straight-forward is that when there are many integers, then the "carry" vector that plays an essential role in the addition proof will not be binary anymore and can not be proven to be small with algebraic techniques. But the same problem appears in the multiplication proof from [25] and we can use the solution there with an "approximate shortness proof" for the carry vector that has negligible additional cost in our case.

The final proof protocol is presented in Figure 12.

E.1 Transaction Proof Size and Parameters

We now concretely calculate the size of the non-interactive version of the proof protocol in Figure 12 via the Fiat-Shamir transform for various ring sizes l^r, numbers of input accounts m, and numbers of output accounts n. First, recall that we have $d = 128$, $l = 32$, and $\log(q) \approx 32$. We set the MLWE and MSIS ranks λ and κ, respectively, such that we get a root Hermite factor of roughly 1.004. This results in $\lambda = \kappa = 10$. A public key pk has size 3.28 Kilobytes, and a coin has size 3.78 KB. The output coins are minted inside the protocol in Figure 12, but their size is not part of the proof size.

Inspection of the protocol in Figure 12 shows that the proof size is given by the sizes of the auxiliary commitment $\vec{\boldsymbol{t}}'$, the masked opening $\vec{\boldsymbol{z}}$, and the polynomial $\boldsymbol{h}_2$. Note that challenges can always be expanded from a seed and don't

contribute to the proof size. Also, in the non-interactive version the polynomial vector $\vec{\boldsymbol{w}}'$ and the garbage polynomial $\boldsymbol{g}_0$ don't have to be transmitted because they can be computed by the verifier and checked with the hash function.

For the auxiliary commitment a common optimization that we also make use of in our proof size calculation is that it is not necessary to transmit the full top part $\vec{\boldsymbol{t}}_0$ and it suffices to send the rounding with respect to some power-of-two, i.e. the quotient modulo 2^D. This is known as the Dilithium public key compression technique [14]. Moreover, the MLWE error for the top part doesn't have to be explicitly opened and no masked copy of it has to be included in $\vec{\boldsymbol{z}}$. Then only a rounding of $\vec{\boldsymbol{w}}'$ with respect to a divisor $2\gamma_2$ of $q-1$ is transmitted and the remainder serves as a masking vector for the MLWE error where another rejection sampling step is necessary. This is the signature compression technique in Dilithium and goes back to the Bai-Galbraith signature scheme [2].

For $\vec{\boldsymbol{z}}$ we use the improved Gaussian rejection sampling from [26]. It leaks one bit of information about the secret vectors, but every coin and public key is only proven twice so there is no problem in this application. Since the implicit opening of the MLWE error due to the Bai-Galbraith compression technique combined with public key compression is bounded in infinity norm by $2\gamma_2$, we also use the infinity norm in assessing the hardness of the MSIS problem, and let the verifier check that each individual polynomial coefficient of $\vec{\boldsymbol{z}}$ is smaller than the worst-case bound 12σ. We aim at having $2\gamma_2$ be roughly equal to 12σ, while making sure that the expected number of Bai-Galbraith rejections stays reasonably small.

For choosing σ we need to bound $\left\|\boldsymbol{\alpha}_i \vec{\boldsymbol{s}}_i^{(\mathrm{in})}\right\|$ and $\|\boldsymbol{c}\vec{\boldsymbol{r}}'\|$. Then we can use the triangular inequality to get a bound on

$$T = \left\| \sum_{i=1}^{m} \boldsymbol{\alpha}_i \vec{\boldsymbol{s}}_i^{(\mathrm{in})} + \sum_{i=1}^{m} \boldsymbol{\beta}_i \vec{\boldsymbol{r}}_i^{(\mathrm{in})} + \sum_{i=1}^{n} \boldsymbol{\gamma}_i \vec{\boldsymbol{r}}_i^{(\mathrm{out})} \right\|$$

and set $\sigma = T$. For this we use the approach from [26, Section 3.2].

With the two optimization from above we find that the auxiliary commitment needs space $\kappa(\log(q)-D)+(7+m+n+\kappa+1+m(2r-1))\log(q)$ bits. Furthermore, the transmitted polynomial vector $\vec{\boldsymbol{z}}$ has length $\lambda+7+m+n+\kappa+1+m(2r-1)$. For computing its bandwidth requirement we compute the entropy of the discrete Gaussian with standard deviation σ and assume that $\vec{\boldsymbol{z}}$ is entropy coded.

