Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings*

Muhammed F. Esgin[1,2**], Ngoc Khanh Nguyen[3,4], and Gregor Seiler[3,4]

[1] Monash University, Australia
[2] Data61, CSIRO, Australia
[3] IBM Research – Zurich, Switzerland
[4] ETH Zurich, Switzerland

Abstract. We propose a very fast lattice-based zero-knowledge proof system for exactly proving knowledge of a ternary solution $\vec{s} \in \{-1, 0, 1\}^n$ to a linear equation $A\vec{s} = \vec{u}$ over $\mathbb{Z}_q$, which improves upon the protocol by Bootle, Lyubashevsky and Seiler (CRYPTO 2019) by producing proofs that are shorter by a factor of 8.
At the core lies a technique that utilizes the module-homomorphic BDLOP commitment scheme (SCN 2018) over the fully splitting cyclotomic ring $\mathbb{Z}_q[X]/(X^d + 1)$ to prove scalar products with the NTT vector of a secret polynomial.

1 Introduction

Zero-knowledge proofs[5] of knowledge are a central building-block in many cryptographic schemes, especially in privacy-preserving protocols (e.g. group signatures). In these protocols there are often underlying basic public-key primitives, such as encryption and signature schemes, and one has to prove certain statements about the ciphertexts and signatures produced by the underlying primitives. In addition to their usefulness in privacy-preserving protocols, zero-knowledge proof systems have also gained a lot of attention in recent years due to their applications in blockchain protocols.

For post-quantum security the underlying public-key primitives have to be built based on quantum-safe computational hardness assumptions, and lattice-based primitives are a leading choice in this regard. Now, when proving statements related to lattice-based primitives, one always ends up proving knowledge of a short solution to a linear system of equations over some prime field $\mathbb{Z}_q$. More precisely, we want to be able to prove knowledge of a ternary solution $\vec{s} \in \{-1, 0, 1\}^n$ to the equation

$$A\vec{s} = \vec{u}, \tag{1}$$

where the matrix $A \in \mathbb{Z}_q^{m \times n}$ and the right hand side $\vec{u} \in \mathbb{Z}_q^m$ are public. There is no loss of generality in Equation (1) in the sense that it encompasses the situations when the secret vector $\vec{s}$ has coefficients from a larger interval, or when the equation in fact describes linear relations between polynomials in some polynomial ring $\mathcal{R}_q$ of higher rank over $\mathbb{Z}_q$, which arise in important so-called ring-based constructions. In the first situation the secret coefficients can be expanded in base 3 and thereby the equation transformed to the above form. In the second situation the matrix A has a certain structure that describes the linear relations over $\mathcal{R}_q$ with respect to some $\mathbb{Z}_q$-basis of $\mathcal{R}_q$. Then the equation is equivalent to an equation

$$\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{u}} \tag{2}$$

with polynomial matrix $\boldsymbol{A}$, polynomial vector $\vec{\boldsymbol{u}}$ and short polynomial vector $\vec{\boldsymbol{s}}$ with coefficients that are ternary polynomials.

* This research was supported by the SNSF ERC starting transfer grant FELICITY.
** Work done while at IBM Research – Zurich
[5] We use the term "proof" instead of the slightly more precise "argument", and mean computationally sound zero-knowledge proof when we just write zero-knowledge proof.

We call a proof system that exactly proves knowledge of a ternary vector $\vec{s}$ as in Equation (1), and hence does not have any knowledge gap, an *exact* proof system. The goal of this paper is to construct an efficient exact lattice-based proof system.

Currently the most efficient lattice-based protocols that include zero-knowledge proofs utilize so-called *approximate* proof systems which are based on the rejection sampling technique by Lyubashevsky [Lyu09, Lyu12]. Examples are the signature schemes [Lyu12, BG14, DKL+18], the group signature schemes [dPLS18, YAZ+19, EZS+19], and the ring signatures [ESLL19, EZS+19]. Efficient approximate proofs work over polynomial rings $\mathcal{R}_q$ and the prover ends up proving knowledge of a vector $\vec{\boldsymbol{s}}^*$ over $\mathcal{R}_q$ fulfilling only the perturbed equation

$$\boldsymbol{A}\vec{\boldsymbol{s}}^* = \bar{\boldsymbol{c}}\vec{\boldsymbol{u}},$$

where $\bar{\boldsymbol{c}}$ is a short polynomial. Moreover, the coefficients of the polynomials in $\vec{\boldsymbol{s}}^*$ are from a much larger range then the ternary coefficients in the vector $\vec{\boldsymbol{s}}$ that the prover actually knows. The most important reason for the practical efficiency of approximate proofs is that they achieve negligible soundness error with only one repetition.

While approximate proofs are sufficient for many applications, their biggest drawback is that one has to account for the longer vector $\vec{\boldsymbol{s}}^*$ when setting parameters for the underlying schemes so that these schemes are still secure with respect to $\vec{\boldsymbol{s}}^*$. Concretely, suppose that as part of a larger protocol one has to encrypt some message and prove linear relations on the message. Then, when using an approximate proof system, one cannot choose a standard and vetted lattice-based encryption scheme such as Kyber [BDK+18], NTRU, or another scheme in round 2 of the NIST PQC standardization effort. This is problematic for both theoretical and practical reasons. Moreover, if some part of the protocol does not require zero-knowledge proofs, then the efficiency of this part still suffers from the other parts involving zero-knowledge proofs because of the described effect on parameters.

Finally, there are applications for which approximate proof systems are not sufficiently expressive. Natural examples are range proofs for integers and proofs of integer relations, which have applications in blockchain protocols. In these protocols one wants to commit to integers, prove that they lie in certain intervals, and prove additive and multiplicative relations between them. All these problems can be directly solved with an exact proof system that is capable of proving linear equations as above [LLNW18], but approximate proof systems alone are not sufficient for this task. One reason is that one has to commit to the integers in their binary or some other small-base representation and then prove that the committed message really is a binary vector, i.e. that it does not have coefficients from a larger set. This cannot directly be achieved with approximate proofs.

Coming back to exact proof systems, there is a long line of research using Stern's protocol [Ste93] in a lattice setting to exactly prove Equations as in (1) [LLNW17]. But even for the smallest equations, which for example arise when proving a Ring-LWE sample, the proofs produced by this approach have several Megabytes in size and hence are not really practical. The reason behind this is that a single protocol execution has a very large soundness error of 2/3, and thus many protocol repetitions (in the order of hundreds) are required to reach a negligible soundness error.

In [BLS19, YAZ+19], the authors use the BDLOP commitment scheme [BDL+18] to construct an exact proof system and achieve proof sizes of several hundred Kilobytes for proving Ring-LWE samples. The results in the present paper can be seen as an extension of the results of [BLS19].

Now, for post-quantum security, even when relying on underlying lattice-based primitives, it is of course not necessary to also built the zero-knowledge proof system with lattice techniques, as long as the proof system does not introduce computational assumptions that are known to be insecure against quantum computers. In fact, there are PCP-type proof systems using Kilian's framework [Kil92], such as Ligero [AHIV17] or Aurora [BCR+19], that are capable of exactly proving linear equations as above, and that are secure assuming only the security of a hash function. These proof systems are even succinct and produce proofs with sizes that are sublinear or even logarithmic in the size of the witness $\vec{s}$, but they have a base cost in the order of 100 KB for Ligero and around 70 KB for Aurora.

The proof system that we present in this paper scales linearly in the witness size but produces proofs of only 47 KB for proving a Ring-LWE sample. So there is a regime of interesting statements where linear-sized proof systems can beat the best logarithmic PCP-type systems in terms of proof size.

For larger equations where we cannot quite achieve proof sizes as small as the PCP-type systems, lattice-based systems still have one big advantage. Namely, they are very computationally lightweight. Implementations of lattice-based cryptography are generally known to be very fast. For example, the fastest lattice-based CCA-secure KEMs have encapsulation and decapsulation times in the order of a few microseconds on standard laptop processors [LS19] and are thus about one order of magnitude faster than a single elliptic curve scalar multiplication. The reason for this very high speed is essentially twofold. Firstly, there is usually no multi-precision arithmetic needed since efficient lattice-based schemes use finite field moduli q that are below 2^{32}. And secondly, the required arithmetic has a high degree of data-level parallelism that is favourable to modern CPUs, which is especially true for schemes whose arithmetic natively supports the Number Theoretic Transform (NTT). The protocols that we present in this paper are no exception to this; they use single-precision 32-bit moduli, are NTT-friendly, and don't introduce any computational tasks that are not also present in standard lattice-based basic encryption or signature schemes. We demonstrate the fast speed of our protocols with an optimized implementation for Intel CPUs that achieves prover and verifier running times of 3.52 and 0.4 milliseconds, respectively, for the case of proving a ternary solution to a linear equation of dimensions 1024×2048.

Contrary to this, existing studies of using logarithmic PCP-type proof systems for proving the linear equations (1) that arise in lattice-based privacy-preserving protocols show that one ends up with prover runtimes in the order of several tens of seconds even for the smallest instances and on very powerful processors [BCOS20]. This also seems to be the case for the logarithmic but not quantum-safe Bulletproofs proof system [dPLS19]. For example, in [BCOS20] the authors construct a lattice-based group signature scheme using Aurora as the proof system. They found that proving a Ring-LWE sample takes 40 seconds on a laptop computer. Even worse, they could not successfully run the full signing algorithm, due to very large memory requirements, even with the help of Google Cloud large-memory compute instances. This is especially problematic since for privacy-preserving protocols to be used in practice, the prover would often need to be run on constraint devices, possibly down to smart cards or TPM chips. We summarize the above comparison in Table 1.

Table 1. Proof length comparison for proving knowledge of a Ring-LWE sample in dimension 1024 modulo a prime $q \approx 2^{32}$. Here the dimensions of the corresponding Equation as in (1) are $m = 1024$ and $n = 2048$. The sizes for the Stern-type proof is taken from [BLS19]. The sizes for Ligero and the scheme from [Beu20] are taken from [Beu20] and are for the parameter $m = 512$.

Stern-type proofs	3522 KB
[BLS19]	384 KB
[Beu20]	233 KB
Ligero [AHIV17]	157 KB
Aurora [BCR+19, BCOS20]	72 KB
Our work	**47 KB**

1.1 Our Approach

The proof system in the present work extends the system from [BLS19]. On a high level, the approach entails committing to a polynomial $\check{\boldsymbol{s}} \in \mathcal{R}_q$ whose NTT basis representation is equal to the secret vector $\vec{s}$, $\mathsf{NTT}(\check{\boldsymbol{s}}) = \vec{s}$. Then, using a product proof protocol that allows to prove multiplicative relations between committed polynomials, the prover shows that $\check{\boldsymbol{s}}(\mathbf{1}-\check{\boldsymbol{s}})(\mathbf{1}+\check{\boldsymbol{s}}) = \mathbf{0}$. This implies that $\vec{s}$ has ternary coefficients since the polynomial product is component-wise in the NTT basis,

$$\mathsf{NTT}(\check{\boldsymbol{s}}(\mathbf{1}-\check{\boldsymbol{s}})(\mathbf{1}+\check{\boldsymbol{s}})) = \vec{s} \circ (\vec{1}-\vec{s}) \circ (\vec{1}+\vec{s}),$$

where $\vec{1} = (1, \ldots, 1)^T$ is the all-ones vector and $\circ$ denotes the component-wise product. What remains is the linear part where the prover needs to show that $\vec{s}$ is a solution to Equation (1). The linear part was the biggest obstacle to smaller proof sizes in [BLS19]. The reason is that while the BDLOP commitment scheme makes it very easy to prove linear relations over the polynomial ring $\mathcal{R}_q$, one needs to be able to prove linear relations between the NTT coefficients corresponding to the committed polynomials when using the above encoding of the secret vector.

Essentially there are two ways to commit to vectors using the BDLOP commitment scheme. Either one commits to polynomials whose coefficient vectors are equal to the secret vectors, or one commits to polynomials whose NTT vectors are the secret vectors. The first way makes it easy to prove structured linear equations as in (2) by directly using the homomorphic property of the commitment scheme. The second way allows for efficient range proofs with the help of an efficient product proof. But we need to prove a linear equation and conduct a range proof at the same time.

In [BLS19] the problem is side-stepped by reusing a masked opening $\boldsymbol{z}$ of the committed polynomial $\check{\boldsymbol{s}}$ with scalar challenge $c \in \mathbb{Z}_q$,

$$\boldsymbol{z} = \boldsymbol{y} + c\check{\boldsymbol{s}},$$

which is sent as part of the product proof. The verifier can apply the NTT to get a masked opening of the secret vector $\vec{s}$, $\mathsf{NTT}(\boldsymbol{z}) = \hat{y} + c\vec{s}$, and then check that $A\mathsf{NTT}(\boldsymbol{z}) = \vec{w} + c\vec{u}$, where $\vec{w} = A\hat{y}$ is sent by the prover before seeing the challenge c. This approach crucially requires that the challenge c is an integer from $\mathbb{Z}_q$ and not a proper polynomial. Otherwise the masked opening $\mathsf{NTT}(\boldsymbol{z})$ of $\vec{s}$ would include a component-wise product that is incompatible with the linear equation. But with only an integer challenge c the protocol is restricted to soundness error $1/q$ and hence needs to be repeated multiple times.

The main new technique in this paper is a more efficient method to directly prove linear relations among NTT coefficients of the message polynomials in the BDLOP commitment scheme. Then the product proof can make use of proper polynomial challenges and our proof system profits from further improvements in the product proof presented recently in [ALS20].

We now go a bit more into detail and describe our method for the linear proof. For concreteness, let us define $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$, where d is a power of two and $X^d + 1$ splits fully into linear factors over $\mathbb{Z}_q$. Then the i-th NTT coefficient of a polynomial $\check{\boldsymbol{s}} \in \mathcal{R}_q$ is equal to the evaluation of $\check{\boldsymbol{s}}$ at the i-th primitive $2d$-th root of unity r_i. Therefore, if $\vec{s} = \mathsf{NTT}(\check{\boldsymbol{s}})$ and $\vec{\gamma} \xleftarrow{\$} \mathbb{Z}_q^d$ is a random vector, we have

$$\begin{aligned}
\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle &= \langle A\vec{s}, \vec{\gamma} \rangle - \langle \vec{u}, \vec{\gamma} \rangle = \langle \vec{s}, A^T\vec{\gamma} \rangle - \langle \vec{u}, \vec{\gamma} \rangle \\
&= \sum_{i=0}^{d-1} \check{\boldsymbol{s}}(r_i) \left(\mathsf{NTT}^{-1}(A^T\vec{\gamma})\right)(r_i) - \langle \vec{u}, \vec{\gamma} \rangle \\
&= \frac{1}{d} \sum_{i=0}^{d-1} \boldsymbol{f}(r_i) = f_0,
\end{aligned}$$

where $\boldsymbol{f} = \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\check{\boldsymbol{s}} - \langle \vec{u}, \vec{\gamma} \rangle \in \mathcal{R}_q$ and $f_0 \in \mathbb{Z}_q$ is the constant coefficient of $\boldsymbol{f}$. The last equality follows from Lemma 2.1. The idea is then to prove that f_0, the constant coefficient of $\boldsymbol{f}$, is zero. This proves that $\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle = 0$. For a uniformly random $\vec{\gamma} \in \mathbb{Z}_q^d$, the probability that $\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle = 0$ when $A\vec{s} \neq \vec{u}$ is $1/q$. Therefore, allowing the verifier to choose a random $\vec{\gamma} \in \mathbb{Z}_q^d$ as a challenge, proving $f_0 = 0$ proves that $A\vec{s} = \vec{u}$ with a soundness error $1/q$.

To prove that $\boldsymbol{f}$ has vanishing constant coefficient, the prover initially commits to $\check{\boldsymbol{s}}$ and a polynomial $\boldsymbol{g}$ with vanishing constant coefficient. The polynomial $\boldsymbol{g}$ will be used to mask $\boldsymbol{f}$. Upon receiving a challenge $\vec{\gamma} \in \mathbb{Z}_q^d$, the prover computes $\boldsymbol{f}$ and sets $\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g}$. Using the given information, we show that the verifier can compute a commitment to $\boldsymbol{f}$ (without requiring it to be sent by the prover). This allows to prove that $\boldsymbol{h}$ is of the correct form and the verifier can simply observe that $\boldsymbol{h}$ has a zero constant coefficient.

The above proof system has a soundness error of $1/q$, which is not negligibly small for typical choices of q. We show in Section 3.2 how to amplify the soundness of this protocol at a low cost using Galois automorphisms. Informally, consider k uniformly random vectors $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1}$ such that $1/q^k$ is negligible.

