Shorter Lattice-Based Zero-Knowledge Proofs via One-Time Commitments *

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

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

Abstract. There has been a lot of recent progress in constructing efficient zero-knowledge proofs for showing knowledge of an $\vec{s}$ with small coefficients satisfying $\boldsymbol{A}\vec{s} = \vec{t}$. For typical parameters, the proof sizes have gone down from several megabytes to a bit under 50KB (Esgin et al., Asiacrypt 2020). These are now within an order of magnitude of the sizes of lattice-based signatures, which themselves constitute proof systems which demonstrate knowledge of something weaker than the aforementioned equation. One can therefore see that this line of research is approaching optimality. In this paper, we modify a key component of these proofs, as well as apply several other tweaks, to achieve a further reduction of around 30% in the proof output size. We also show that this savings propagates itself when these proofs are used in a general framework to construct more complex protocols.

1 Introduction

Zero-knowledge proofs and commit-and-prove protocols form the foundations of virtually all privacy-based protocols. In preparing for the (eventual) coming of quantum computing, there has been a lot of focus in recent years on building such protocols based on quantum-safe assumptions. Quantum-safe PCP / IOP schemes whose security is just based on the collision-resistance of cryptographic hash functions have existed for decades, and there has been a lot of recent work around them. The main feature of these constructions is that their outputs are sublinear in the statement size. The main downside is that they can be very slow and memory intensive. Furthermore, there is a lower bound of around 100KB for proofs of statements that have small size. So it's quite likely that they are not the best solution for all scenarios.

In the last few years, new techniques emerged in lattice cryptography that made it a promising foundation for quantum-safe privacy-based protocols. These techniques have been used for blockchain applications [EZS+19], verifiable random functions [EKS+20], and for proofs of arithmetic statements [LNS20]. In all of these scenarios, lattice schemes appear to be the best solutions available. For example, a commit-and-prove protocol for integer products is significantly smaller and several orders of magnitude faster than other quantum-safe solutions. These results show that lattices might eventually be very reasonable substitutes for the classical cryptography that is currently embedded in those protocols.

At the core of many of the recent privacy-based protocols is the BDLOP commitment scheme [BDL+18] which is able to commit to an arbitrary message vector $\vec{\boldsymbol{m}}$ modulo q. In [BDL+18], it was also shown how one can prove knowledge of $\vec{\boldsymbol{m}}$ by only proving knowledge of the commitment randomness $\vec{\boldsymbol{r}}$. The proof technique was essentially using the "Fiat-Shamir with Aborts" [Lyu09] framework to keep the proofs small and avoid leaking any information about $\vec{\boldsymbol{r}}$. If one uses the Gaussian rejection sampling procedure [Lyu12], then the magnitude of each coefficient of the $\vec{\boldsymbol{r}}'$

* This is the full version of [LNS21] presented at PKC 2021.

output in the proof is around $12 \cdot \kappa \|\vec{\boldsymbol{r}}\|$; where the κ is some constant that comes from the challenge and the 12 comes from a Gaussian tail bound needed for (statistical) zero-knowledge.

The increased coefficient size means that the proof is noticeably larger than the randomness itself (and gives rise to a vicious cycle of having to increase the modulus and dimension of the underlying lattice). It nevertheless seems necessary because leaking some information about the randomness can be dangerous. For example, if one were to repeatedly perform proofs of knowledge for the same commitment and leaking something about the same randomness each time, eventually the entire randomness could be recovered by even a passive observer.

1.1 One-time Commitments

If one looks closer at how the BDLOP commitment is being used in many of the privacy-based protocols, one would notice that the scheme is used to commit to some intermediate value, give a proof-of-knowledge of the value (i.e. proof of knowledge of the commitment randomness), and then discards the commitment. So only *one* proof of knowledge is performed. In this case, it's not immediately clear whether some leakage of the randomness vector is problematic. Still, it seems somewhat risky to only perform a minimal perturbation on the randomness and hope that this is good enough for LWE to remain secure. Instead of relying completely on heuristic security, it would be good to have a technique which lowers the proof size, and at the same time concretely allows one to understand exactly how the LWE problem is affected by the leakage.

For efficiency purposes, the BDLOP commitment instantiation is only computationally hiding (i.e. based on LWE), so one cannot use prior techniques to prove that enough entropy remains in the randomness after the leakage. While there are techniques that show that LWE-based encryption schemes can tolerate some leakage, the results are asymptotic and it's unclear what the actual practical implication of the leakage is [DGK+10,GKPV10,BD20]. There has also been some work examining the practical aspects of leakage in the Ring-LWE setting [DDGR20], and the security of the scheme is barely affected when the leakage is small.

We show that a particular rejection sampling strategy information-theoretically leaks just one bit of randomness, and allows us to exactly quantify what this leakage is. More specifically, in addition to the public LWE samples, there is also a short public vector $\vec{\boldsymbol{z}}$, and we require that the whoe LWE secret (i.e. the secret concatenated with the error vector) has a non-negative inner product with it. Because the LWE secret is uniformly distributed around 0, the probability that the inner product will be non-negative is greater than 1/2, and so this extra restriction loses (less than) one bit of entropy.

We observe that the leakage essentially transforms the LWE problem instance into (a much less leaky version of) an extended-LWE one, which was shown to be equivalent to LWE in [AP12]. The decisional extended-LWE problem asks to distinguish between the distributions $(\boldsymbol{B}, \boldsymbol{B}\vec{\boldsymbol{r}}, \vec{\boldsymbol{z}}, \langle \vec{\boldsymbol{r}}, \vec{\boldsymbol{z}} \rangle)$ and $(\boldsymbol{B}, \vec{\boldsymbol{u}}, \vec{\boldsymbol{z}}, \langle \vec{\boldsymbol{r}}, \vec{\boldsymbol{z}} \rangle)$, where $\vec{\boldsymbol{r}}$ is sampled from the secret/noise domain of the LWE problem, $\vec{\boldsymbol{u}}$ is uniformly random, and $\vec{\boldsymbol{z}}$ is sampled from some efficiently sampleable distribution. One caveat is that for efficiency, we use a structured matrix $\boldsymbol{B}$, and the proof in [AP12] does not carry over to our setting. [3]

Furthermore, it's clear that there is an equivalence between the *search* versions of extended Ring-LWE and Ring-LWE – and so a separation between the decisional versions would be quite interesting

[3] For the readers familiar with the *sample-preserving* reduction between search and decisional LWE problems [MM11], the underlying obstacles for that reduction and the extended-LWE reduction not carrying over to the Ring-LWE setting are similar.

because all the best current lattice algorithms work by trying to solve the search problem. In short, we do not believe that this one bit of leakage has any practical consequences on the hardness of the Ring-LWE problem. It would actually be interesting if this assumption found even more applications, beside the original ones (e.g. [OPW11]) that inspired the Extended-LWE assumption, for improving the efficiency of lattice schemes. [4]

1.2 Technical Overview

The last prover move of a Schnorr-type Σ-protocol is a value $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \vec{\boldsymbol{v}}$, where $\vec{\boldsymbol{y}}$ is a "masking" vector that the prover created during the first move and $\vec{\boldsymbol{v}}$ is the combination of the secret and the challenge (usually their product, but this is unimportant for our purposes and we can consider the whole $\vec{\boldsymbol{v}}$ to be secret). If we would like $\vec{\boldsymbol{z}}$ to have small norm, then we cannot choose the coefficients of $\vec{\boldsymbol{y}}$ to be so large that the sum $\vec{\boldsymbol{y}} + \vec{\boldsymbol{v}}$ statistically hides $\vec{\boldsymbol{v}}$. Instead we use rejection sampling to force the distribution of $\vec{\boldsymbol{z}}$ to be independent of $\vec{\boldsymbol{v}}$. In particular, if we would like the distribution of the output to be f, while the distribution of $\vec{\boldsymbol{z}}$ is g, then one should output $\vec{\boldsymbol{z}}$ with probability $f(\vec{\boldsymbol{z}})/(M \cdot g(\vec{\boldsymbol{z}}))$, where M is some positive integer set so that this ratio is never larger than 1. Since $1/M$ is the probability that something will be output, we also want to have M as small as possible. Combining these two requirements, it's easy to see that M should be set to $\max_{\vec{\boldsymbol{z}}}(g(\vec{\boldsymbol{z}})/f(\vec{\boldsymbol{z}}))$.

If we follow [Lyu12] where the target distribution f is a discrete Gaussian with standard deviation $\mathfrak{s}$ and the distribution g is a shifted Gaussian (by $\vec{\boldsymbol{v}}$) with the same standard deviation, then via [Lyu12, Lemma 4.5], the value for M is derived as

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

To obtain the inequality, one uses a standard 1-dimensional tail bound for the inner product of a discrete Gaussian with arbitrary vector (c.f. [Lyu12, Lemma 4.3]). And in fact, it is this tail bound that contributes the most to M. For example, if we would like to have $M = \exp(1)$, then we would need to set $\mathfrak{s} \approx 12\|\vec{\boldsymbol{v}}\|$. On the other hand, if we knew that $\langle\vec{\boldsymbol{z}},\vec{\boldsymbol{v}}\rangle \geq 0$, then we could set $\mathfrak{s} \approx \|\vec{\boldsymbol{v}}\|/\sqrt{2}$, a decrease of around a factor of 17. So the intuition is for the prover to throw away the potential outputs $\vec{\boldsymbol{z}}$ that have the aforementioned inner product be negative-valued. The effect of this is that it leaks information about $\vec{\boldsymbol{z}}$ – in particular, the fact that its inner product with the secret is positive.

Interestingly, the new value of $\mathfrak{s}$ is identical to what one would get by using bimodal rejection sampling, as in the BLISS signature scheme [DDLL13]. We cannot directly apply the technique from that paper because it required that the public value $\boldsymbol{B}_0$ be set up such that $\boldsymbol{B}_0\vec{\boldsymbol{r}} = q \pmod{2q}$. Intuitively, with this setup, leaking the sign of the secret $\vec{\boldsymbol{r}}$ doesn't matter because both $\vec{\boldsymbol{r}}$ and $-\vec{\boldsymbol{r}}$ satisfy the equality. Since we do not have this setup, the bimodal technique from BLISS would leak some information. At this point, we do not see a way to quantify the leakage stemming from the bimodal distribution, unlike the fairly simple leakage that we proposed instead. We should also mention that a recent work [TWZ20] showed how to apply the bimodal technique to the BDLOP commitment scheme without any leakage. Their transformation is not for free, though – it increases the length of the output, and also has a technical subtlety that makes it difficult to apply to our current result. The issue is that a part of our security proof requires that the challenge space has

[4] Indeed, much of the progress in constructions of practical classical cryptography has come from making stronger, but still plausible, assumptions that stem from discrete log or factoring.

Proving	[ENS20]	[LNS20]	**this work**	heuristic masking with no rejection sampling
Knowledge of an LWE Sample	47KB	47KB	**33.3** KB	27 KB
Integer Addition	–	24.8KB	**16.9** KB	13.8 KB
Integer Multiplication	–	40.2KB	**28.2** KB	23 KB

Fig. 1. Proof size comparisons for secure commit-and prove protocols for proving knowledge of a (Module)-LWE sample of total dimension (secret and error vector) 2048, and 128-bit integer addition and multiplication. In the last column we list the proof sizes when only masking the secret vector $\boldsymbol{c}\vec{\boldsymbol{r}}$ with a uniformly random masking vector that has the same infinity norm bound than the secret vector and not performing any rejection sampling. This is possibly still secure, but lacks a security proof at this point.

a particular distribution over $\{-1, 0, 1\}$ (it's necessary for Lemma 2.1, which is crucial for the product proof over fully-splitting rings). In the proof in [TWZ20], on the other hand, the challenges are masked by the prover and so their distribution is possibly determined in an adversarial fashion.

1.3 Applications and Paper Overview

A good benchmark for the progress of the development of lattice-based proof techniques is the simple proof of knowledge of a trinary secret vector $\vec{\boldsymbol{s}}$ satisfying $\boldsymbol{B}\vec{\boldsymbol{s}} = \vec{\boldsymbol{t}}$. Up until a few years ago, for a particular real-world parametrization of the problem ($\boldsymbol{B}$ being a 1024×2048 dimensional matrix over $\mathbb{Z}_q$ for $q \approx 2^{32}$), the smallest proof used "Stern proofs" [Ste93] ported to lattices [LNSW13], and had proof sizes of several megabytes. Switching the proof strategy to showing that a coefficient s_i is trinary by using a commit-and-prove strategy by proving the algebraic equation $(s_i - 1) \cdot s_i \cdot (s_i + 1) = 0$ lowered the proof sizes to around 300KB [YAZ+19,BLS19]. Then even more recently, by utilizing more efficient product proofs [EZS+19,ALS20], it was shown in [ENS20] how to obtain proofs of size under 50KB.

In Section 3, we apply our new technique to the aforementioned benchmark, as well as to the recent framework in [LNS20] for proving integer relations. Appendix B shows how this framework can be further optimised using commitment compression from [DKL+18]. In total, we obtain a size reduction of around 30% for these proofs (see Figure 1). As a side note, the proofs in [ENS20,LNS20] used the uniform distribution instead of discrete Gaussians. The reason was that Gaussians did not provide much of a size reduction (a few kilobytes) and are a bit more cumbersome to work with. But in light of our modified rejection sampling, which only appears to apply to the Gaussian technique, we believe that it is now worthwhile to switch in order to take advantage of the additional size reduction.

To see how far our new rejection sampling procedure is from the (almost) best possible, we also compute the proof size for the case in which we don't do *any* rejection sampling when giving a proof of knowledge of the commitment randomness. In particular, we heuristically mask the secret by adding a random masking vector whose coefficients are as big as those of the vector. This undoubtedly leaks something about the randomness, but it may still be reasonable to hope that the message in the commitment remains hidden because there is still enough entropy in the LWE secret (i.e. the randomness of the commitment).[5] This strategy leads to proof sizes that are around

[5] Even smaller sizes would be of course obtained if one does no masking at all, but then the scheme would be clearly insecure.

20% smaller. We do not see how one can get any sort of security reduction for this approach, but we don't see an attack either. It is therefore an interesting open problem whether this approach can lead to something with a security proof. This would lead to essentially optimal parameters for this approach of creating zero-knowledge proofs.

Since our basic proofs are fairly close to optimal (as far as this line of research goes), we find it worthwhile to give further applications of them. Combined with a few extra tricks we develop along the way, we believe this leads to the shortest current constructions for certain primitives. Section 4 describes how one constructs a verifiable decryption scheme for Kyber [BDK+18], which is a finalist of the NIST PQC competition. This approach can be easily extended to other lattice-based KEM finalists, such as Saber [DKRV18] or NTRU [HPS98]. Complementing the previous section, which requires range proofs in the infinity norm, in Appendix E we further consider range proofs with respect to the Euclidean norm. Concretely, we show how to prove that a vector $\vec{v}$ has $\|\vec{v}\| < a$ for some integer a. In Appendix F, we describe an alternative approach for creating an important ingredient in some proofs – namely, *approximate* range proofs [BL17,BN20,LNS20] – by using bimodal Gaussians [DDLL13] which further increases the efficiency of the aforementioned protocol. We also remark that the latter two sections make use of the improved framework from [LNS20] defined in Section 3.

2 Preliminaries

2.1 Notation

Denote $\mathbb{Z}_p$ to be the ring of integers modulo p. Let q be an odd prime. We write $\vec{v} \in R^m$ to denote vectors over a ring R and matrices over R will be written as regular capital letters M. Define I_n to be the $n \times n$ identity matrix over $\mathbb{Z}_q$. For an integer a, we define a vector $\vec{a} = (a, \ldots, a)$ unless stated otherwise. By default, all vectors are column vectors. We write $\vec{v}||\vec{w}$ for a usual concatenation of $\vec{v}$ and $\vec{w}$ (which is still a column vector). For $\vec{v}, \vec{w} \in \mathbb{Z}_q^k$, $\vec{v} \circ \vec{w}$ is the usual component-wise multiplication. For simplicity, we denote $\vec{u}^2 = \vec{u} \circ \vec{u}$. We write $x \leftarrow S$ when $x \in S$ is sampled uniformly at random from the finite set S and similarly $x \leftarrow D$ when x is sampled according to the distribution D. We write $[n]$ to denote the set $\{1, \ldots, n\}$. Given two functions $f, g : \mathbb{N} \to [0, 1]$, we write $f(\mu) \approx g(\mu)$ if $|f(\mu) - g(\mu)| < \mu^{-\omega(1)}$. A function f is negligible if $f \approx 0$. We write $\mathsf{negl}(n)$ to denote an unspecified negligible function in n.

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

Modular reductions. We define $r' = r \bmod^{\pm} q$ to be the unique element r' in the range $-\frac{q-1}{2} \leq r' \leq \frac{q-1}{2}$ such that $r' \equiv r \pmod q$. We also denote $r' = r \bmod^{+} q$ to be the unique element r' in the range $0 \leq r' < q$ such that $r' \equiv r \pmod q$. When the exact representation is not important, we simply write $r \bmod q$.

Sizes of elements. For an element $w \in \mathbb{Z}_q$, we write $\|w\|_\infty$ to mean $|w \bmod^{\pm} q|$. Define the ℓ_∞ and ℓ_p norms for $\boldsymbol{w} = w_0 + w_1 X + \ldots + w_{d-1} X^{d-1} \in \mathcal{R}$ as follows:

$$\|\boldsymbol{w}\|_\infty = \max_j \|w_j\|_\infty, \quad \|\boldsymbol{w}\|_p = \sqrt[p]{\|w_0\|_\infty^p + \ldots + \|w_{d-1}\|_\infty^p}.$$

If $\vec{\boldsymbol{w}} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m) \in \mathcal{R}^m$, then

$$\|\vec{\boldsymbol{w}}\|_\infty = \max_j \|\boldsymbol{w}_j\|_\infty, \quad \|\vec{\boldsymbol{w}}\|_p = \sqrt[p]{\|\boldsymbol{w}_1\|^p + \ldots + \|\boldsymbol{w}_k\|^p}.$$

By default, we denote $\|\vec{\boldsymbol{w}}\| := \|\vec{\boldsymbol{w}}\|_2$.

2.2 Cyclotomic Rings

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

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

where ζ^j $(j \in \mathbb{Z}_{2l}^\times)$ ranges over all the l primitive $2l$-th roots of unity.

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

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

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

2.3 Automorphisms

The ring $\mathcal{R}_q$ has a group of automorphisms $\mathrm{Aut}(\mathcal{R}_q)$ that is isomorphic to $\mathbb{Z}_{2d}^\times \cong \mathbb{Z}_2 \times \mathbb{Z}_{d/2}$,

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

where σ_i is defined by $\sigma_i(X) = X^i$. Note that for $i \in \mathbb{Z}_{2d}^\times$ and odd j it holds that $(\sigma_i(X - \zeta^j)) = (X - \zeta^{ji^{-1}})$ in $\mathcal{R}_q$ (as ideals), and for $\boldsymbol{f} \in \mathcal{R}_q$,

$$\sigma_i\left(\boldsymbol{f} \bmod (X - \zeta^j)\right) = \sigma_i(\boldsymbol{f}) \bmod \left(X - \zeta^{ji^{-1}}\right).$$

Let k be a divisor of l and $\sigma := \sigma_{2l/k+1} \in \mathsf{Aut}(\mathcal{R}_q)$. Then, we can write

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

2.4 Challenge Space

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

Consider the coefficients of the polynomial $\boldsymbol{c} \bmod (X^{d/l} - \zeta^j)$ for $\boldsymbol{c} \leftarrow C$. Then, all coefficients follow the same distribution over $\mathbb{Z}_q$. Let us write Y for the random variable over $\mathbb{Z}_q$ that follows this distribution. Attema et al. [ALS20] give an upper bound on the maximum probability of Y.

Lemma 2.1. *Let the random variable Y over $\mathbb{Z}_q$ be defined as above. Then for all $x \in \mathbb{Z}_q$,*

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

In particular, [ALS20,ENS20] computed that for $q \approx 2^{32}$, the maximum probability for each coefficient of $c \bmod X^{d/l} - \zeta^j$ is around $2^{-31.4}$. In general, we will call this probability p.

An immediate consequence of Lemma 2.1 is that polynomial $\boldsymbol{c} \leftarrow C$ is invertible in $\mathcal{R}_q$ with overwhelming probability as long as parameters q, d, l are selected so that $q^{-d/l}$ is negligible.

Let k be a divisor of d such that $q^{-kd/l}$ is negligible and set $\sigma = \sigma_{2l/k+1}$. Let us define a probability distribution $\tilde{C}$ over $\mathcal{R}^k$ which first samples $\boldsymbol{c} = \boldsymbol{c}_0 + \boldsymbol{c}_1 X + \ldots + \boldsymbol{c}_{k-1} X^{k-1} \leftarrow C$ and outputs $(\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{k-1})$ where each $\boldsymbol{c}_i$ is defined as $\boldsymbol{c}_i = \sum_{j=0}^{d/k-1} c_{jk+i} X^{jk}$. Clearly, we have

$$\boldsymbol{c} = \sum_{i=0}^{k-1} \boldsymbol{c}_i X^i.$$

2.5 Module-SIS and Module-LWE Problems

Security of the [BDL+18] commitment scheme used in our protocols relies on the well-known computational lattice problems, namely Module-LWE (M-LWE) and Module-SIS (M-SIS) [LS15]. Both problems are defined over $\mathcal{R}_q$.