In Figure 3 we list the transaction proof sizes for a wide variation of anonymity set sizes and the practial useful situation of 1 or 2 input accounts and 2 output accounts. For example, in the casee of 2 input and 2 output accounts and anonymity set size 1024, our protocol has a proof size of 26.49 Kilobytes. This is almost five times smaller than the proof size from the MatRiCT protocol, although there the anonymity set size is only 100. Our advantage grows towards larger anonymity sets, since our protocol scales quasi-logarithmically instead of polylogarithmically. So, for anonymity set size 2^{25} our protocol is more than 18 times smaller. Figure 4 gives proof sizes for the fixed anonymity set size 1024 but varying number of input accounts, up to 100. Concretely, for $m = 100$ input

accounts we achieve a proof size of 344.26 KB, which is about 3.4 times smaller than the MatRiCT protocol.

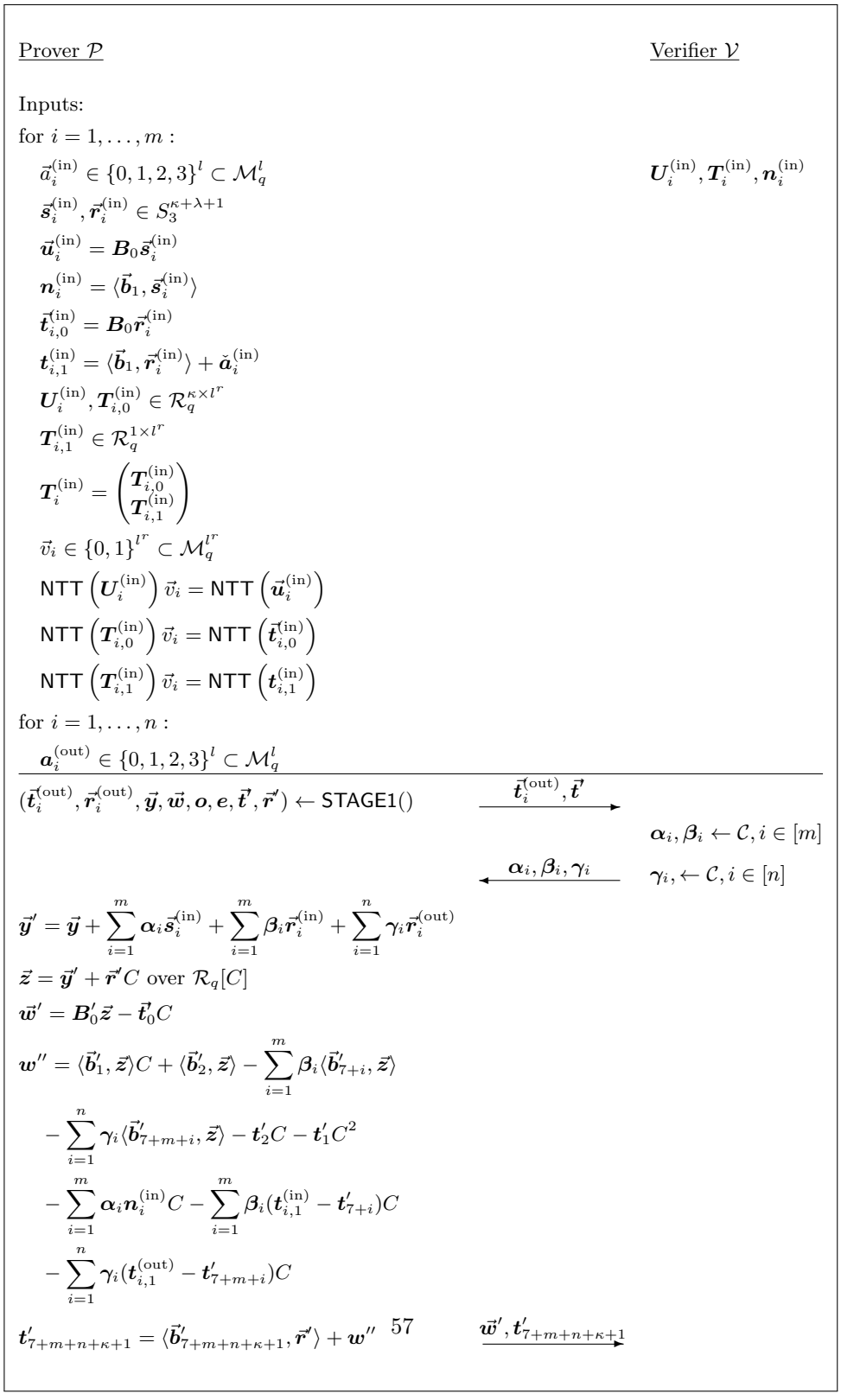

Fig. 12. Our interactive protocol for computing a transaction together with the proof for it. The prover function $\mathsf{STAGE1}()$ is given in Figure 14.

Prover $\mathcal{P}$ | Verifier $\mathcal{V}$