Similarly as before, we can write

$$\boldsymbol{f}_i := d\mathsf{NTT}^{-1}(A^T\vec{\gamma}_i)\check{\boldsymbol{s}} - \langle\vec{u},\vec{\gamma}_i\rangle$$

and thus the constant coefficient of $\boldsymbol{f}_i$ is $\langle A\vec{s}-\vec{u},\vec{\gamma}_i\rangle$. For each $i=0,\ldots,k-1$, we will define maps $\mathsf{L}_i\colon \mathcal{R}_q \to \mathcal{R}_q$ which satisfies the following property. Denote $\boldsymbol{p} = \mathsf{L}_i(\boldsymbol{f}_i)$ and $(p_0,\ldots,p_{d-1})$ to be the coefficient vector of $\boldsymbol{p}$. Then, $p_0 = \ldots = p_{i-1} = p_{i+1} = \ldots = p_{k-1} = 0$ and $p_i = \langle A\vec{s}-\vec{u},\vec{\gamma}_i\rangle$. We can observe that if $A\vec{s}=\vec{u}$ then $\boldsymbol{f}$ defined now as

$$\boldsymbol{f} = \mathsf{L}_0(\boldsymbol{f}_0) + \ldots + \mathsf{L}_{k-1}(\boldsymbol{f}_{k-1})$$

has the first k coefficients equal to 0. Therefore, we can construct a protocol for proving this similarly as above. On the other hand, when $A\vec{s}\neq\vec{u}$ then the probability that all the first k coefficients of $\boldsymbol{f}$ are equal to zero is $1/q^k$. The advantage of this approach over the standard way of having k-parallel repetitions is that the size of the commitment part of the non-interactive proof remains the same as that of a single protocol run. Therefore, the overall cost is significantly reduced.

We believe that this protocol can be useful in other settings, where one wants to switch from the the coefficient basis to the NTT basis.

Another obstacle against practical efficiency (as encountered in [BLS19, YAZ+19]) is that a proof of such a non-linear relation as in (1) requires communication of "garbage terms". These garbage terms end up being a substantial cost in the proofs in [BLS19, YAZ+19]. In [ALS20], a better product proof is presented that reduces the cost of the garbage terms, also using Galois automorphisms.

Applications. Having an efficient proof system to prove knowledge of $\vec{s}\in\{-1,0,1\}^n$ satisfying (1) paves the way for various efficient zero-knowledge proofs that can be used in many applications. In order to show the effectiveness of our new techniques, we present two example applications with concrete parameters in Appendix B. The first one is to prove knowledge of secrets in LWE samples. This is an important proof system to be used, for example, with fully homomorphic encryption (FHE) schemes. The goal here is to prove that $\vec{u}$ is a proper LWE sample such that $\vec{u} = A'\vec{s}' + \vec{e} \bmod q$ for $\vec{s}',\vec{e}\in\{-1,0,1\}^k$, which is equivalent to proving $\vec{u} = (A' \,\|\, I_k)\cdot\vec{s} \mod q$ for $\vec{s} = (\vec{s}',\vec{e}) \in \{-1,0,1\}^{2k}$. As shown in Table 1, our proof system achieves an improvement of 8× in terms of proof length over the state-of-the-art result by Bootle, Lyubashevsky and Seiler [BLS19], and is dramatically shorter than the Stern-based proofs.

Our other example application is a proof of plaintext knowledge. In this case, the goal is to create a ciphertext and a zero-knowledge proof to prove that the ciphertext is a proper encryption of a message known by the prover. Proofs of plaintext knowledge have applications, for example, in the settings of verifiable encryption, verifiable secret sharing and group signatures.

Being a very core proof system, there are many other applications beyond the two examples above, where our main protocol and our new techniques can be useful. For example, one can apply our unstructured linear proof to prove that one vector is a NTT representation of a polynomial (written as a vector of coefficients). Indeed, the matrix A in (1) simply becomes a Vandermonde matrix. Furthermore, one can see [YAZ+19] for various applications that all build on a similar core proof system.

2 Preliminaries

2.1 Notation

Table 2 summarizes the notation and parameters that will appear in this paper.

Let q be an odd prime, and $\mathbb{Z}_q$ denote the ring of integers modulo q. We write $[a,b[= \{a,a+1,\ldots,b-1\} \subset \mathbb{Z}$ for the half-open interval of integers between a and b. For $r\in\mathbb{Z}$, we define $r \bmod q$ to be the unique element in the interval $[-\frac{q-1}{2},\frac{q-1}{2}]$ that is congruent to r modulo q. We write $\vec{v}\in\mathbb{Z}_q^m$ to denote vectors over $\mathbb{Z}_q$ and matrices over $\mathbb{Z}_q$ will be written as regular capital letters M. By default, all vectors are column vectors. We write $\vec{v}\,\|\,\vec{w}$ for the concatenation of $\vec{v}$ and $\vec{w}$ (which is still a column vector).

Let d be a power of two and denote $\mathcal{R}$ and $\mathcal{R}_q$ to be the rings $\mathbb{Z}[X]/(X^d+1)$ and $\mathbb{Z}_q[X]/(X^d+1)$, respectively. Bold lower-case letters $\boldsymbol{p}$ denote elements in $\mathcal{R}$ or $\mathcal{R}_q$ and bold lower-case letters with arrows

Parameter	Explanation
d	Degree of the polynomial $X^d + 1$, power of two
q	Rational prime modulus
$\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$	The field over which the linear system is defined
$m \in \mathbb{Z}$	The number of rows in the linear system
$n \in \mathbb{Z}$	The number of columns in the linear system
$\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$	The ring of integers in the $2d$-th cyclotomic number field
$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$	The ring of integers $\mathcal{R}$ modulo q
$k \in \mathbb{Z}$	Repetition rate
$\sigma = \sigma_{2d/k+1}$	Automorphism in $\mathsf{Aut}(\mathcal{R}_q)$ of order k
$\mathcal{C} \subset \mathcal{R}$	Challenge set
C	Probability distribution over $\mathcal{C}$ for challenges
T	Bound for honest prover's $\boldsymbol{c}\vec{\boldsymbol{r}}$ in the infinity norm
δ_1	width of the uniform distribution for sampling $\vec{\boldsymbol{y}}$
λ	M-LWE dimension
κ	M-SIS dimension
χ	Error distribution on $\mathcal{R}$ in the M-LWE problem

Table 2. Overview of Parameters and Notation

$\vec{\boldsymbol{b}}$ represent column vectors with coefficients in $\mathcal{R}$ or $\mathcal{R}_q$. We also use bold upper-case letters for matrices $\boldsymbol{B}$ over $\mathcal{R}$ or $\mathcal{R}_q$. For a polynomial denoted as a bold letter, we write its i-th coefficient as the corresponding regular font letter with subscript i, e.g. $f_0 \in \mathbb{Z}_q$ is the constant coefficient of $\boldsymbol{f} \in \mathcal{R}_q$.

We write $x \xleftarrow{\$} S$ when $x \in S$ is sampled uniformly at random from the finite set S and similarly $x \xleftarrow{\$} D$ when x is sampled according to the distribution D.

Norms and Sizes. For an element $w \in \mathbb{Z}_q$, we write $|w|$ to mean $|w \bmod q|$. Define the ℓ_∞ and ℓ_2 norms for $\boldsymbol{w} \in \mathcal{R}_q$ as follows,

$$\|\boldsymbol{w}\|_\infty = \max_i |w_i| \quad \text{and} \quad \|\boldsymbol{w}\|_2 = \sqrt{|w_0|^2 + \ldots + |w_{d-1}|^2}.$$

Similarly, for $\vec{\boldsymbol{w}} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k) \in \mathcal{R}^k$, we define

$$\|\vec{\boldsymbol{w}}\|_\infty = \max_i \|\boldsymbol{w}_i\|_\infty \quad \text{and} \quad \|\vec{\boldsymbol{w}}\|_2 = \sqrt{\|\boldsymbol{w}_1\|_2^2 + \ldots + \|\boldsymbol{w}_k\|_2^2}.$$

2.2 Prime Splitting and Galois Automorphisms

Let l be a power of two dividing d and suppose $q - 1 \equiv 2l \pmod{4l}$. Then, $\mathbb{Z}_q$ contains primitive $2l$-th roots of unity but no elements with order a higher power of two, and the polynomial $X^d + 1$ factors into l irreducible binomials $X^{d/l} - \zeta$ modulo q where ζ runs over the $2l$-th roots of unity in $\mathbb{Z}_q$ [LS18, Theorem 2.3].

The ring $\mathcal{R}_q$ has a group of automorphisms $\mathsf{Aut}(\mathcal{R}_q)$ that is isomorphic to $\mathbb{Z}_{2d}^\times$,

$$i \mapsto \sigma_i \colon \mathbb{Z}_{2d}^\times \to \mathsf{Aut}(\mathcal{R}_q),$$

where σ_i is defined by $\sigma_i(X) = X^i$. In fact, these automorphisms come from the Galois automorphisms of the $2d$-th cyclotomic number field which factor through $\mathcal{R}_q$.

The group $\mathsf{Aut}(\mathcal{R}_q)$ acts transitively on the prime ideals $(X^{d/l} - \zeta)$ in $\mathcal{R}_q$ and every σ_i factors through field isomorphisms

$$\mathcal{R}_q/(X^{d/l} - \zeta) \to \mathcal{R}_q/(\sigma^i(X^{d/l} - \zeta)).$$

Concretely, for $i \in \mathbb{Z}_{2d}^\times$ it holds that

$$\sigma_i(X^{d/l} - \zeta) = (X^{id/l} - \zeta) = (X^{d/l} - \zeta^{i^{-1}})$$

To see this, observe that the roots of $X^{d/l} - \zeta^{i^{-1}}$ (in an appropriate extension field of $\mathbb{Z}_q$) are also roots of $X^{id/l} - \zeta$. Then, for $f \in \mathcal{R}_q$,

$$\sigma_i\left(f \bmod (X^{d/l} - \zeta)\right) = \sigma_i(f) \bmod (X^{d/l} - \zeta^{i^{-1}}).$$

The cyclic subgroup $\langle 2l+1 \rangle < \mathbb{Z}_{2d}^\times$ has order d/l [LS18, Lemma 2.4] and stabilizes every prime ideal $(X^{d/l} - \zeta)$ since ζ has order $2l$. The quotient group $\mathbb{Z}_{2d}^\times / \langle 2l+1 \rangle$ has order l and hence acts simply transitively on the l prime ideals. Therefore, we can index the prime ideals by $i \in \mathbb{Z}_{2d}^\times / \langle 2l+1 \rangle$ and write

$$\left(X^d + 1\right) = \prod_{i \in \mathbb{Z}_{2d}^\times / \langle 2l+1 \rangle} \left(X^{d/l} - \zeta^i\right)$$

Now, the product of the $k \mid l$ prime ideals $(X^{d/l} - \zeta^i)$ where i runs over $\langle 2l/k+1 \rangle / \langle 2l+1 \rangle$ is given by the ideal $(X^{kd/l} - \zeta^k)$. So, we can partition the l prime ideals into l/k groups of k ideals each, and write

$$\left(X^d + 1\right) = \prod_{j \in \mathbb{Z}_{2d}^\times / \langle 2l/k+1 \rangle} \left(X^{kd/l} - \zeta^{jk}\right) = \prod_{j \in \mathbb{Z}_{2d}^\times / \langle 2l/k+1 \rangle} \prod_{i \in \langle 2l/k+1 \rangle / \langle 2l+1 \rangle} \left(X^{\frac{d}{l}} - \zeta^{ij}\right).$$

Another way to write this, which we will use in our protocols, is to note that $\mathbb{Z}_{2d}^\times / \langle 2l/k+1 \rangle \cong \mathbb{Z}_{2l/k}^\times$ and the powers $(2l/k+1)^i$ for $i = 0, \ldots, k-1$ form a complete set of representatives for $\langle 2l/k+1 \rangle / \langle 2l+1 \rangle$. So, if $\sigma = \sigma_{2l/k+1} \in \mathsf{Aut}(\mathcal{R}_q)$, then

$$\left(X^d + 1\right) = \prod_{j \in \mathbb{Z}_{2l/k}^\times} \prod_{i=0}^{k-1} \sigma^i \left(X^{\frac{d}{l}} - \zeta^j\right),$$

and the prime ideals are indexed by $(i, j) \in I = \{0, \ldots, k-1\} \times \mathbb{Z}_{2l/k}^\times$.

The fully splitting case. In this paper our main attention lies on the setup where $q \equiv 1 \pmod{2d}$ and hence q splits completely. In this case there is a primitive $2d$-th root of unity $\zeta \in \mathbb{Z}_q$ and

$$(X^d + 1) = \prod_{i \in \mathbb{Z}_{2d}^\times} (X - \zeta^i).$$

Then, for a divisor k of d and $\sigma = \sigma_{2d/k+1}$ of order k, we have the partitioning

$$(X^d + 1) = \prod_{j \in \mathbb{Z}_{2d}^\times / \langle 2d/k+1 \rangle} \prod_{i \in \langle 2d/k+1 \rangle} (X - \zeta^{ij}) = \prod_{j \in \mathbb{Z}_{2d/k}^\times} \prod_{i=0}^{k-1} \sigma^i (X - \zeta^j)$$

2.3 The Number Theoretic Transform

The Number Theoretic Transform (NTT) of a polynomial $\boldsymbol{f} \in \mathcal{R}_q$ is defined by

$$\mathsf{NTT}(\boldsymbol{f}) = (\hat{\boldsymbol{f}}_i)_{i \in \mathbb{Z}_{2l}^\times} \in \prod_{i \in \mathbb{Z}_{2l}^\times} \mathbb{Z}_q[X]/(X^{d/l} - \zeta^i) \cong (\mathbb{F}_{q^{d/l}})^l$$

where $\hat{\boldsymbol{f}}_i = \boldsymbol{f} \bmod (X^{d/l} - \zeta^i)$. We write $\mathsf{NTT}^{-1}(\hat{\boldsymbol{f}}) = \boldsymbol{f}$ for the inverse map, which exists due to the Chinese remainder theorem. Note that for $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{R}_q$, $\mathsf{NTT}(\boldsymbol{fg}) = \mathsf{NTT}(\boldsymbol{f}) \circ \mathsf{NTT}(\boldsymbol{g})$ where $\circ$ denotes the coefficient-wise multiplication of vectors.

The (scaled) sum of the NTT coefficients of a polynomial $\boldsymbol{f} \in \mathcal{R}_q$ is equal to its first d/l coefficients. This will be later used when proving unstructured linear relations over $\mathbb{Z}_q$.

Lemma 2.1. *Let $\boldsymbol{f} \in \mathcal{R}_q$. Then $\frac{1}{l}\sum_{i\in\mathbb{Z}_{2l}^\times} \hat{\boldsymbol{f}}_i = f_0 + f_1 X + \cdots + f_{d/l-1}X^{d/l-1}$, when we lift the $\hat{\boldsymbol{f}}_i$ to $\mathbb{Z}_q[X]$.*

Proof. Write $\boldsymbol{f}(X) = \boldsymbol{f}_0(X^{d/l}) + \boldsymbol{f}_1(X^{d/l})X + \cdots + \boldsymbol{f}_{d/l-1}(X^{d/l})X^{d/l-1}$ Then, it suffices to prove

$$\frac{1}{l}\sum_{i\in\mathbb{Z}_{2l}^\times} \boldsymbol{f}_j(\zeta^i) = f_j$$

for all $j = 0, \ldots, d/l-1$, which is the sum over the coefficients of a fully splitting length-l NTT. We find

$$\sum_{i\in\mathbb{Z}_{2l}^\times} \boldsymbol{f}_j(\zeta^i) = \sum_{i\in\mathbb{Z}_{2l}^\times}\sum_{\nu=0}^{l-1} f_{\nu d/l+j}\zeta^{i\nu} = \sum_{\nu=0}^{l-1} f_{\nu d/l+j}\sum_{i\in\mathbb{Z}_{2l}^\times}\zeta^{i\nu}$$

and it remains to show that for every $\nu \in \{1, \ldots, l-1\}$, $\sum_{i\in\mathbb{Z}_{2l}^\times}\zeta^{i\nu} = 0$. Indeed,

$$\sum_{i\in\mathbb{Z}_{2l}^\times}\zeta^{i\nu} = \sum_{i=0}^{l-1}\zeta^{(2i+1)\nu} = \zeta^\nu\sum_{i=0}^{l-1}\zeta^{2i\nu} = \zeta^\nu\frac{\zeta^{2l\nu}-1}{\zeta^{2\nu}-1} = 0$$

since $\zeta^{2l\nu} = 1$. □

2.4 Challenge Space

Let $\mathcal{C} = \{-1, 0, 1\}^d \subset \mathcal{R}_q$ be the set of ternary polynomials, which have coefficients in $\{-1, 0, 1\}$. We define $C\colon \mathcal{C} \to [0, 1]$ to be the probability distribution on $\mathcal{C}$ such that the coefficients of a challenge $\boldsymbol{c} \xleftarrow{\$} C$ are independently identically distributed with $\Pr(0) = 1/2$ and $\Pr(1) = \Pr(-1) = 1/4$.