Definition 2.2 (M-SIS$_{\kappa,m,B}$). *Given $\boldsymbol{A} \leftarrow \mathcal{R}_q^{\kappa \times m}$, the* Module-SIS *problem with parameters $\kappa, m > 0$ and $0 < B < q$ asks to find $\vec{\boldsymbol{z}} \in \mathcal{R}_q^m$ such that $\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{0}}$ over $\mathcal{R}_q$ and $0 < \|\vec{\boldsymbol{z}}\| \leq B$. An algorithm $\mathcal{A}$ is said to have advantage ϵ in solving M-SIS$_{\kappa,m,B}$ if*

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

Definition 2.3 (M-LWE$_{m,\lambda,\chi}$). *The* Module-LWE *problem with parameters $m, \lambda > 0$ and an error distribution χ over $\mathcal{R}$ asks the adversary $\mathcal{A}$ to distinguish between the following two cases: 1) $(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})$ for $\boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \lambda}$, a secret vector $\vec{\boldsymbol{s}} \leftarrow \chi^\lambda$ and error vector $\vec{\boldsymbol{e}} \leftarrow \chi^m$, and 2) $(\boldsymbol{A}, \vec{\boldsymbol{b}}) \leftarrow \mathcal{R}_q^{m \times \lambda} \times \mathcal{R}_q^m$. Then, $\mathcal{A}$ is said to have advantage ϵ in solving M-LWE$_{m,\lambda,\chi}$ if*

$$\begin{aligned} \Big| \Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \lambda};\ \vec{\boldsymbol{s}} \leftarrow \chi^\lambda;\ \vec{\boldsymbol{e}} \leftarrow \chi^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})\right] \\ - \Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{m \times \lambda};\ \vec{\boldsymbol{b}} \leftarrow \mathcal{R}_q^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \vec{\boldsymbol{b}})\right] \Big| \geq \epsilon. \end{aligned} \tag{3}$$

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

2.6 Probability Distributions

For sampling randomness in the commitment scheme that we use, and to define a variant of the Ring Learning with Errors problem, we need to define an error distribution χ^d on $\mathcal{R}$. In this paper we sample the coefficients of the random polynomials in the commitment scheme using the distribution χ on $\{-1, 0, 1\}$ where ± 1 both have probability 5/16 and 0 has probability 6/16. 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.

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

Definition 2.4. *The* discrete Gaussian distribution *on $\mathcal{R}^\ell$ centered around $\vec{\boldsymbol{v}} \in \mathcal{R}^\ell$ with standard deviation $\mathfrak{s} > 0$ is given by*

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

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

2.7 Approximate Range Proofs

Baum and Lyubashevsky [BL17] showed that if $B\vec{s}$ has small coefficients, for a vector $\vec{s}$ over $\mathbb{Z}_q$ and uniformly random binary matrix B, then with high probability $\vec{s}$ must have small coefficients as well. More recently, Lyubashevsky et al. [LNS20] generalise their result for $B\vec{s} + \vec{e}$ where $\vec{e}$ is an arbitrary vector over $\mathbb{Z}_q$. We extend the lemma for the case when B is sampled from a distribution centered at 0. The main advantage of this approach is that the infinity norm of $B\vec{s}$ decreases significantly, which is essential for the rejection sampling.

Concretely, we define the probability distribution $\mathsf{C} : \{-1, 0, 1\} \to [0, 1]$ such that $\Pr_{c \leftarrow \mathsf{C}}[c = 0] = p$ and $\Pr_{c \leftarrow \mathsf{C}}[c = 1] = \Pr_{c \leftarrow \mathsf{C}}[c = -1] = (1 - p)/2$ for some $p \in [0, 1]$. Then, we have the following lemma.

Lemma 2.5. *Let $\vec{s} \in \mathbb{Z}_q^m$ and $\vec{y} \in \mathbb{Z}_q^n$. Then*

$$\Pr\left[\|B\vec{s} + \vec{y}\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty : B \leftarrow \mathsf{C}^{n \times m}\right] \leq \max\{p, 1 - p\}^n.$$

We provide the proof of Lemma 2.5 in Appendix A.

2.8 Commit-and-Prove

Let R_L be a polynomial-time verifiable relation containing (ck, x, w). We will call ck the commitment key, x the statement and w the witness. Also, we define a language L_{ck} as the set of statements x for which there exists a witness w such that $(ck, x, w) \in R_L$.

We define the commit-and-prove functionality similarly as in [EG14,CLOS02] for a relation R_L. Roughly speaking, we want to commit to messages $m_1, \ldots, m_n$ and prove certain statements about them. Therefore, $w = (m_1, \ldots, m_n)$ constitutes a witness for $x \in L_{ck}$.

Formally, a commit-and-prove functionality (CP) consists of four algorithms $CP = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$. We require $\mathsf{Com}, \mathsf{Verify}$ to be deterministic whereas $\mathsf{Gen}, \mathsf{Prove}$ are probabilistic.

- $\mathsf{Gen}(1^\mu)$: Given a security parameter μ, generates a commitment key ck. The commitment key specifies a message space $\mathcal{M}_{ck}$ a randomness space $\mathcal{R}_{ck}$ and commitment space $\mathcal{C}_{ck}$.
- $\mathsf{Com}_{ck}(m; r)$: Given a commitment key ck, a message $m \in \mathcal{M}_{ck}$ and randomness $r \in \mathcal{R}_{ck}$ returns a commitment $c \in \mathcal{C}_{ck}$.
- $\mathsf{Prove}_{ck}\left(x, ((m_1, r_1), \ldots, (m_n, r_n))\right)$: Given a commitment key ck, statement x and commitment openings $m_i \in \mathcal{M}_{ck}, r_i \in \mathcal{R}_{ck}$ and $(ck, x, (m_1, \ldots, m_n)) \in R_L$ returns a proof π.
- $\mathsf{Verify}_{ck}\left(x, c_1, \ldots, c_n, \pi\right)$: Given a commitment key ck, a statement x, a proof π and commitments $c_i \in \mathcal{C}_{ck}$, outputs 1 (accept) or 0 (reject).

Definition 2.6 (Correctness). *The commit-and-prove functionality CP has statistical correctness with correctness error $\rho : \mathbb{N} \to [0, 1]$ if for all adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Gen}(1^\mu); (x, m_1, r_1 \ldots, m_n, r_n) \leftarrow \mathcal{A}(ck); c_i = \mathsf{Com}_{ck}(m_i; r_i); \\ \pi \leftarrow \mathsf{Prove}_{ck}\left(x, ((m_1, r_1), \ldots, (m_n, r_n))\right) : \mathsf{Verify}_{ck}\left(x, c_1, \ldots, c_n, \pi\right) = 0 \end{array}\right] \leq \rho(\mu) \tag{4}$$

where $\mathcal{A}$ outputs $m_i \in \mathcal{M}_{ck}, r_i \in \mathcal{R}_{ck}$ so that $(ck, x, (m_1, \ldots, m_n)) \in R_L$.

Definition 2.7 (Knowledge Soundness). *The commit-and-prove functionality CP is knowledge sound with knowledge error $\epsilon : \mathbb{N} \to [0, 1]$ if for all PPT $\mathcal{A}$ there exists an expected polynomial time extractor $\mathcal{E}$ so that :*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Gen}(1^\mu); (x, c_1, \ldots, c_n, \pi) \leftarrow \mathcal{A}(ck); ((m_1^*, r_1^*) \ldots, (m_n^*, r_n^*)) \leftarrow \mathcal{E}(c_1, \ldots, c_n) : \\ \mathsf{Verify}_{ck}\left(x, c_1, \ldots, c_n, \pi\right) = 1 \wedge ((ck, x, (m_1^*, \ldots, m_n^*)) \notin R_L \vee \exists i, \mathsf{Com}(m_i^*; r_i^*) \neq c_i) \end{array}\right] \tag{5}$$

is less or equal to $\epsilon(\mu)$, where $\mathcal{E}$ outputs $m_i^* \in \mathcal{M}_{ck}$ and $r_i^* \in \mathcal{R}_{ck}$.

In lattice-based zero-knowledge proofs, it is sometimes useful to relax the definition of knowledge soundness by only requiring $r_1^*, \ldots, r_n^* \in \bar{\mathcal{R}}_{ck}$ where $\mathcal{R}_{ck} \subseteq \bar{\mathcal{R}}_{ck}$. However, the definition still makes sense if one can argue that the extracted commitments are still binding. This is what we do by additionally defining notions of *weak opening* (Definition B.3).

The next property is a new notion called *simulatability*. Informally, it means that there exists an efficient simulator $\mathcal{S}$ which can simulate both the commitment generation and the proof at the same time.

Definition 2.8 (Simulatability). *The commit-and-prove functionality CP is simulatable if there exist PPT simulators* SimCom *and* $\mathsf{SimProve}$ *such that for all PPT adversaries $\mathcal{A}$:*

$$\begin{array}{l} \Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Gen}(1^\mu); (x, m_1, \ldots, m_n) \leftarrow \mathcal{A}(ck); r_1, \ldots, r_n \leftarrow \xi; \forall i, c_i = \mathsf{Com}_{ck}(m_i, r_i); \\ \pi \leftarrow \mathsf{Prove}_{ck}(x, (m_1, r_1), \ldots, (m_n, r_n)) : (ck, x, (m_1, \ldots, m_n)) \in R_L \wedge \mathcal{A}(c_1, \ldots, c_n, \pi) = 1 \end{array}\right] \\ \approx \Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Gen}(1^\mu); (x, m_1, \ldots, m_n) \leftarrow \mathcal{A}(ck);, c_1, \ldots, c_n \leftarrow \mathsf{SimCom}_{ck}(x); \\ \pi \leftarrow \mathsf{SimProve}_{ck}(x, c_1, \ldots, c_n) : (ck, x, (m_1, \ldots, m_n)) \in R_L \wedge \mathcal{A}(c_1, \ldots, c_n, \pi) = 1 \end{array}\right] \end{array} \tag{6}$$

where ξ is a probability distribution on $\mathcal{R}_{ck}$.

The difference between simulatability and zero-knowledge is that randomness $r_1, \ldots, r_n$ is directly generated from ξ as it would in the real-world protocol rather than chosen from adversary. This property becomes crucial when using the BDLOP commitments [BDL+18].

2.9 Commitment Scheme

We recall the BDLOP commitment scheme from [BDL+18] used in our constructions as well as previous works [ALS20,ENS20,LNS20]. Suppose that we want to commit to a message vector $\vec{\boldsymbol{m}} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) \in \mathcal{R}_q^n$ for $n \geq 1$ and that module ranks of κ and λ are required for M-SIS and M-LWE security, respectively. Then, in the key generation, a matrix $\boldsymbol{B}_0 \leftarrow \mathcal{R}_q^{\kappa \times (\kappa+\lambda+n)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_n \leftarrow \mathcal{R}_q^{\kappa+\lambda+n}$ are generated and output as public parameters. Note that one could choose to generate $\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_n$ in a more structured way as in [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 $\vec{\boldsymbol{r}} \leftarrow \chi^{d \cdot (\kappa+\lambda+n)}$. Now, there are two parts of the commitment scheme: the binding part and the message encoding part. In particular, we compute

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

for $i \in [n]$, where $\vec{\boldsymbol{t}}_0$ forms the binding part and each $\boldsymbol{t}_i$ encodes a message polynomial $\boldsymbol{m}_i$.

We recall the notion of a weak opening [ALS20].

Definition 2.9. *A* weak opening *for the commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1 \parallel \cdots \parallel \boldsymbol{t}_n$ consists of d polynomials $\bar{\boldsymbol{c}}_i \in \mathcal{R}_q$, a randomness vector $\vec{\boldsymbol{r}}^*$ over $\mathcal{R}_q$ and messages $\boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_n^* \in \mathcal{R}_q$ such that*

$$\begin{aligned} &\|\bar{\boldsymbol{c}}_i\|_1 \leq 2\mathsf{k} \text{ and } \bar{\boldsymbol{c}}_i \bmod X - \zeta^{2i+1} \neq 0 \text{ for all } 0 \leq i < d, \\ &\|\bar{\boldsymbol{c}}_i \vec{\boldsymbol{r}}^*\|_2 \leq 2\beta \text{ for all } 0 \leq i < d, \\ &\boldsymbol{B}_0 \vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0, \\ &\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}_i^* = \boldsymbol{t}_i \text{ for } i \in [n] \end{aligned}$$

Attema et al. show that the commitment scheme is still binding with respect to weak openings.

2.10 Framework by Lyubashevsky et al. [LNS20]

Recently, Lyubashevsky et al. [LNS20] proposed a general framework for proving linear and multiplicative relations between committed messages. Concretely, suppose that the prover $\mathcal{P}$ has a vector of n messages $\vec{m} = (\vec{m}_1, \ldots, \vec{m}_n) \in \mathbb{Z}_q^{nl}$. Let pp be a public set of polynomials $P : \left(\mathbb{Z}_q^l\right)^n \to \mathbb{Z}_q^l$ of n variables over $\mathbb{Z}_q^l$ with standard component-wise addition and multiplication and define $\alpha = \max_{P \in \mathsf{pp}} \deg(P)$ [6]. For readability, we will often "concatenate" polynomials in pp, i.e. we will write $\mathsf{pp} = \{P_1, \ldots, P_t\}$ where each $P_i : \left(\mathbb{Z}_q^l\right)^n \to \mathbb{Z}_q^{u_i l}$ and $u_1, \ldots, u_t > 0$.

For the linear relations, let us set $\mathsf{ulp} = (A, \vec{u}) \in \mathbb{Z}_q^{vl \times nl} \times \mathbb{Z}_q^{vl}$. In practice, A can have arbitrary number of rows but then we would have to pad rows with zeroes in order to get a multiple of l.

Overall, [LNS20] provides a protocol $\boldsymbol{\pi} = (\mathsf{Com}_{n,\alpha}, \Pi_n^\alpha(\mathsf{pp}, \mathsf{ulp}))$ where the prover $\mathcal{P}$ first generates the BDLOP commitments to $\vec{m}$ (sub-protocol $\mathsf{Com}_{n,\alpha}$) and then wants to prove that $\vec{m} \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$ (sub-protocol $\Pi_n^\alpha(\mathsf{pp}, \mathsf{ulp})$) where

$$\mathcal{L}_n(\mathsf{pp}, \mathsf{ulp}) := \{\vec{m} \in \mathbb{Z}_q^{nl} : \forall P \in \mathsf{pp}, P(\vec{m}) = \vec{0} \text{ and } A\vec{m} = \vec{u}\}. \tag{7}$$

[6] Although Lyubashevsky et al. only consider the case $\alpha \leq 3$, it can be easily generalised by sending more garbage commitments.

In this paper we are only interested in applying the LNS framework only for the case $l = d$, i.e. $X^d + 1$ splits completely into linear factors modulo q, unless stated otherwise.

Let us formulate the protocol π in terms of the commit-and-prove functionality from Section 2.8. First, the relation R_{LNS} we are interested in here is

$$(ck, (\mathsf{pp}, \mathsf{ulp}), \vec{m}) \in R_{LNS} \iff \vec{m} \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp}).$$

We define a CP functionality $LNS = (\mathsf{LNSGen}, \mathsf{LNSCom}, \mathsf{LNSProve}^U, \mathsf{LNSVerify}^U)$ for the relation R_{LNS} as follows:

- $\mathsf{LNSGen}(1^\mu)$: It outputs a commitment key ck specifies the message space $\mathcal{M}_{ck} = \mathbb{Z}_q^{nd}$, randomness space $\mathcal{R}_{ck} = \{-1, 0, 1\}^{d\cdot(\kappa+\lambda+n+\alpha)}$[7] and the commitment space $\mathcal{C}_{ck} = \mathcal{R}_q^{\kappa+n}$. It also generates the matrix $\boldsymbol{B}_0 \leftarrow \mathcal{R}_q^{\kappa\times(\kappa+\lambda+n+\alpha)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{n+\alpha} \leftarrow \mathcal{R}_q^{\kappa+\lambda+n+\alpha}$.
- $\mathsf{LNSCom}_{ck}((\vec{m}_1, \ldots, \vec{m}_n); \vec{\boldsymbol{r}})$ outputs $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$ where

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

- $\mathsf{LNSProve}^U_{ck}((\mathsf{pp}, \mathsf{ulp}), \vec{m}_1, \ldots, \vec{m}_n, \vec{\boldsymbol{r}})$: It runs the non-interactive version of the protocol $\Pi_n^\alpha(\mathsf{pp}, \mathsf{ulp})$ (e.g. [LNS20, Fig. 8]), using the Fiat-Shamir transform, and outputs the proof π. Letter U denotes that the algorithm uses uniform rejection sampling.
- $\mathsf{LNSVerify}^U_{ck}\Big((\mathsf{pp}, \mathsf{ulp}), \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n, \pi\Big)$: Check the verification equations of $\Pi_n^\alpha(\mathsf{pp}, \mathsf{ulp})$ (e.g. [LNS20, Fig. 9]).

Eventually, Lyubashevsky et al. show that the commit-and-prove functionality LNS defined above for the relation R_{LNS} is both knowledge sound with negligible knowledge error under the Module-SIS assumption and simulatable under the Module-LWE assumption where the randomness $\vec{\boldsymbol{r}}$ distribution is defined over $\xi = \chi^{d\cdot(\kappa+\lambda+n+\alpha)}$.

3 Opening Proof with Improved Rejection Sampling

In lattice-based zero-knowledge proofs, e.g.[BLS19,ALS20], 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, $\boldsymbol{c} \leftarrow C$ is a challenge polynomial, and $\vec{\boldsymbol{y}}$ is a "masking" vector. In order to remove the dependency of $\vec{\boldsymbol{z}}$ on $\vec{\boldsymbol{r}}$, one applies the *rejection sampling* technique.

Lemma 3.1 ([BLS19]). *Let $V \subseteq \mathcal{R}^\ell$ be a set of polynomials with norm at most T and $\rho\colon V \to [0,1]$ be a probability distribution. Also, write $\mathfrak{s} = 11T$ and $M = 3$. Now, sample $\vec{\boldsymbol{v}} \leftarrow \rho$ and $\vec{\boldsymbol{y}} \leftarrow D_{\mathfrak{s}}^{\ell d}$, set $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \vec{\boldsymbol{v}}$, and run $b \leftarrow \mathsf{Rej}_0(\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \mathfrak{s})$ as defined in Fig. 2. Then, the probability that $b = 0$ is at least $(1 - 2^{-100})/M$ and the distribution of $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{z}})$, conditioned on $b = 0$, is within statistical distance of $2^{-100}/M$ of the product distribution $\rho \times D_{\mathfrak{s}}^{\ell d}$.*

[7] Note that the length of $\vec{\boldsymbol{r}}$ is not $\kappa + \lambda + n$ as in Section 2.9 since the prover will later in the protocol commit to α garbage polynomials using the same $\vec{\boldsymbol{r}}$.

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

Fig. 2. Two rejection sampling algorithms: the one used in previous works (left) and the one proposed in this section (right).

Let us recall how parameters $\mathfrak{s}$ and M are usually selected. Namely, the repetition rate M is chosen to be an upper-bound on:

$$\frac{D^{\ell d}_{\mathfrak{s}}(\vec{\boldsymbol{z}})}{D^{\ell d}_{\vec{\boldsymbol{v}},\mathfrak{s}}(\vec{\boldsymbol{z}})} = \exp\left(\frac{-2\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{v}} \rangle + \|\vec{\boldsymbol{v}}\|^2}{2\mathfrak{s}^2}\right) \leq \exp\left(\frac{24\mathfrak{s}\|\vec{\boldsymbol{v}}\| + \|\vec{\boldsymbol{v}}\|^2}{2\mathfrak{s}^2}\right) = M. \tag{8}$$

For the inequality we used the tail bound which says that with probability at least $1 - 2^{100}$ we have $|\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{v}} \rangle| < 12\mathfrak{s}\|\vec{\boldsymbol{v}}\|$ for $\vec{\boldsymbol{z}} \leftarrow D^{\ell d}_{\mathfrak{s}}$ [Ban93,Lyu12]. Hence, by setting $\mathfrak{s} = 11\|\vec{\boldsymbol{v}}\|$ we obtain $M \approx 3$.

In this section we propose a new way to apply rejection sampling. Namely, we force $\vec{\boldsymbol{z}}$ to satisfy $\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{v}} \rangle \geq 0$, otherwise we abort. With this additional assumption, we can set M in the following way:

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

Hence, for $M \approx 3$ one would select $\mathfrak{s} = 0.675 \cdot \|\vec{\boldsymbol{v}}\|$. Note that the probability for $\vec{\boldsymbol{z}} \leftarrow D^{\ell d}_{\sigma}$ that $\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{v}} \rangle \geq 0$ is at least $1/2$. Hence, the expected number of rejections would be at most $2M = 6$. On the other hand, if one aims for $M = 6$ repetitions using (8), then $\mathfrak{s} = 6.74 \cdot \|\vec{\boldsymbol{v}}\|$. Thus, we manage to reduce the standard deviation by around a factor of 10.

