Practical Sublinear Proofs for R1CS from Lattices*

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

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

Abstract. We propose a practical sublinear-size zero-knowledge proof system for Rank-1 Constraint Satisfaction (R1CS) based on lattices. The proof size scales asymptotically with the square root of the witness size. Concretely, the size becomes 2-3 times smaller than Ligero (ACM CCS 2017), which also exhibits square root scaling, for large instances of R1CS. At the core lies an interactive variant of the Schwartz-Zippel Lemma that might be of independent interest.

1 Introduction

Zero-Knowledge proof systems are an important tool in the construction of many cryptographic protocols, especially in the area of privacy-preserving cryptography. This paper is about zero-knowledge proof systems based on techniques and hardness assumptions from lattice cryptography. In recent years there has been a lot of progress in the construction of lattice-based proof systems whose proof sizes scale linearly with the statement size [ESS+19, EZS+19, ALS20, ENS20, LNS20]. The concrete proof sizes for typical statements have been reduced by a factor of about 100 over earlier proof systems. This in turn has made it possible to construct efficient advanced quantum-safe privacy-preserving schemes, for example group and ring signature schemes, that achieve or get near to practically acceptable bandwidth requirements [ESLL19, LNPS21, LNS21b].

On the other hand, the linear scaling of the proof systems implies that they are only practical for proving relatively small statements and great care needs to be taken to minimize the statement sizes when using them in the construction of advanced schemes. For example, the linear-size proof systems can not be used to construct efficient group signature schemes on top of vetted lattice-based basic signature schemes such as the NIST PQC finalists Dilithium [DKL+18] and Falcon [FHK+18]. Dilithium and Falcon involve a hash function that is modeled as a random oracle and proving a preimage to such a hash function would lead to a very large proof size.

For solving this problem and more generally for being able to prove arbitrary circuit satisfaction with lattice-based proof systems, practically efficient sublinear-size proof systems are needed. There are several proposals of asymptotically sublinear lattice-based proof systems in the literature [BBC+18, BLNS20,

* This work is supported by the EU H2020 ERC Project 101002845 PLAZA.

ACK21, AL21], but their concrete proof sizes are not analyzed in the papers and they are not practically efficient. These sublinear-size lattice-based proof systems use adaptations and extensions of techniques from discrete-log-based proof systems. In particular several forms of "folding" stemming from the two-tiered commitment scheme [Gro11] and Bulletproofs [BBB+18]. While folding techniques are very effective in the discrete-log setting and retain asymptotic efficiency in the lattice setting, they do not play nicely with the shortness requirement in lattice cryptography. This leads to a concrete blow-up of the proof size. We exemplify this in the case of lattice-based Bulletproofs.On a high level, it must be possible to invert the folding in the extraction such that the extracted solution vector is still short. For general (short) challenges this will not be the case. In [BLNS20, ACK21] monomial challenges X^i are used that result in a large soundness error which can not be boosted [AF21]. But even when ignoring this problem, the length of the extracted solution vector grows by a factor of $12d^3$ for every level of folding where d is the dimension of the polynomial ring. Then the parameters must be chosen such that the Module Short Integer Solution problem (Module-SIS) is hard with respect to the length of the extracted solution vector, resulting in the need for large integer moduli q. It follows that the length of the extracted solution becomes prohibitively large for less than 10 foldings. When choosing an optimal number of foldings the required modulus q still needs to be in the order of several hundred bits and the proof size turns out to be in excess of 100 Megabytes for typical example applications.

In light of these problems, we construct the first concretely efficient sublinear-size lattice-base zero-knowledge proof system in this paper. Our proof system uses new techniques that avoid any folding and the proof size scales with the square root of the statement size. We apply it for proving R1CS [BCG+13] where it is most efficient and achieves optimal sizes for numbers of constraints above 2^{20}. Because of the square root scaling, we compare our proof system to the PCP-type Ligero proof system [AHIV17], and more specifically to the straightforward extension Ligero-R1CS from [BCR+19] to the R1CS language, which also exhibits square root scaling and is faster and less memory-demanding than other PCP-type proof systems. In the setting over a finite field of size 128 bits our system results in a proof size of 10.79 Megabytes for 2^{24} constraints, whereas Ligero results in 31.83 Megabytes, for the same field size, number of constraints, and comparable soundness error around 2^{-110}.