In [ALS20] it is shown that if $\boldsymbol{c} \xleftarrow{\$} C$ then the distribution of $\boldsymbol{c} \bmod X^{kd/l} - \zeta^k$ is almost uniform.

Lemma 2.2. *Let $\boldsymbol{c} \xleftarrow{\$} C$. The coefficients of $\boldsymbol{c} \bmod X^{kd/l} - \zeta^k$ are independently identically distributed, say with distribution X. Then, for $x \in \mathbb{Z}_q$,*

$$\Pr(X = x) \leq \frac{1}{q} + \frac{2l/k}{q}\sum_{j\in\mathbb{Z}_q^*/\langle\zeta^k\rangle}\prod_{i=0}^{l/k-1}\left|\frac{1}{2} + \frac{1}{2}\cos(2\pi j\zeta^{ki}/q)\right|. \tag{3}$$

For example, by numerical computing the probability in Lemma 2.2, one finds for $d = 128$, $q \approx 2^{32}$ fully splitting, i.e. $l = d$, and $k = 4$, that the maximum probability for the coefficients of $\boldsymbol{c} \bmod X^4 - \zeta^4$ is bounded by $2^{-31.4}$.

2.5 Module-SIS and Module-LWE Problems

We employ the computationally binding and computationally hiding commitment scheme from [BDL+18] in our protocols, and rely on the well-known Module-LWE (MLWE) and Module-SIS (MSIS) [PR06, LPR10, LS15] problems to prove the security of our constructions. Both problems are defined over a ring $\mathcal{R}_q$ for a positive modulus $q \in \mathbb{Z}^+$. For the Module-SIS problem we use the variant with respect to the infinity norm.

Definition 2.3 ($\mathsf{MSIS}_{n,m,\beta_{\text{SIS}}}$)**.** *The goal in the* Module-SIS *problem with parameters $n, m > 0$ and $\beta_{\text{SIS}} > q$ is to find, for a given matrix $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$, $\vec{\boldsymbol{x}} \in \mathcal{R}_q^m$ such that $\boldsymbol{A}\vec{\boldsymbol{x}} = \vec{\boldsymbol{0}}$ over $\mathcal{R}_q$ and $0 < \|\vec{\boldsymbol{x}}\|_\infty \leq \beta_{\text{SIS}}$. We say that a PPT adversary $\mathcal{A}$ has advantage ϵ in solving $\mathsf{MSIS}_{n,m,\beta_{\text{SIS}}}$ if*

$$\Pr\left[0 < \|\vec{\boldsymbol{x}}\|_\infty \leq \beta_{\text{SIS}} \wedge \boldsymbol{A}\vec{\boldsymbol{x}} = \vec{\boldsymbol{0}} \text{ over } \mathcal{R}_q \,\middle|\, \boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m};\ \vec{\boldsymbol{x}} \leftarrow \mathcal{A}(\boldsymbol{A})\right] \geq \epsilon.$$

Definition 2.4 ($\mathsf{MLWE}_{n,m,\chi}$)**.** *In the* Module-LWE *problem with parameters $n, m > 0$ and an error distribution χ over $\mathcal{R}$, the PPT adversary $\mathcal{A}$ is asked to distinguish $(\boldsymbol{A}, \vec{\boldsymbol{t}}) \xleftarrow{\$} \mathcal{R}_q^{m\times n} \times \mathcal{R}_q^m$ from $(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})$ for $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times n}$, a secret vector $\vec{\boldsymbol{s}} \xleftarrow{\$} \chi^n$ and error vector $\vec{\boldsymbol{e}} \xleftarrow{\$} \chi^m$. We say that $\mathcal{A}$ has advantage ϵ in solving* $\mathsf{MLWE}_{n,m,\chi}$ *if*

$$\left|\Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times n};\ \vec{\boldsymbol{s}} \xleftarrow{\$} \chi^n;\ \vec{\boldsymbol{e}} \xleftarrow{\$} \chi^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})\right]\right. \tag{4}$$
$$\left. - \Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times n};\ \vec{\boldsymbol{t}} \xleftarrow{\$} \mathcal{R}_q^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \vec{\boldsymbol{t}})\right]\right| \geq \epsilon.$$

For our practical security estimations of these two problems against known attacks, the parameter m in both of the problems does not play a crucial role. Therefore, we sometimes simply omit m and use the notations $\mathsf{MSIS}_{n,B}$ and $\mathsf{MLWE}_{n,\chi}$. The parameters κ and λ denote the *module ranks* for MSIS and MLWE, respectively.

2.6 Error Distribution, Discrete Gaussians and Rejection Sampling

For sampling randomness in the commitment scheme that we use, and to define the particular variant of the Module-LWE problem that we use, we need to specify the error distribution χ^d on $\mathcal{R}$. In general any of the standard choices in the literature is fine. So, for example, χ can be a narrow discrete Gaussian distribution or the uniform distribution on a small interval. In the numerical examples in Section 4.2 we assume that χ is the computationally simple centered binomial distribution on $\{-1, 0, 1\}$ where ± 1 both have probability $5/16$ and 0 has probability $6/16$. This distribution is chosen (rather than the more "natural" uniform one) because it is easy to sample given a random bitstring by computing $a_1 + a_2 - b_1 - b_2 \bmod 3$ with uniformly random bits a_i, b_i.

Rejection Sampling. In our zero-knowledge proof, 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 $\vec{\boldsymbol{z}}$ cannot be used to gain any information on the prover's secret. During the protocol, the prover computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ where $\vec{\boldsymbol{r}}$ is the randomness used to commit to the prover's secret, $\boldsymbol{c} \xleftarrow{\$} C$ is a challenge polynomial, and $\vec{\boldsymbol{y}}$ is a "masking" vector. To remove the dependency of $\vec{\boldsymbol{z}}$ on $\vec{\boldsymbol{r}}$, we use the rejection sampling technique by Lyubashevsky [Lyu08, Lyu09, Lyu12]. In the two variants of this technique the masking vector is either sampled uniformly from some bounded region or using a discrete Gaussian distribution.

Although the Gaussian variant allows to sample $\vec{\boldsymbol{y}}$ from narrower distributions for acceptable rejection rates, we use the uniform variant in this paper. The reasons for this are that, firstly, uniform sampling is much faster in implementations and much easier to protect against side-channel attacks, and, secondly, uniform sampling allows to be combined with the compression techniques from [BG14, DKL+18], which make up for the disadvantage concerning the width of the distribution.

The gist of uniform rejection sampling is the following. Let T be a bound on the infinity norm of $\boldsymbol{c}\vec{\boldsymbol{r}}$ and let the coefficients of the polynomials of $\vec{\boldsymbol{y}}$ be sampled from the interval $[-\delta_1, \delta_1[$. Then, the conditioned distribution of the coefficients of $\vec{\boldsymbol{z}}$ given that $\|\vec{\boldsymbol{z}}\|_\infty < \delta_1 - T$ is the uniform distribution on $]-(\delta_1 - T), \delta_1 - T[$, independent of $\boldsymbol{c}\vec{\boldsymbol{r}}$.

2.7 Commitment Scheme

In our protocol, we use a variant of the commitment scheme from [BDL+18], which allows to commit to a vector of messages in $\mathcal{R}_q$. Suppose that we want to commit to a message vector $\vec{\boldsymbol{m}} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_l)^T \in \mathcal{R}_q^l$ and that module ranks of κ and λ are required for MSIS and MLWE security, respectively. Then, in the key generation, a uniformly random matrix $\boldsymbol{B}_0 \xleftarrow{\$} \mathcal{R}_q^{\kappa\times(\lambda+\kappa+l)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_l \xleftarrow{\$} \mathcal{R}_q^{\lambda+\kappa+l}$ are generated and output as public parameters. In practice, one may choose to generate $\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_l$ in a more structured way as in [BDL+18] 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 MLWE problems. We refer to [EZS+19, Appendix C] for a more detailed discussion.

To commit to the message $\vec{\boldsymbol{m}}$, we first sample $\vec{\boldsymbol{r}} \xleftarrow{\$} \chi^{(\lambda+\kappa+l)d}$. Now, there are two parts of the commitment scheme; the binding part and the message encoding part. Particularly, we compute

$$\begin{aligned}\vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0\vec{\boldsymbol{r}},\\ \boldsymbol{t}_i &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}\rangle + \boldsymbol{m}_i \quad \text{for } i = 1,\dots,l,\end{aligned}$$

where $\vec{\boldsymbol{t}}_0$ forms the binding part and each $\boldsymbol{t}_i$ encodes a message polynomial $\boldsymbol{m}_i$. The commitment scheme is computationally hiding under the Module-LWE assumption and computationally binding under the Module-SIS assumption; see [BDL+18].

The utility of the commitment scheme for zero-knowledge proof systems stems from the fact that one can compute module homomorphisms on committed messages. For example, let $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ be from $\mathcal{R}_q$. Then

$$\boldsymbol{a}_1\boldsymbol{t}_1 + \boldsymbol{a}_2\boldsymbol{t}_2 = \langle \boldsymbol{a}_1\vec{\boldsymbol{b}}_1 + \boldsymbol{a}_2\vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}}\rangle + \boldsymbol{a}_1\boldsymbol{m}_1 + \boldsymbol{a}_2\boldsymbol{m}_2$$

is a commitment to the message $\boldsymbol{a}_1\boldsymbol{m}_1+\boldsymbol{a}_2\boldsymbol{m}_2$ with matrix $\boldsymbol{a}_1\vec{\boldsymbol{b}}_1+\boldsymbol{a}_2\vec{\boldsymbol{b}}_2$. This module homomorphic property together with a proof that a commitment is a commitment to the zero polynomial allows to prove linear relations among committed messages over $\mathcal{R}_q$.

2.8 Opening and Product Proof

We use the opening proof from [ALS20, Figure 2] that we sketch now. Suppose that the prover knows an opening to the commitment

$$\begin{aligned}\vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0\vec{\boldsymbol{r}},\\ \boldsymbol{t}_1 &= \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle + \boldsymbol{m}_1.\end{aligned}$$

As in previous opening proofs the prover gives an approximate proof for the first equation. To this end, the prover and verifier engage in k parallel executions of a sigma protocol with challenges $\sigma^i(\boldsymbol{c})$, $i = 0,\dots,k-1$, that are the rotations of a global challenge $\boldsymbol{c} \xleftarrow{\$} C$. Concretely, in the first flow, the prover samples k short masking vectors $\vec{\boldsymbol{y}}_i$ from the discrete Gaussian distribution $D_{\mathfrak{s}}^{(\lambda+\kappa+1)d}$ and sends commitments $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$ over to the verifier. The verifier replies with the challenge $\boldsymbol{c}$. Then the prover applies rejection sampling, and, if this does not reject, sends $\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$. The verifier checks that the $\vec{\boldsymbol{z}}_i$ are short and the equations $\boldsymbol{B}_0\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0$.

Now, unlike in previous protocols, the algebraic setup is such that it is not possible to extract a pair of accepting transcript with invertible challenge difference $\bar{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c}'$. Instead, extraction works by piecing together l/k accepting transcripts where for each ideal $(X^{kd/l} - \zeta^{kj})$, there is a transcript pair with challenge difference $\bar{\boldsymbol{c}}_j \bmod (X^{kd/l} - \zeta^{kj}) \neq 0$. For this to work out it is required that the maximum probability p over $\mathbb{Z}_q$ of the coefficients of $\boldsymbol{c} \bmod (X^{kd/l} - \zeta^k)$, as given by Lemma 2.2, is such that $p^{kd/l}$ is negligible. For example, if $d = 128$, $q \approx 2^{32}$ fully splits so that $l = d$, and $k = 4$, then $p^{kd/l} = p^4 \approx 2^{-128}$.

Next, the analysis of the protocol given in [ALS20, Theorem 4.4] shows that it is possible to extract a weak opening from a prover with non-negligible high success probability, as given in the following definition.

Definition 2.5. *A* weak opening *for the commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1$ consists of l polynomials $\sigma^i(\bar{\boldsymbol{c}}_j) \in \mathcal{R}_q$, a randomness vector $\vec{\boldsymbol{r}}^*$ over $\mathcal{R}_q$ and a message $\boldsymbol{m}_1^* \in \mathcal{R}_q$ such that*

$$\begin{aligned}&\left\|\sigma^i(\bar{\boldsymbol{c}}_j)\right\|_1 \leq 2d \text{ and } \sigma^i(\bar{\boldsymbol{c}}_j) \bmod \sigma^i(X^{d/l} - \zeta^j) \neq 0 \text{ for all } (i,j) \in I,\\ &\left\|\sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{r}}^*\right\|_2 \leq 2\beta \text{ for all } (i,j) \in I,\\ &\boldsymbol{B}_0\vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0,\\ &\langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^*\rangle + \boldsymbol{m}_1^* = \boldsymbol{t}_1.\end{aligned}$$

The commitment scheme is binding with respect to weak openings, c.f. [ALS20, Lemma 4.3]. Furthermore, in the extraction it is also possible to obtain vectors $\vec{\boldsymbol{y}}_i^*$ such that every accepting transcript satisfies the following

$$\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}^* + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}^*,$$

when it contains the same prover commitments $\vec{\boldsymbol{w}}_i$ that were used in the extraction.

We also apply the product proof from [ALS20, Figure 4], adapted to the case of a cubic relation, to prove that our secret vector has ternary coefficients. In addition to the opening proof, the product proof only requires two additional commitments to garbage terms.

3 Proving Unstructured Linear Relations over $\mathbb{Z}_q^n$

Our goal for this section is to construct an efficient protocol for proving unstructured linear relations among committed $\mathbb{Z}_q$-elements. By this we mean that we want to be able to commit to a vector $\vec{s} \in \mathbb{Z}_q^n$ and prove that it fulfills an arbitrary linear equation $A\vec{s} = \vec{u}$ for a public matrix $A \in \mathbb{Z}_q^{m \times n}$ and vector $\vec{u} \in \mathbb{Z}_q^m$. We borrow LWE terminology and call the linear equation "unstructured" to highlight the fact that A can be an arbitrary matrix over $\mathbb{Z}_q$ that does not necessarily express linear relations over some ring of higher rank.

Proofs of linear relations are useful for applications in lattice cryptography only if it is possible to amend them by a proof of shortness. So, we will also want to be able to prove that the vector $\vec{s}$ is short. As opposed to the so-called *approximate* proofs that are ubiquitous in lattice cryptography and where the prover only proves knowledge of a vector that is much longer than the one it actually knows, we are interested in *exact* proofs of shortness. These have the advantage that the parameters of underlying cryptographic schemes do not have to account for the longer vectors that can be extracted from a prover, i.e. the schemes do not need to be secure with respect to the longer vectors. This results in more efficient schemes. For example, one interesting goal of this line of research is to construct a proof of plaintext knowledge or a verifiable encryption scheme for a standard unmodified lattice-based public-key encryption scheme. In particular, for one of the schemes submitted to the NIST PQC standardization effort.

The most efficient lattice-based exact proofs of shortness work by encoding the vector $\vec{s}$ in the NTT representations $\mathsf{NTT}(\check{\boldsymbol{s}}_i)$ of possibly several polynomials $\check{\boldsymbol{s}}_i \in \mathcal{R}_q$. In the first step, we restrict to the case where q splits completely in $\mathcal{R}$. Then $\mathsf{NTT}(\check{\boldsymbol{s}}_i)$ is a vector in $\mathbb{Z}_q^d$.

Now, for simplicity, assume that n is divisible by d. Suppose the prover $\mathcal{P}$ knows an opening to a commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1 \parallel \cdots \parallel \boldsymbol{t}_{n/d}$ to n/d secret polynomials $\check{\boldsymbol{s}}_1, \ldots, \check{\boldsymbol{s}}_{n/d} \in \mathcal{R}_q$. More precisely,

$$\begin{aligned}\vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0\vec{\boldsymbol{r}},\\ \boldsymbol{t}_i &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \check{\boldsymbol{s}}_i \text{ for } i \in \{1, \ldots, n/d\}.\end{aligned}$$

Then, the goal of $\mathcal{P}$ is to prove that the vector

$$\vec{s} = \mathsf{NTT}(\check{\boldsymbol{s}}_1) \parallel \cdots \parallel \mathsf{NTT}(\check{\boldsymbol{s}}_{n/d}) \in \mathbb{Z}_q^n$$

satisfies the linear equation $A\vec{s} = \vec{u}$ over $\mathbb{Z}_q$ where $A \in \mathbb{Z}_q^{m \times n}$ and $\vec{u} \in \mathbb{Z}_q^m$ are public.

Firstly, we describe the main ideas and present a protocol which achieves soundness error $1/q$. Then, in Section 3.2 and Appendix A we present two methods to efficiently decrease the soundness error to negligible quantities. The latter one, however, is only interesting when the secret vector $\vec{s}$ is strictly shorter than d. In that case, we make use of non-fully splitting rings $\mathcal{R}_q$.