Subset Rejection Sampling. In order to prove security of our new rejection sampling algorithm, we need the following modification of the rejection sampling lemma by Lyubashevsky [Lyu12].

Lemma 3.2 (Subset Rejection Sampling). *Let V be an arbitrary set and $h : V \to \mathbb{R}$, $f : \mathbb{Z}^m \to \mathbb{R}$ be probability distributions. Also define a family of set $S_v \subseteq \mathbb{Z}^m$ for $v \in V$ and $S = \{(z, v) \subseteq V \times \mathbb{Z}^m : z \in S_v\}$. Suppose $g_v : \mathbb{Z}^m \to \mathbb{R}$ is a family distributions indexed by all $v \in V$ and there exist $M, \gamma \geq 0$ which satisfy:*

$$\begin{aligned} &\forall v \in V, z \in S_v : Mg_v(z) \geq f(z) \\ &\forall v \in V : \sum_{z \in S_v} f(z) \geq \gamma. \end{aligned} \tag{10}$$

Then the distributions of the output of $\mathcal{A}$ and $\mathcal{F}$, defined in Fig. 3, are identical. Moreover, the probability that $\mathcal{A}$ and $\mathcal{F}$ output something is at least $\frac{\gamma}{M}$.

$\underline{\mathcal{A}}$	$\underline{\mathcal{F}}$
01 $v \leftarrow h$	01 $v \leftarrow h$
02 $z \leftarrow g_v$	02 $z \leftarrow f$
03 if $z \notin S_v$ then abort	03 if $z \notin S_v$ then abort
04 output (z, v) with probability $\frac{f(z)}{Mg_v(z)}$	04 output (z, v) with probability $\frac{1}{M}$.

Fig. 3. Algorithms $\mathcal{A}$ and $\mathcal{F}$ for Lemma 3.2.

Proof. Let $v \in V$. If $z \in S_v$, the probability that $\mathcal{A}$ outputs z is equal to $g_v(z) \cdot \frac{f(z)}{Mg_v(z)} = \frac{f(z)}{M}$. Otherwise, the probability that $\mathcal{A}$ outputs $z \notin S_v$ is exactly 0. Hence

$$\Pr[\mathcal{A} \text{ outputs something}] = \sum_{v \in V} h(v) \sum_{z \in S_v} \frac{f(z)}{M} \geq \frac{\gamma}{M}.$$

Moreover, the probability that $\mathcal{F}$ outputs something is also $\frac{\sum_{(z,v)\in S} h(v)f(z)}{M}$. Hence:

$$\begin{aligned}
\Delta(\mathcal{A}, \mathcal{F}) &= \frac{1}{2}\left(\sum_{(z,v)\in S} |\mathcal{A}(z,v) - \mathcal{F}(z,v)|\right) \\
&= \frac{1}{2}\sum_{v \in V} h(v)\left(\sum_{z \in S_v}\left|g_v(z) \cdot \frac{f(z)}{Mg_v(z)} - \frac{f(z)}{M}\right|\right) \\
&= \frac{1}{2}\sum_{v \in V} h(v)\left(\sum_{z \in S_v}\left|\frac{f(z)}{M} - \frac{f(z)}{M}\right|\right) = 0.
\end{aligned}$$

□

Later on, we will consider the special case when $f := D^m_{\mathfrak{s}}, g_{\vec{v}} := D^m_{\vec{v},\mathfrak{s}}$ for $\vec{v} \in V \subseteq \mathbb{Z}^m$ and

$$S_v := \{\vec{z} \in \mathbb{Z}^m : \langle \vec{v}, \vec{z} \rangle \geq 0\}.$$

Then, the probability that $\mathcal{A}$ outputs something is at least:

$$\frac{1}{M}\sum_{\vec{v} \in V} h(\vec{v}) \Pr_{\vec{z} \leftarrow D^m_{\mathfrak{s}}}[\langle \vec{v}, \vec{z} \rangle \geq 0] \geq \frac{1}{2M}.$$

Therefore, we can set $\gamma = 1/2$. Here, we used the fact that $\Pr_{\vec{z} \leftarrow D^m_{\mathfrak{s}}}[\langle \vec{v}, \vec{z} \rangle > 0] = \Pr_{\vec{z} \leftarrow D^m_{\mathfrak{s}}}[\langle \vec{v}, \vec{z} \rangle < 0]$. We also highlight that the value M we chose in (9) indeed satisfies Equation 10.

Extended M-LWE. One observes that with the new approach, the verifier learns some new information about the secret. Indeed, if a prover $\mathcal{P}$ returns $\vec{\boldsymbol{z}}$ then the verifier $\mathcal{V}$ knows that $\langle \vec{\boldsymbol{z}}, \vec{\boldsymbol{v}} \rangle \geq 0$. However, later on we will show that the opening proof from [ALS20] using the new rejection sampling is still simulatable assuming that a new problem, which we call *Extended M-LWE*, is computationally hard. For readability, we will describe it in a "knapsack" form.

Definition 3.3 (Extended M-LWE$_{m,k,\lambda,\chi,\xi_1,\xi_2}$). *The* Extended Module-LWE *problem with parameters $m, \lambda > 0$ and error distributions χ, ξ_1 and ξ_2 over $\mathcal{R}$ and $\mathcal{R}^k$ respectively, asks the adversary $\mathcal{A}$ to distinguish between the following two cases:*

1. $\left(\boldsymbol{B}, \boldsymbol{B}\vec{\boldsymbol{r}}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, \vec{\boldsymbol{z}}, \mathsf{sign}\left(\left\langle \vec{\boldsymbol{z}}, \begin{pmatrix} \boldsymbol{c}_1\vec{\boldsymbol{r}} \\ \vdots \\ \boldsymbol{c}_k\vec{\boldsymbol{r}} \end{pmatrix} \right\rangle\right)\right)$ *for* $\boldsymbol{B} \leftarrow \mathcal{R}_q^{m\times(m+\lambda)}$*, a secret vector* $\vec{\boldsymbol{r}} \leftarrow \chi^{m+\lambda}$ *and* $(\vec{\boldsymbol{z}}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) \leftarrow \xi_1^{k(m+\lambda)} \times \xi_2$
2. $\left(\boldsymbol{B}, \vec{\boldsymbol{u}}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, \vec{\boldsymbol{z}}, \mathsf{sign}\left(\left\langle \vec{\boldsymbol{z}}, \begin{pmatrix} \boldsymbol{c}_1\vec{\boldsymbol{r}} \\ \vdots \\ \boldsymbol{c}_k\vec{\boldsymbol{r}} \end{pmatrix} \right\rangle\right)\right)$ *for* $\boldsymbol{B} \leftarrow \mathcal{R}_q^{m\times(m+\lambda)}, \vec{\boldsymbol{u}} \leftarrow \mathcal{R}_q^m$ *and* $(\vec{\boldsymbol{z}}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) \leftarrow \xi_1^{k(m+\lambda)} \times \xi_2$

where $\mathsf{sign}(a) = 1$ *if* $a \geq 0$ *and* 0 *otherwise. Then,* $\mathcal{A}$ *is said to have advantage* ϵ *in solving Extended M-LWE*$_{m,k,\lambda,\chi,\xi_1,\xi_2}$ *if*

$$\begin{aligned}\Big|\Pr\Big[b=1 \,\Big|\, \boldsymbol{B} \leftarrow \mathcal{R}_q^{m\times(m+\lambda)};\, \vec{\boldsymbol{r}} \leftarrow \chi^{m+\lambda};\, \vec{\boldsymbol{z}} \leftarrow \xi_1^{k(m+\lambda)};\, \vec{\boldsymbol{c}} \leftarrow \xi_2;\, b \leftarrow \mathcal{A}(\boldsymbol{B}, \boldsymbol{B}\vec{\boldsymbol{r}}, \vec{\boldsymbol{z}}, \vec{\boldsymbol{c}}, s)\Big] \\ - \Pr\Big[b=1 \,\Big|\, \boldsymbol{B} \leftarrow \mathcal{R}_q^{m\times\lambda};\, \vec{\boldsymbol{u}} \leftarrow \mathcal{R}_q^m;\, \vec{\boldsymbol{z}} \leftarrow \xi_1^{k(m+\lambda)}; \vec{\boldsymbol{c}} \leftarrow \xi_2\,;\, b \leftarrow \mathcal{A}(\boldsymbol{B}, \vec{\boldsymbol{u}}, \vec{\boldsymbol{z}}, \vec{\boldsymbol{c}}, s)\Big]\Big| \geq \epsilon.\end{aligned}$$

where

$$s = \mathsf{sign}\left(\left\langle \vec{\boldsymbol{z}}, \begin{pmatrix} \boldsymbol{c}_1\vec{\boldsymbol{r}} \\ \vdots \\ \boldsymbol{c}_k\vec{\boldsymbol{r}} \end{pmatrix} \right\rangle\right) \text{ and } \vec{\boldsymbol{c}} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k).$$

To simplify notation, we will write Extended M-LWE$_{m,\lambda}$ *to denote Extended M-LWE*$_{m,1,\lambda,\chi^d,D_{\mathfrak{s}}^d,C}$ *where* χ *is defined in Section 2.6.*

We note that the LWE problem with various side information has already been discussed in e.g. [DGK+10,AP12,DDGR20]. As far as we are aware, our new variant of M-LWE is the closest to the Extended LWE problem defined by Alperin-Sheriff and Peikert [AP12][8]. Indeed, in Appendix D we show that hardness of the non-algebraic version of our Extended M-LWE (i.e. without any polynomial ring structure) can be reduced to plain LWE using similar techniques as in [AP12].

Hardness of Ring/Module-LWE is often analysed as an LWE problem since, so far, the best known attacks do not make use of the algebraic structure of the polynomial ring [ADPS15]. Then, the only two attacks which would be relevant to our Module-LWE problem are be the primal and dual attacks [Alb17,AGVW17,APS15]. Interestingly, the concrete analysis suggests that solving search-LWE (using the primal attack) is more efficient than solving the decisional version of LWE (using the dual attack). Thus, we believe that the search Extended-MLWE problem should still be hard with one bit, i.e. s, leaked since the vector $\vec{\boldsymbol{r}}$ has high enough entropy.

3.1 Concrete Instantiation

We describe how to apply our rejection sampling in the opening proof by Attema et al. [ALS20]. For readability, we consider the simple version of the protocol without any commitment compression [DKL+18] or Galois automorphisms [LNS20, Appendix A.3] for boosting soundness. We discuss how to apply all those improvements in Section 3.2 and Appendix B.

[8] In [AP12] the hint is the inner product $\langle \vec{r}, \vec{z} \rangle$ of the secret vector $\vec{r}$ and some $\vec{z}$ sampled from a given distribution D.

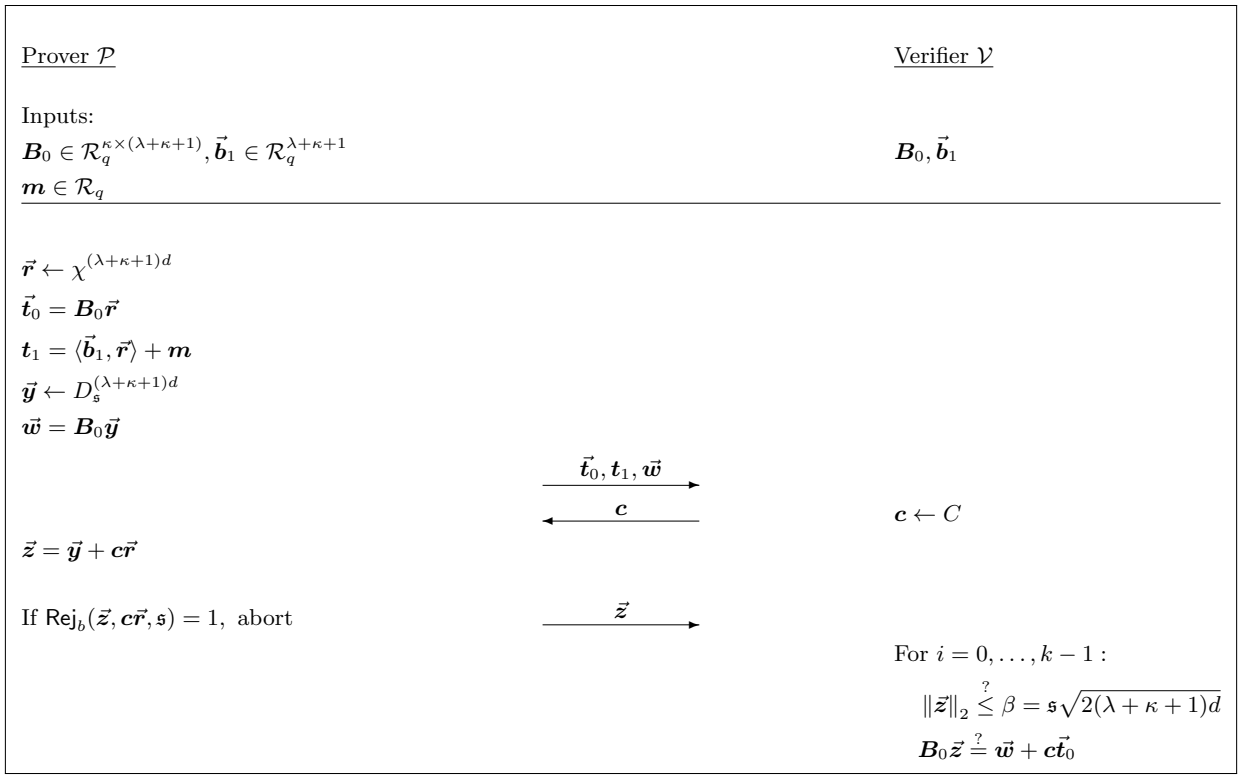

Fig. 4. Opening proof for the commitment scheme. If $b = 0$ then the protocol is identical to the one described in [ALS20] and uses Rej_0 defined in Fig. 2. On the other hand, if $b = 1$ then we apply the new rejection sampling algorithm Rej_1 in Fig. 2.

One observes that the protocol presented in Fig. 4 is not very meaningful since it only shows that prover $\mathcal{P}$ has a polynomial $\boldsymbol{m} \in \mathcal{R}_q$. However, it is a key ingredient to prove linear [ENS20] and multiplicative [ALS20] relations between committed messages.

Formally, let us define the following the commit-and-prove functionality $CP = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$:

- $\mathsf{Gen}(1^\mu)$: Given a security parameter μ, generates a commitment key ck which specifies a message space $\mathcal{M}_{ck} = R_q$, a randomness space $\mathcal{R}_{ck} = \{-1, 0, 1\}^{(\lambda+\kappa+1)d}$ and commitment space $\mathcal{C}_{ck} = \mathcal{R}_q^{\kappa+1}$. It also generates the matrix $\boldsymbol{B}_0 \leftarrow \mathcal{R}_q^{\kappa\times(\kappa+\lambda+1)}$ and the vector $\vec{\boldsymbol{b}}_1 \leftarrow \mathcal{R}_q^{\kappa+\lambda+1}$.
- $\mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}})$: Given a commitment key ck, a message $\boldsymbol{m} \in \mathcal{M}_{ck}$ and randomness $\vec{\boldsymbol{r}} \in \mathcal{R}_{ck}$ returns a commitment $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = (\boldsymbol{B}_0\vec{\boldsymbol{r}}, \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle + \boldsymbol{m}) \in \mathcal{C}_{ck}$.
- $\mathsf{Prove}_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}})$: It first generates $\vec{\boldsymbol{y}} \leftarrow D_{\mathfrak{s}}^{\kappa+\lambda+1}$ and computes $\boldsymbol{c} = H(\boldsymbol{B}_0\vec{\boldsymbol{y}})$. Then, it computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and gets $b \leftarrow \mathsf{Rej}_1(\vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}}, \mathfrak{s})$. If $b = 0$, it outputs $\pi = (\boldsymbol{c}, \vec{\boldsymbol{z}})$.
- $\mathsf{Verify}_{ck}\left(x, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi\right)$: Parse $\pi = (\boldsymbol{c}, \vec{\boldsymbol{z}})$. If $\|\vec{\boldsymbol{z}}\| \leq \mathfrak{s}\sqrt{2(\lambda+\kappa+1)d}$ and $\boldsymbol{c} = H(\vec{\boldsymbol{B}}_0\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0)$, return 1. Otherwise, return 0.

Here, $H : \{0,1\}^* \to \{-1, 0, 1\}^d \subset \mathcal{R}_q$ is a random oracle which generates output from the distribution C (see Section 2.4). The language R_L, for which CP is defined, is trivial: $(ck, x, \boldsymbol{m}) \in R_L \iff \boldsymbol{m} \in \mathcal{R}_q$.

Correctness and knowledge soundness of CP can be proven almost identically as in [ALS20, Theorem 4.4]. Hence, we will only focus on simulatability.

Theorem 3.4. *Suppose Extended M-LWE$_{\kappa+1,\lambda}$ is computationally hard. Then, the commit-and-prove functionality $CP = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$ defined above for the language R_L is simulatable in the random oracle model H.*

Proof. Let us consider the following hybrid algorithms.

- $\mathsf{Prove}^1_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}})$: It first generates $\vec{\boldsymbol{y}} \leftarrow D_{\mathfrak{s}}^{\kappa+\lambda+1}, \boldsymbol{c} \leftarrow C$. Then, it computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and gets $b \leftarrow \mathsf{Rej}_1(\vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}}, \mathfrak{s})$. If $b = 1$, it outputs $\pi = (\boldsymbol{c}, \vec{\boldsymbol{z}})$ and programs $\boldsymbol{c} = H(\boldsymbol{B}_0\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0)$.
- $\mathsf{Prove}^2_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}})$: It first generates $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{\kappa+\lambda+1}, \boldsymbol{c} \leftarrow C$. If $\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}} \rangle \geq 0$ then with probability $1/M$ it outputs $\pi = (\boldsymbol{c}, \vec{\boldsymbol{z}})$ and programs $\boldsymbol{c} = H(\boldsymbol{B}_0\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0)$.

It is easy to see that the difference between Prove and Prove^1 is that the algorithm programs the random oracle at one particular value $\boldsymbol{B}\vec{\boldsymbol{y}}$ (without checking whether it was already set). Hence, by arguing similarly as for zero-knowledge in [KLS18,DKL+18] we have that for all PPT adversaries $\mathcal{A}$:

$$\begin{aligned} &\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = \mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}}) : (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right] \\ &\approx \Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = \mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^1_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}}) : (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right]. \end{aligned} \tag{11}$$

For the next hybrid, we apply Lemma 3.2. Namely, let $m = (\lambda + \kappa + 1)d$ and $V = \{\boldsymbol{c}\vec{\boldsymbol{r}} : \boldsymbol{c} \in \{-1, 0, 1\}^d, \vec{\boldsymbol{r}} \in \{-1, 0, 1\}^m\}$. Set the probability distribution $h : V \to \mathbb{R}$ as

$$h(\vec{\boldsymbol{v}}) = \Pr[\boldsymbol{c}\vec{\boldsymbol{r}} = \vec{\boldsymbol{v}} : \boldsymbol{c} \leftarrow C, \vec{\boldsymbol{r}} \leftarrow \chi^m].$$

Next, we set $f := D_{\mathfrak{s}}^m, g_{\vec{\boldsymbol{v}}} := D_{\vec{\boldsymbol{v}},\mathfrak{s}}^m$ for $\vec{\boldsymbol{v}} \in V$ and

$$S_{\vec{\boldsymbol{v}}} := \{\vec{\boldsymbol{z}} \in \mathcal{R}_q^{\lambda+\kappa+1} : \langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{z}} \rangle \geq 0\}.$$

Then, by Lemma 3.2 we have:

$$\begin{aligned} &\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = \mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^1_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}}) : (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right] \\ &= \Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = \mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^2_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}}) : (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right] \end{aligned} \tag{12}$$

for all adversaries $\mathcal{A}$. Now we define two algorithms SimCom and $\mathsf{SimProve}$ responsible for the simulation.

- $\mathsf{SimCom}_{ck}(x)$: It samples $\vec{\boldsymbol{t}}_0$ and $\boldsymbol{t}_1$ uniformly at random from $\mathcal{R}_q^\kappa$ and $\mathcal{R}_q$ respectively and returns $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$.
- $\mathsf{SimProve}_{ck}\left(x, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1\right)$: It first generates $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{\kappa+\lambda+1}, \vec{\boldsymbol{r}}^* \leftarrow \chi^{\lambda+\kappa+1}$ and $\boldsymbol{c} \leftarrow C$. Then, if $\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}}^* \rangle \geq 0$ then with probability $1/M$ it outputs $\pi = (\boldsymbol{c}, \vec{\boldsymbol{z}})$ and programs $\boldsymbol{c} = H(\boldsymbol{B}_0\vec{\boldsymbol{z}} - \boldsymbol{c}\vec{\boldsymbol{t}}_0)$.