The prover and verifier run the OOOM protocol from Figure 15; As outputs they both get the matrices $\hat{P}_{i,j}$, the prover gets the vectors $\vec{x}_{i,j}$ and the verifier the commitments $\boldsymbol{t}'_{7+m+n+\kappa+1+mr+1}, \ldots, \boldsymbol{t}'_{7+m+n+\kappa+1+mr+m(r-1)}$ to these vectors.

$\boldsymbol{\psi}_i \leftarrow$ | $\boldsymbol{\psi}_i \leftarrow \mathcal{R}_q, i \in [m+n+mr+2]$

$(\boldsymbol{g}_0, \boldsymbol{t}'_5, \boldsymbol{t}'_6, \boldsymbol{t}'_7, \boldsymbol{h}_2) \leftarrow \mathsf{GARBAGE}()$ $\xrightarrow{\boldsymbol{g}_0, \boldsymbol{t}'_5, \boldsymbol{t}'_6, \boldsymbol{t}'_7, \boldsymbol{h}_2}$

$\boldsymbol{c} \leftarrow$ | $\boldsymbol{c} \leftarrow \mathcal{C}$

if $\mathsf{Rej}(\vec{\boldsymbol{z}}(\boldsymbol{c}), \vec{\boldsymbol{z}}(\boldsymbol{c}) - \vec{\boldsymbol{y}}, \sigma) = 1$,

then abort $\xrightarrow{\vec{\boldsymbol{z}}(\boldsymbol{c})}$ $\mathsf{VERIFY}()$

Fig. 13. Transaction Proof, Part II. The prover function $\mathsf{GARBAGE}()$ is given in Figure 16 and computes the garbage polynomials and commitments. The verifier function $\mathsf{VERIFY}()$ for checking the response by the prover is given in Figure 17.

$\underline{\mathsf{STAGE1}(\vec{a}_i^{(\text{in})}, \vec{a}_i^{(\text{out})}, \vec{v}_i)}$