Although we present all of our protocols so that only *non-aborting* protocol transcripts are simulatable, there is a standard generic method to simulate aborts of an *interactive* protocol as given in [BCK+14], which is also used, e.g., in [ESS+19]. In particular, for all but the last move of the prover, the prover sends $\mathsf{aCom}(M)$ instead of the transmitted text M for an auxiliary commitment aCom. In the last move, all of these committed texts are revealed unless aborted. In the case of abort, the prover just sends an error message $\perp$. The abort can easily be simulated in this case by relying on the hiding property of aCom. We refer to [BCK+14, ESS+19] for more details. Also, note that the simulation of aborts is not important for most of the practical applications as the protocol is made non-interactive and the simulation of aborts is not needed in that case.

3.1 Basic Protocol

Let us assume that $n = d$ and denote $\check{\boldsymbol{s}} := \check{\boldsymbol{s}}_1$. We show how to deal with the case $n > d$ in Section 3.3. The first protocol relies on the following simple observation. Suppose that $A\vec{s} = \vec{u}$. This means that for all $\vec{\gamma} \in \mathbb{Z}_q^m$, we have $\langle A\vec{s} - \vec{u}, \vec{\gamma}\rangle = 0$. On the contrary, if $A\vec{s} \neq \vec{u}$, then for a uniformly random $\vec{\gamma} \in \mathbb{Z}_q^m$, $\langle A\vec{s} - \vec{u}, \vec{\gamma}\rangle = 0$ only with probability $1/q$. Hence, $\vec{\gamma}$ will become a challenge generated from the verifier. Using Lemma 2.1, we rewrite the inner product,

$$\begin{aligned}\langle A\vec{s} - \vec{u}, \vec{\gamma}\rangle &= \langle A\vec{s}, \vec{\gamma}\rangle - \langle \vec{u}, \vec{\gamma}\rangle = \langle \vec{s}, A^T\vec{\gamma}\rangle - \langle \vec{u}, \vec{\gamma}\rangle \\ &= \sum_{j \in \mathbb{Z}_{2d}^\times} \boldsymbol{s}(\zeta^j)\left(\mathsf{NTT}^{-1}(A^T\vec{\gamma})\right)(\zeta^j) - \langle \vec{u}, \vec{\gamma}\rangle = \frac{1}{d}\sum_{j \in \mathbb{Z}_{2d}^\times} \boldsymbol{f}(\zeta^j) = f_0,\end{aligned}$$

where $\boldsymbol{f} \in \mathcal{R}_q$ is the polynomial defined by $\boldsymbol{f} := \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\check{\boldsymbol{s}} - \langle \vec{u}, \vec{\gamma}\rangle$ and $f_0 \in \mathbb{Z}_q$ is the constant coefficient of $\boldsymbol{f}$. So, by utilizing the polynomial product in $\mathcal{R}_q$, it is possible to compute a scalar product over $\mathbb{Z}_q$ with a vector encoded in the NTT representation of the polynomial. We observe that the verifier can compute a commitment to $\boldsymbol{f}$. Indeed, note that

$$\mathsf{NTT}^{-1}(dA^T\vec{\gamma})\boldsymbol{t}_1 - \langle \vec{u}, \vec{\gamma}\rangle = \langle \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\vec{\boldsymbol{b}_1}, \vec{\boldsymbol{r}}\rangle + \boldsymbol{f}.$$

Hence, $\mathcal{V}$ can compute the commitment

$$\boldsymbol{\tau} = \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\boldsymbol{t}_1 - \langle \vec{u}, \vec{\gamma}\rangle. \tag{5}$$

Now, $\mathcal{P}$ needs to prove that $\boldsymbol{f}$ has a zero constant coefficient. The idea is to first send a commitment $\boldsymbol{t}_2$ to a random polynomial $\boldsymbol{g}$ with a zero constant coefficient before $\vec{\gamma}$ is generated. Intuitively, $\boldsymbol{g}$ is introduced to mask $\boldsymbol{f}$. After getting $\vec{\gamma}$, $\mathcal{P}$ sends $\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g}$ and the verifier can check that $h_0 = 0$. Note that by knowing $\boldsymbol{\tau}, \boldsymbol{t}_2$ and $\boldsymbol{h}$, the verifier can compute a commitment $\boldsymbol{\tau} + \boldsymbol{t}_2 - \boldsymbol{h}$ to the zero polynomial $\mathbf{0}$. Hence, in the final stage, $\mathcal{P}$ needs to prove that this polynomial is indeed a commitment to $\mathbf{0}$ in the usual way.

The full protocol is presented as follows. First, the prover $\mathcal{P}$ generates a random polynomial $\boldsymbol{g} \in \mathcal{R}_q$ with zero constant coefficient and computes a commitment to $\boldsymbol{g}$ defined as $\boldsymbol{t}_2 = \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}}\rangle + \boldsymbol{g}$. The prover also starts the opening proof with soundness error $1/q$ for the commitments and samples a vector of small polynomials $\vec{\boldsymbol{y}}$ and computes the commitment $\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$. Then, $\mathcal{P}$ sends $\boldsymbol{t}_2$ and $\vec{\boldsymbol{w}}$ to the verifier. Next, $\mathcal{V}$ generates and sends a uniformly random vector $\vec{\gamma} \in \mathbb{Z}_q^m$. $\mathcal{P}$ can then compute the polynomial $\boldsymbol{f}$ defined above and $\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g}$. Furthermore, it sets $\boldsymbol{v} = \langle \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\vec{\boldsymbol{b}_1} + \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{y}}\rangle$ and sends $\boldsymbol{h}, \boldsymbol{v}$ to $\mathcal{V}$. Then, the verifier generates a challenge $\boldsymbol{c} \xleftarrow{\$} C$ and sends it to the prover. Eventually, $\mathcal{P}$ sends a response $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$.

The verifier $\mathcal{V}$ first checks that $\vec{\boldsymbol{z}}$ consists of small polynomials and that $\boldsymbol{h}$ has constant coefficient equal to 0. Also, $\mathcal{V}$ checks that $\boldsymbol{B}_0\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}_0}$ and

$$\langle \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\vec{\boldsymbol{b}_1} + \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{z}}\rangle = \boldsymbol{v} + \boldsymbol{c}\,(\boldsymbol{\tau} + \boldsymbol{t}_2 - \boldsymbol{h})$$

where $\boldsymbol{\tau}$ is computed as in Equation (5).

One can observe that if $A\vec{s} \neq \vec{u}$ then the constant coefficient of $\boldsymbol{f}$ becomes a uniformly random element of $\mathbb{Z}_q$, outside the control of the prover. Thus, also the constant coefficient of $\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g}$ will be uniformly random because the constant coefficient of $\boldsymbol{g}$ is independent of the constant coefficient of $\boldsymbol{f}$. In particular, it will be non-zero with probability $1 - 1/q$ and this can be detected by the verifier. Therefore, the probability that a malicious prover manages to cheat is essentially $1/q$.

3.2 Boosting Soundness by Mapping Down

More abstractly, in the above protocol we checked $\langle A\vec{s} - \vec{u}, \vec{\gamma}\rangle = 0$ by investigating whether $\mathsf{L}(\vec{\gamma})$ has a zero constant coefficient where $\mathsf{L} : \mathbb{Z}_q^m \to \mathcal{R}_q$ is defined as

$$\mathsf{L}(\vec{\gamma}) := \mathsf{NTT}^{-1}(dA^T\vec{\gamma})\check{\boldsymbol{s}} - \langle \vec{u}, \vec{\gamma}\rangle. \tag{6}$$

As we observed earlier, the constant coefficient of $\mathsf{L}(\vec{\gamma})$ is indeed $\langle A\vec{s} - \vec{u}, \vec{\gamma}\rangle$.

Now, suppose we can define k functions $\mathsf{L}_0, \ldots, \mathsf{L}_{k-1}$ with the following property. For any $0 \leq \mu < k$ and $\vec{\gamma}_\mu \in \mathbb{Z}_q^m$, $\boldsymbol{p} = L_\mu(\vec{\gamma}_\mu) \in \mathcal{R}_q$ is a polynomial such that $p_0 = \ldots = p_{\mu-1} = p_{\mu+1} = \ldots = p_{k-1} = 0$ and $p_\mu = \langle A\vec{s} - \vec{u}, \vec{\gamma}_\mu\rangle$. This would mean that for $0 \leq \mu < k$, the μ-th coefficient related to X^μ of the polynomial

$$\boldsymbol{f} = \mathsf{L}_0(\vec{\gamma}_0) + \mathsf{L}_1(\vec{\gamma}_1) + \ldots + \mathsf{L}_{k-1}(\vec{\gamma}_{k-1})$$

is equal to $\langle A\vec{s} - \vec{u}, \vec{\gamma_\mu}\rangle$. In particular, if $A\vec{s} = \vec{u}$, then $f_0 = f_1 = \ldots = f_{k-1} = 0$. Thus, in order to decrease the soundness error, we can let the verifier $\mathcal{V}$ send k independently uniformly random vectors $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1}$ and then $\mathcal{P}$ proves that $\boldsymbol{f} \in \mathcal{R}_q$ has the first k coefficients equal to zero. Note that we still need to find a way for $\mathcal{V}$ to compute a commitment to $\boldsymbol{f}$ from $\vec{\boldsymbol{t}}_1$ and $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1}$.

Constructing L_μ. Let $\mathcal{S}_q$ be the $\mathbb{Z}_q$-submodule of $\mathcal{R}_q$ generated by X^k, i.e.

$$\mathcal{S}_q = \{p_0 + p_1 X^k + \cdots + p_{d/k-1} X^{d-k} \in \mathcal{R}_q\} \subset \mathcal{R}_q.$$

We have $\mathcal{S}_q \cong \mathbb{Z}_q[X]/(X^{d/k} + 1)$. From Galois theory, there is a corresponding subgroup H of $\mathsf{Aut}(\mathcal{R}_q)$ of order k such that $\sigma(\boldsymbol{p}) = \boldsymbol{p}$ for all $\sigma \in H$ if and only if $\boldsymbol{p} \in \mathcal{S}_q$. It is easy to see that this group is generated by $\sigma = \sigma_{2d/k+1} \in \mathsf{Aut}(\mathcal{R}_q)$, which is the same automorphism that we use in the automorphism opening proof. In fact, this follows from the fact that $\mathrm{ord}(\sigma) = k$ and $\sigma(X^k) = X^{k(2d/k+1)} = X^k$.

We have the trace map $\mathsf{Tr}\colon \mathcal{R}_q \to \mathcal{S}_q$ given by

$$\mathsf{Tr}(\boldsymbol{p}) = \sum_{\nu=0}^{k-1} \sigma^\nu(\boldsymbol{p}).$$

Notice that the constant coefficient of $\mathsf{Tr}(\boldsymbol{p})$ is equal to kp_0. Now define L_μ by

$$\mathsf{L}_\mu(\vec{\gamma}) = \frac{1}{k} X^\mu \mathsf{Tr}(\mathsf{L}(\vec{\gamma})) = \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\mathsf{NTT}^{-1}(dA^T\vec{\gamma})\check{\boldsymbol{s}} - \langle \vec{u}, \vec{\gamma}\rangle\right).$$

If $\boldsymbol{p} = \mathsf{L}_\mu(\vec{\gamma})$, then $\boldsymbol{p}$ is of the form

$$\boldsymbol{p} = p_\mu X^\mu + p_{k+\mu} X^{k+\mu} + \cdots + p_{d-k+\mu} X^{d-k+\mu}$$

and thus has the property that the first k coefficients except the μ-th coefficient are zero. Moreover, it is clear from above that $p_\mu = \langle A\vec{s} - \vec{u}, \vec{\gamma}\rangle$.

Finally, given the commitment $\boldsymbol{t}_1$ to $\boldsymbol{s}$, the verifier can compute a commitment to $\boldsymbol{f} = \mathsf{L}_0(\vec{\gamma}_0) + \cdots + \mathsf{L}_{k-1}(\vec{\gamma}_{k-1})$ via

$$\begin{aligned}\boldsymbol{\tau} &= \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\boldsymbol{t}_1 - \langle \vec{u}, \vec{\gamma}_\mu\rangle\right) \\ &= \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\langle \mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle\right) + \boldsymbol{f}. \end{aligned} \tag{7}$$

The Protocol. We present the full protocol in Figure 1 with the verification algorithm given in Figure 2.

It is natural to separate the commitment $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ to the secret polynomial $\check{\boldsymbol{s}}$ from our protocol for proving the linear relation. Then, $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ is given as input to the protocol, which also proves knowledge of an opening to the external commitment. Now, for efficiency reasons, one wants to avoid sending a completely fresh commitment to the masking polynomial $\boldsymbol{g}$ and instead reuse the top part $\vec{\boldsymbol{t}}_0$ of the commitment to $\check{\boldsymbol{s}}$, but this creates a problem with the standard notion of a zero-knowledge proof. Namely, with this approach

it is required that the randomness vector $\vec{r}$, which is a part of the witness, is really random so that the commitment to $\boldsymbol{g}$ is hiding, but the zero-knowledge definition demands simulatability for any (fixed) witness. Hence, we don't take this approach and also send the commitment to $\check{\boldsymbol{s}}$ as part of our protocol. Then, our protocol only shows knowledge of a solution $\vec{s}$ to the linear equation $A\vec{s} = \vec{u}$. This in itself is not a solution to a hard problem but our protocol is still useful because it can be combined with a shortness proof that simultaneously shows that $\vec{s}$ is short (see Section 4). In isolation our protocol is best viewed as a so-called commit-and-proof protocol [CLOS02], which is interesting even without involving a hard problem because of the commitment that can later be used outside of the protocol.

The Prover $\mathcal{P}$ starts by generating a uniformly random polynomial $\boldsymbol{g}$ satisfying $g_0 = \ldots = g_{k-1} = 0$ and then computes the commitment

$$\begin{aligned}\vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0\vec{\boldsymbol{r}}\\ \boldsymbol{t}_1 &= \langle\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle + \check{\boldsymbol{s}}\\ \boldsymbol{t}_2 &= \langle\vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}}\rangle + \boldsymbol{g}.\end{aligned}$$

Now the prover needs to start an opening proof with soundness $1/q^k$. Also, it is going to prove a relation which involves the k automorphisms σ^i. Therefore, it uses the automorphism opening proof from [ALS20] and samples vectors $\vec{\boldsymbol{y}}_0, \ldots, \vec{\boldsymbol{y}}_{k-1}$ of short polynomials that are going to be used to mask $\vec{\boldsymbol{r}}$ k times with challenges of the form $\sigma^i(\boldsymbol{c})$. Also, $\mathcal{P}$ computes $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$. The prover sends $\vec{\boldsymbol{t}}_0$, $\boldsymbol{t}_1$, $\boldsymbol{t}_2$ and $\vec{\boldsymbol{w}}_i$ to $\mathcal{V}$.

Next, the verifier selects uniformly random vectors $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1} \in \mathbb{Z}_q^m$ and sends them to $\mathcal{P}$. Then, the prover computes

$$\boldsymbol{f} = \sum_{\mu=0}^{k-1} \mathsf{L}_\mu(\vec{\gamma}_\mu) = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu\left(\mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\check{\boldsymbol{s}} - \langle\vec{u}, \vec{\gamma}_\mu\rangle\right).$$

By construction, $f_0 = \ldots = f_{k-1} = 0$. Note that $\mathcal{V}$ can compute a commitment $\boldsymbol{\tau}$ to $\boldsymbol{f}$ as explained above. Now the prover sets $\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g}$ and computes for $i = 0, \ldots, k-1$,

$$\boldsymbol{v}_i = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu\left(\langle\mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}}_{i-\nu \bmod k}\rangle\right) + \langle\vec{\boldsymbol{b}}_2, \vec{\boldsymbol{y}}_i\rangle.$$

It sends $\boldsymbol{h}$ and $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{k-1}$. The verifier sends a random challenge polynomial $\boldsymbol{c} \xleftarrow{\$} C$. Eventually, $\mathcal{P}$ computes $\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ for $i = 0, \ldots, k-1$ and sends $\vec{\boldsymbol{z}}_0, \ldots, \vec{\boldsymbol{z}}_{k-1}$.

The Verifier $\mathcal{V}$ first checks that for all $i = 0, \ldots, k-1$, $\vec{\boldsymbol{z}}_i$ is short, and

$$\boldsymbol{B}_0\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0.$$

Then, $\mathcal{V}$ checks that $h_0, \ldots, h_{k-1}$ are all equal to zero and computes $\boldsymbol{\tau}$ as in (7). Finally, the verifier checks whether for all $i = 0, \ldots, k-1$,

$$\begin{aligned}&\sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu\left(\langle\mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{z}}_{i-\nu \bmod k}\rangle\right) + \langle\vec{\boldsymbol{b}}_2, \vec{\boldsymbol{z}}_i\rangle\\ &= \boldsymbol{v}_i + \sigma^i(\boldsymbol{c})(\tau + \boldsymbol{t}_2 - \boldsymbol{h})\end{aligned}$$

to test whether $\boldsymbol{\tau} + \boldsymbol{t}_2 - \boldsymbol{h}$ really is a commitment to zero.