For the sake of contradiction suppose there exists a PPT adversary $\mathcal{A}$ such that

$$\begin{aligned} &\Big|\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = \mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^2_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}}) : (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right] \\ &- \Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) \leftarrow \mathsf{SimCom}_{ck}(x); \\ \pi \leftarrow \mathsf{SimProve}_{ck}(x, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1); (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right]\Big| = \epsilon. \end{aligned} \tag{13}$$

Let us construct an adversary $\mathcal{B}$ which solves the Extended M-LWE$_{\kappa+1,\lambda}$ also with probability ϵ using the algorithm $\mathcal{A}$. Concretely, suppose that $\mathcal{B}$ is given a tuple $\left((\boldsymbol{B}_0||\vec{\boldsymbol{b}}_1), (\vec{\boldsymbol{t}}_0||\boldsymbol{u}_1), \vec{\boldsymbol{z}}, \boldsymbol{c}, \mathsf{sign}(\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}}\rangle)\right)$ for $\vec{\boldsymbol{z}} \leftarrow D_{\mathfrak{s}}^{(\lambda+\kappa+1)d}, \boldsymbol{c} \leftarrow C$ and $\vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}$. Firstly, $\mathcal{A}$ outputs a pair $(x, \boldsymbol{m})$. Then, $\mathcal{B}$ sets $\boldsymbol{t}_1 = \boldsymbol{u}_1 + \boldsymbol{m}$. Finally, if $\mathsf{sign}(\langle \vec{\boldsymbol{z}}, \boldsymbol{c}\vec{\boldsymbol{r}}\rangle) \geq 0$ then $\mathcal{B}$ sets $\pi = (\boldsymbol{c}, \vec{\boldsymbol{z}})$ and with probability $1/M$ sends $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi)$. Otherwise, it aborts. At the end, $\mathcal{B}$ outputs the bit sent from $\mathcal{A}$.

First, suppose that $\vec{\boldsymbol{t}}_0 = \boldsymbol{B}\vec{\boldsymbol{r}}$ and $\boldsymbol{u}_1 = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle$. Then, $(\boldsymbol{t}_0, \boldsymbol{t}_1)$ constructed by $\mathcal{B}$ is indeed equal to $\mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}})$. Then, the way π is built is identical as in Prove^2. In this case, the probability that $\mathcal{B}$ outputs bit 1 is equal to

$$\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) = \mathsf{Com}_{ck}(\boldsymbol{m}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^2_{ck}(x, \boldsymbol{m}, \vec{\boldsymbol{r}}) : (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right].$$

On the other hand, assume that $\vec{\boldsymbol{t}}_0$ and $\boldsymbol{u}_1$ are chosen uniformly at random and also independently of $\vec{\boldsymbol{r}}$. Then, $\boldsymbol{t}_1$ is random as well. Hence, the probability that $\mathcal{B}$ outputs 1 is indeed equal to

$$\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (x, \boldsymbol{m}) \leftarrow \mathcal{A}(ck); (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1) \leftarrow \mathsf{SimCom}_{ck}(x); \\ \pi \leftarrow \mathsf{SimProve}_{ck}(x, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1); (ck, x, \boldsymbol{m}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \pi) = 1 \end{matrix}\right].$$

Thus, $\mathcal{B}$ can efficiently distinguish between the two Extended M-LWE cases with probability ϵ. Since, we assumed that the problem is computationally hard, this implies that ϵ is negligible. Then, the statement holds by the hybrid argument. □

3.2 Boosting Soundness and Decreasing Standard Deviation

The protocol in Fig. 4 has soundness error around $q^{-d/l}$ which is not necessarily negligible. In order to boost soundness, Attema et al. [ALS20] apply Galois automorphisms. We recall the extended opening proof protocol below.

Prover $\mathcal{P}$ first generates $\vec{\boldsymbol{y}}_0, \ldots, \vec{\boldsymbol{y}}_{k-1} \leftarrow D_{\mathfrak{s}}^{(\lambda+\kappa+1)d}$ and $\vec{\boldsymbol{r}}, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1$ as before. Next, it outputs $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \vec{\boldsymbol{w}}_0, \ldots, \vec{\boldsymbol{w}}_{k-1})$ where $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$. After receiving a challenge $\boldsymbol{c} \leftarrow C$ from the verifier, $\mathcal{P}$ computes

$$\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}} \text{ for } i = 0, \ldots, k-1$$

where $\sigma := \sigma_{2d/k+1}$ where k is a divisor of d. Then, the prover applies rejection sampling $\mathsf{Rej}_1(\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \mathfrak{s})$ where $\vec{\boldsymbol{z}} = \vec{\boldsymbol{z}}_0 \parallel \cdots \parallel \vec{\boldsymbol{z}}_{k-1}$ and $\vec{\boldsymbol{v}} = \sigma^0(\boldsymbol{c})\vec{\boldsymbol{r}} \parallel \cdots \parallel \sigma^{k-1}(\boldsymbol{c})\vec{\boldsymbol{r}}$. If it does not abort, then $\mathcal{P}$ outputs $\vec{\boldsymbol{z}}$. Finally, the verifier checks that $\vec{\boldsymbol{z}}$ is small and

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

for $i = 0, \ldots, k-1$. As argued by Attema et al., this protocol has soundness around $q^{-dk/l}$.

More recently, Lyubashevsky et al. [LNS20, Appendix A.6] (also mentioned in [ENS20]) improved this opening proof by applying a simple modification. Suppose $X^n + 1$ splits completely modulo q, i.e. $l = d$. Let us write the challenge $\boldsymbol{c} = \boldsymbol{c}_0 + \boldsymbol{c}_1 X + \ldots + \boldsymbol{c}_{k-1}X^{k-1} \leftarrow C$ where

$$\boldsymbol{c}_i = \sum_{j=0}^{d/k-1} c_{jk+i}X^{jk}.$$

By definition of $\sigma = \sigma_{2d/k+1}$, we have that $\sigma(\boldsymbol{c}_i) = \boldsymbol{c}_i$ for each i. Therefore, we have:

$$\sigma^i(\boldsymbol{c}) = \sum_{j=0}^{k-1} \sigma^i(X^j)\boldsymbol{c}_j.$$

The modified opening proof protocol is presented as follows. Prover $\mathcal{P}$ samples $\vec{\boldsymbol{y}}'_0, \ldots, \vec{\boldsymbol{y}}'_{k-1}$ from $D_{\mathfrak{s}}^{(\lambda+\kappa+1)d}$ as before. Then, the prover sends $\vec{\boldsymbol{w}}'_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}'_i$. After getting a challenge $\boldsymbol{c} \leftarrow C$, it computes $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{k-1}$ as above and calculates $\vec{\boldsymbol{z}}'_i$ as:

$$\begin{pmatrix} \vec{\boldsymbol{z}}'_0 \\ \vec{\boldsymbol{z}}'_1 \\ \vdots \\ \vec{\boldsymbol{z}}'_{k-1} \end{pmatrix} = \begin{pmatrix} \vec{\boldsymbol{y}}'_0 \\ \vec{\boldsymbol{y}}'_1 \\ \vdots \\ \vec{\boldsymbol{y}}'_{k-1} \end{pmatrix} + \begin{pmatrix} \boldsymbol{c}_0\vec{\boldsymbol{r}} \\ \boldsymbol{c}_1\vec{\boldsymbol{r}} \\ \vdots \\ \boldsymbol{c}_{k-1}\vec{\boldsymbol{r}} \end{pmatrix}.$$

Since each $\boldsymbol{c}_i$ has only at most d/k non-zero coefficients, we manage to decrease the standard deviation possibly by a factor of k (in practice the improvement is smaller if one upper-bounds $\|\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}\|$ more cleverly).

Eventually, the prover applies rejection sampling $\mathsf{Rej}_1(\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \mathfrak{s})$ where $\vec{\boldsymbol{z}} = \vec{\boldsymbol{z}}'_0 \parallel \cdots \parallel \vec{\boldsymbol{z}}'_{k-1}$ and $\vec{\boldsymbol{v}} = \boldsymbol{c}_0\vec{\boldsymbol{r}} \parallel \cdots \parallel \boldsymbol{c}_{k-1}\vec{\boldsymbol{r}}$. After receiving vectors $\vec{\boldsymbol{z}}'_j$, $\mathcal{V}$ first checks whether

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

for $i = 0, \ldots, k-1$ and that each $\vec{\boldsymbol{z}}_i$ is small.

Note that by computing

$$\vec{\boldsymbol{y}}_i = \sum_{j=0}^{k-1} \sigma^i(X^j)\vec{\boldsymbol{y}}'_j, \;\; \vec{\boldsymbol{z}}_i = \sum_{j=0}^{k-1} \sigma^i(X^j)\vec{\boldsymbol{z}}'_j$$

and $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$ for $i = 0, \ldots, k-1$, we have:

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

which is the exact verification equation as in [ALS20]. This observation is crucial in [LNS20] in order to still be able to prove linear and multiplicative relations using techniques from [ALS20,ENS20].

Lyubashevsky et al. bound $\|\vec{\boldsymbol{v}}\|$ by first finding α so that

$$\Pr\left[\exists i, \|\boldsymbol{c}_i\vec{\boldsymbol{r}}\|_\infty > \alpha : \boldsymbol{c} \leftarrow C, \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}\right] < 2^{-128}$$

and then setting the bound $\|\vec{\boldsymbol{v}}\| \leq \alpha\sqrt{\lambda+\kappa+1}$. In Appendix C we describe a more optimal way to compute an upper-bound on $\|\vec{\boldsymbol{v}}\|$ using almost identical methods as in [DDLL13].

3.3 Applications

We apply the new rejection sampling technique in the protocol by Esgin et al. [ENS20, Appendix B] to prove knowledge of secrets in LWE samples. Concretely, for $n = 2048$, we want to prove knowledge of a ternary vector $\vec{s} \in \{-1, 0, 1\}^n$ such that

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

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. For fair comparison, we will use the protocol described in [ENS20, Fig. 3] with the following two modifications (i) we do the Gaussian rejection sampling according to Rej_1 instead of the uniform one and (ii) we apply the commitment compression techniques as in Appendix B.

We set parameters $(q, d, l, k) = (\approx 2^{32}, 128, 128, 4)$ similarly as in [ENS20]. Esgin et al. choose the expected number of repetitions to be 18.87. Since sampling from a discrete Gaussians is much less efficient than from a uniform distribution, for fairness we set $\mathfrak{s} = T$ and $M \approx 3.3$ where T is the upper-bound on $\|\vec{\boldsymbol{v}}\|$ where $\vec{\boldsymbol{v}} = \boldsymbol{c}_0\vec{\boldsymbol{r}} \parallel \cdots \parallel \boldsymbol{c}_{k-1}\vec{\boldsymbol{r}}$. Esgin et al. use the fact that $\|\vec{\boldsymbol{v}}\|_\infty \leq d/k = 32$ and thus they set $T = 32\sqrt{(\lambda+\kappa+3+16)d}$. We, however, apply the bound described in Appendix C and observe that for (κ, λ) selected in the next paragraph, our bound on $\|\vec{\boldsymbol{v}}\|$ is around five times smaller than in [ENS20].

Now we set λ and κ such that M-LWE and M-SIS are hard against known attacks. We measure the hardness with the root Hermite factor δ and aim for $\delta \approx 1.0043$ [ENS20,EZS+19,BLS19]. By assuming that the Extended M-LWE is almost equally hard as M-LWE, we set $\lambda = 10$ as in [ENS20]. On the other hand, for the M-SIS hardness we manage to set $\kappa = 8$ due to having smaller standard deviation $\mathfrak{s}$. Hence, without using the compression techniques, we obtain the proof of size 41.35KB compared to 47KB by [ENS20]. After applying the additional improvements described in Appendix B we reduce the proof size to 33.6KB.

Similarly, we can consider the LNS functionality defined in Section 2.10 where $\mathsf{LNSProve}^U$ use our new Gaussian rejection sampling in Fig. 2 instead of a uniform one. We will denote this variant of the protocol as $\mathsf{LNSProve}^D$ and the corresponding CP functionality for the language R_L as

$$LNS^D = (\mathsf{LNSGen}, \mathsf{LNSCom}, \mathsf{LNSProve}^D, \mathsf{LNSVerify}^D).$$

The total proof size (when using Discrete Gaussians and without any Dilithium compression techniques) is about

$$(n+\kappa+\alpha+1)d\log q \;+\; k(\lambda+\kappa+n+\alpha)d\cdot\log(12\mathfrak{s}) \quad \text{bits.} \tag{14}$$

Here, n, α are defined in Section 2.10, k is a divisor of d such that q^{-k} is negligible and $\mathfrak{s}$ is the standard deviation used for the rejection sampling. For efficiency, we also apply Dilithium compression methods described in Appendix B.

We present the proof sizes for proving n-bit integer addition and multiplication using LNS^D in Figure 1.

4 Verifiable Decryption

In this section we apply the improved LNS framework with the rejection sampling from Section 3 to the problem of constructing a verifiable decryption scheme. We restrict our attention the Kyber key encapsulation scheme [BDK+18] and its NIST level 1 parameter set Kyber512. Kyber is a finalist in the NIST PQC standardization effort. Our techniques work equally well for any of the other lattice-based KEMs in round 3 of the NIST process, i.e. Saber [DKRV18] and NTRU [HPS98]. Kyber512 uses module rank 2 over the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^{256}+1)$ with modulus $q = 3329$. The public key is given by an MLWE vector $\vec{\boldsymbol{t}} = \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}}$ where $\boldsymbol{A} \in \mathcal{R}_q^{2\times 2}$ is a uniform public matrix and

$\vec{s}$ and $\vec{e}$ are short secret vectors with coefficients in the interval $[-2, 2]$. The encryption of a binary polynomial $\boldsymbol{m} \in \{0,1\}^{256}$ encoding a 256-bit key consists of the rounded vector and polynomial

$$\vec{u} = \mathsf{Compress}_{10}\left(\boldsymbol{A}^T\vec{s}' + \vec{e}_u\right) = \boldsymbol{A}^T\vec{s}' + \vec{e}'_u$$
$$\boldsymbol{v} = \mathsf{Compress}_4\left(\langle\vec{t}, \vec{s}'\rangle + \boldsymbol{e}_v + \left\lceil\frac{q}{2}\right\rfloor \boldsymbol{m}\right) = \langle\vec{t}, \vec{s}'\rangle + \boldsymbol{e}'_v + \left\lceil\frac{q}{2}\right\rfloor \boldsymbol{m}$$

where $\vec{s}'$, $\vec{e}_u$ and $\boldsymbol{e}_v$ are again short, the functions $\mathsf{Compress}_{10}$ and $\mathsf{Compress}_4$ compress to 10 and 4 bits per coefficient, and $\vec{e}'_u$, $\boldsymbol{e}'_v$ include the errors coming from the compression. Finally, decryption uses the observation

$$\boldsymbol{v} - \langle\vec{s}, \vec{u}\rangle = \langle\vec{e}, \vec{s}'\rangle - \langle\vec{s}, \vec{e}'_u\rangle + \boldsymbol{e}'_v + \left\lceil\frac{q}{2}\right\rfloor \boldsymbol{m},$$

which implies

$$\left\|\boldsymbol{v} - \langle\vec{s}, \vec{u}\rangle - \left\lceil\frac{q}{2}\right\rfloor \boldsymbol{m}\right\|_\infty < \frac{q}{4}$$

with overwhelming probability. In fact, the decryption algorithm will recover $\boldsymbol{m}$ precisely if this norm bound is true. In the scheme there is no guarantee for this bound and encrypted keys can fail to be decryptable with probability around 2^{-139}.

Now, for a verifiable decryption scheme we need to be able to prove knowledge of a vector $\vec{s}$ and polynomials $\boldsymbol{m}$, $\boldsymbol{x}$ such that

$$\boldsymbol{v} - \langle\vec{s}, \vec{u}\rangle - \left\lceil\frac{q}{2}\right\rfloor \boldsymbol{m} = \boldsymbol{x} \tag{15}$$
$$\vec{s} \in \{-2, -1, 0, 1, 2\}^{512} \tag{16}$$
$$\boldsymbol{m} \in \{0,1\}^{256} \tag{17}$$
$$\|\boldsymbol{x}\|_\infty < \frac{q}{4} \tag{18}$$

The first three properties (15), (16), (17) can in principle directly be proven with the LNS framework. For the fourth one we can use a small-base decomposition approach for doing range proofs. A small problem is posed by the magnitude of the Kyber modulus $q = 3329$. While it is possible to instantiate the LNS framework in a way that allows to directly prove linear equations modulo such small primes, this results in quite large soundness error and many repetitions in the opening proof. To circumvent this problem and arrive at a more efficient protocol, we use the framework with a much larger modulus q' and lift Equation (15) to the integers. This means that we instead prove

$$\boldsymbol{v} - \langle\vec{s}, \vec{u}\rangle - \left\lceil\frac{q}{2}\right\rfloor \boldsymbol{m} + \boldsymbol{d}q \equiv \boldsymbol{x} \pmod{q'} \tag{19}$$

for another secret polynomial $\boldsymbol{d} \in \mathbb{Z}[X]/(X^{256}+1)$ whose range $\|\boldsymbol{d}\|_\infty \leq 2^{10}$ we also need to prove. Note that Equation (19) for $q' > 2^{23}$ together with the range proofs implies Equation (15) since from the ranges of the individual terms we know that the equation must hold over the integers, which in turn implies (15) since the additional term, which is a multiple of q, vanishes modulo q.

4.1 Range Proofs

In the protocol sketched above there are the two range proofs $\|\boldsymbol{d}\|_\infty \leq 2^{10}$ and $\|\boldsymbol{x}\|_\infty < q/4$. For the first range proof it is actually enough to prove that $q\,\|\boldsymbol{d}\|_\infty < q'/4$. We use a 64-bit prime q' for the LNS framework protocol. Then, there is enough head-room between the actual range 2^{10} of

$\boldsymbol{d}$ and $q'/(4q) > 2^{50}$ so that we can use the approximate shortness proof. On the other hand, for proving the range of $\boldsymbol{x}$ it is important to not have any slack in the proof. So here we decompose $\boldsymbol{x}$ in base 5. We choose base 5, since for proving the coefficients of $\vec{\boldsymbol{s}}$ to lie in the interval $[-2, 2]$ we already have degree-5 product relations and hence can prove the base-5 decomposition without any additional garbage commitments. The interval of the coefficients of $\boldsymbol{x}$ has length $q/2 = 1664$ and hence we need 5 base-5 digits for each coefficient. Now, since 1664 is not a power of 5 we write each coefficient of $\boldsymbol{x}$ in the form

$$a_0 + a_1 5 + a_2 5^2 + a_3 5^3 + a_4 260$$

with $a_4 \in 0, \ldots, 4$. This decomposition is not unique but precisely maps to the integers in the interval $[0, 1664]$.

4.2 Proof Size

We compute the size of the verifiable decryption scheme for Kyber512 from above. As commitment messages we have the vector $\boldsymbol{s}$ of $\mathbb{Z}$-dimension 512, the polynomial $\boldsymbol{m}$ of dimension 256, the masking polynomial for the approximate range proof for $\boldsymbol{d}$, and the expansion of $\boldsymbol{x}$ in base-5, which has $\mathbb{Z}$-dimension $5 \cdot 256 = 1280$. This amounts to a total message dimension of $n = 2176$. We then computed that the full LNS protocol with 64-bit modulus, MLWE rank 20 and MSIS rank 5 has a proof size of 43.6KB.

Acknowledgements

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

References

ADPS15. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. *IACR Cryptol. ePrint Arch.*, 2015:1092, 2015.

AGVW17. Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to LWE. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 297–322. Springer, 2017.

Alb17. Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in helib and SEAL. In *EUROCRYPT (2)*, volume 10211 of *Lecture Notes in Computer Science*, pages 103–129, 2017.

ALS20. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 470–499. Springer, 2020.

AP12. Jacob Alperin-Sheriff and Chris Peikert. Circular and KDM security for identity-based encryption. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 334–352. Springer, 2012.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `https://eprint.iacr.org/2015/046`.

Ban93. Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, Dec 1993.

BD20. Zvika Brakerski and Nico Döttling. Hardness of LWE on general entropic distributions. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 551–575. Springer, 2020.

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.

BL17. Carsten Baum and Vadim Lyubashevsky. Simple amortized proofs of shortness for linear relations over polynomial rings. *IACR Cryptology ePrint Archive*, 2017:759, 2017.

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 176–202. Springer, 2019.

BN20. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In *Public Key Cryptography (1)*, volume 12110 of *Lecture Notes in Computer Science*, pages 495–526. Springer, 2020.

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.

DDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO (1)*, pages 40–56, 2013.

DGK+10. Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, 2010.

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.

DKRV18. Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In *AFRICACRYPT*, pages 282–305, 2018.

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

EG14. Alex Escala and Jens Groth. Fine-tuning groth-sahai proofs. In *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 630–649. Springer, 2014.

EKS+20. Muhammed F. Esgin, Veronika Kuchta, Amin Sakzad, Ron Steinfeld, Zhenfei Zhang, Shifeng Sun, and Shumo Chu. Practical post-quantum few-time verifiable random function with applications to algorand. Cryptology ePrint Archive, Report 2020/1222, 2020. `https://eprint.iacr.org/2020/1222`.

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. *IACR Cryptol. ePrint Arch.*, 2020:518, 2020. To appear at ASIACRYPT 2020. `https://eprint.iacr.org/2020/518`.

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.

GKPV10. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010.

HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, 2018.

LNS20. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. *IACR Cryptology ePrint Archive*, 2020. ACM CCS 2020. `http://eprint.iacr.org/2020/1183`.

LNS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. PKC 2021, 2021.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC*, pages 107–124, 2013.

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

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.

MM11. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484. Springer, 2011.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

OPW11. Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2011.

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

TWZ20. Yang Tao, Xi Wang, and Rui Zhang. Short zero-knowledge proof of knowledge for lattice-based commitment. In *PQCrypto*, volume 12100 of *Lecture Notes in Computer Science*, pages 268–283. Springer, 2020.

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.

A Proof of Lemma 2.5

We first prove the following lemma.

Lemma A.1. *Let q be an odd positive integer and C be the probability distribution defined in Section 2.7. Then, for any vector $\vec{s} \in \mathbb{Z}_q^n$ and integer $y \in \mathbb{Z}_q$, we have*

$$\Pr_{\vec{c} \leftarrow \mathsf{C}^n}\left[\|\langle \vec{c}, \vec{s}\rangle + y\rangle\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty\right] \leq \max\{p, 1-p\}.$$

Proof. Let s_i be the coefficient of $\vec{s}$ so that $\|s_i\|_\infty = \|\vec{s}\|_\infty$. Then, one can write $\langle \vec{c}, \vec{s}\rangle + y = s_i c_i + r$ for some $r \in \mathbb{Z}_q$. We consider two cases.

Case 1: $\|r\|_\infty \geq \frac{1}{2}\|\vec{s}\|_\infty$. Then, c_i would have to be either 1 or -1 for any chance of $s_i c_i + r$ to be less than $\frac{1}{2}\|\vec{s}\|_\infty$. This implies that

$$\Pr_{\vec{c} \leftarrow \mathsf{C}^n}\left[\|\langle \vec{c}, \vec{s}\rangle + y\rangle\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty \middle| \|r\|_\infty \geq \frac{1}{2}\|\vec{s}\|_\infty\right] \leq \Pr_{c_i \leftarrow \mathsf{C}}[|c_i| = 1] = 1 - p.$$

Case 2: $\|r\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty$. We will prove that

$$\|r + bs_i\|_\infty \geq \frac{1}{2}\|s_i\|_\infty$$

for any $b \in \{-1, 1\}$. Therefore, we have

$$\Pr_{\vec{c} \leftarrow \mathsf{C}^n}\left[\|\langle \vec{c}, \vec{s}\rangle + y\rangle\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty \middle| \|r\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty\right] \leq \Pr_{c_i \leftarrow \mathsf{C}}[c_i = 0] = p$$

which will complete the proof of the lemma.

First, we can assume that $|s_i| \leq q/2$ and $|r| < |s_i|/2$. Thus, $\|r + bs_i\|_\infty$ is either equal to $|r + bs_i|$ or $|r + bs_i \pm q|$. In the former case, we immediately have $|r + bs_i| \geq |bs_i| - |r_i| > |s_i|/2$. For the

latter case, we can assume for the sake of contradiction that $u = r + bs_i \pm q$ where $|u| < |s_i|/2$. Therefore,

$$q = |\pm q| = |r + bs_i - u| \leq |r| + |bs_i| + |u| < |s_i|/2 + |s_i| + |s_i|/2 = 2|s_i| \leq q.$$

□

This lemma can be then easily generalised to the matrix setting. Hence, the statement holds.

B Opening Proof with Commitment Compression

B.1 Low/High Order Bits

In order to reduce the size of the commitment, we need some algorithms that extract "higher-order" and "lower-order" bits of elements in $\mathbb{Z}_q$. The goal is that when given an arbitrary element $r \in \mathbb{Z}_q$ and another small element $z \in \mathbb{Z}_q$, we would like to be able to recover the higher order bits of $r + z$ without needing to store z. The algorithms are exactly as in [DKL+18], and we repeat them for completeness in Fig. 5. They are described as working on integers modulo q, but one can extend it to (vectors of) polynomials in $\mathcal{R}_q$ by simply being applied individually to each coefficient.

$\underline{\mathsf{Power2Round}_q(r, D)}$
00 $r := r \bmod^+ q$
01 $r_0 := r \bmod^\pm 2^D$
02 **return** $(r - r_0)/2^D$

$\underline{\mathsf{UseHint}_q(h, r, \alpha)}$
03 $m := (q-1)/\alpha$
04 $(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$
05 **if** $h = 1$ and $r_0 > 0$ **return** $(r_1 + 1) \bmod^+ m$
06 **if** $h = 1$ and $r_0 \leq 0$ **return** $(r_1 - 1) \bmod^+ m$
07 **return** r_1

$\underline{\mathsf{MakeHint}_q(z, r, \alpha)}$
08 $r_1 := \mathsf{HighBits}_q(r, \alpha)$
09 $v_1 := \mathsf{HighBits}_q(r + z, \alpha)$
10 **return** $[\![r_1 \neq v_1]\!]$

$\underline{\mathsf{Decompose}_q(r, \alpha)}$
11 $r := r \bmod^+ q$
12 $r_0 := r \bmod^\pm \alpha$
13 **if** $r - r_0 = q - 1$
14 **then** $r_1 := 0$; $r_0 := r_0 - 1$
15 **else** $r_1 := (r - r_0)/\alpha$
16 **return** (r_1, r_0)

$\underline{\mathsf{HighBits}_q(r, \alpha)}$
17 $(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$
18 **return** r_1

$\underline{\mathsf{LowBits}_q(r, \alpha)}$
19 $(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$
20 **return** r_0

Fig. 5. Supporting algorithms for commitment compression.

Lemma B.1. *Suppose that q and α are positive integers satisfying $q > 2\alpha$, $q \equiv 1 \pmod{\alpha}$ and α even. Let $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{z}}$ be vectors of elements in R_q where $\|\vec{\boldsymbol{z}}\|_\infty \leq \alpha/2$, and let $\vec{\boldsymbol{h}}, \vec{\boldsymbol{h}'}$ be vectors of bits. Then the* $\mathsf{HighBits}_q$, $\mathsf{MakeHint}_q$, *and* $\mathsf{UseHint}_q$ *algorithms satisfy the following properties:*

1. $\mathsf{UseHint}_q(\mathsf{MakeHint}_q(\vec{\boldsymbol{z}}, \vec{\boldsymbol{r}}, \alpha), \vec{\boldsymbol{r}}, \alpha) = \mathsf{HighBits}_q(\vec{\boldsymbol{r}} + \vec{\boldsymbol{z}}, \alpha)$.

2. *Let* $\vec{\boldsymbol{v}}_1 = \mathsf{UseHint}_q(\vec{\boldsymbol{h}}, \vec{\boldsymbol{r}}, \alpha)$. *Then* $\|\vec{\boldsymbol{r}} - \vec{\boldsymbol{v}}_1 \cdot \alpha\|_\infty \leq \alpha + 1$.
3. *For any* $\vec{\boldsymbol{h}}, \vec{\boldsymbol{h}}'$, *if* $\mathsf{UseHint}_q(\vec{\boldsymbol{h}}, \vec{\boldsymbol{r}}, \alpha) = \mathsf{UseHint}_q(\vec{\boldsymbol{h}}', \vec{\boldsymbol{r}}, \alpha)$, *then* $\vec{\boldsymbol{h}} = \vec{\boldsymbol{h}}'$.

Lemma B.2. *If* $\|\vec{\boldsymbol{s}}\|_\infty \leq \beta$ *and* $\|\mathsf{LowBits}_q(\vec{\boldsymbol{r}}, \alpha)\|_\infty < \alpha/2 - \beta$, *then*

$$\mathsf{HighBits}_q(\vec{\boldsymbol{r}}, \alpha) = \mathsf{HighBits}_q(\vec{\boldsymbol{r}} + \vec{\boldsymbol{s}}, \alpha).$$

B.2 Opening Proof

We apply the compression techniques from [DKL+18] in the opening proof by Attema et al. [ALS20]. We already implement the rejection sampling technique from Section 3. The protocol has soundness error around $q^{-kd/l}$ where k is a divisor of d. We recall that $\sigma := \sigma_{2d/k+1}$ and we only consider the case when $l = d$. We present the protocol in Fig. 6.

To begin with, the prover $\mathcal{P}$ generates a BDLOP commitment $(\vec{\boldsymbol{t}}_0, \boldsymbol{t})$ to a message $\boldsymbol{m} \in \mathcal{R}_q$:

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

We can reduce the size of the commitment by not sending the low-order bits of $\boldsymbol{t}_0$. Concretely, we write

$$\vec{\boldsymbol{t}}_0 = \vec{\boldsymbol{t}}_{0,1} \cdot 2^D + \vec{\boldsymbol{t}}_{0,0} \text{ where } \|\vec{\boldsymbol{t}}_{0,0}\|_\infty < 2^{D-1}$$

and only send $\left(\vec{\boldsymbol{t}}_{0,1}, \boldsymbol{t}_1\right)$. Also, the matrix $\boldsymbol{B}_0$ can be written as $\boldsymbol{B}_0 = (\boldsymbol{B} \parallel \boldsymbol{I}_\kappa)$ as in [BDL+18]. Thus, we can write $\vec{\boldsymbol{t}}_0 = \boldsymbol{B}\vec{\boldsymbol{r}}_1 + \vec{\boldsymbol{r}}_2$ where $\vec{\boldsymbol{r}} = \vec{\boldsymbol{r}}_1 \parallel \vec{\boldsymbol{r}}_2$.

The next step is that $\mathcal{P}$ generates $\vec{\boldsymbol{y}}_i \leftarrow D_{\mathfrak{s}}^{\lambda+\kappa+1}$ and computes $\vec{\boldsymbol{w}}_i = \boldsymbol{B}\vec{\boldsymbol{y}}_i$. Finally, it outputs $\vec{\boldsymbol{w}}_{i,1} = \mathsf{HighBits}_q(\vec{\boldsymbol{w}}_i, 2\gamma)$ for $i = 0, \ldots, k-1$. Here, γ corresponds to γ_2 in Dilithium.

Next, the verifier sends a challenge polynomial $\boldsymbol{c} \leftarrow C$. Then, $\mathcal{P}$ computes

$$\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}} + \sigma^i(\boldsymbol{c})\,\vec{\boldsymbol{r}}_1 \text{ for } i = 0, \ldots, k-1$$

and applies the rejection sampling Rej_1. If the algorithm does not abort then the prover checks whether for any i: $\|\sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_{0,0}\|_\infty \geq \gamma_2$ or $\vec{\boldsymbol{l}}_{i,0} \geq \gamma - \delta$ or $\vec{\boldsymbol{l}}_{i,1} \neq \vec{\boldsymbol{w}}_i$ where

$$(\vec{\boldsymbol{l}}_{i,1}, \vec{\boldsymbol{l}}_{i,0}) = \mathsf{Decompose}_q(\vec{\boldsymbol{w}}_i - \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}_2, 2\gamma)$$

and δ is the L_∞ bound on $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}_2$. If not, then it outputs $(\vec{\boldsymbol{z}}_i, \vec{\boldsymbol{h}}_i)$ where

$$\vec{\boldsymbol{h}}_i = \mathsf{MakeHint}_q\left(-\sigma^i(\boldsymbol{c})\boldsymbol{t}_{0,0}, \vec{\boldsymbol{w}}_i - \sigma^i(\boldsymbol{c}) \cdot \vec{\boldsymbol{r}}_2 + \sigma^i(\boldsymbol{c})\boldsymbol{t}_{0,0}, 2\gamma\right) \text{ for } i = 0, 1, \ldots, k-1.$$

At the end, $\mathcal{V}$ checks that coefficients of $\vec{\boldsymbol{z}}_i$ are small and for $i = 0, \ldots, k-1$:

$$\vec{\boldsymbol{w}}_{i,1} \stackrel{?}{=} \mathsf{UseHint}_q\left(\vec{\boldsymbol{h}}_i, \boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c}) \cdot 2^D \boldsymbol{t}_{0,1}, 2\gamma\right).$$

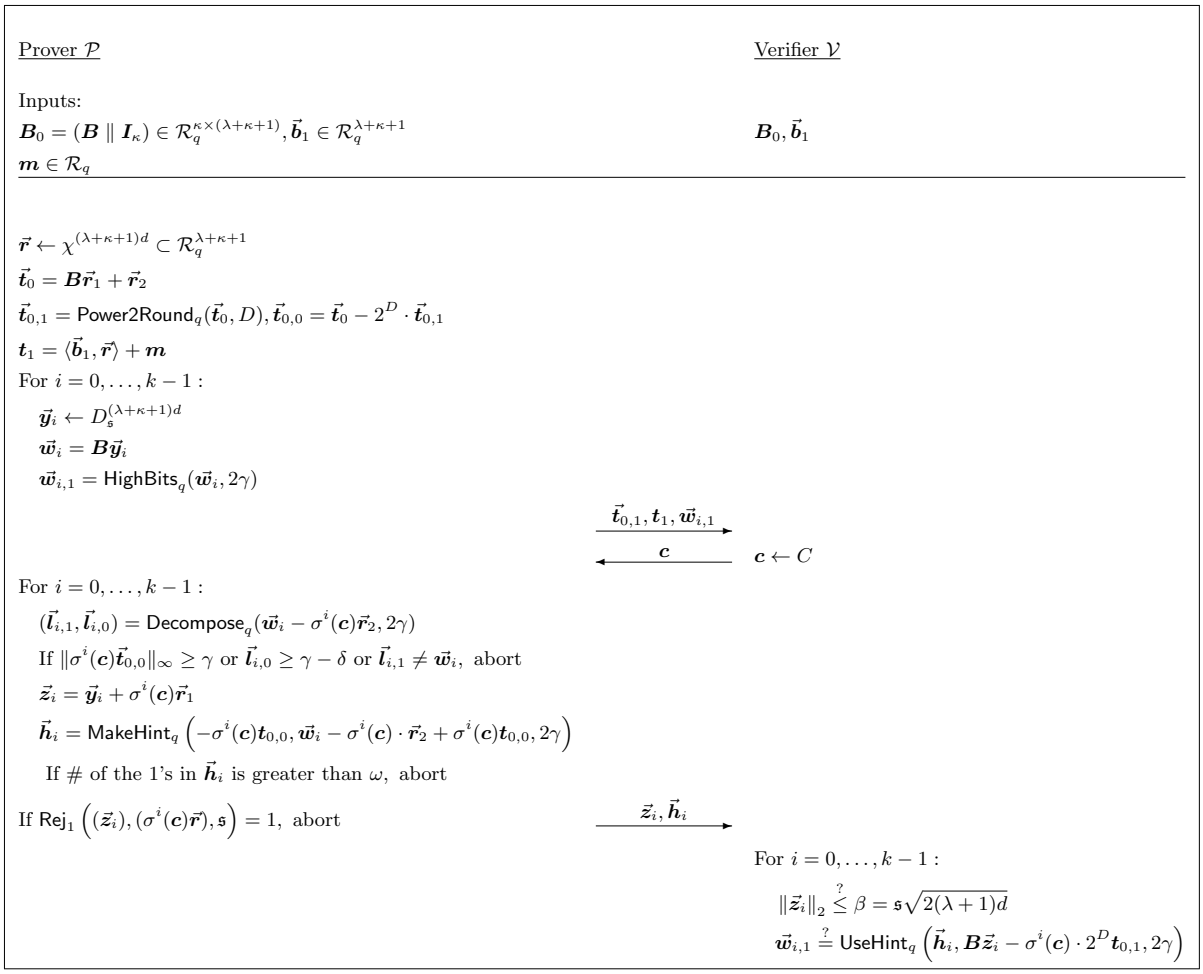

Fig. 6. Automorphism opening proof for the commitment scheme using compression techniques from Dilithium [DKL+18].

B.3 Security analysis.

Correctness. It follows directly from Lemmas B.1 and B.2:

$$\begin{aligned}
&\mathsf{UseHint}_q\left(\vec{\boldsymbol{h}}_i, \boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c}) \cdot 2^D\boldsymbol{t}_{0,1}, 2\gamma\right) \\
&\quad = \mathsf{HighBits}_q\left(\boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c}) \cdot \left(2^D\boldsymbol{t}_{0,1} + \vec{\boldsymbol{t}}_{0,0}\right), 2\gamma\right) \\
&\quad = \mathsf{HighBits}_q\left(\boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0, 2\gamma\right) \\
&\quad = \mathsf{HighBits}_q\left(\vec{\boldsymbol{w}}_i - \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}_2, 2\gamma\right) \\
&\quad = \vec{\boldsymbol{w}}_{i,1}.
\end{aligned}$$

Soundness. In the soundness argument, we will extract an *extended weak opening* which is a slight modification of Definition B.3. We first recall the notion of a weak opening [ALS20].

Definition B.3. *A* weak opening *for the commitment* $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1 \parallel \cdots \parallel \boldsymbol{t}_n$ *consists of* d *polynomials* $\bar{\boldsymbol{c}}_i \in \mathcal{R}_q$*, a randomness vector* $\vec{\boldsymbol{r}}^*$ *over* $\mathcal{R}_q$ *and messages* $\boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_n^* \in \mathcal{R}_q$ *such that*

$$\begin{aligned}
&\|\bar{\boldsymbol{c}}_i\|_1 \leq 2\mathsf{k} \text{ and } \bar{\boldsymbol{c}}_i \bmod X - \zeta^{2i+1} \neq 0 \text{ for all } 0 \leq i < d,\\
&\|\bar{\boldsymbol{c}}_i \vec{\boldsymbol{r}}^*\|_2 \leq 2\beta \text{ for all } 0 \leq i < d,\\
&\boldsymbol{B}_0 \vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0,\\
&\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}_i^* = \boldsymbol{t}_i \text{ for } i \in [n]
\end{aligned}$$

Attema et al. show that the commitment scheme is still binding with respect to weak openings.

Definition B.4. *An extended weak opening for the commitment* $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_{0,1} \parallel \boldsymbol{t}_1 \parallel \cdots \parallel \boldsymbol{t}_n$ *consists of* d *polynomials* $\bar{\boldsymbol{c}}_i \in \mathcal{R}_q$*, a randomness vector* $\vec{\boldsymbol{r}}^*$ *over* $\mathcal{R}_q$*, messages* $\boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_n^* \in \mathcal{R}_q$ *and additional vector* $\vec{\boldsymbol{u}}^*$ *such that*

$$\begin{aligned}
&\|\bar{\boldsymbol{c}}_i\|_1 \leq 2d \text{ and } \bar{\boldsymbol{c}}_i \bmod X - \zeta^{2i+1} \neq 0 \text{ for all } 0 \leq i < d,\\
&\|\bar{\boldsymbol{c}}_i \vec{\boldsymbol{r}}^*\|_\infty \leq 2\beta \text{ and } \|\bar{\boldsymbol{c}}_i \vec{\boldsymbol{u}}^*\|_\infty \leq 4\gamma + 2 \text{ for all } 0 \leq i < d,\\
&\boldsymbol{B} \vec{\boldsymbol{r}}^* + \vec{\boldsymbol{u}}^* = 2^D \vec{\boldsymbol{t}}_{0,1},\\
&\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}_i^* = \boldsymbol{t}_i \text{ for } i \in [n].
\end{aligned}$$

Identically as in [ALS20] we can argue that the commitment scheme is binding with respect to extended weak openings if the infinity-norm M-SIS is hard.

Definition B.5 (Infinity-norm $\mathsf{MSIS}_{n,m,\beta_{\mathrm{SIS}}}$). *The goal in the infinity-norm* Module-SIS *problem with parameters* $n, m > 0$ *and* $\beta_{\mathrm{SIS}} > q$ *is to find, for a given matrix* $\boldsymbol{A} \leftarrow \mathcal{R}_q^{n \times m}$*,* $\vec{\boldsymbol{x}} \in \mathcal{R}_q^m$ *such that* $\boldsymbol{A}\vec{\boldsymbol{x}} = \vec{\boldsymbol{0}}$ *over* $\mathcal{R}_q$ *and* $0 < \|\vec{\boldsymbol{x}}\|_\infty \leq \beta_{\mathrm{SIS}}$*. We say that a PPT adversary* $\mathcal{A}$ *has advantage* ϵ *in solving* $\mathsf{MSIS}_{n,m,\beta_{\mathrm{SIS}}}$ *if*

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

For our practical security estimations, the parameter m does not play a crucial role. Therefore, we sometimes simply omit m and use the notation $\mathsf{MSIS}_{n,\beta_{\mathrm{SIS}}}$

Lemma B.6. *The commitment scheme is binding with respect to extended weak openings if the infinity-norm* $\mathsf{MSIS}_{\kappa,\beta_{\mathrm{SIS}}}$ *is hard where*

$$\beta_{\mathrm{SIS}} = 8d \cdot \max\{\beta, 2\gamma + 1\}.$$

More precisely, from two different extended weak openings $((\bar{\boldsymbol{c}}_i), \vec{\boldsymbol{r}}^*, \boldsymbol{m}^*)$ *and* $((\bar{\boldsymbol{c}}_i'), \vec{\boldsymbol{r}}^{*\prime}, \boldsymbol{m}^{*\prime}, \vec{\boldsymbol{u}}^*)$ *with* $\boldsymbol{m}_j^* \neq \boldsymbol{m}_j^{*\prime}$ *for some* $j \in [n]$*, one can immediately compute an infinity-norm Module-SIS solution for* $\boldsymbol{B}_0 = (\boldsymbol{B} \parallel \boldsymbol{I}_\kappa)$ *of norm at most* β_{SIS}*, where* $\boldsymbol{I}_\kappa$ *is the identity matrix over* $\mathcal{R}_q^{\kappa \times \kappa}$*.*

The soundness argument works similarly as in [ALS20, Theorem 2.1]. Concretely, the extractor $\mathcal{E}$ repeatedly runs $\mathcal{P}$ with freshly sampled challenges until it hits an accepting transcript. Let $\vec{\boldsymbol{w}}_{i,1}$, $\boldsymbol{c}, \vec{\boldsymbol{h}}_i$ and $\vec{\boldsymbol{z}}_i$ be the prover commitments, challenge and prover replies in this transcript, respectively. Then, $\mathcal{E}$ wants to get d/k more accepting transcripts such that for each of the d/k ideals $(X^k - \zeta^{jk})$, $j \in \mathbb{Z}_{2d/k}^\times$, there is a transcript whose challenge differs from $\boldsymbol{c}$ modulo the ideal. Moreover, these transcripts need to contain the same prover commitments $\vec{\boldsymbol{w}}_{i,1}$ as in the first accepting transcript.