for $i = 1, \ldots, n$:
  $\vec{\boldsymbol{r}}_i^{(\text{out})} \leftarrow S_3^{\kappa+\lambda+1}$
  $\vec{\boldsymbol{t}}_{i,0}^{(\text{out})} = \boldsymbol{B}_0 \vec{\boldsymbol{r}}_i^{(\text{out})}$
  $\boldsymbol{t}_{i,1}^{(\text{out})} = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}_i^{(\text{out})} \rangle + \boldsymbol{a}_i^{(\text{out})}$
$\vec{\boldsymbol{y}} \leftarrow D_\sigma^{\kappa+\lambda+7+m+n+\kappa+1+m(2r-1)}$
$\vec{\boldsymbol{w}} = \boldsymbol{B}'_0 \vec{\boldsymbol{y}}$
$\boldsymbol{o} \leftarrow \{\boldsymbol{p} \in \mathcal{R}_q \mid p_0 = \cdots = p_3 = 0\}$
$\boldsymbol{e} = (\boldsymbol{X} - \boldsymbol{4})^{-1} (\sum_{i=1}^m \boldsymbol{a}_i^{(\text{in})} - \sum_{i=1}^n \boldsymbol{a}_i^{(\text{out})})$
for $i = 1, \ldots, m$:
  $\vec{v}_i = \vec{v}_{i,1} \otimes \cdots \otimes \vec{v}_{i,r} \in \left(\mathcal{M}_q^l\right)^{\otimes r}$
$\vec{\boldsymbol{r}}' \leftarrow S_3^{\kappa+\lambda+7+m+n+\kappa+1+m(2r-1)}$
$\vec{\boldsymbol{t}}'_0 = \boldsymbol{B}'_0 \vec{\boldsymbol{r}}'$
$\boldsymbol{t}'_1 = \langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{r}}' \rangle$
$\boldsymbol{t}'_2 = \langle \vec{\boldsymbol{b}}'_2, \vec{\boldsymbol{r}}' \rangle + \langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{y}} \rangle$
$\boldsymbol{t}'_3 = \langle \vec{\boldsymbol{b}}'_3, \vec{\boldsymbol{r}}' \rangle + \boldsymbol{o}$
$\boldsymbol{t}'_4 = \langle \vec{\boldsymbol{b}}'_4, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{e}}$
for $i = 1, \ldots, m$:
  $\boldsymbol{t}'_{7+i} = \langle \vec{\boldsymbol{b}}'_{7+i}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{a}}_i^{(\text{in})}$
for $i = 1, \ldots, n$:
  $\boldsymbol{t}'_{7+m+i} = \langle \vec{\boldsymbol{b}}'_{7+m+i}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{a}}_i^{(\text{out})}$
for $i = 1, \ldots, \kappa$:
  $\boldsymbol{t}'_{7+m+n+i} = \langle \vec{\boldsymbol{b}}'_{7+m+n+i}, \vec{\boldsymbol{r}}' \rangle + \hat{\boldsymbol{w}}_i$
for $i = 0, \ldots, r-1$, for $j = 1, \ldots, m$:
  $\boldsymbol{t}'_{7+m+n+\kappa+1+mi+j} = \langle \vec{\boldsymbol{b}}'_{7+m+n+\kappa+1+mi+j}, \vec{\boldsymbol{r}}' \rangle + \check{\boldsymbol{v}}_{j,i}$

Fig. 14. Prover Stage I: Mint output coins, sample masking vectors, and compute auxiliary commitment.