Security Analysis.

Theorem 3.1. *The protocol in Figure 1 is complete, computational honest verifier zero-knowledge under the Module-LWE 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 $\boldsymbol{c} \bmod X^k - \zeta^k$ as in Lemma 2.2.*

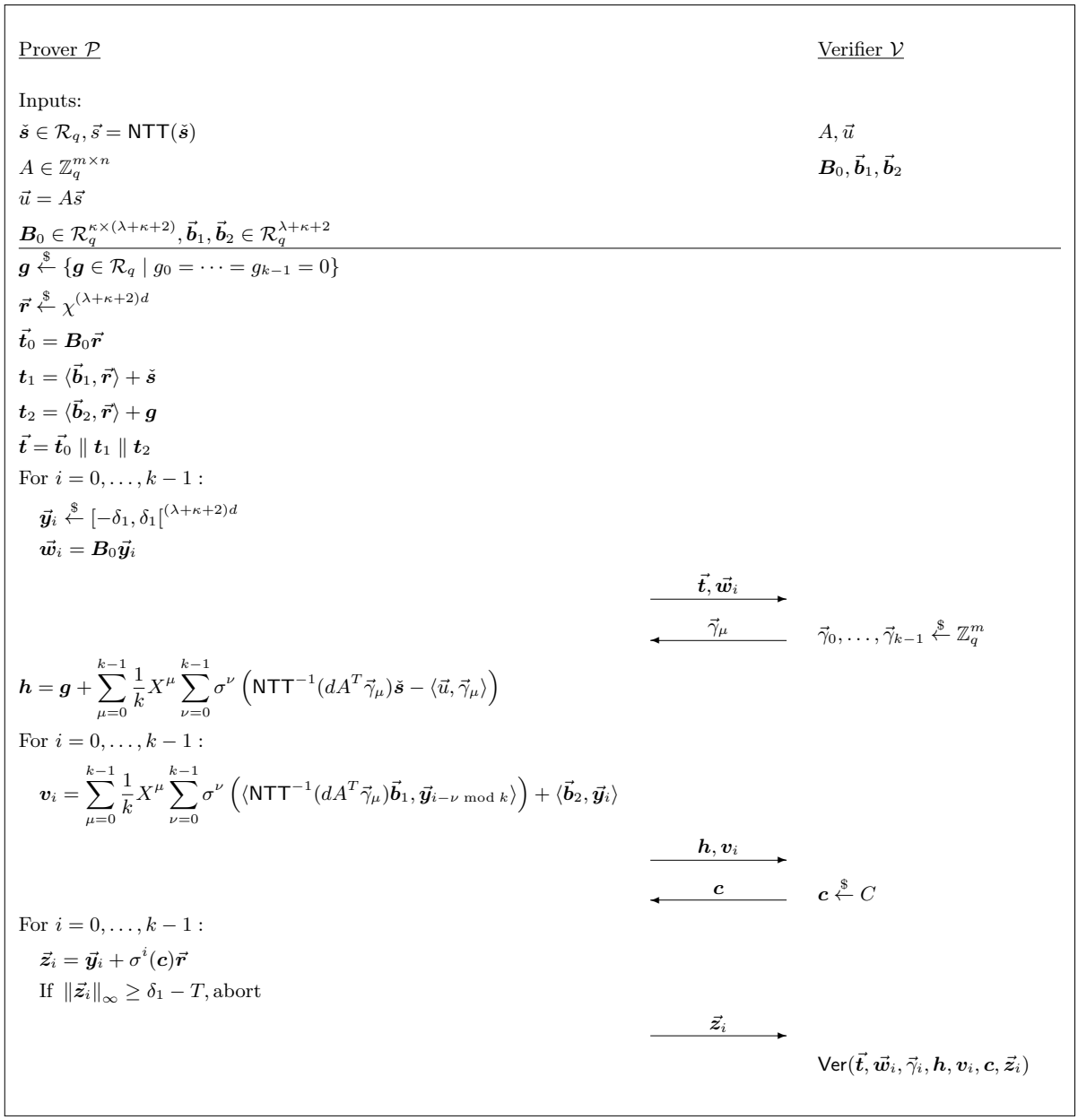

Fig. 1. Automorphism proof of knowledge of a solution to an unstructured linear equation over $\mathbb{Z}_q$. Verification equations are described in Figure 2.

Then, for completeness, unless the honest prover $\mathcal{P}$ aborts due to the rejection sampling, it always convinces the honest verifier $\mathcal{V}$.

For 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 $\mathcal{V}$ for every statement $A\vec{s} = \vec{u}$. 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 ε in distinguishing $\mathsf{MLWE}_{\lambda,\chi}$.

For soundness, there is an extractor $\mathcal{E}$ with the following properties. When given rewindable black-box access to a deterministic prover $\mathcal{P}^*$ that sends the commitment $\vec{\boldsymbol{t}}$ in the first round and convinces $\mathcal{V}$ with probability $\varepsilon \geq q^{-k} + p^k$, $\mathcal{E}$ either outputs a weak opening for $\vec{\boldsymbol{t}}$ with message $\check{\boldsymbol{s}}^*$ such that $A\mathsf{NTT}(\check{\boldsymbol{s}}^*) = \vec{u}$,

$\mathsf{Ver}(\vec{\boldsymbol{t}}, \vec{\boldsymbol{w}}_i, \vec{\gamma}_i, \boldsymbol{h}, \boldsymbol{v}_i, \boldsymbol{c}, \vec{\boldsymbol{z}}_i)$

01 For $i = 0, \ldots, k-1$:

02 $\quad \|\vec{\boldsymbol{z}}_i\|_\infty \overset{?}{<} \beta = \delta_1 - T$

03 $\quad \boldsymbol{B}_0 \vec{\boldsymbol{z}}_i \overset{?}{=} \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0$

04 $h_0 \overset{?}{=} \ldots \overset{?}{=} h_{k-1} \overset{?}{=} 0$

05 $\boldsymbol{\tau} = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\boldsymbol{t}_1 - \langle \vec{u}, \vec{\gamma}_\mu \rangle\right)$

06 For $i = 0, \ldots, k-1$:

07 $\quad \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\langle \mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\boldsymbol{b}_1, \vec{\boldsymbol{z}}_{i-\nu \bmod k} \rangle\right) + \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{z}}_i \rangle \overset{?}{=} \boldsymbol{v}_i + \sigma^i(\boldsymbol{c})(\boldsymbol{\tau} + \boldsymbol{t}_2 - \boldsymbol{h})$

Fig. 2. Verification equations for Figure 1.

or a $\mathsf{MSIS}_{\kappa,8d\beta}$ *solution for* $\boldsymbol{B}_0$ *in expected time at most* $1/\varepsilon + (d/k)(\varepsilon - p^k)^{-1}$ *when running* $\mathcal{P}^*$ *once is assumed to take unit time.*

Proof. Completeness. It follows by careful inspection that the verification equations are always true for the messages sent by $\mathcal{P}$.

Zero-Knowledge. We can simulate a non-aborting transcript between the honest prover and the honest verifier in the following way. First, in a non-aborting transcript the vectors $\vec{\boldsymbol{z}}_i$ are independently uniformly random in $]-(\delta_1-T), \delta_1-T[^{(\lambda+\kappa+2)d}$ and also independent from $\boldsymbol{c}\vec{\boldsymbol{r}}$. So the simulator can just sample $\vec{\boldsymbol{z}}_i \overset{\$}{\leftarrow}]-(\delta_1-T), \delta_1-T[^{(\lambda+\kappa+2)d}$ and $\boldsymbol{c} \overset{\$}{\leftarrow} C$. The polynomial $\boldsymbol{h}$ is such that $h_0 = \cdots = h_{k-1} = 0$ in honest transcripts and the other coefficients are uniformly random because of the additive term $\boldsymbol{g}$. Hence, the simulator samples $\boldsymbol{h} \overset{\$}{\leftarrow} \{\boldsymbol{h} \in \mathcal{R}_q \mid h_0 = \cdots = h_{k-1} = 0\}$. Then, the challenges $\vec{\gamma}_\mu \in \mathbb{Z}_q^m$ are independently uniformly random and the simulator samples them in this way. Now, we turn to the commitment $\vec{\boldsymbol{t}}$. In the honest execution the randomness vector $\vec{\boldsymbol{r}}$ is statistically independent from the $\vec{\boldsymbol{z}}_i$ and the other messages already handled, i.e. $\boldsymbol{c}$, $\boldsymbol{h}$, $\vec{\gamma}_\mu$. So, it follows that the additive term $\boldsymbol{B}_0\vec{\boldsymbol{r}} \parallel \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}} \rangle \parallel \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}} \rangle$ is indistinguishable from uniform given $\vec{\boldsymbol{z}}_i, \boldsymbol{c}, \boldsymbol{h}, \vec{\gamma}_\mu$ if MLWE_λ is hard. Hence $\vec{\boldsymbol{t}}$ is indistinguishable from uniform since the committed polynomials $\check{\boldsymbol{s}}$ and $\boldsymbol{g}$ are also independent from $\vec{\boldsymbol{r}}$ in the honest execution. Therefore, we let the simulator sample a uniformly random $\vec{\boldsymbol{t}} \in \mathcal{R}_q^{\kappa+2}$. Now, in an honest transcript, the remaining messages $\vec{\boldsymbol{w}}_i$ and $\boldsymbol{v}_i$ are all uniquely determined from the already handled messages and the verification equations because of completeness. We see that if the simulator computes these messages so that the verification equations become true, then the resulting transcript is indistinguishable from the honest transcript. More precisely, if there exists some distinguisher that is able to distinguish a simulated transcript from an honest transcript, then it must be able to distinguish the MLWE samples in the commitment $\vec{\boldsymbol{t}}$ from uniform with the same advantage.

Soundness. First, the extractor opens the commitments $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$. From [ALS20, Theorem 4.4], unless $\mathcal{E}$ finds an $\mathsf{MSIS}_{\kappa,8d\beta}$ solution, the extractor can compute vectors $\vec{\boldsymbol{y}}^*$ and $\vec{\boldsymbol{r}}^*$ such that for every accepting transcript with first messages $\boldsymbol{t}_2$ and $\vec{\boldsymbol{w}}_i$,

$$\boldsymbol{z}_i = \vec{\boldsymbol{y}}_i^* + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}^*.$$

The expected runtime for this equals the runtime in the theorem statement. Then let $\check{\boldsymbol{s}}^* \in \mathcal{R}_q$ and $\boldsymbol{g}^* \in \mathcal{R}_q$ be the extracted messages, which are defined by

$$\boldsymbol{t}_1 = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^* \rangle + \check{\boldsymbol{s}}^* \text{ and } \boldsymbol{t}_2 = \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{g}^*.$$

Now substituting these expressions into τ gives

$$\boldsymbol{\tau} = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\langle \mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^* \rangle\right) + \boldsymbol{f}^*,$$

where

$$\boldsymbol{f}^* = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\mathsf{NTT}^{-1}(dA^T\vec{\gamma}_\mu)\check{\boldsymbol{s}}^* - \langle \vec{u}, \vec{\gamma}_\mu \rangle\right).$$

From the discussion in this section we know that $f_\mu^* = \langle A\vec{s}^* - \vec{u}, \vec{\gamma}_\mu \rangle$ for $\mu = 0, \ldots, k-1$, $\vec{s}^* = \mathsf{NTT}(\check{s}^*)$. Next we find from the last verification equations,

$$\begin{aligned}&\left(\sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\langle \mathsf{NTT}^{-1}(dA^T \vec{\gamma}_\mu) \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}}^*_{i-\nu \bmod k} \rangle \right) + \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{y}}^* \rangle - \boldsymbol{v}_i \right) \\ &= \sigma^i(\boldsymbol{c}) \left(\boldsymbol{f}^* + \boldsymbol{g}^* - \boldsymbol{h} \right). \end{aligned} \tag{8}$$

for all $i = 0, \ldots, k-1$. The coefficients of these linear polynomials in $\sigma^i(\boldsymbol{c})$ are independent from $\boldsymbol{c}$ in a random accepting transcript. We bound the success probability ε of the prover under the assumption $A\vec{s}^* \neq \vec{u}$. In this case the coefficients f_μ^* for $\mu = 0, \ldots, k-1$ are uniformly random elements in $\mathbb{Z}_q$ in a random transcript. Hence, $f_\mu^* + g_\mu^*$ is uniformly random since $\boldsymbol{g}^*$ is independent from the $\vec{\gamma}_\mu$. Also we know that $h_\mu = 0$ in every accepting transcript. So, suppose $f_\mu^* + g_\mu^* - h_\mu^* = f_\mu^* + g_\mu^* \neq 0$ for some μ. Then there exists some $j \in \mathbb{Z}_{2d}^\times$ with $\boldsymbol{f}^* + \boldsymbol{g}^* - \boldsymbol{h} \bmod (X - \zeta^j) \neq 0$. Therefore, there is only one possible value modulo $(X^k - \zeta^{jk})$ for the challenge in such a transcript, otherwise Equation 8 cannot be true for all i. Since the maximum probability of every coefficient of $\boldsymbol{c} \bmod (X^k - \zeta^{jk})$ is less than p we see that the success probability is bounded by

$$\begin{aligned}\varepsilon = \Pr\left[\text{accepting}\right] &< \left(\frac{1}{q}\right)^k + \Pr\left[\text{accepting} \,\middle|\, f_\mu^* + g_\mu^* \neq 0 \text{ for some } \mu\right] \\ &\leq \left(\frac{1}{q}\right)^k + p^k.\end{aligned}$$

This is in contradiction to the bound in the theorem statement and thus it must hold $A\vec{s}^* = \vec{u}$. □

3.3 General Case

Previously, we assumed that $n = d$ so that $\vec{s} = \mathsf{NTT}(\check{\boldsymbol{s}}) = \mathsf{NTT}(\check{\boldsymbol{s}}_1)$. When $n > d$, we slightly modify our approach. We have $\vec{s} = \mathsf{NTT}(\check{\boldsymbol{s}}_1) \parallel \cdots \parallel \mathsf{NTT}(\check{\boldsymbol{s}}_{n/d})$ and now also define polynomials $\boldsymbol{\psi}_j$ such that

$$A^T \vec{\gamma} = \mathsf{NTT}(\boldsymbol{\psi}_1) \parallel \cdots \parallel \mathsf{NTT}(\boldsymbol{\psi}_{n/d}).$$

Then the inner product $\langle A\vec{s}, \vec{\gamma} \rangle = \langle \vec{s}, A^T \vec{\gamma} \rangle$ can be written as a sum of smaller inner products. We find

$$\begin{aligned}\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle &= \sum_{j=1}^{n/d} \langle \mathsf{NTT}(\check{\boldsymbol{s}}_j), \mathsf{NTT}(\boldsymbol{\psi}_j) \rangle - \langle \vec{u}, \vec{\gamma} \rangle \\ &= \sum_{j=1}^{n/d} \sum_{i \in \mathbb{Z}_{2d}^\times} \check{\boldsymbol{s}}_j(\zeta^i) \boldsymbol{\psi}_j(\zeta^i) - \langle \vec{u}, \vec{\gamma} \rangle = \frac{1}{d} \sum_{i \in \mathbb{Z}_{2d}^\times} \left(\sum_{j=1}^{n/d} d\check{\boldsymbol{s}}_j \boldsymbol{\psi}_j - \langle \vec{u}, \vec{\gamma} \rangle \right)(\zeta^i).\end{aligned}$$

Next, similarly as before, we incorporate more challenges. So, for $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1} \in \mathbb{Z}_q^m$ we write

$$A^T \vec{\gamma}_\mu = \mathsf{NTT}(\boldsymbol{\psi}_1^{(\mu)}) \| \cdots \| \mathsf{NTT}(\boldsymbol{\psi}_{n/d}^{(\mu)})$$

and then set

$$\boldsymbol{f} = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\sum_{j=1}^{n/d} d\boldsymbol{\psi}_j^{(\mu)} \boldsymbol{s}_j - \langle \vec{u}, \vec{\gamma}_\mu \rangle \right).$$

It holds that for $\mu = 0, \ldots, k-1$, $f_\mu = \langle A\vec{s} - \vec{u}, \vec{\gamma}_\mu \rangle$. Now, note that $\boldsymbol{\tau}$ defined as

$$\begin{aligned}
\boldsymbol{\tau} &= \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\sum_{j=1}^{n/d} d\boldsymbol{\psi}_j^{(\mu)} \boldsymbol{t}_j - \langle \vec{u}, \vec{\gamma}_\mu \rangle \right) \\
&= \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\sum_{j=1}^{n/d} \langle d\boldsymbol{\psi}_j^{(\mu)} \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{r}} \rangle + d\boldsymbol{\psi}_j^{(\mu)} \check{\boldsymbol{s}}_j - \langle \vec{u}, \vec{\gamma} \rangle \right) \\
&= \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\left\langle \sum_{j=1}^{n/d} d\boldsymbol{\psi}_j^{(\mu)} \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{r}} \right\rangle \right) + \boldsymbol{f}
\end{aligned}$$

is indeed a commitment to $\boldsymbol{f}$ and can be computed by the verifier.