To this end, for every j, $\mathcal{E}$ repeatedly rewinds the prover to just after the first flow and sends a random challenge that is different from $\boldsymbol{c}$ modulo $(X^k - \zeta^{jk})$ until the resulting transcript with challenge $\boldsymbol{c}_j$ and replies $\vec{\boldsymbol{h}}_{ij}, \vec{\boldsymbol{z}}_{ij}$ is accepting. We write $\bar{\boldsymbol{c}}_j = \boldsymbol{c} - \boldsymbol{c}_j$ for the challenge differences. By construction, $\bar{\boldsymbol{c}}_j \bmod (X^k - \zeta^{jk}) \neq 0$.

The expected runtime for the whole process is as follows. Suppose the first transcript takes expected time $1/\varepsilon$. Next, when restricting to challenges that are different modulo one of the ideals $(X^k + \zeta^{jk})$, the remaining success probability is at least $\varepsilon - \mathsf{p}^k$ (where p is defined in Section 2.4). So in expected time at most

$$\frac{1}{\varepsilon} + \frac{d}{k}\frac{1}{\varepsilon - \mathsf{p}^k}$$

the extractor has the $1 + d/k$ accepting transcripts.

Now fix an index $(e, f) \in I = \{0, \ldots, k-1\} \times \mathbb{Z}^{\times}_{2d/k}$ and consider the associated prime ideal $\mathfrak{p}_{ef} = \sigma^e(X - \zeta^f)$ dividing $(X^{kd} - \zeta^{fk})$. One of the permutations of $\bar{\boldsymbol{c}}_f$ is non-zero modulo $\mathfrak{p}_{ef}$. So there exists at least one $e' = e'(e, f) \in \{0, \ldots, k-1\}$ such that $\sigma^{e'}(\bar{\boldsymbol{c}}_f) \bmod \mathfrak{p}_{ef} \neq 0$. Now, we set

$$\vec{\boldsymbol{r}}^*_{ef} = \frac{\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f}}{\sigma^{e'}(\bar{\boldsymbol{c}}_f)} \bmod \sigma^e\left(X - \zeta^f\right).$$

Next, let $\vec{\boldsymbol{r}}^* \in \mathcal{R}_q^{\lambda+\kappa+1}$ be such that $\vec{\boldsymbol{r}}^* \equiv \vec{\boldsymbol{r}}^*_{ef} \pmod{\sigma^e(X - \zeta^f)}$ for all $(e, f) \in I$. We claim $\sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{r}}^* = \vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}_{ij}$ for all $(i, j) \in I$, unless we find an infinity-norm Module-SIS solution for $(\boldsymbol{B} \parallel \boldsymbol{I}_\kappa)$. From the verification equations and Lemma B.1 we have

$$\begin{aligned}
\|\boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c}) \cdot 2^D\vec{\boldsymbol{t}}_{0,1} - 2\gamma \cdot \vec{\boldsymbol{w}}_{i,1}\|_\infty &\leq 2\gamma + 1 \\
\|\boldsymbol{B}\vec{\boldsymbol{z}}_{ij} - \sigma^i(\boldsymbol{c}_j) \cdot 2^D\vec{\boldsymbol{t}}_{0,1} - 2\gamma \cdot \vec{\boldsymbol{w}}_{i,1}\|_\infty &\leq 2\gamma + 1.
\end{aligned}$$

for all $(i, j) \in I$. Hence, by the triangle inequality:

$$\boldsymbol{B}(\vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}_{ij}) + \vec{\boldsymbol{u}}_{ij} = \sigma^i(\bar{\boldsymbol{c}}_j) \cdot 2^D\vec{\boldsymbol{t}}_{0,1} \tag{20}$$

for some vector $\vec{\boldsymbol{u}}_{ij} \in \mathcal{R}_q^\kappa$ such that $\|\vec{\boldsymbol{u}}_{ij}\|_\infty \leq 4\gamma + 2$. Similarly for $(e', f) \in I$ we can conclude:

$$\boldsymbol{B}(\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f}) + \vec{\boldsymbol{u}}_{e'f} = \sigma^{e'}(\bar{\boldsymbol{c}}_f) \cdot 2^D\vec{\boldsymbol{t}}_{0,1}$$

for some $\vec{\boldsymbol{u}}_{e'f} \in \mathcal{R}_q^\kappa$ such that $\|\vec{\boldsymbol{u}}_{e'f}\|_\infty \leq 4\gamma + 2$. Therefore, either

$$\begin{cases}
\sigma^{e'}(\bar{\boldsymbol{c}}_f)(\vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}_{ij}) = \sigma^i(\bar{\boldsymbol{c}}_j)(\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f}) \\
\sigma^{e'}(\bar{\boldsymbol{c}}_f)\vec{\boldsymbol{u}}_{ij} = \sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{u}}_{e'f}
\end{cases} \tag{21}$$

or we have found a non-trivial infinity-norm Module-SIS solution for $\boldsymbol{B}_0 = (\boldsymbol{B} \parallel \boldsymbol{I}_\kappa)$

$$\begin{pmatrix}
\sigma^{e'}(\bar{\boldsymbol{c}}_f)(\vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}_{ij}) - \sigma^i(\bar{\boldsymbol{c}}_j)(\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f}) \\
\sigma^{e'}(\bar{\boldsymbol{c}}_f)\vec{\boldsymbol{u}}_{ij} - \sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{u}}_{e'f}
\end{pmatrix}$$

with the infinity norm at most

$$8d \cdot \max\{\beta, 2\gamma + 1\}.$$

Then,

$$\begin{aligned}\sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{r}}^* &\equiv \sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{r}}^*_{ef} \\ &\equiv \sigma^i(\bar{\boldsymbol{c}}_j)\frac{\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f}}{\sigma^{e'}(\bar{\boldsymbol{c}}_f)} \\ &\equiv \vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}_{ij} \pmod{\sigma^e(X - \zeta^f)},\end{aligned}$$

and the claim follows from the Chinese remainder theorem.

Next, let us also define

$$\vec{\boldsymbol{u}}^*_{ef} = \frac{\vec{\boldsymbol{u}}_{e'f}}{\sigma^{e'}(\bar{\boldsymbol{c}}_f)} \bmod \sigma^e\left(X - \zeta^f\right)$$

and set $\vec{\boldsymbol{u}}^* \in \mathcal{R}_q^\kappa$ be such that $\vec{\boldsymbol{u}}^* \equiv \vec{\boldsymbol{u}}^*_{ef} \pmod{\sigma^e(X - \zeta^f)}$ for all $(e, f) \in I$. By (21) we can also conclude that $\vec{\boldsymbol{u}}_{ij} = \sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{u}}^*$ for $(i, j) \in I$. Therefore, the following holds:

$$\boldsymbol{B}\sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{r}}^* + \sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{u}}^* = \sigma^i(\bar{\boldsymbol{c}}_j) \cdot 2^D\vec{\boldsymbol{t}}_{0,1}$$

for all $(i, j) \in I$. In particular, we get

$$\boldsymbol{B}\vec{\boldsymbol{r}}^* + \vec{\boldsymbol{u}}^* = 2^D\vec{\boldsymbol{t}}_{0,1}.$$

Finally, we compute the extracted message $\boldsymbol{m}^*$ which we set to fulfil the equation

$$\boldsymbol{t}_1 = \langle\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^*\rangle + \boldsymbol{m}^*.$$

We conclude that the extractor has obtained an extended weak opening $(\sigma^i(\bar{\boldsymbol{c}}_j), \vec{\boldsymbol{r}}^*, \vec{\boldsymbol{u}}^*, \boldsymbol{m}^*)$ for the commitment $\vec{\boldsymbol{t}}$. In particular, it is true that $\left\|\sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{r}}^*\right\|_\infty \le 2\beta$ and $\left\|\sigma^i(\bar{\boldsymbol{c}}_j)\vec{\boldsymbol{u}}^*\right\|_\infty \le 4\gamma + 2$ for all $(i, j) \in I$.

We turn to the $\vec{\boldsymbol{y}}^*_i$. Set them to be the vectors defined by

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

Now, consider an arbitrary accepting transcript with the same prover commitments $\vec{\boldsymbol{w}}_i$ as above, but possibly a different challenge $\boldsymbol{c}'$ and different last messages $\vec{\boldsymbol{z}}'_i$. Then, for a moment write $\vec{\boldsymbol{z}}'_i = \vec{\boldsymbol{y}}^{*\prime}_i + \sigma^i(\boldsymbol{c}')\vec{\boldsymbol{r}}^*$. We aim to show $\vec{\boldsymbol{y}}^*_i = \vec{\boldsymbol{y}}^{*\prime}_i$. From the verification equations for $\vec{\boldsymbol{z}}_i, \vec{\boldsymbol{z}}'_i$ and Lemma B.1 we have:

$$\begin{aligned}\|\boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c}) \cdot 2^D\vec{\boldsymbol{t}}_{0,1} - 2\gamma \cdot \vec{\boldsymbol{w}}_{i,1}\|_\infty &\le 2\gamma + 1 \\ \|\boldsymbol{B}\vec{\boldsymbol{z}}'_i - \sigma^i(\boldsymbol{c}') \cdot 2^D\vec{\boldsymbol{t}}_{0,1} - 2\gamma \cdot \vec{\boldsymbol{w}}_{i,1}\|_\infty &\le 2\gamma + 1.\end{aligned}$$

Then, by the triangle inequality we have:

$$\boldsymbol{B}(\vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}'_i) + \vec{\boldsymbol{u}}'_i = \sigma^i(\bar{\boldsymbol{c}})2^D\vec{\boldsymbol{t}}_{0,1}.$$

for all $i \in \{0, \ldots, k-1\}$ where $\bar{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c}'$ and $\|\vec{\boldsymbol{u}}'_i\|_\infty \le 4\gamma + 2$. Combining this with (20), unless we find a Module-SIS solution for $(\boldsymbol{B} \parallel \boldsymbol{I}_\kappa)$,

$$\sigma^{e'}(\bar{\boldsymbol{c}}_f)(\vec{\boldsymbol{z}}_i - \vec{\boldsymbol{z}}'_i) = \sigma^i(\bar{\boldsymbol{c}})(\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f}).$$

Therefore, since $\vec{\boldsymbol{z}}_{e'} - \vec{\boldsymbol{z}}_{e'f} = \sigma^{e'}(\bar{\boldsymbol{c}}_f)\vec{\boldsymbol{r}}^*$,

$$\sigma^{e'}(\bar{\boldsymbol{c}}_f)(\vec{\boldsymbol{y}}^*_i - \vec{\boldsymbol{y}}^{*\prime}_i) = 0.$$

Recall $\sigma^{e'}(\bar{\boldsymbol{c}}_f) \not\equiv 0 \pmod{\mathfrak{p}_{ef}}$. Hence, $\vec{\boldsymbol{y}}^*_i \equiv \vec{\boldsymbol{y}}^{*\prime}_i \pmod{\mathfrak{p}_{ef}}$, and thus $\vec{\boldsymbol{y}}^*_i = \vec{\boldsymbol{y}}^{*\prime}_i$.

Simulatability. One can show similarly as in the proof of Theorem 3.4 and [DKL+18, Appendix B] that the protocol is simulatable under the Extended M-LWE$_{\kappa+1,k,\lambda,\chi^d,D_s^d,C'}$ where probability distribution C' over $\mathcal{R}^k$ is defined as: first sample $\boldsymbol{c} \leftarrow C$ and output $(\sigma^0(\boldsymbol{c}), \ldots, \sigma^{k-1}(\boldsymbol{c}))$. We only remark that after perfectly simulating $\vec{\boldsymbol{z}}_i, \vec{\boldsymbol{t}}_{0,1}, \vec{\boldsymbol{t}}_{0,0}$ one can then set the hint

$$\vec{\boldsymbol{h}}_i := \mathsf{MakeHint}_q\left(-\sigma^i(\boldsymbol{c})\boldsymbol{t}_{0,0}, \boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c}) \cdot 2^D \boldsymbol{t}_{0,1}, 2\gamma\right).$$

Also, the simulator can manually check conditions on $\vec{\boldsymbol{l}}_{i,1}$ and $\vec{\boldsymbol{l}}_{i,0}$ since $\vec{\boldsymbol{w}}_i - \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}_2 = \boldsymbol{B}\vec{\boldsymbol{z}}_i - \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0$.

B.4 Setting Parameters

The only additional parameters which get introduced when using the commitment compression techniques are γ, δ, ω and D where $2\gamma | q-1$. First, γ needs be set such that $\mathsf{MSIS}_{\kappa,\beta_{\text{SIS}}}$ is hard where

$$\beta_{\text{SIS}} = 8d \cdot \max\{\beta, 2\gamma + 1\}.$$

Next, δ is the bound on $\|\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}_2\|_\infty$. For Lemma B.2 to work, we need $\delta < \gamma$. Also, in order to apply Lemma B.1, we need

$$\|\sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_{0,0}\|_\infty \leq \gamma$$

for all $i = 0, 1, \ldots, k-1$ and $\boldsymbol{c} \leftarrow C$. Since $\|\vec{\boldsymbol{t}}_{0,0}\|_\infty \leq 2^{D-1}$, we only need to define γ such that

$$d2^{D-1} \leq \delta.$$

As for ω, the parameter is set so that heuristically the Hamming weight of $\vec{\boldsymbol{h}}_i$ is greater than ω with probability less than 1% as in [DKL+18].

When computing proof sizes using the commitment compression, we already include the additional optimisations described in Section 3.2, especially reducing the Hamming weight of a challenge from d to d/k.

Proving	$\log(q)$	d	k	λ	κ	$\log(\gamma)$	δ	D	proof size
Knowledge of an LWE Sample	32	128	4	10	9	17	32	14	33.3 KB
Integer Addition	30	128	4	10	10	17	32	14	16.9 KB
Integer Multiplication	30	128	4	10	10	17	32	14	28.2 KB

Fig. 7. Proof size for proving knowledge of a LWE sample, 128-bit integer addition and multiplication using commitment compression.

C Further Reducing the Standard Deviation

We recall that Lyubashevsky et al. [LNS20] bound $T = \|\vec{\boldsymbol{v}}\|$, where $\vec{\boldsymbol{v}} = \boldsymbol{c}_0\vec{\boldsymbol{r}} \,\|\, \cdots \,\|\, \boldsymbol{c}_{k-1}\vec{\boldsymbol{r}}$ and $\boldsymbol{c}_i, \vec{\boldsymbol{r}}$ are defined in Section 3.2, by first finding α so that

$$\Pr\left[\exists i, \|\boldsymbol{c}_i\vec{\boldsymbol{r}}\|_\infty > \alpha : \boldsymbol{c} \leftarrow C, \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}\right] < 2^{-128}$$

and then setting the bound $\|\vec{\boldsymbol{v}}\| \leq \alpha\sqrt{\lambda+\kappa+1}$. In this section, we show an alternative way to compute an upper-bound on $\|\vec{\boldsymbol{v}}\|$ using similar methods as in [DDLL13].

We introduce the following definition. Firstly, for a polynomial $\boldsymbol{a} \in \mathcal{R}_q$, we define $\mathsf{Rot}(\boldsymbol{a}) \in \mathbb{Z}_q^{d\times d}$ as

$$\mathsf{Rot}(\boldsymbol{a}) = \begin{pmatrix} a_0 & -a_{n-1} & \cdots & -a_1 \\ a_1 & a_0 & \cdots & -a_2 \\ \vdots & \vdots & \cdots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{pmatrix}.$$

Then, for $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}_q$, the coefficient vector of $\boldsymbol{ab}$ can be calculated as $\mathsf{Rot}(\boldsymbol{a})\vec{b}$ where $\vec{b} = (b_0, b_1, \ldots, b_{d-1})$ is the coefficient vector of $\boldsymbol{b}$.

Next, for $A \in \mathbb{Z}_q^{n\times m}$ and $\alpha \leq n$, we define:

$$N_\alpha(A) = \max_{I\subseteq[m],|I|=\alpha} \sum_{i\in I}\sum_{j\in I} |T_{i,j}| \text{ where } T = A^T A.$$

Then, we have the following simple lemma.

Lemma C.1. *For any matrix $A \in \mathbb{Z}_q^{n\times m}$ and $\vec{c} \in \{-1,0,1\}^m$ such that $\|\vec{c}\|_1 \leq \alpha$. Then, $\|A\vec{c}\| \leq \sqrt{N_\alpha(A)}$.*

Proof. Let $J \subseteq [m]$ be the set of indices j such that $c_j \in \{-1,1\}$. We can then easily expand J to a set I such that $J \subseteq I$ and $|I| = \alpha$. Hence, if we define $T = A^T A$ then we have:

$$\|A\vec{c}\|^2 = \vec{c}^t A^t A\vec{c} \leq \sum_{i\in J}\sum_{j\in J} |T_{i,j}| \leq \sum_{i\in I}\sum_{j\in I} |T_{i,j}| \leq N_\alpha(A).$$

□

Let us fix $i \in \{0,1,\ldots,k-1\}$. We will bound $\|\boldsymbol{c}_i\vec{\boldsymbol{r}}\|$ in the following way. Let $\vec{c}$ be the coefficients of $\vec{\boldsymbol{c}}_i$. Then, we have $\|\vec{c}\|_1 \leq d/k$. Then, we set the matrix R as

$$R = \begin{pmatrix} \mathsf{Rot}(\boldsymbol{r}_0) \\ \vdots \\ \mathsf{Rot}(\boldsymbol{r}_{\lambda+\kappa}) \end{pmatrix} \in \mathbb{Z}_q^{(\lambda+\kappa+1)d\times d}.$$

Clearly, we have $\|\boldsymbol{c}_i\vec{\boldsymbol{r}}\| = \|R\vec{c}\|$. Therefore, by Lemma C.1, $\|R\vec{c}\| \leq \sqrt{N_{d/k}(R)}$.

The only question left is how to efficiently bound $\sqrt{N_{d/k}(R)}$. We apply the approximation by Ducas et al. [DDLL13, Section 4.1]. Concretely, we compute the matrix $T = R^T R$ and put absolute values for all entries in T. Next, for each column i of T, we compute the sum s_i of d/k largest elements in the i-th column. Eventually, we output the sum of d/k largest values in the vector $(s_0, s_1, \ldots, s_{d-1})$.

This approximation can be described even easier when considering the algebraic structure of R. Indeed, note that

$$T = R^T R = \sum_{i=0}^{\lambda+\kappa} \mathsf{Rot}(\boldsymbol{r}_i)^T \mathsf{Rot}(\boldsymbol{r}_i) = \sum_{i=0}^{\lambda+\kappa} \mathsf{Rot}\left(\sigma_{-1}(\boldsymbol{r}_i)\boldsymbol{r}_i\right) = \mathsf{Rot}\left(\sum_{i=0}^{\lambda+\kappa} \sigma_{-1}(\boldsymbol{r}_i)\boldsymbol{r}_i\right).$$

Therefore, if we put the absolute value in all entries of T, then T becomes a circulant matrix and thus the d/k largest values in each column are the same. Hence, we can describe the bound on $N_{d/k}(R)$ as $N_{d/k}(R) \leq d/k \cdot (\text{ sum of the } d/k \text{ largest coefficients of } \sum_{i=0}^{\lambda+\kappa} \sigma_{-1}(\boldsymbol{r}_i)\boldsymbol{r}_i \text{ in the absolute value})$. Finally, we bound $\|\vec{\boldsymbol{v}}\|$ as

$$\|\vec{\boldsymbol{v}}\|^2 = \sum_{i=0}^{k-1} \|\boldsymbol{c}_i \vec{\boldsymbol{r}}\|^2 \leq k \cdot N_{d/k}(R).$$

Identically as in [DDLL13], we will force the randomness $\vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+1)d}$ to satisfy $N_{d/k}(R) \leq t$ for some threshold t. In concrete instantiations, e.g. Section 3.3, we will heuristically choose t so that the probability of $N_{d/k}(R) \leq t$ is around 99%.