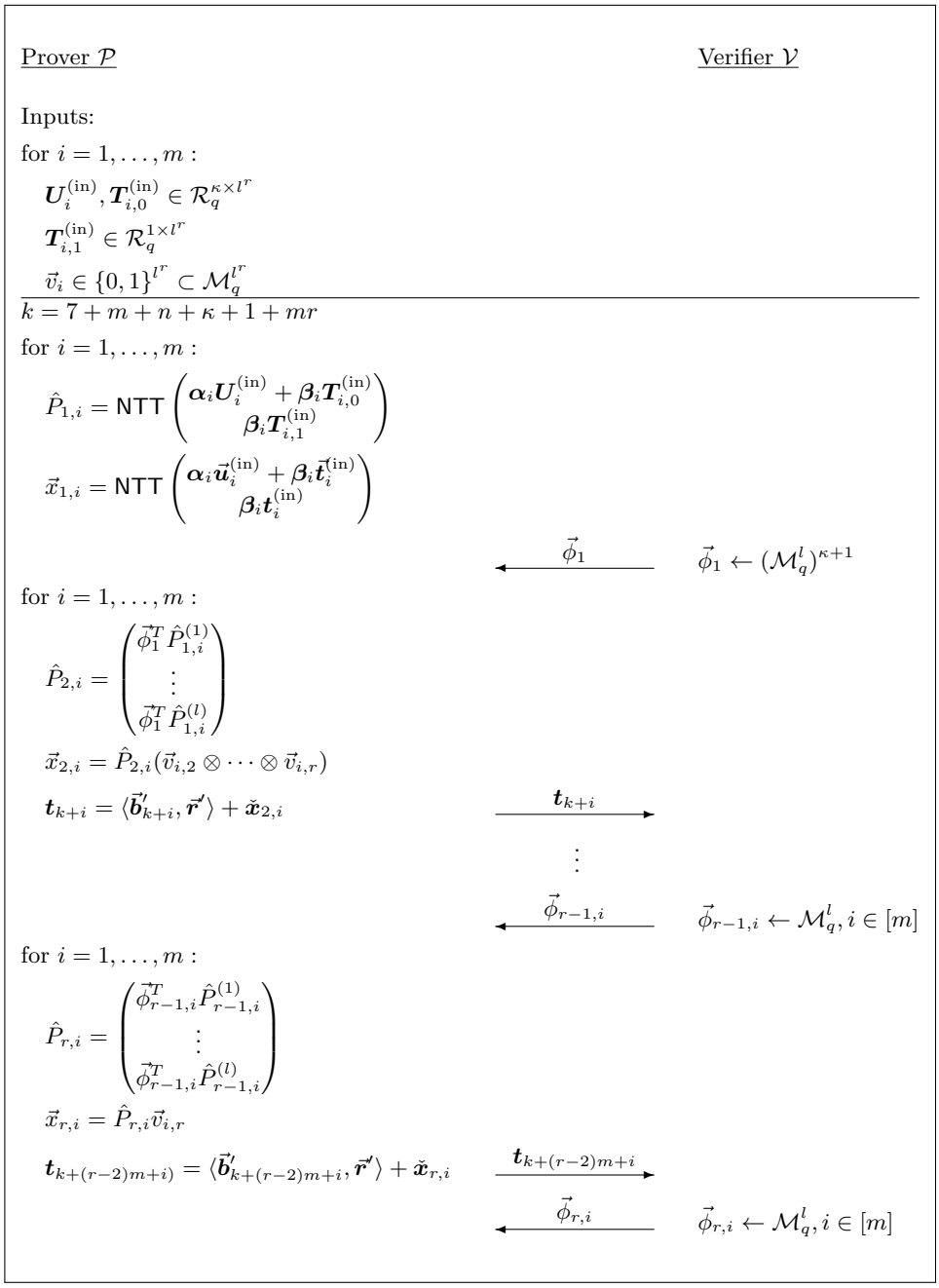

Fig. 15. One-Out-Of-Many Recursion for Transaction Proof.

GARBAGE()

$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}}' + \vec{\boldsymbol{r}}' C$ over $\mathcal{R}_q[C]$

for $i = 1, \ldots, 7 + m + n + \kappa + 1 + m(2r-1)$:

  $\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}'_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{t}'_i C$

$\boldsymbol{g} = \boldsymbol{\psi}_1 \prod_{j=-1}^{1} (\boldsymbol{f}_4 - \boldsymbol{j}C) C$

$k = 7$

for $i = 1, \ldots, m + n$:

  $\boldsymbol{g} \mathrel{+}= \boldsymbol{\psi}_{1+i} \prod_{j=0}^{3} (\boldsymbol{f}_{k+i} - \boldsymbol{j}C)$

$k = 7 + m + n + \kappa + 1$

for $i = 0, \ldots, r-1$, for $j = 1, \ldots, m$:

  $\boldsymbol{g} \mathrel{+}= \boldsymbol{\psi}_{1+m+n+im+j} \prod_{j=0}^{1} (\boldsymbol{f}_{k+im+j} - \boldsymbol{j}C) C^2$

$k = 7 + m + n$

$$\vec{\boldsymbol{f}}' = \begin{pmatrix} \langle \vec{\boldsymbol{b}}'_{k+1}, \vec{\boldsymbol{y}}' \rangle \\ \vdots \\ \langle \vec{\boldsymbol{b}}'_{k+\kappa+1}, \vec{\boldsymbol{y}}' \rangle \end{pmatrix} - \sum_{i+1}^{m} \mathsf{NTT}^{-1} \left(\vec{\boldsymbol{x}}_{1,i} \right) C$$

$\boldsymbol{h} = \sum_{i=1}^{m} \boldsymbol{f}_{k+\kappa+1+i} \boldsymbol{f}_{k+\kappa+1+rm+i} + \sum_{i=1}^{\kappa+1} \check{\boldsymbol{\phi}}_{1,i} \boldsymbol{f}'_i C$

$k = 7 + m + n + \kappa + 1$

for $i = 1, \ldots, r-2$, for $j = 1, \ldots, m$:

  $\boldsymbol{h} \mathrel{+}= \boldsymbol{f}_{k+im+j} \boldsymbol{f}_{k+rm+im+j} + \check{\boldsymbol{\phi}}_{i+1,j} \boldsymbol{f}_{k+rm+(i-1)m+j} C$

for $i = 1, \ldots, m$:

  $\boldsymbol{h} \mathrel{+}= \mathsf{NTT}^{-1} \left(\hat{P}^T_{r,i} \vec{\phi}_{r,i} \right) \boldsymbol{f}_{k+(r-1)m+i} C - \check{\boldsymbol{\phi}}_{r,i} \boldsymbol{f}_{k+rm+(r-2)m+i} C$

$\boldsymbol{h} \mathrel{+}= \check{\boldsymbol{\psi}}'_{m+n+rm+2} (\sum_{i+1}^{m} \boldsymbol{f}_{7+i} - \sum_{i=1}^{n} \boldsymbol{f}_{7+m+i}) C + \boldsymbol{f}_4 C$

$\boldsymbol{h} \mathrel{-}= \boldsymbol{f}_3 C$

$\boldsymbol{g} = \boldsymbol{g}_0 + \boldsymbol{g}_1 C + \boldsymbol{g}_2 C^2 + \boldsymbol{g}_3 C^3$

$\boldsymbol{h} = \boldsymbol{h}_0 + \boldsymbol{h}_1 C + \boldsymbol{h}_2 C^2$

$\boldsymbol{t}'_7 = \langle \vec{\boldsymbol{b}}'_7, \vec{\boldsymbol{r}}' \rangle + \boldsymbol{g}_3 + \boldsymbol{h}_1$

$\boldsymbol{t}'_6 = \langle \vec{\boldsymbol{b}}'_6, \vec{\boldsymbol{r}}' \rangle + \boldsymbol{g}_2 + \boldsymbol{h}_0 + \langle \vec{\boldsymbol{b}}'_7, \vec{\boldsymbol{y}}' \rangle$

$\boldsymbol{t}'_5 = \langle \vec{\boldsymbol{b}}'_5, \vec{\boldsymbol{r}}' \rangle + \boldsymbol{g}_1 + \langle \vec{\boldsymbol{b}}'_6, \vec{\boldsymbol{y}}' \rangle$

$\boldsymbol{g}_0 = \boldsymbol{g}_0 + \langle \vec{\boldsymbol{b}}'_5, \vec{\boldsymbol{y}}' \rangle$

Fig. 16. Garbage Commitments

$\underline{\mathsf{VERIFY}()}$

$\|\vec{\boldsymbol{z}}\| \leq B$

$\boldsymbol{B}'_0\vec{\boldsymbol{z}} \stackrel{?}{=} \vec{\boldsymbol{w}}' + \boldsymbol{c}\vec{\boldsymbol{t}}'_0$

$h_{2,0} \stackrel{?}{=} h_{2,1} \stackrel{?}{=} h_{2,2} \stackrel{?}{=} h_{2,3} \stackrel{?}{=} 0$

for $i = 1, \ldots, 7 + m + n + \kappa + 1 + m(2r-1)$:

  $\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}'_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{t}'_i\boldsymbol{c}$

$\boldsymbol{g} = \boldsymbol{\psi}_1 \prod_{j=-1}^{1}(\boldsymbol{f}_4 - \boldsymbol{j}\boldsymbol{c})\boldsymbol{c}$

$k = 7$

for $i = 1, \ldots, m + n$:

  $\boldsymbol{g} \mathrel{+}= \boldsymbol{\psi}_{1+i} \prod_{j=0}^{3}(\boldsymbol{f}_{k+i} - \boldsymbol{j}\boldsymbol{c})$

$k = 7 + m + n + \kappa + 1$

for $i = 0, \ldots, r-1$, for $j = 1, \ldots, m$:

  $\boldsymbol{g} \mathrel{+}= \boldsymbol{\psi}_{1+m+n+im+j} \prod_{j=0}^{1}(\boldsymbol{f}_{k+im+j} - \boldsymbol{j}\boldsymbol{c})\boldsymbol{c}^2$