4 Main Protocol

In this section we present our main protocol for proving knowledge of a ternary solution $\vec{s} \in \{-1, 0, 1\}^n$ to an arbitrary linear equation $A\vec{s} = \vec{u}$ over $\mathbb{Z}_q$. The protocol is essentially an amalgamation of the linear proof from Section 3 and the product proof from [ALS20]. We use a fully splitting prime q and automorphisms to boost the soundness. So, at a high level the prover commits to n/d polynomials $\check{\boldsymbol{s}}_j$ whose NTT coefficients are the coefficients of $\vec{s}$. That is,

$$\vec{s} = \begin{pmatrix} \mathsf{NTT}(\check{\boldsymbol{s}}_1) \\ \vdots \\ \mathsf{NTT}(\check{\boldsymbol{s}}_{n/d}) \end{pmatrix}.$$

Then the prover uses a generalization of the product proof to many cubic relations to show that

$$\check{\boldsymbol{s}}_j(\check{\boldsymbol{s}}_j + \boldsymbol{1})(\check{\boldsymbol{s}}_j - \boldsymbol{1}) = \boldsymbol{0}$$

for all j. This shows that $\mathsf{NTT}(\check{\boldsymbol{s}}_j) \in \{-1, 0, 1\}^d$ since the polynomial product in $\mathcal{R}_q$ is coefficient-wise in the NTT representation. This is the technique that was used in [BLS19].

In parallel, the prover uses the linear proof for the general case from Section 3.3, to show that the polynomials $\check{\boldsymbol{s}}_j$ really give a solution to the linear equation. The complete protocol is given in Figure 3 and it is proven secure in Theorem 4.1.

4.1 Security Analysis

Theorem 4.1. *The protocol in Figure 3 is complete, computational honest verifier zero-knowledge under the Module-LWE 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 $\boldsymbol{c} \bmod X^k - \zeta^k$ as in Lemma 2.2.*

Then, for completeness, in case the honest prover $\mathcal{P}$ does not abort due to rejection sampling, it always convinces the honest verifier $\mathcal{V}$.

For 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 $\mathcal{V}$ for every statement $A\vec{s} = \vec{u}$, $\vec{s} \in \{-1, 0, 1\}^n$. 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 also has advantage ε in distinguishing $\mathsf{MLWE}_{\lambda,\chi}$.

For soundness, there is an extractor $\mathcal{E}$ with the following properties. When given rewindable black-box access to a deterministic prover $\mathcal{P}^*$ that convinces $\mathcal{V}$ with probability $\varepsilon > (3p)^k$, $\mathcal{E}$ either outputs a solution $\vec{s}^* \in \{-1, 0, 1\}^n$ to $A\vec{s}^* = \vec{u}$, or a $\mathrm{MSIS}_{\kappa,8d\beta}$ solution for $\boldsymbol{B}_0$ in expected time at most $1/\varepsilon + (\varepsilon - p^k)^{-1}$ when running $\mathcal{P}^*$ once is assumed to take unit time.

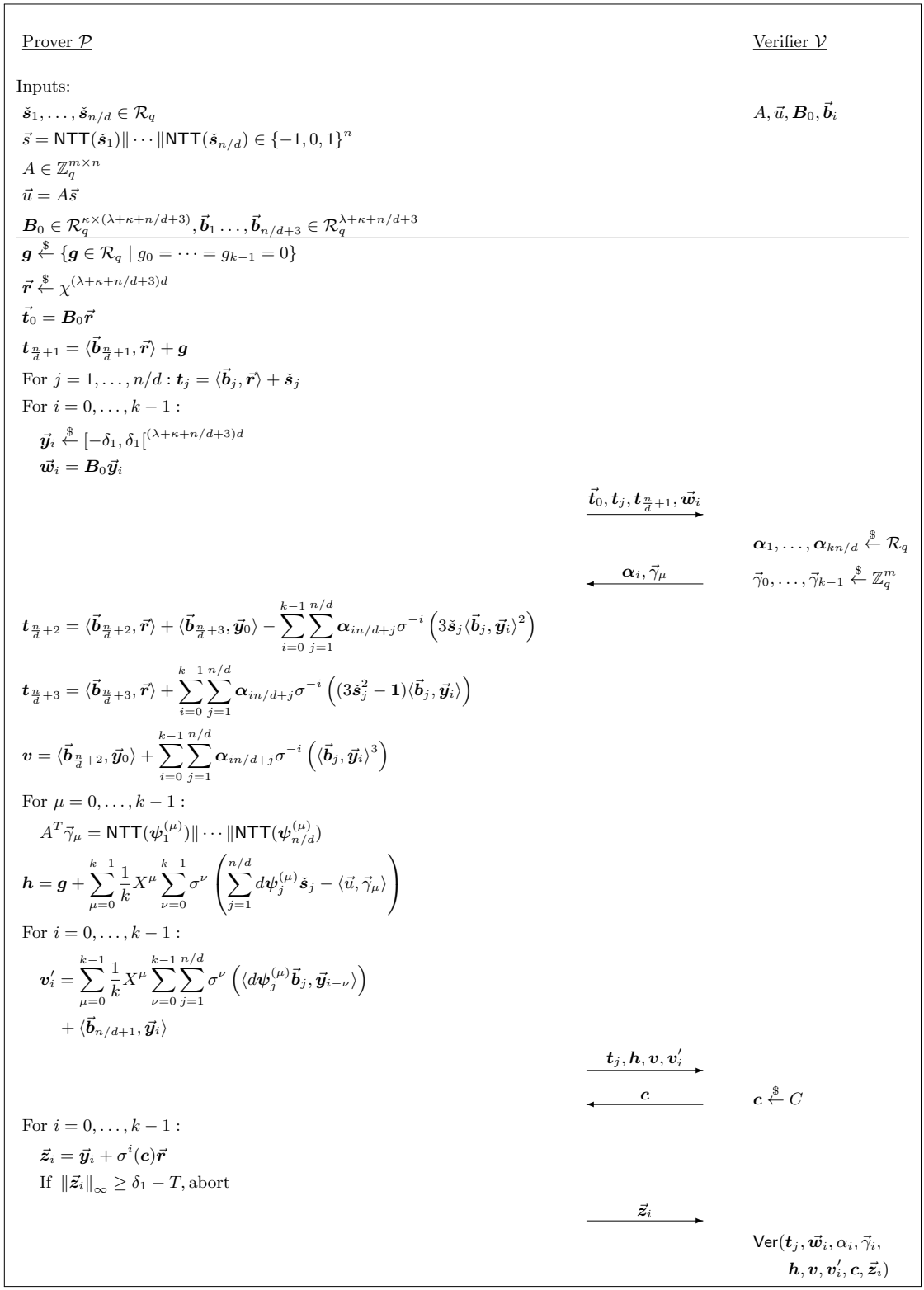

Fig. 3. Proof of knowledge of a ternary solution to an unstructured linear equation over $\mathbb{Z}_q$. Verification equations are defined in Figure 4.

$\mathsf{Ver}(\boldsymbol{t}_j, \vec{\boldsymbol{w}}_i, \boldsymbol{\alpha}_i, \vec{\gamma}_i, \boldsymbol{h}, \boldsymbol{v}, \boldsymbol{v}'_i, \boldsymbol{c}, \vec{\boldsymbol{z}}_i)$

For $i = 0, \ldots, k-1$:

$\quad \|\vec{\boldsymbol{z}}_i\|_2 \overset{?}{<} \beta = \delta_1 - T$

$\quad \boldsymbol{B}_0 \vec{\boldsymbol{z}}_i \overset{?}{=} \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0$

For $i = 0, \ldots, k-1$:

$\quad$ For $j = 1, \ldots, n/d$:

$\quad\quad \boldsymbol{f}_j^{(i)} = \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_j$

$\boldsymbol{f}_{\frac{n}{d}+2} = \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+2}, \vec{\boldsymbol{z}}_0 \rangle - \boldsymbol{c}\boldsymbol{t}_{\frac{n}{d}+2}$

$\boldsymbol{f}_{\frac{n}{d}+3} = \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+3}, \vec{\boldsymbol{z}}_0 \rangle - \boldsymbol{c}\boldsymbol{t}_{\frac{n}{d}+3}$

$\sum_{i=0}^{k-1} \sum_{j=1}^{n/d} \boldsymbol{\alpha}_{in/d+j} \sigma^{-i} \left(\boldsymbol{f}_j^{(i)} (\boldsymbol{f}_j^{(i)} + \sigma^i(\boldsymbol{c}))(\boldsymbol{f}_j^{(i)} - \sigma^i(\boldsymbol{c})) \right) + \boldsymbol{f}_{\frac{n}{d}+2} + \boldsymbol{c}\boldsymbol{f}_{\frac{n}{d}+3} \overset{?}{=} \boldsymbol{v}$

For $\mu = 0, \ldots, k-1$:

$\quad h_\mu \overset{?}{=} 0$

$\quad A^T \vec{\gamma}_\mu = \mathsf{NTT}(\boldsymbol{\psi}_1^{(\mu)}) \| \cdots \| \mathsf{NTT}(\boldsymbol{\psi}_{n/d}^{(\mu)})$

$\boldsymbol{\tau} = \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(\sum_{j=1}^{n/d} d\boldsymbol{\psi}_j^{(\mu)} \boldsymbol{t}_j - \langle \vec{u}, \vec{\gamma}_\mu \rangle \right)$

For $i = 0, \ldots, k-1$:

$\quad \sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sum_{j=1}^{n/d} \sigma^\nu \left(d\boldsymbol{\psi}_j^{(\mu)} \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{z}}_{i-\nu \bmod k} \rangle \right) + \langle \vec{\boldsymbol{b}}_{n/d+1}, \vec{\boldsymbol{z}}_i \rangle$

$\quad\quad \overset{?}{=} \boldsymbol{v}'_i + \sigma^i(\boldsymbol{c})(\boldsymbol{\tau} + \boldsymbol{t}_{n/d+1} - \boldsymbol{h})$

Fig. 4. Verification equations for Figure 3.

Proof. Completeness. It follows by careful inspection that the verification equations are always true for the messages sent by $\mathcal{P}$.

Zero-Knowledge. We can simulate a non-aborting transcript between the honest prover and the honest verifier in the following way. First, in a non-aborting transcript the vectors $\vec{\boldsymbol{z}}_i$ are independently uniformly random in $]-(\delta_1 - T), \delta_1 - T[^{(\lambda+\kappa+n/d+3)d}$ and also independent from $\boldsymbol{c}\vec{\boldsymbol{r}}$. So the simulator can just sample $\vec{\boldsymbol{z}}_i \overset{\$}{\leftarrow}]-(\delta_1 - T), \delta_1 - T[^{(\lambda+\kappa+n/d+3)d}$ and $\boldsymbol{c} \overset{\$}{\leftarrow} C$. The polynomial $\boldsymbol{h}$ is such that $h_0 = \cdots = h_{k-1} = 0$ in honest transcripts and the other coefficients are uniformly random because of the additive term $\boldsymbol{g}$. Hence, the simulator samples $\boldsymbol{h} \overset{\$}{\leftarrow} \{\boldsymbol{h} \in \mathcal{R}_q \mid h_0 = \cdots = k_{k-1} = 0\}$. Then, the challenges $\boldsymbol{\alpha}_i \in \mathcal{R}_q$ and $\vec{\gamma}_\mu \in \mathbb{Z}_q^m$ are independently uniformly random and the simulator samples them in this way. Next, all the commitment messages $\vec{\boldsymbol{t}}_0$, $\boldsymbol{t}_j$, $j = 1, \ldots, n/d+3$, are computationally indistinguishable from uniformly random polynomials if MLWE_λ is hard since the randomness vector $\vec{\boldsymbol{r}}$ is statistically independent from the $\vec{\boldsymbol{z}}_i$. In fact, they include independent Module-LWE samples. So the simulator can just take uniformly random $\vec{\boldsymbol{t}}_0 \in \mathcal{R}_q^\kappa$, $\boldsymbol{t}_j \in \mathcal{R}_q$. Now, in an honest transcript, the remaining messages $\vec{\boldsymbol{w}}_i$, $\boldsymbol{v}$, $\boldsymbol{v}'_i$ are all uniquely determined by the verification equations because of completeness. We see that if the simulator computes these messages so that the verification equations become true, then the resulting transcript is indistinguishable from an honest transcript.

Soundness. First the extractor opens the commitments $\boldsymbol{t}_j$, $j = 1, \ldots, n/d+3$. From [ALS20, Theorem 4.4], unless $\mathcal{E}$ has found a $\mathsf{MSIS}_{\kappa, 8d\beta}$ solution, the extractor can compute vectors $\vec{\boldsymbol{y}}_i^*$ and $\vec{\boldsymbol{r}}^*$ such that for every accepting transcript with first messages $\boldsymbol{t}_j$, $\vec{\boldsymbol{w}}_i$, it holds $\boldsymbol{z}_i = \vec{\boldsymbol{y}}_i^* + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}^*$. The expected runtime for this is equal to the runtime given in the theorem statement. Then let $\check{\boldsymbol{s}}_j^*$, $\boldsymbol{g}^*$, $\boldsymbol{m}_{n/d+2}^*$ and $\boldsymbol{m}_{n/d+3}^*$ be the extracted messages, which are such that

$$
\begin{aligned}
\boldsymbol{t}_j &= \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{r}}^* \rangle + \check{\boldsymbol{s}}_j^* \quad \text{for } j = 1, \ldots, n/d, \\
\boldsymbol{t}_{\frac{n}{d}+1} &= \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+1}, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{g}^*, \\
\boldsymbol{t}_{\frac{n}{d}+2} &= \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+2}, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}_{\frac{n}{d}+2}^*, \\
\boldsymbol{t}_{\frac{n}{d}+3} &= \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+3}, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}_{\frac{n}{d}+3}^*.
\end{aligned}
$$

Now substituting these expressions into $\boldsymbol{f}_j^{(i)}$, $\boldsymbol{f}_{\frac{n}{d}+2}$, $\boldsymbol{f}_{\frac{n}{d}+3}$ as computed in the verification algorithm gives

$$\begin{aligned}\boldsymbol{f}_j^{(i)} &= \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}_i^* \rangle - \sigma^i(\boldsymbol{c})\check{\boldsymbol{s}}_j^*,\\ \boldsymbol{f}_{\frac{n}{d}+2} &= \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+2}, \vec{\boldsymbol{y}}_0^* \rangle - \boldsymbol{c}\boldsymbol{m}_{\frac{n}{d}+2}^*\\ \boldsymbol{f}_{\frac{n}{d}+3} &= \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+3}, \vec{\boldsymbol{y}}_0^* \rangle - \boldsymbol{c}\boldsymbol{m}_{\frac{n}{d}+3}^*.\end{aligned}$$

Next, the verification equation in Line 9 of the verification algorithm reads

$$\begin{aligned}&\left(\sum_{i=0}^{k-1}\sum_{j=1}^{n/d} \boldsymbol{\alpha}_{in/d+j}\sigma^{-i}\left(\langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}_i^*\rangle^3\right) + \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+2}, \vec{\boldsymbol{y}}_0^*\rangle - \boldsymbol{v}\right)\\ &- \boldsymbol{c}\left(\sum_{i=0}^{k-1}\sum_{j=1}^{n/d} \boldsymbol{\alpha}_{in/d+j}\sigma^{-i}\left(3\langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}_i\rangle^2\check{\boldsymbol{s}}_j^*\right) - \langle \vec{\boldsymbol{b}}_{\frac{n}{d}+3}, \vec{\boldsymbol{y}}_0^*\rangle + \boldsymbol{m}_{\frac{n}{d}+2}^*\right)\\ &+ \boldsymbol{c}^2\left(\sum_{i=0}^{k-1}\sum_{j=1}^{n/d} \boldsymbol{\alpha}_{in/d+j}\sigma^{-i}\left(\langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}_i^*\rangle(3(\check{\boldsymbol{s}}_j^*)^2 - \mathbf{1})\right) - \boldsymbol{m}_{\frac{n}{d}+3}^*\right)\\ &- \boldsymbol{c}^3\left(\sum_{i=0}^{k-1}\sum_{j=1}^{n/d} \boldsymbol{\alpha}_{in/d+j}\sigma^{-i}\left(\check{\boldsymbol{s}}_j^*(\check{\boldsymbol{s}}_j^* - \mathbf{1})(\check{\boldsymbol{s}}_j^* + \mathbf{1})\right)\right) = \mathbf{0}.\end{aligned}$$

If we assume that $\check{\boldsymbol{s}}_j^*(\check{\boldsymbol{s}}_j^* - \mathbf{1})(\check{\boldsymbol{s}}_j^* + \mathbf{1}) \neq 0$ for some j, then following the same argument as in [ALS20, Theorem 5.1], the success probability of the prover must be bounded by

$$\varepsilon \leq \sum_{i=0}^{k}\binom{k}{i}\left(\frac{1}{q}\right)^i\left(1-\frac{1}{q}\right)^{k-i}2^{k-i}q^ip^k < (3p)^k.$$

This is not the case and therefore $\vec{s}_j^* = \mathsf{NTT}(\check{\boldsymbol{s}}_j^*) \in \{-1,0,1\}^d$ for all j.