D Extended LWE

In this section we analyse hardness of the *Extended LWE* (ELWE) problem analogous to the one in Definition 3.3 but without any polynomial structure. The problem can thus be phrased as follows. Let χ, ξ_1 and ξ_2 be efficiently sampleable probability distributions over $\mathbb{Z}$, $\mathbb{Z}^\ell$ and $\mathbb{Z}^{\ell \times m}$ respectively. Then, the challenger generates matrices $B \leftarrow \mathbb{Z}_q^{(m-n)\times m}, C \leftarrow \xi_2$ and vectors $\vec{r} \leftarrow \chi^m, \vec{z} \leftarrow \xi_1$ and $\vec{u} \leftarrow \mathbb{Z}_q^{m-n}$. Then, it flips a bit $b \leftarrow \{0,1\}$ and if $b = 1$ then it sets $\vec{u} = B\vec{r}$. Eventually, the challenger outputs

$$(B, \vec{u}, \vec{z}, C, \mathsf{sign}(\langle \vec{z}, C\vec{r} \rangle)) .$$

We say an adversary $\mathcal{A}$ has advantage ϵ if it can distinguish whether $b = 0$ or $b = 1$ with probability ϵ.

Assume that the challenger sends $\langle \vec{z}, C\vec{r} \rangle \in \mathbb{Z}$ in the clear instead of the sign. Then, the "no-noise" version of the Extended-LWE problem defined by Alpen-Sheriff and Peikert [AP12] can be seen as the Extended-LWE with $\ell = m$ and C being the identity matrix.

We will prove using similar techniques as in [AP12] that hardness of ELWE can be reduced to plain LWE.

Lemma D.1. *Let $T \in \mathbb{R}_+$, χ, ξ_1 and ξ_2 be efficiently sampleable probability distributions over $\mathbb{Z}$, $\mathbb{Z}^\ell$ and $\mathbb{Z}^{\ell \times m}$ respectively such that for $\vec{r} \leftarrow \chi^m, \vec{z} \leftarrow \xi_1, C \leftarrow \xi_2$, the probability that $|\langle \vec{z}, C\vec{r} \rangle| < T$ is overwhelming. Suppose there is a PPT adversary $\mathcal{A}$ which has advantage ϵ when solving ELWE. Then, there is also a PPT adversary $\mathcal{S}$ which has advantage $\epsilon/(2T-1) - \mathsf{negl}(n)$ in solving Decisional-LWE.*

Proof. Adversary $\mathcal{S}$ works as follows. It receives an LWE instance in knapsack form: $B \in \mathbb{Z}_q^{(m-n)\times m}, \vec{u} \in \mathbb{Z}_q^{m-n}$. It then samples $\vec{z} \leftarrow \chi_1, C \leftarrow \chi_2, \vec{r}' \leftarrow \chi^m$ and $\vec{v} \leftarrow \mathbb{Z}_q^{m-n}$ and sets:

$$B' = B - \vec{v}\vec{z}^T C \text{ and } \vec{u}' = \vec{u} - \vec{v} \cdot \langle \vec{z}, C\vec{r}' \rangle.$$

Eventually, $\mathcal{S}$ sends $(B', \vec{u}', \vec{z}, C, \langle \vec{z}, \vec{r}' \rangle)$ to $\mathcal{A}$ and returns the bit output by $\mathcal{A}$.

First, suppose that $B, \vec{u}$ were chosen independently and uniformly at random. Then, $\vec{B}'$ and $\vec{u}'$ are also uniformly random. As a consequence, $\mathcal{S}$ perfectly simulates the case $b = 0$. Next, suppose that $\vec{u} = B\vec{r}$. Then, we have:

$$\vec{u}' = B\vec{r} - \vec{v} \cdot \langle \vec{z}, C\vec{r}' \rangle = B'\vec{r} + \vec{v} \cdot \langle \vec{z}, C\vec{r} - C\vec{r}' \rangle.$$

If $\langle \vec{z}, C\vec{r} \rangle = \langle \vec{z}, C\vec{r}' \rangle \pmod{q}$ then $\vec{u}' = B'\vec{r}$ over $\mathbb{Z}_q$ and thus $\mathcal{S}$ perfectly simulates the case $b = 1$. However, if $\langle \vec{z}, C\vec{r} - C\vec{r}' \rangle$ is non-zero modulo q then for fixed $B, C, \vec{z}, \vec{r}, \vec{r}'$ we have that $\vec{u}$ is uniformly random since $\vec{v}$ is. Consequently, because $\vec{r}, \vec{r}'$ are identically distributed, it follows that $\mathcal{S}$ perfectly simulates the case $b = 0$.

Finally, the probability that $\langle \vec{z}, C\vec{r} \rangle = \langle \vec{z}, C\vec{r}' \rangle$ modulo q is at least $1/(2T-1) - \mathsf{negl}(n)$. The statement now follows from simple calculation. □

The value T can be easily found using the tail bound inequalities e.g. for χ defined as in Section 2.6, $\xi_1 = D_{\mathfrak{s}}^{\ell}$ and $\xi_2 = \mathsf{C}^{\ell \times m}$ where C is defined in 2.7.

E Proving Shortness in the L_2 Norm

In this section we present how to efficiently prove that a vector $\vec{v} \in \mathbb{Z}_q^{\ell d}$, for some $\ell > 0$, is *short* in the L_2 norm. Recall that proving $\|\vec{v}\|_\infty < a$ for some a can be simply done by decomposing each entry of $\vec{v}$ with respect to base b and showing that the extended vector has coefficients in $\{0, \ldots, b-1\}$ e.g. as in Section 4. We show below that the argument for proving $\|\vec{v}\|^2 < a$ is slightly more involved. As usual, we work over the fully-splitting ring $\mathcal{R}_q$ with $l = d$.

Overview. Given a vector $\vec{v} \in \mathbb{Z}^{\ell d}$, we want to prove that $\|\vec{v}\|^2 < a = 2^{\mathsf{a}} < q/2$ for some public positive integer a. By definition of the L_2 norm, we have the following:

$$\|\vec{v}\|^2 = \left(1\ 1 \cdots\ 1\right) \left[\vec{v} \circ \vec{v}\right]. \tag{22}$$

The strategy can be split into four steps.

Step 1. The prover computes $\vec{u} = \vec{v}^2$ and by sending commitments to $\vec{v}, \vec{u}$, it will convince the verifier that the equation holds over $\mathbb{Z}_q$. This is a proof of a simple multiplicative relation.

Step 2. By doing an approximate range proof (see Section 2.7), $\mathcal{P}$ will prove that $\vec{v}$ has relatively small coefficients. Concretely, the prover samples $\vec{y} \leftarrow D_{\mathfrak{s}'}^{\tau d}$ and given a challenge matrix $B \leftarrow \mathsf{C}^{\tau d \times \ell d}$, it computes $\vec{z} = \vec{y} + B\vec{v}$ and applies rejection sampling. After sending $\vec{z}$, the verifier will check that $\|\vec{z}\|_\infty < \frac{1}{2}\sqrt{q/(2\ell d)}$. If $\vec{z}$ indeed satisfies this inequality, then by Lemma 2.5, we have $\|\vec{v}\|_\infty < \sqrt{q/(2\ell d)}$ with probability

$$1 - \max\{p, 1-p\}^{\tau d}.$$

Hence, $\|\vec{v}^2\|_\infty < q/(2\ell d)$ and consequently $\vec{u} = \vec{v}^2$ over integers.

Step 3. Let $\vec{w} = (w_0, \ldots, w_{d-1}) \in \{0,1\}^d$ be the binary decomposition of $\|\vec{v}\|^2 \in \mathbb{N}$. Then, by sending the commitment to $\vec{w}$, $\mathcal{P}$ would prove that

$$\left(1\ 1 \cdots\ 1\right) \vec{u} - \left(1\ 2 \cdots\ 2^{d-1}\right) \vec{w} = \left(1\ 1 \cdots\ 1\right) \vec{u} - \|\vec{v}\|^2 = 0 \pmod{q}. \tag{23}$$

Step 4. Lastly, $\mathcal{P}$ shows that $\|\vec{v}\|^2 < 2^{\mathsf{a}}$ having sent the commitment to its binary decomposition $\vec{w}$. Thus, we simply prove that $\vec{w} \in \{0,1\}^{\mathsf{a}} \times \{0\}^{d-\mathsf{a}}$ or alternatively: $\vec{w} \circ (\vec{w} - \vec{\mathsf{a}}) = \vec{0}$ where $\vec{\mathsf{a}} = (1, \ldots, 1, 0, \ldots, 0)$ has the first a coefficients equal to 1 and 0 everywhere else. Then, since we proved that $\|\vec{u}\|_\infty$ and $\|\vec{w}\|_\infty$ are sufficiently small, we would conclude that (23) also holds over integers. Indeed, by Steps 1 and 2 the verifier is convinced that $\|\vec{u}\|_\infty = \|\vec{v}^2\|_\infty < q/(2\ell d)$. Then,

$$\| \left(1\ 1 \cdots\ 1\right) \vec{u}\|_\infty < \| \left(1\ 1 \cdots\ 1\right) \vec{u}\|_\infty < \ell d \cdot q/(2\ell d) = q/2$$

and $\| \left(1\ 2 \cdots\ 2^{d-1}\right) \vec{w}|_\infty < q/2$. Hence, by Step 4 we get that Equation 23 holds over integers.

We remark that if someone wants to prove smallness of the L_2 norm for an integer a not being a power-of-two, then in Step 4 one would apply the range proof protocol from [LNS20] instead.

E.1 The Protocol

We present the protocol for proving $\|\vec{v}\|^2 < a = 2^{\mathsf{a}}$ in Fig.8. Prover $\mathcal{P}$ starts by sampling vectors $\vec{y} \leftarrow D^{\tau d}_{\mathfrak{s}'}$ from a discrete Gaussian with standard deviation $\mathfrak{s}'$. Then, it generates a BDLOP commitment to

$$\vec{m} = (\vec{v}, \vec{u}, \vec{y}, \vec{w}) \in \mathbb{Z}_q^{(2\ell+\tau+1)d}$$

and sends it to $\mathcal{V}$. Next, the verifier sends a challenge matrix $B \leftarrow \mathsf{C}^{\tau d \times \ell d}$. Then, $\mathcal{P}$ computes $\vec{z}$ as

$$\vec{z} = \vec{y} + B\vec{v} \bmod q \tag{24}$$

and applies rejection sampling. If it does not abort, $\mathcal{P}$ outputs $\vec{z}$.

Now we define $\mathsf{pp}, \mathsf{ulp}$ in order to apply the LNS framework. For the multiplicative relations, we define the following polynomials:

$$P_1(\vec{m}) = \vec{u} - \vec{v}^2 \text{ and } P_2(\vec{m}) = \vec{w} \circ (\vec{w} - \vec{\mathsf{a}}).$$

As for linear relations, we want to prove Equations (23) and 24 over $\mathbb{Z}_q$. They can all be combined into a single equation

$$B'\vec{m} = \begin{pmatrix} 0 \\ \vec{z} \end{pmatrix}$$

for some public matrix $B' \in \mathbb{Z}_q^{(\tau d+1)\times(2\ell+\tau+1)d}$. Thus, we set

$$\mathsf{pp} = \{P_1, P_2\} \text{ and } \mathsf{ulp} = \left(B', (0||\vec{z})\right). \tag{25}$$

Eventually, the prover runs $\mathit{\Pi}^2_{2\ell+\tau+1}(\mathsf{pp}, \mathsf{ulp})$. At the end, the verifier checks the verification equations for $\mathit{\Pi}^2_{2\ell+\tau+1}(\mathsf{pp}, \mathsf{ulp})$ as well as $\|\vec{z}\|_\infty < \frac{1}{2}\sqrt{q/(2\ell d)}$.

E.2 Proving the L_2 Norm as a Commit-and-Prove Functionality

We can define the protocol in Fig. 8 as a "commit-and-prove" functionality $CP = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$ using random oracles $G : \{0,1\}^* \to \{-1,0,1\}^{\tau d \times \ell d}, H : \{0,1\}^* \to \{-1,0,1\}^d$ as follows. Here, G and H output according to the distribution $\mathsf{C}^{\tau d \times \ell d}$ and C respectively. First, we define a relation R_L as:

$$(ck, a, \vec{v}) \iff \vec{v} \in \mathbb{Z}_q^{\ell d} \wedge \|\vec{v}\| < a = 2^{\mathsf{a}} < q/2.$$

Next,

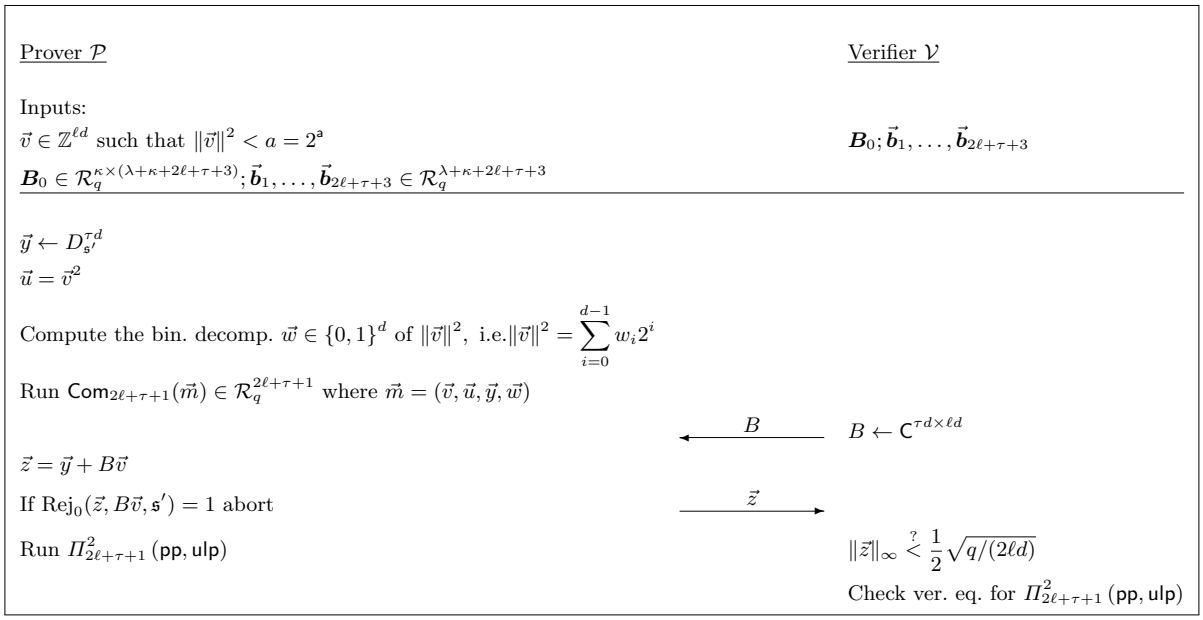

Fig. 8. Protocol for proving that a vector $\vec{v} \in \mathbb{Z}^{\ell d}$ satisfies $\|\vec{v}\|^2 < a$. We use the interactive protocol $\boldsymbol{\pi} = (\mathsf{Com}_{n,\alpha}, \Pi_n^\alpha(\mathsf{pp}, \mathsf{ulp}))$ for $n = 2\ell + \tau + 1$ and $\alpha = 2$ (see Section 2.10) where the latter subprotocol uses the rejection sampling from Section 3. Relations pp and ulp are set as in (25). Rejection sampling algorithm Rej_0 is defined in Fig. 2.

- $ck \leftarrow \mathsf{Gen}(1^\mu)$ specifies the public parameters e.g. q, message space $\mathcal{M}_{ck} = \mathbb{Z}_q^{\ell d}$, randomness space $\mathcal{R}_{ck} = \{-1, 0, 1\}^{(\lambda+\kappa+2\ell+\tau+3)d} \subset \mathcal{R}_q^{\lambda+\kappa+2\ell+\tau+3}$ and commitment space $\mathcal{C}_{ck} = \mathcal{R}_q^{\kappa+\ell}$.
  It also generates the matrix $\boldsymbol{B}_0 \leftarrow \mathcal{R}_q^{\kappa \times (\lambda+\kappa+n+2)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{n+2} \leftarrow \mathcal{R}_q^{\lambda+\kappa+n+2}$ for $n = 2\ell + \tau + 1$.
- $\mathsf{Com}_{ck}(\vec{v}_1, \ldots, \vec{v}_\ell; \vec{\boldsymbol{r}})$: output $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell)$ where

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

- $\mathsf{Prove}_{ck}(a, \vec{v}, \vec{\boldsymbol{r}})$: generate $\vec{y} = (\vec{y}_1, \ldots, \vec{y}_\tau) \leftarrow D_{\mathfrak{s}'}^{\tau d}$ and compute $(\vec{u}_1, \ldots, \vec{u}_\ell) = \vec{u} = \vec{v}^2$. Also, binary decompose $\|\vec{v}\|^2$ into a vector $\vec{w} = (w_0, w_1, \ldots, w_{d-1}) \in \{0,1\}^d$. Next, calculate:

$$\begin{aligned} \boldsymbol{t}_{\ell+i} &= \langle \vec{\boldsymbol{b}}_{\ell+i}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{u}_i) \bmod q \text{ for } i \in [\ell] \\ \boldsymbol{t}_{2\ell+i} &= \langle \vec{\boldsymbol{b}}_{2\ell+i}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{y}_i) \bmod q \text{ for } i \in [\tau] \\ \boldsymbol{t}_{2\ell+\tau+1} &= \langle \vec{\boldsymbol{b}}_{2\ell+\tau+1}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{w}) \bmod q. \end{aligned}$$

  After that, obtain the challenge matrix $B = G(\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+1})$ and compute $\vec{z} = \vec{y} + B\vec{u}$. If $1 \leftarrow \mathsf{Rej}_0(\vec{z}, B\vec{u}, \mathfrak{s}')$, then run

$$\pi' \leftarrow \mathsf{LNSProve}^D_{ck'}((\mathsf{pp}, \mathsf{ulp}), \vec{m}, \vec{\boldsymbol{r}})$$

  where: (i) $\mathsf{pp}, \mathsf{ulp}$ are defined in (25), (ii) ck' denotes a message space $\mathbb{Z}_q^{nd}$, randomness space $\mathcal{R}_{ck} = \{-1, 0, 1\}^{d \cdot (\kappa+\lambda+n+\alpha)}$ and the commitment space $\mathcal{R}_q^{\kappa+n}$ for $n = 2\ell + \tau + 1$ and $\alpha = 2$. Finally, output

$$\pi = (\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+1}, B, \vec{z}, \pi').$$

– $\mathsf{Verify}_{ck}\left(a, \vec{v}, (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell), (\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+3}, B, \vec{z}, \pi')\right)$: return 1 if $\|\vec{z}\|_\infty < \frac{1}{2}\sqrt{q/(2\ell d)}$ and $\mathsf{LNSVerify}^D_{ck'}\left((\mathsf{pp}, \mathsf{ulp}), \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{2\ell+\tau+1}, \pi'\right) = 1$.

Security Analysis. We prove the following properties of CP defined above.

Theorem E.1. *Let q, k be selected such that q^{-k} is negligible. Then, the commit-and-prove $CP = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$ defined above has correctness error $1 - 1/(M' \cdot 2M)$, where M' is the repetition rate for the first rejection sampling Rej_0. Also, it has negligible soundness error under the M-SIS$_{\kappa,8d\beta}$ assumption, where $\beta = \mathfrak{s}\sqrt{2(\kappa+\lambda+2\ell+\tau+1)}$, and is simulatable under the Extended M-LWE$_{\kappa+2\ell+\tau+1,k,\lambda,\chi^d,D^d_{\mathfrak{s}},\tilde{C}}$ assumption* [9].

Proof. Firstly, correctness follows directly from Lemma 3.1 and further analysis in Section 3. As for soundness, suppose there exists a PPT adversary which finds $(a, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi)$ such that $\mathsf{Verify}_{ck}\left(a, (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell), \pi\right) = 1$. Let us parse $\pi = (\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+3}, B, \vec{z}, \pi')$. Consequently, $\mathcal{A}$ found $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{2\ell+\tau+1}$ and π' which satisfy

$$\mathsf{LNSVerify}^D_{ck'}\left((\mathsf{pp}, \mathsf{ulp}), \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{2\ell+\tau+1}, \pi'\right) = 1.$$

Therefore, we can construct an efficient extractor $\mathcal{E}$, similarly as in Section B or [LNS20,ENS20], which finds a weak opening (Definition B.3) to $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{2\ell+\tau+1})$ under the assumption that M-SIS is hard. In particular, it finds $\vec{\boldsymbol{r}}^*$ such that $\boldsymbol{B}_0\vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0$ and sets $\vec{v}^* = (\vec{v}^*_1, \ldots, \vec{v}^*_\ell)$ where

$$\vec{v}^*_i := \mathsf{NTT}\left(\boldsymbol{t}_i - \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^* \rangle\right) \text{ for } i \in [\ell].$$

Similarly it computes $\vec{u}^*, \vec{w}^*, \vec{y}^*$. As shown by [LNS20], these vectors indeed satisfy:

$$\vec{m}^* = (\vec{v}^*, \vec{u}^*, \vec{y}^*, \vec{w}^*) \in \mathcal{L}_{2\ell+\tau+1}(\mathsf{pp}, \mathsf{ulp})$$

As a consequence, we have

$$\vec{u}^* = \vec{v}^* \circ \vec{v}^* \bmod q \text{ and } \vec{w}^* \in \{0,1\}^{\mathsf{a}} \times \{0\}^{d-\mathsf{a}}$$

Also, by Lemma 2.5 we obtain

$$\|\vec{v}\|_\infty < \sqrt{q/(2\ell d)}$$

with probability at least $1 - \max\{p, 1-p\}^{\tau d}$. Therefore,

$$\|\vec{v}^* \cdot \vec{v}^*\|_\infty < \sqrt{q/(2\ell d)}^2 < q/2.$$

This implies that the equation above holds over integers.