$k = 7 + m + n$

$$\vec{\boldsymbol{f}}'_0 = \begin{pmatrix} \boldsymbol{f}_{7+m+n+1} \\ \vdots \\ \boldsymbol{f}_{7+m+n+\kappa} \end{pmatrix} + \boldsymbol{B}'_0\vec{\boldsymbol{z}} - \sum +_{i=1}^{m} \boldsymbol{\beta}_i \vec{\boldsymbol{t}}^{(\mathrm{out})}_{i,0} + \boldsymbol{c}\vec{\boldsymbol{t}}'_0$$

$$\boldsymbol{f}'_1 = \boldsymbol{f}_{7+m+n+\kappa+1} + \langle \vec{\boldsymbol{b}}'_1, \vec{\boldsymbol{z}} \rangle \boldsymbol{c} + \langle \vec{\boldsymbol{b}}'_2, \vec{\boldsymbol{z}} \rangle - \sum_{i=1}^{m} \boldsymbol{\beta}_i \langle \vec{\boldsymbol{b}}'_{7+i}, \vec{\boldsymbol{z}} \rangle - \sum_{i=1}^{n} \boldsymbol{\gamma}_i \langle \vec{\boldsymbol{b}}'_{7+m+i}, \vec{\boldsymbol{z}} \rangle$$
$$-\boldsymbol{t}'_2\boldsymbol{c} - \boldsymbol{t}'_1\boldsymbol{c}^2 - \sum_{i=1}^{m} \boldsymbol{\alpha}_i \boldsymbol{n}^{(\mathrm{in})}_i \boldsymbol{c} + \sum_{i=1}^{m} \boldsymbol{\beta}_i \boldsymbol{t}'_{7+i}\boldsymbol{c} - \sum_{i=1}^{n} \boldsymbol{\gamma}_i (\boldsymbol{t}^{(\mathrm{out})}_{i,1} - \boldsymbol{t}'_{7+m+i})\boldsymbol{c}$$

$$\boldsymbol{h} = \sum_{i=1}^{m} \boldsymbol{f}_{k+\kappa+1+i}\boldsymbol{f}_{k+\kappa+1+rm+i} - \sum_{i=1}^{\kappa} \check{\boldsymbol{\phi}}_{1,i}\vec{\boldsymbol{f}}'_{0,i}\boldsymbol{c} - \check{\boldsymbol{\phi}}_{1,\kappa+1}\boldsymbol{f}'_1\boldsymbol{c}$$

$k = 7 + m + n + \kappa + 1$

for $i = 1, \ldots, r-2$, for $j = 1, \ldots, m$:

  $\boldsymbol{h} \mathrel{+}= \boldsymbol{f}_{k+im+j}\boldsymbol{f}_{k+rm+im+j} + \check{\boldsymbol{\phi}}_{i+1,j}\boldsymbol{f}_{k+rm+(i-1)m+j}\boldsymbol{c}$

for $i = 1, \ldots, m$:

  $\boldsymbol{h} \mathrel{+}= \mathsf{NTT}^{-1}\left(\hat{P}^T_{r,i}\vec{\phi}_{r,i}\right)\boldsymbol{f}_{k+(r-1)m+i}\boldsymbol{c} - \check{\boldsymbol{\phi}}_{r,i}\boldsymbol{f}_{k+rm+(r-2)m+i}\boldsymbol{c}$

$\boldsymbol{h} \mathrel{+}= \check{\boldsymbol{\psi}}'_{m+n+rm+2}(\sum_{i+1}^{m} \boldsymbol{f}_{7+i} - \sum_{i=1}^{n} \boldsymbol{f}_{7+m+i})\boldsymbol{c} + \boldsymbol{f}_4\boldsymbol{c}$

$\boldsymbol{h} \mathrel{-}= \boldsymbol{f}_3\boldsymbol{c}$

$\boldsymbol{g}_0 \stackrel{?}{=} \boldsymbol{g} + (\boldsymbol{h} - \boldsymbol{h}_2\boldsymbol{c}^2)\boldsymbol{c}^2 + \boldsymbol{f}_5 + \boldsymbol{f}_6\boldsymbol{c} + \boldsymbol{f}_7\boldsymbol{c}^2$

Fig. 17. Verification Equations