Now substituting $\boldsymbol{t}_j$ into τ gives

$$\boldsymbol{\tau} = \sum_{\mu=0}^{k-1}\frac{1}{k}X^\mu\sum_{\nu=0}^{k-1}\sigma^\nu\left(\sum_{j=1}^{n/d}\langle d\boldsymbol{\psi}_j^{(\mu)}\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^*\rangle\right) + \boldsymbol{f}^*.$$

where

$$\boldsymbol{f}^* = \sum_{\mu=0}^{k-1}\frac{1}{k}X^\mu\sum_{\nu=0}^{k-1}\sigma^\nu\left(\sum_{j=1}^{n/d} d\boldsymbol{\psi}_j^{(\mu)}\check{\boldsymbol{s}}_j^* - \langle \vec{u}, \vec{\gamma}_\mu\rangle\right)$$

We know that $f_\mu^* = \langle A\vec{s}^* - \vec{u}, \vec{\gamma}_\mu\rangle$ for $\mu = 0,\ldots,k-1$ and $\vec{s}^* = \vec{s}_j^*\|\cdots\|\vec{s}_j^*$. Next we find from the last verification equations,

$$\left(\sum_{\mu=0}^{k-1}\frac{1}{k}X^\mu\sum_{\nu=0}^{k-1}\sum_{j=1}^{n/d}\sigma^\nu\left(\langle d\boldsymbol{\psi}_j^{(\mu)}\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}}_{i-\nu \bmod k}^*\rangle\right) + \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{y}}^*\rangle - \boldsymbol{v}_i'\right) \tag{9}$$

$$= \sigma^i(\boldsymbol{c})\left(\boldsymbol{f}^* + \boldsymbol{g}^* - \boldsymbol{h}\right). \tag{10}$$

for all $i = 0,\ldots,k-1$. The coefficients of these linear polynomials in $\sigma^i(\boldsymbol{c})$ are independent from $\boldsymbol{c}$ in a random accepting transcript. With the same reasoning as in the proof of Theorem 3.1 it follows that if $A\vec{s}^* \neq \vec{u}$, then

$$\varepsilon < \left(\frac{1}{q}\right)^k + p^k$$

in contradiction to the bound in the statement. Hence $A\vec{s}^* = \vec{u}$.

4.2 Proof Size

We now look at the size of the non-interactive proof outputs via the Fiat-Shamir transform of the protocol in Figure 3. First, note that for the non-interactive proof the messages $\boldsymbol{w}_i$, $\boldsymbol{v}$ and $\boldsymbol{v}_i$ need not be included in the output as they are uniquely determined by the remaining components. Further, the challenges can be generated from a small seed of 256 bits, which itself is generated as the hash of some components. Therefore, the contribution of the challenges to the total proof length is extremely small and thus we neglect it.

As "full-sized" elements of $\mathcal{R}_q$, we have $\vec{\boldsymbol{t}}_0$, $\boldsymbol{t}_j$, and $\boldsymbol{h}$ (in fact, $\boldsymbol{h}$ is missing k coefficients, but that is a negligible consideration). Therefore, we have in total

$$\kappa + n/d + 3 + 1$$

full-sized elements of $\mathcal{R}_q$, which altogether costs

$$(\kappa + n/d + 4)\, d\lceil \log q \rceil \quad \text{bits.}$$

Now, the only remaining part are the vectors $\vec{\boldsymbol{z}}_i$. Since the k vectors $\vec{\boldsymbol{z}}_i$ of length $(\lambda + \kappa + n/d + 3)d$ over $\mathbb{Z}_q$ are bounded by δ_1 in infinity norm, they require

$$k(\lambda + \kappa + n/d + 3)d\lceil \log 2\delta_1 \rceil \quad \text{bits}$$

in the proof. It is easy to see that no coefficient of the product $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ can exceed d for any $0 \le i \le k-1$. Hence, we set $T = d$.

In conclusion, the overall proof length is about

$$(\kappa + n/d + 4)\, d\lceil \log q \rceil + k\,(\lambda + \kappa + n/d + 3)\, d\lceil \log 2\delta_1 \rceil \quad \text{bits,} \tag{11}$$

Proof length optimizations. The size of the non-interactive proof can be reduced with a number of standard techniques. The first techniques are the two compression techniques from the Bai-Galbraith [BG14] and Dilithium [DKL+18] signature schemes. They reduce the size of the masked openings $\vec{\boldsymbol{z}}_i$ and the top part $\vec{\boldsymbol{t}}_0$ of the commitment. As we have mentioned in Section 2.7, the commitment matrix $\boldsymbol{B}_0$ can be decomposed as $\boldsymbol{B}_0 = (\boldsymbol{B}_0', \boldsymbol{I})$, where $\boldsymbol{I}$ is the identity matrix of dimension κ. Then we can similarly decompose $\vec{\boldsymbol{r}} = \vec{\boldsymbol{r}}_1 \parallel \vec{\boldsymbol{r}}_2$ and write

$$\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}} = \boldsymbol{B}_0'\vec{\boldsymbol{r}}_1 + \vec{\boldsymbol{r}}_2.$$

Now, we see that when we give an approximate proof for this equation that proves $\bar{\boldsymbol{c}}\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0'\bar{\vec{\boldsymbol{z}}}_1 + \bar{\vec{\boldsymbol{z}}}_2$, we are essentially proving knowledge of a short vector $\bar{\vec{\boldsymbol{z}}}_1$ such that $\boldsymbol{B}_1\bar{\vec{\boldsymbol{z}}}_1$ is close to $\bar{\boldsymbol{c}}\vec{\boldsymbol{t}}_0$. But this can be achieved in zero-knowledge without sending both masked openings $\vec{\boldsymbol{z}}_1$ and $\vec{\boldsymbol{z}}_2$ as follows. Let $\vec{\boldsymbol{y}}$ be the masking vector for $\vec{\boldsymbol{r}}_1$, $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}_1$, and $\vec{\boldsymbol{w}} = \boldsymbol{B}_0'\vec{\boldsymbol{y}}$. Then decompose $\vec{\boldsymbol{w}}$ as quotient and remainder modulo $\alpha = 2\delta_2 \approx \delta_1$,

$$\vec{\boldsymbol{w}} = \alpha\vec{\boldsymbol{w}}_1 + \vec{\boldsymbol{w}}_0$$

where $\|\vec{\boldsymbol{w}}_0\|_\infty \le \delta_2$. It follows that

$$\boldsymbol{B}_0'\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0 = \alpha\vec{\boldsymbol{w}}_1 + \vec{\boldsymbol{w}}_0 - \boldsymbol{c}\vec{\boldsymbol{r}}_2.$$

Hence, when we keep the remainder $\vec{\boldsymbol{w}}_0$ secret, it can serve as the masking vector for $\vec{\boldsymbol{r}}_2$. Moreover, by rejecting if $\|\vec{\boldsymbol{w}}_0 - \boldsymbol{c}\vec{\boldsymbol{r}}_2\|_\infty \ge \delta_2 - T$, this doesn't leak information about $\vec{\boldsymbol{r}}_2$ and the equation can be checked by the verifier by decomposing $\boldsymbol{B}_0'\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0$. The second compression technique starts with the same observation that it suffices to prove knowledge of a preimage of $\boldsymbol{B}_0'$ that is close to $\bar{\boldsymbol{c}}\vec{\boldsymbol{t}}_0$. When we decompose

$$\vec{\boldsymbol{t}}_0 = \vec{\boldsymbol{t}}_{0,1}2^D + \vec{\boldsymbol{t}}_{0,0},$$

then it is actually sufficient to only send the high bits $\vec{\boldsymbol{t}}_{0,1}$, because the low bits only introduce an additional noise term $\bar{\boldsymbol{c}}\vec{\boldsymbol{t}}_{0,0}$ in the verification equation that can be handled with sending a few hint bits. We defer to the Dilithium paper for the details.

Another optimization we employ is in the calculation of a maximum absolute coefficient in $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$. In our applications, we aim to minimize d and set $d = 128$. Now in this case, a coefficient of $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ is the sum of 128 coefficients with i.i.d. $P(-1) = P(1) = 5/32$ and $P(0) = 22/32$.[6] If we calculate the convolution of this distribution, we find that a coefficient is bigger than 78 in absolute value with probability less than 2^{-114}. Hence, by a union bound the probability that any of the coefficients in $\left(\sigma^0(\boldsymbol{c})\vec{\boldsymbol{r}}, \ldots, \sigma^{k-1}(\boldsymbol{c})\vec{\boldsymbol{r}}\right)$ is bigger than 78 will still be negligibly small. Therefore, we can set $T = 78$ instead of $T = d = 128$.

The previous optimization can be combined with the following optimization. Instead of using the k rotations $\sigma^i(\boldsymbol{c})$ of one dense challenge for the masked openings $\vec{\boldsymbol{z}}_i$, we can write $\boldsymbol{c}$ in the subring generated by X^k and fixed by σ, i.e. $\boldsymbol{c} = \boldsymbol{c}_0 + \boldsymbol{c}_1 X + \cdots + \boldsymbol{c}_{k-1} X^{k-1}$ where the polynomials $\boldsymbol{c}_j$ are sparse with only at most d/k nonzero coefficients. Then all the images $\sigma^i(\boldsymbol{c})$ are just different linear combinations of the $\boldsymbol{c}_j$. So it suffices to use these sparse challenges $\boldsymbol{c}_j$ in the masked openings that are transmitted and the verifier can then recombine them to obtain the masked openings with challenges $\sigma^i(\boldsymbol{c})$ as needed in the protocols.

References

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM Conference on Computer and Communications Security*, pages 2087–2104. ACM, 2017.

ALS20. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 470–499. Springer, 2020.

APS15. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

BCK+14. Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 551–572. Springer, 2014.

BCOS20. Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum snarks for RSIS and RLWE and their applications to privacy. In *PQCrypto*, volume 12100 of *Lecture Notes in Computer Science*, pages 247–267. Springer, 2020.

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT (1)*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019.

BDK+18. Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P*, pages 353–367, 2018.

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, pages 368–385, 2018.

Beu20. Ward Beullens. Sigma protocols for mq, PKP and sis, and fishy signature schemes. In *EUROCRYPT (3)*, volume 12107 of *Lecture Notes in Computer Science*, pages 183–211. Springer, 2020.

BG14. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO (1)*, pages 176–202. Springer, 2019.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503. ACM, 2002.

CvH91. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

DKL+18. 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.

[6] Recall that a coefficient of $\boldsymbol{c}$ is zero with probability 1/2 and a coefficient of $\vec{\boldsymbol{r}}$ is zero with probability 6/16. The probabilities of ± 1 are always equal to each other.

dPLS18. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM CCS*, pages 574–591. ACM, 2018.

dPLS19. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for FHE and ring-lwe ciphertexts. In *Public Key Cryptography (1)*, volume 11442 of *Lecture Notes in Computer Science*, pages 344–373. Springer, 2019.

ESLL19. Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 115–146. Springer, 2019.

ESS+19. Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS*, volume 11464 of *Lecture Notes in Computer Science*, pages 67–88. Springer, 2019.

EZS+19. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *CCS*, pages 567–584. ACM, 2019. Full version at `https://eprint.iacr.org/2019/1287.pdf`.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992.

LLNW17. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based prfs and applications to e-cash. In *ASIACRYPT (3)*, volume 10626 of *Lecture Notes in Computer Science*, pages 304–335. Springer, 2017.

LLNW18. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 700–732. Springer, 2018.

LN17. Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT*, 2017.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.

LS18. Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT (1)*, pages 204–224. Springer, 2018.

LS19. Vadim Lyubashevsky and Gregor Seiler. NTTRU: truly fast NTRU using NTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):180–201, 2019.

Lyu08. Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography - PKC 2008*, pages 162–179, 2008.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166, 2006.

RST01. 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.

Sei18. Gregor Seiler. Faster AVX2 optimized NTT multiplication for ring-lwe lattice cryptography. *IACR Cryptology ePrint Archive*, 2018:39, 2018. http://eprint.iacr.org/2018/039.

Ste93. Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO*, pages 13–21, 1993.

YAZ+19. 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.

Auxiliary Supporting Material

A Boosting Soundness by Going Up

We now present the second method to decrease the soundness error of the protocol from Section 3.1. This method is efficient if there are fewer secret coefficients than the ring dimension, i.e. if $n < d$ such as $n = 32$

and $d = 128$. Then it is better not to choose a completely splitting prime q so that the opening proof has negligible soundness error with only one repetition ($k = 1$). So, assume $q - 1 \equiv 2l \pmod{4l}$ with $l < d$, and $n = l$. In this case, the analysis of the basic protocol from Section 3.1 does not apply directly and we cannot use automorphisms to boost soundness by mapping down to a smaller ring. Instead, we go the other direction. The prime q splits completely in the subring $\mathcal{S}_q = \{p_0 + p_1 X^{d/l} + \cdots + p_{l-1} X^{d-d/l} \in \mathcal{R}_q\} \cong \mathbb{Z}_q[X]/(X^l + 1)$ of $\mathcal{R}_q$. So we choose the secret polynomial $\check{\boldsymbol{s}}$ such that it lies in $\mathcal{S}_q$, which is the case if and only if the NTT vector $\mathsf{NTT}(\check{\boldsymbol{s}})$ lies in the subvector space $\mathbb{Z}_q^l$ of $(\mathbb{F}_{q^{d/l}})^l$. Then $\check{\boldsymbol{s}}$ encodes the l coefficients of $\vec{s}$. Our protocol assumes that there is a proof for this property. This can for example be part of the shortness proof since $\check{\boldsymbol{s}}(\check{\boldsymbol{s}} - \mathbf{1})(\check{\boldsymbol{s}} + \mathbf{1}) = \mathbf{0}$ shows that $\mathsf{NTT}(\check{\boldsymbol{s}})$ even lies in $\{-1, 0, 1\}^l \subset \mathbb{Z}_q^l \subset (\mathbb{F}_{q^{d/l}})^l$. With this setup the basic protocol using $\vec{\gamma} \in \mathbb{Z}_q^m$ proves the linear relation $A\vec{s} = \vec{u}$ with soundness error $1/q$. But now we can let $\vec{\gamma}$ be uniformly random over $\mathbb{F}_{q^{d/l}}$ and directly get negligible soundness error. Indeed, note that by Lemma 2.1,

$$\begin{aligned}
\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle_{\mathbb{F}_{q^{d/l}}} &= \langle \vec{s}, A^T\vec{\gamma} \rangle_{\mathbb{F}_{q^{d/l}}} - \langle \vec{u}, \vec{\gamma} \rangle_{\mathbb{F}_{q^{d/l}}} \\
&= \sum_{j \in \mathbb{Z}_{2l}^\times} \left(\check{\boldsymbol{s}} \mathsf{NTT}^{-1}(A^T\vec{\gamma}) \bmod (X^{d/l} - \zeta^j) \right) - \langle \vec{u}, \vec{\gamma} \rangle_{\mathbb{F}_{q^{d/l}}} \\
&= \frac{1}{l} \sum_{j \in \mathbb{Z}_{2l}^\times} (\boldsymbol{f} \bmod (X^{d/l} - \zeta^j)) = f_0 + f_1 X + \cdots + f_{d/l-1} X^{d/l-1},
\end{aligned}$$

where the scalar product is over the finite field $\mathbb{F}_{q^{d/l}}$ and the polynomial $\boldsymbol{f} \in \mathcal{R}_q$ is defined by $\boldsymbol{f} = \check{\boldsymbol{s}} \mathsf{NTT}^{-1}(lA^T\vec{\gamma}) - \langle \vec{u}, \vec{\gamma} \rangle$. The protocol is given in Figure 5.