Let $t = \left(1\ 2 \cdots\ 2^{d-1}\right)\vec{w}^*$. Then, we know that $t < 2^{\mathsf{a}} = a < q/2$. Now, we just need to prove that the L_2 norm of $\vec{v}^*$ is exactly $\sqrt{t}$. From the linear relations we obtain:

$$\left(1\ 1 \cdots\ 1\right)\vec{u}^* - \left(1\ 2 \cdots\ 2^{d-1}\right)\vec{w}^* = \left(1\ 1 \cdots\ 1\right)\vec{u}^* - t = 0 \pmod q. \tag{26}$$

[9] We recall that distribution $\tilde{C}$ is defined in Section 2.4.

On the other hand, we have $\|\vec{u}\|_\infty = \|\vec{v}^* \circ \vec{v}^*\|_\infty < q/(2\ell d)$. Therefore,

$$\| \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix} \vec{u}^* \|_\infty < \ell d \cdot q/(2\ell d) < q/2.$$

We conclude that over integers:

$$\|\vec{v}^*\|^2 = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix} [\vec{v}^* \circ \vec{v}^*] = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix} \vec{u}^* = t < a.$$

We only sketch out the proof of simulatability since it is almost identical to the proof of Theorem 3.4. Firstly, define the hybrid algorithms Prove^1 and Prove^2 where the first one differs from Prove by programming the random oracle queries and the latter one additionally generates $\vec{z} \leftarrow D_{\mathfrak{s}}^{\tau d}$, sets $\vec{y} = \vec{z} - B\vec{u}$ and outputs the proof with probability $1/M'$. Then, by Lemma 3.1 we have that for all PPT adversaries $\mathcal{A}$:

$$\begin{aligned} &\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (a, \vec{v}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) = \mathsf{Com}_{ck}(\vec{v}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}_{ck}(a, \vec{v}, \vec{\boldsymbol{r}}) : (ck, a, \vec{v}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi) = 1 \end{matrix}\right] \\ &\approx \Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (a, \vec{v}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) = \mathsf{Com}_{ck}(\vec{v}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^2_{ck}(a, \vec{v}, \vec{\boldsymbol{r}}) : (ck, a, \vec{v}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi) = 1 \end{matrix}\right]. \end{aligned} \tag{27}$$

On the other hand, since LNS functionality is simulatable, we can construct algorithms $\mathsf{LNSSimProve}$ and $\mathsf{LNSSimCom}$ which can simulate honest transcripts. Hence, we define simulators SimCom and $\mathsf{SimProve}$ as follows:

- $\mathsf{SimCom}_{ck}(a)$: Sample $\vec{\boldsymbol{t}}_0 \leftarrow \mathcal{R}_q^\kappa$ and $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell \leftarrow \mathcal{R}_q$ uniformly at random and output $\vec{\boldsymbol{t}} = (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell)$.
- $\mathsf{SimProve}_{ck}(a, \vec{\boldsymbol{t}})$: Pick $\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+1} \leftarrow \mathcal{R}_q$ uniformly random, $B \leftarrow \mathsf{C}^{\tau d \times \ell d}$ and $\vec{z} \leftarrow D_{\mathfrak{s}'}^{\tau d}$. Then, get
$$\pi' \leftarrow \mathsf{LNSSimProve}_{ck'}\left((\mathsf{pp}, \mathsf{ulp}), \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{2\ell+\tau+1}\right)$$
and output
$$\pi = (\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+1}, B, \vec{z}, \pi')$$
with probability $1/M$.

We claim that for all PPT adversaries $\mathcal{A}$:

$$\begin{aligned} &\Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (a, \vec{v}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+n+2)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) = \mathsf{Com}_{ck}(\vec{v}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^2_{ck}(a, \vec{v}, \vec{\boldsymbol{r}}) : (ck, a, \vec{v}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi) = 1 \end{matrix}\right] \\ &\approx \Pr\left[\begin{matrix} ck \leftarrow \mathsf{Gen}(1^\mu); (a, \vec{v}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+n+2)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) = \mathsf{SimCom}_{ck}(a); \\ \pi \leftarrow \mathsf{SimProve}_{ck}(a, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) : (ck, a, \vec{v}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi) = 1 \end{matrix}\right]. \end{aligned} \tag{28}$$

for $n = 2\ell + \tau + 1$.

Suppose there is an efficient adversary $\mathcal{A}$ which distinguishes the two cases with probability ϵ. We construct $\mathcal{B}$ which will try to win the simulatability game of LNS^D as follows. When $\mathcal{A}$ outputs $(a, \vec{v})$, $\mathcal{B}$ samples $\vec{z} \leftarrow D_{\mathfrak{s}'}^{\tau d}$ and $B \leftarrow \mathsf{C}^{\tau d \times 6\ell d}$ and computes (i) $\vec{u} = \vec{v}^2$, (ii) binary decomposition $\vec{w} \in \{0,1\}^d$ of $\|\vec{v}\|^2$ and (iii) $y = \vec{z} - B\vec{u}$. Then, it constructs the multiplicative and linear relations as in (25) and outputs $((\mathsf{pp}, \mathsf{ulp}), \vec{m})$ where $\vec{m} = (\vec{v}, \vec{u}, \vec{y}, \vec{w})$. When $\mathcal{B}$ is given $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{2\ell+\tau+1}, \pi')$, it outputs $\vec{\boldsymbol{t}} = (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell)$ as a "commitment output" and $\pi = (\boldsymbol{t}_{\ell+1}, \ldots, \boldsymbol{t}_{2\ell+\tau+1}, B, \vec{z}, \pi')$ to $\mathcal{A}$. Eventually, $\mathcal{B}$ passes the bit output by $\mathcal{A}$.

Suppose $\mathcal{B}$ interacts with $\mathsf{LNSCom}_{ck'}$ and $\mathsf{LNSProve}^D_{ck'}$. recall that ck' denotes a message space $\mathbb{Z}_q^{nd}$, randomness space $\mathcal{R}_{ck} = \{-1, 0, 1\}^{d\cdot(\kappa+\lambda+n+\alpha)}$ and the commitment space $\mathcal{R}_q^{\kappa+n}$ for $n = 2\ell + \tau + 1$ and $\alpha = 2$. Then, the vector $\vec{\boldsymbol{t}} = (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell)$ output by $\mathcal{B}$ is indeed a commitment $\mathsf{Com}_{ck}(\vec{v}; \vec{\boldsymbol{r}})$ and π is the proof output by Prove^2. Hence, we have:

$$\begin{aligned}
&\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Gen}(1^\mu); (a, \vec{v}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+n+2)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) = \mathsf{Com}_{ck}(\vec{v}; \vec{\boldsymbol{r}}); \\ \pi \leftarrow \mathsf{Prove}^2_{ck}(a, \vec{v}, \vec{\boldsymbol{r}}) : (ck, a, \vec{v}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi) = 1 \end{array}\right] \\
&= \Pr\left[\begin{array}{l} ck' \leftarrow \mathsf{Gen}(1^\mu); ((\mathsf{pp}, \mathsf{ulp}), \vec{m}) \leftarrow \mathcal{B}(ck'); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+n+2)d}; \\ (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n) \leftarrow \mathsf{LNSCom}_{ck'}(\vec{m}; \vec{\boldsymbol{r}});\ \pi \leftarrow \mathsf{LNSProve}^D_{ck'}((\mathsf{pp}, \mathsf{ulp}), \vec{m}, \vec{\boldsymbol{r}}) : \\ (ck, (\mathsf{pp}, \mathsf{ulp}), \vec{m}) \in R_{LNS} \wedge \mathcal{B}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n, \pi) = 1 \end{array}\right]
\end{aligned} \tag{29}$$

Now, assume that $\mathcal{B}$ interacts with $\mathsf{LNSSimCom}_{ck'}$ and $\mathsf{LNSSimProve}^D_{ck'}$. We recall from [ALS20] that $\mathsf{LNSSimCom}_{ck'}$ simply outputs $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n) \leftarrow \mathcal{R}_q^{\kappa+n}$. Hence, the vector $\vec{\boldsymbol{t}}$ sent from $\mathcal{B}$ to $\mathcal{A}$ indeed has the same distribution as SimCom_{ck}. Similarly, the proof π constructed from $\mathcal{B}$ is generated exactly as in $\mathsf{SimProve}_{ck}$. Therefore:

$$\begin{aligned}
&\Pr\left[\begin{array}{l} ck \leftarrow \mathsf{Gen}(1^\mu); (a, \vec{v}) \leftarrow \mathcal{A}(ck); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+n+2)d}; (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) = \mathsf{SimCom}_{ck}(a); \\ \pi \leftarrow \mathsf{SimProve}_{ck}(a, \vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell) : (ck, a, \vec{v}) \in R_L \wedge \mathcal{A}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_\ell, \pi) = 1 \end{array}\right] \\
&= \Pr\left[\begin{array}{l} ck' \leftarrow \mathsf{Gen}(1^\mu); ((\mathsf{pp}, \mathsf{ulp}), \vec{m}) \leftarrow \mathcal{B}(ck'); \vec{\boldsymbol{r}} \leftarrow \chi^{(\lambda+\kappa+n+2)d}; \\ (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n) \leftarrow \mathsf{LNSSimCom}_{ck'}(\mathsf{pp}, \mathsf{ulp});\ \pi \leftarrow \mathsf{LNSSimProve}^D_{ck'}(\mathsf{pp}, \mathsf{ulp}) : \\ (ck, (\mathsf{pp}, \mathsf{ulp}), \vec{m}) \in R_{LNS} \wedge \mathcal{B}(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n, \pi) = 1 \end{array}\right].
\end{aligned} \tag{30}$$

Since the LNS functionality is simulatable under the Extended M-LWE assumption, we conclude that ϵ has to be negligible. Thus, the statement holds by the hybrid argument. □

E.3 Proof Size

We provide concrete parameters and proof sizes in Tables 9 and 10. For the LNS^D framework protocol, we apply Equation 14 for $\alpha = 2$ and $n = 2\ell + \tau + 1$. Also, we have to take into account the sizes of $\vec{z}$. Hence, the total proof size is at most:

$$(2\ell + \tau + \kappa + 4)d \log q \ + \ k(\lambda + \kappa + 2\ell + \tau + 3)d \cdot \log(12\mathfrak{s}) + \tau d \cdot \log\left(12\mathfrak{s}'\right)$$

bits.

For the first rejection sampling we choose $\mathfrak{s}' = 11T'$ where T' is the upper-bound on $\|B\vec{v}\|$. We compute T' using the standard inequality $\|B\vec{v}\| \leq s_1(B) \cdot \|\vec{v}\|$ where $s_1(B)$ is the singular value/operator norm of B. Note that each entry of B is sampled uniformly and independently from C and the distribution is actually 0-subgaussian with parameter $s = \sqrt{2\pi}$ [10]. Hence, it follows from [MP12, Lemma 2.9] that

$$s_1(B) \leq \sqrt{\tau d} + \sqrt{\ell d} + 6$$

except with probability 2^{-163}. Thus, we set $T' = (\sqrt{\tau d} + \sqrt{\ell d} + 6) \cdot a$.

In order for the verification equations to hold in an honest execution, we need to select q such that

$$6 \cdot 11(\sqrt{\tau d} + \sqrt{\ell d} + 6) \cdot a = 6 \cdot 11T' = 6\mathfrak{s}' < \frac{1}{2}\sqrt{q/(2\ell d)}. \tag{31}$$

[10] Here, we use the definition of a subgaussian distribution from [MP12]

For the second rejection sampling, we compute $\mathfrak{s}$ as in Section 3.3. Therefore, the repetition rates for both rejection samplings are $M' \approx 3$ and $2M \approx 3.3$.

Last but not least, we apply the Dilithium compression techniques from Appendix B in order to further reduce the proof size.

a	ℓ	$\log(q)$	k	κ	λ	ARP	Bimodal	proof size
2^{10}	8	64	2	4	21	1	no	42.7KB
2^{20}	8	64	2	4	21	2	yes	47.76KB
2^{30}	8	128	1	2	40	1	no	58.8KB
2^{40}	8	128	1	2	40	1	no	58.95KB
2^{50}	8	128	1	2	40	2	no	63.03KB

Fig. 9. Proof size comparison for proving $\|\vec{v}\|^2 < a$, where $\vec{v}$ is a 1024-dimensional vector, using the protocol in Fig. 8. For all parameter sets, we choose $(d, l, \tau, p) = (128, 128, 1, 0.5)$. ARP denotes the number of approximate range proofs, i.e. whether we only prove shortness of $\vec{v}$ or we also prove for $\vec{u}$.

a	without compression	with compression
2^{10}	42.7KB	40.8KB
2^{20}	47.76KB	43.2KB
2^{30}	58.8KB	58.9KB
2^{40}	58.95KB	58.9KB
2^{50}	63.03KB	59.5KB

Fig. 10. Proof size comparison for proving $\|\vec{v}\|^2 < a$ with and without Dilithium compression techniques (Appendix B). We observe that for larger modulus $q \geq 2^{64}$, the advantage of compressing the commitment is not very significant.

E.4 Reducing the Modulus

Note that Equation 31 implies that the higher value a, the larger modulus q we need to choose. Indeed, with the approach above, even for $a = 2^{20}$ we already cannot afford to select $q \approx 2^{64}$ and thus we need to raise it to $q \approx 2^{128}$ (in order to apply Lemma 2.1). One method, which does not completely circumvent this problem but allows us to pick slightly smaller q, is to also apply an approximate range proof on $\vec{u}$. Concretely, the prover additionally samples $\vec{y}'' \leftarrow D^{\tau d}_{\mathfrak{s}''}$ from a Discrete Gaussian with standard deviation $\mathfrak{s}''$ and given a challenge matrix $B'' \leftarrow \mathsf{C}^{\ell d \times \tau d}$ it computes $\vec{z}'' = \vec{y}'' + B\vec{u}$. Then, $\mathcal{P}$ applies rejection sampling and outputs $\vec{z}''$ if it does not reject. Eventually, the prover needs to prove $\vec{z}'' = \vec{y}'' + B\vec{u}$ which is simply another linear proof.

We show how this technique compares to the previous approach. First, we choose the standard deviation $\mathfrak{s}'' = 11T''$ where T' is an upper-bound on $\|B\vec{u}\|$. Similarly as before, we pick $T'' = (\sqrt{\tau d} + \sqrt{\ell d} + 6) \cdot a^2$. Then, in order to prove that $\vec{u} = \vec{v}^2$ and (23) over integers, we need: (i) $12\mathfrak{s}'' < q/2$, (ii) $(12\mathfrak{s}')^2 < q/2$ and (iii) $2 \cdot 6\mathfrak{s}'' < q/(2\ell d)$.

Therefore, we can choose $2 \cdot 6 \cdot 11 \cdot (\sqrt{\tau d} + \sqrt{\ell d} + 6)$ times larger modulus q than before. The clear disadvantages of this approach are: (i) committing to one more vector $\vec{y} \in \mathbb{Z}_q^{\tau d}$ and sending a masked opening $\vec{z}''$ and (ii) slower run-time due to having an additional rejection sampling.

Another way, which allows us to further reduce the standard deviation $\mathfrak{s}', \mathfrak{s}''$ and consequently choose smaller q, is to apply Bimodal Gaussians [DDLL13]. Informally, instead of calculating $\vec{z} = \vec{y} + B\vec{v}$, we commit to a random bit b and compute

$$\vec{z} = \vec{y} + (-1)^b B\vec{v}.$$

Eventually, we apply rejection sampling and prove that this equation holds over $\mathbb{Z}_q$. In Appendix F we show how this can be done within the LNS framework. Overall, this approach reduces the standard deviation by a factor of around 12 at the cost of committing to 2τ more vectors in $\mathbb{Z}_q^d$.

F Bimodal Gaussians and Approximate Range Proofs

In Section 2.7 we described how to prove that coefficients of $\vec{s} \in \mathbb{Z}_q^{\ell d}$ are relatively small compared to q. We recall that the prover $\mathcal{P}$ first samples $\vec{y}$ from a discrete Gaussian $D_{\mathfrak{s}'}^{\tau d}$ and sends the BDLOP commitments to $\vec{s}, \vec{y}$. For simplicity, let us set $\tau = 1$ and $p = 0.5$. Then, given a challenge matrix $B \leftarrow \mathsf{C}^{d \times \ell d}$, where its coefficients are sampled from C, $\mathcal{P}$ computes $\vec{z} = \vec{y} + B\vec{s}$ and applies the standard rejection sampling, i.e. Rej_0 from Fig. 2. If it does not abort, it sends $\vec{z}$ and next proves the linear relation: $\vec{z} = \vec{y} + B\vec{s}$.

In order to reduce the standard deviation $\mathfrak{s}'$, we apply the Bimodal Gaussians technique [DDLL13,TWZ20]. Concretely, the prover $\mathcal{P}$ starts by sampling $\vec{y}$ from a discrete Gaussian $D_{\mathfrak{s}'}^{m}$ and sending the BDLOP commitments to $\vec{s}, \vec{y}$ as before. Additionally, it samples a bit $b \leftarrow \{0, 1\}$ uniformly at random and commits to $\vec{b} = (b, b, \ldots, b)$. Then, given a challenge matrix $B \leftarrow \mathsf{C}^{d \times \ell d}$, the prover computes

$$\vec{z} = \vec{y} + (-1)^b \cdot B\vec{s} \tag{32}$$

and applies the rejection sampling $\mathsf{Rej}_2(\vec{z}, B\vec{s}, \mathfrak{s}')$ from Fig. 11. Eventually, $\mathcal{P}$ needs to prove Equation 32. This can be done within the LNS framework as follows. First, the prover computes $\vec{u} = B\vec{s}$ and proves it (unstructured linear proof). Next, it proves that $\vec{z} = \vec{y} + \vec{b} \circ \vec{u}$ (product proof).

$\mathrm{Rej}_2(\vec{z}, \vec{v}, \mathfrak{s})$
01 $u \leftarrow [0, 1)$
02 If $u > \frac{1}{M} \cdot \exp\left(\frac{\|\vec{v}\|^2}{2\mathfrak{s}^2}\right) / \cosh\left(\frac{\langle \vec{z}, \vec{v} \rangle}{\mathfrak{s}^2}\right)$
03 return 1
04 Else
05 return 0

Fig. 11. Bimodal Gaussian rejection sampling [DDLL13].

Last but not least, $\mathcal{P}$ needs to convince the verifier that $\vec{b}$ indeed consists of only 1's or -1's. Note that showing $(\vec{b} - \vec{1}) \circ (\vec{b} + \vec{1}) = \vec{0}$ only proves that $\vec{b} \in \{-1, 1\}^d$. We additionally prove that $\mathsf{NTT}^{-1}\left(\vec{b}\right) \in \mathbb{Z}_q$. As discussed in e.g. [dPLS18], $\boldsymbol{m} \in \mathbb{Z}_q$ if and only if $\sigma_5(\boldsymbol{m}) = \sigma_{-1}(\boldsymbol{m}) = \boldsymbol{m}$. Note that in the fully-splitting ring R_q, where $l = d$, each ring automorphism σ_i permutes the NTT coefficients of the polynomials and thus can represented as a permutation matrix, say $A_i \in \mathbb{Z}_q^{d \times d}$. Therefore, the prover additionally needs to check that $A_5\vec{b} = A_{-1}\vec{b} = \vec{b} \bmod q$ which is again a linear proof.

In the knowledge extraction argument, suppose that one can efficiently extract $\vec{z}^*, \vec{y}^*, \vec{s}^*, \vec{b}^*, \vec{u}^*$ which satisfy all the linear and multiplicative relations above. Then, we conclude that $\vec{z}^* = \vec{y}^* + (-1)^{b^*} \cdot B\vec{s}^*$ for some fixed $b^* \in \{0, 1\}$. Note that since the probability distribution C is symmetric around 0, the matrix $(-1)^{b^*} \cdot B$ is still randomly generated from $\mathsf{C}^{d \times \ell d}$ for a fixed bit b^*. Hence, we can apply Lemma 2.5.

With this technique, we significantly reduce the standard deviation $\mathfrak{s}'$ as described in [DDLL13] at the cost of committing to two more vectors $\vec{b}, \vec{u} \in \mathbb{Z}_q^{\tau d}$. Concretely, for a repetition rate M' we would set $\mathfrak{s}'$ which satisfies:

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

where T' is the upper-bound on the norm of $\|B\vec{s}\|$.