B Applications

B.1 Proving Knowledge of LWE Secrets

As also considered in [BLS19], the first application of our proofs is to prove knowledge of secrets in LWE samples. For a fair comparison, we consider the same setting as in [BLS19]. That is, for $n = 2048$, we want to prove knowledge of a ternary vector $\vec{s} \in \{-1, 0, 1\}^n$ such that

$$\vec{u} = (A' \,\|\, I_m) \cdot \vec{s} \pmod{q},$$

where I_m is the m-dimensional identity matrix, $A' \in \mathbb{Z}_q^{m \times (n-m)}$ is a public matrix chosen uniformly at random and q is a modulus of about 32 bits (i.e., $\log q \approx 32$). Note that $\vec{s}$ here corresponds to the concatenation of a secret vector and an error vector of 1024 dimension each in the usual LWE setting. Let us denote $A = (A' \,\|\, I_m)$. This setting is now precisely the one of the protocol in Figure 1 with $\vec{u} = A\vec{s} \bmod q$, $n = 2048$ and $q \approx 2^{32}$.

First, for about 128 bits of security we set $k = 128/\log q = 4$. Then, to optimize the proof length, we need to set $d = \dim(\mathcal{R}_q)$ as small as possible. This is due to the fact that regardless of what level of security is desired, each "garbage term", namely $\boldsymbol{t}_{n/d+1}, \ldots, \boldsymbol{t}_{n/d+3}, \boldsymbol{h}$, requires $d \log q$ bits of storage. Using Lemma 2.2, the smallest possible d we can choose is 128, thus we set $d = 128$. Next, we need to set the width δ_1 of the masking vectors $\vec{\boldsymbol{y}}_i$ to be large enough as to achieve an acceptable rejection rate. We incorporate the optimization techniques from Section 4.2. In particular, we have $T = 32$ for the bound on the secrets $\boldsymbol{c}_j \vec{\boldsymbol{r}}_1$ in the masked openings. With this we find that $\delta = 2^{18}$ gives an expected number of 18.87 rejections. For the parameters in the compression techniques we chose $\delta_2 = (q-1)/2^{15} \approx 2^{17}$ and $D = 14$. The remaining task is to choose λ and κ to make M-LWE and M-SIS hard in practice against known attacks.

As in prior works (cf. [ESS+19, ESLL19, BLS19]), we measure the hardness of these problems in terms of root Hermite factor δ, aim for $\delta \approx 1.0043$ and follow an estimation strategy as in the recent works [ESS+19, ESLL19], where the authors also aimed for about 128-bit security. Particularly, we use the "LWE estimator" in [APS15] and the methodology and scripts from [DKL+18] to estimate the concrete SIS security in the infinity norm. We can reach the desired security level with $\lambda = 10$ and $\kappa = 9$ for a root Hermite factor

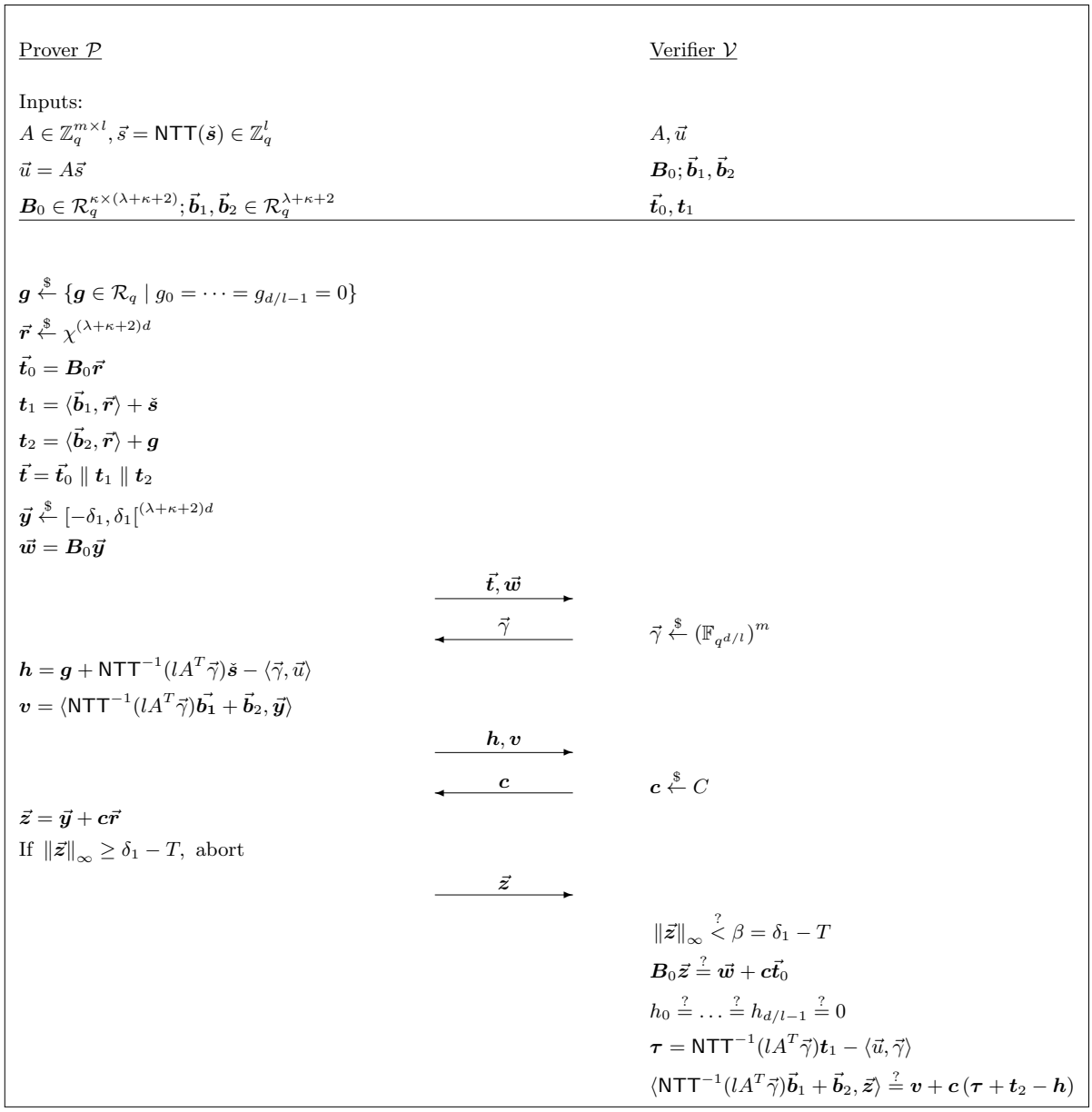

Fig. 5. Simple proof of unstructured linear relations among $l \mid d$ committed integers. The prime q is such that $q \equiv 2l \pmod{4l}$ and hence splits into l prime ideals in the ring $\mathcal{R}$.

(for both SIS and LWE) of $\delta \approx 1.0043$. An advantage of our construction here over [BLS19] is that setting of overall SIS/LWE dimension (which needs to be a multiple of d) is more flexible as we can use a relatively small d of 128. Finally, with these parameters we obtain a proof size of 47 KB. As a result, we achieve an improvement of more than $8\times$ over the proposal in [BLS19] in terms of proof length. Our proof system is fast enough so that a higher rejection rate than 18.37 would be acceptable in practice. For example, $\delta_1 = 2^{17}$ and $\delta_2 \approx 2^{16}$ would result in an expected number of 356.38 rejections. But the proof size would only go down to 45.3 KB, so this small reduction is not worth the much larger running time.

B.2 Proof of Plaintext Knowledge

In a proof of plaintext knowledge (also called a verifiable encryption), the goal is to produce a ciphertext and a zero-knowledge proof such that the decryption of a valid ciphertext is guaranteed to yield a plaintext known by the prover.

The only lattice-based verifiable encryption scheme with a satisfactory practical efficiency is presented in [LN17]. Although this proposal is very efficient in practice, it has some undesirable properties. First, the guarantee on the message $\vec{\boldsymbol{m}}'$ decrypted from a valid ciphertext is *relaxed* in a way that $\vec{\boldsymbol{m}}'$ only satisfies an "approximate" lattice relation. Second, the running time of the decryption algorithm is dependant on the running time of the prover and only the *expected* number of decryption tries is theoretically investigated.

Our proofs from previous sections can help mitigate these drawbacks at the cost of larger proofs. However, unlike the other previous approaches such as [YAZ+19] that can provide an *exact* proof of plaintext knowledge, we believe our results are of practical relevance.

Let us first recall a Module-LWE encryption scheme similar to Kyber [BDK+18] for a message $\boldsymbol{m} \in \mathcal{R}_p$ for $p \in \mathbb{Z}^+$. The secret keys are sampled as $\vec{\boldsymbol{s}}_1, \vec{\boldsymbol{s}}_2 \xleftarrow{\$} S_1^\ell$, where S_1 is the set of polynomials in $\mathcal{R}_q$ with infinity norm at most 1, and the public keys are $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{\ell \times \ell}$ and $\vec{\boldsymbol{t}} = \boldsymbol{A}\vec{\boldsymbol{s}}_1 + \vec{\boldsymbol{s}}_2$. An encryption $(\vec{\boldsymbol{v}}, \boldsymbol{w})$ of a plaintext $\boldsymbol{m} \in \mathcal{R}_p$ satisfies

$$\begin{pmatrix} \vec{\boldsymbol{v}} \\ \boldsymbol{w} \end{pmatrix} = \begin{pmatrix} p\boldsymbol{A}^\top & p\boldsymbol{I}_\ell & 0\,0 \\ p\vec{\boldsymbol{t}}^\top & 0^{\ell\times\ell} & p\;1 \end{pmatrix} \cdot \begin{pmatrix} \vec{\boldsymbol{r}} \\ \vec{\boldsymbol{e}} \\ \boldsymbol{e}' \\ \boldsymbol{m} \end{pmatrix} \mod q, \tag{12}$$

where $\vec{\boldsymbol{r}}, \vec{\boldsymbol{e}} \xleftarrow{\$} S_1^\ell$, $\boldsymbol{e}' \xleftarrow{\$} S_1$ and $\boldsymbol{I}_\ell$ is the $\ell \times \ell$ identity matrix over $\mathcal{R}$. The decryption in this case works by computing

$$\boldsymbol{m} = \boldsymbol{w} - \vec{\boldsymbol{s}}_1^\top \vec{\boldsymbol{v}} \mod q \mod p.$$

For a successful decryption, we require

$$q/2 > \left\| p(\vec{\boldsymbol{s}}_2^\top \vec{\boldsymbol{r}} + \boldsymbol{e}' - \vec{\boldsymbol{s}}_1^\top \vec{\boldsymbol{e}}) + \boldsymbol{m} \right\|_\infty. \tag{13}$$

For simplicity, we consider $\|\boldsymbol{m}\|_\infty = 1$, i.e., $p = 3$. It is easy to adjust also to the setting where $\boldsymbol{m}$ is a binary polynomial.

What we need now is to construct a non-interactive protocol that proves knowledge of $(\vec{\boldsymbol{r}}, \vec{\boldsymbol{e}}, \boldsymbol{e}', \boldsymbol{m})$ with $\|\vec{\boldsymbol{r}}\|_\infty = \|\vec{\boldsymbol{e}}\|_\infty = \|\boldsymbol{e}'\|_\infty = \|\boldsymbol{m}\|_\infty = 1$ that satisfies the relation in (12).

If we expand the matrix in the middle of the relation (12) to its representative matrix over $\mathbb{Z}_q$ and denote it by A, and denote the concatenated coefficient vector of $(\vec{\boldsymbol{r}}, \vec{\boldsymbol{e}}, \boldsymbol{e}', \boldsymbol{m})$ by $\vec{s}$, then we again end up with a relation suitable for the protocol in Figure 1. In this case $\vec{s} \in \mathbb{Z}_q^{(2\ell+2)d}$, i.e., $n = (2\ell+2)d$. As in the previous application, let us consider the setting of $q \approx 2^{32}$ (i.e., $\log q = 32$ and $k = 4$) and $d = 128$. From the previous section, we know that a module rank of $\ell = 10$ for the M-LWE encryption would be sufficient with $q \approx 2^{32}$ and $d = 128$. As a result, we get $n = 2816$, which is close to the value of $n = 2048$ in the previous section. The same module ranks of $\lambda = 10$ and $\kappa = 9$ suffice for the zero-knowledge proof with $\delta \approx 1.0045$ in this case as well.

For this parameter setting, correctness of decryption (i.e., the inequality in (13)) is easily satisfied. Plugging in this parameter setting into (11) (with the described optimizations), we end up with a proof length of 60.89 KB.

B.3 Other Applications

The two applications from Section B show how effective our new techniques are. There are actually various other applications, where our unstructured linear equation proof and our techniques can be useful. Some examples include group signatures [CvH91], ring signatures [RST01], *exact* range proofs, cryptographic

accumulators and proof of message-signature pairs. These examples are all studied in [YAZ+19], where each of them build mainly on a zero-knowledge proof of a relation similar to that of our unstructured linear equation proof. Since this core proof can be realized more efficiently in practice using our novel techniques, we expect more efficient applications to follow.

Particularly, we believe that our zero-knowledge proofs and techniques can be useful in more efficient group signatures that do not rely on *relaxed* zero-knowledge proofs as in [dPLS18, EZS+19]. Although these two works [dPLS18, EZS+19] offer relatively efficient constructions in practice, they have certain drawbacks. More specifically, the opening algorithm in [dPLS18] relies on the decryption algorithm of the verifiable encryption in [LN17], and therefore its worst-case running time for adversarially-generated group signatures is not clear. This case of opening adversarially-generated signatures is not at all supported in [EZS+19], and the group public key length in [EZS+19] grows linearly in the group size, rendering the scheme unsuitable for large groups. Therefore, extending of our techniques here to build a group signature seems to be an interesting future research direction.

C Implementation

We have implemented our proof system for our benchmark application of proving LWE-samples in dimension 1024 modulo $q = 1073479681$. Our implementation achieves a prover runtime of 3.52 ms and a verifier runtime of 0.4 ms. These runtimes are the medians of 500 executions each on a single core of an Intel Skylake CPU running at 3.5 GHz. We used many of the implementation techniques that have been used to speed up the Kyber [BDK+18], Dilithium [DKL+18] and NTTRU [LS19] lattice-based schemes. The software is optimized for x86 CPUs supporting the AVX2 instruction set and especially all of the polynomial arithmetic over $\mathcal{R}_q$ is fully vectorized. Most parts of the AVX2 code are written using C intrinsics, while the AVX2 NTT is implemented in assembly language. As in many lattice-based construction based on the Module-LWE and Module-SIS problems, among the most time-consuming tasks are the expansion of the commitment matrices $\boldsymbol{B}_0$ and $\vec{\boldsymbol{b}}_i$ and polynomial multiplication.

The matrix $\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa \times (\lambda+\kappa+n/d+3)}$ is chosen with the structure $\boldsymbol{B}_0 = (\boldsymbol{I}_\kappa \mid \boldsymbol{B}_0')$ with $\boldsymbol{B}_0' \in \mathcal{R}_q^{\kappa \times (\lambda+n+2)}$. Likewise, the vectors $\vec{\boldsymbol{b}}_i$, $i = 1, \ldots, n/d+3$, are of the form $\vec{\boldsymbol{b}}_i = \vec{\boldsymbol{0}}_\kappa \parallel \vec{\boldsymbol{e}}_i \parallel \vec{\boldsymbol{b}}_i'$ where $\vec{\boldsymbol{e}}_i$ is the i-th standard basis vector of length $n/d+3$ and $\vec{\boldsymbol{b}}_i' \in \mathcal{R}_q^\lambda$. We expand all uniformly random polynomials in $\boldsymbol{B}_0$ and $\vec{\boldsymbol{b}}_i$ from the same 32-bit seed using the output stream of AES-256 in counter mode, where we use a fresh nonce for each polynomial. Our vectorized rejection sampling implementation samples up to 8 uniformly random coefficients simultaneously and the AES implementation is based on the AES-NI instructions.

For fast polynomial multiplication, whenever possible, we keep polynomials in the NTT basis representation to save NTT operations, and also sample uniformly random polynomials directly in the NTT basis. Specifically, uniform polynomials that are sent as part of the proofs are sent in the NTT basis. Our code also includes fast in-place implementations of the Galois automorphisms that operate in the NTT basis.

The AVX2 optimized NTT implementation operates on dense vectors of 8 32-bit coefficients. It uses the modified Montgomery reduction algorithm from [Sei18] and [LS19] to reduce intermediate 64 bit products. This allows to handle dense vector registers more efficiently and saves multiplication instructions that lie on the critical path. At the core of the code lies a vectorized interleaved butterfly implementation that processes 32 coefficients in 4 vector registers at a time. The instruction for parallel Montgomery and single-word reductions in a butterfly operation are very carefully scheduled, taking into account the front-end decoding throughput from the L1 cache and the back-end execution resources. A full NTT execution runs in 540 cycles on an Intel Skylake core.

For the hash function that is needed in the Fiat-Shamir transform to obtain all the challenge polynomials, we use SHAKE128.