Outside of lattice-based cryptography there has been tremendous progress in the construction of practical zero-knowledge proof systems and they have progressed to the point where they can be used routinely to prove relatively large arithmetic circuits with practical costs. When restricting to (plausibly) quantum-safe protocols, the PCP-type systems like Ligero++ [BFH+20] or Aurora [BCR+19] achieve proof sizes that scale poly-logarithmically with the witness size and have small concrete base sizes in the order of 100 Kilobytes. Moreover, these systems only rely on unstructured quantum-safe hardness assumptions (hash functions). It is clear that the polylogarithmic proof systems with small concrete costs like e.g. Aurora offer much smaller proof sizes for large

statements than our square-root sized proof system. We use the comparison with Ligero to be able to claim practicality of our proof system. Namely, that our proof system has very small constants for a proof system that asymptotically scales with the square root of the witness size. It is an important and interesting open research question whether it will be possible to improve upon the polylogarithmic PCP-type systems by relying on structured quantum-safe assumptions as for example lattice-based assumptions, which for example has been achieved for basic signature schemes where lattice-based signatures are more efficient than hash-based ones.

Next to the conventional publicly verifiable proof systems this paper is about, there has recently been much work on (lattice-based) proof systems in the designated verifier preprocessing model. For example, [GMNO18], MAC'n'Cheese [BMRS21], Wolverine [WYKW21], QuickSilver [YSWW21], and [ISW21]. These proof systems achieve very practical sizes but are not directly comparable to publicly verifiable protocols.

1.1 Technical Overview

Our proof system from this paper is constructed in two stages and uses the protocols from [ALS20, ENS20] as a building block. First, we construct an exact binary amortized opening proof for many lattice-based hashes. Then we use this proof to prove an opening to a Merkle hash tree via induction over the levels of the tree. Both proofs can be amended to also prove linear and product relations among the preimage coefficients. We now give some more details about the techniques.

Our sublinear-size proof system is presented as a protocol for proving preimages to many collision-resistant hashes $\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$ over a cyclotomic polynomial ring, typically $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^{128}+1)$ with fully splitting prime $q \approx 2^{128}$. The preimages $\vec{\boldsymbol{s}}_i$ are binary and lie in $\{0,1\}^m \subset \mathcal{R}_q^{m/128}$ where m is a multiple of 128. The hashes can be commitments if parts of the $\vec{\boldsymbol{s}}_i$ are random, but our proof system does not rely on this. Concretely, there are n hashes to m bits each, and we want $m \approx n$ and a proof size that is linear in n. Then our proof system scales with the square root of the witness size. We start from an amortized approximate opening proof for all the hashes that is a variant of the protocol in [LNS21a]. In the protocol the prover sends an amortized masked opening

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + x_1\vec{\boldsymbol{s}}_1 + \cdots + x_n\vec{\boldsymbol{s}}_n,$$

where $\vec{\boldsymbol{y}}$ is the masking vector and $x_i \in \mathbb{Z}_q$ are integer challenges. We forget the polynomial structure and let $\vec{s}_i$ be the coefficient vectors corresponding to the $\vec{\boldsymbol{s}}_i$. We then enhance the protocol with a binary proof that shows that all the $\vec{s}_i$ are binary vectors $\vec{s}_i \in \{0,1\}^m$. To this end, we construct the polynomial (in the x_i)

$$f(x_1,\ldots,x_n) = \langle \vec{\varphi}, \vec{z} \circ ((x_1+\cdots+x_n)\vec{1} - \vec{z})\rangle$$

for a challenge vector $\vec{\varphi}$. Here $\circ$ denotes the componentwise product. The terms divisible by x_i^2 for $i \in \{1,\ldots,n\}$ are given by $\langle \vec{\varphi}, \vec{s}_i \circ (\vec{1} - \vec{s}_i)\rangle$ and vanish when

$\vec{s}_i$ is binary, which we want to prove. There are now two problems that we need to overcome to make this work. First, there is a quadratic number $(n^2+n+2)/2$ of terms that we would need to commit to in order to prove that the interesting terms divisible by x_i^2 vanish. These are called garbage commitments and they would be very expensive and not result in a sublinear-size proof system. Secondly, it is not clear how to prove hat $\vec{z}$ is always of the same form with fixed masking vector $\vec{y}$ so that the polynomial f is really independent of the challenges. We solve the first problem with a technique that can be seen as a multi-round interactive variant of the Schwartz-Zippel lemma. The high-level idea is that we prove

$$f(x_1,\dots,x_n) = f_0 + f_1(x_1) + f_2^{(x_1)}(x_2) + \cdots + f_n^{(x_1,\dots,x_{n-1})}(x_n), \qquad (1)$$

where $f_0 \in \mathbb{Z}_q$ and $f_i^{(x_1,\dots,x_{i-1})} \in \mathbb{Z}_q[x_i]$ is a degree-one polynomial in x_i with zero constant coefficient, depending on $x_1,\dots,x_{i-1}$. More precisely, we do not prove that the $f_i^{(x_1,\dots,x_{i-1})}(x_i)$ are in fact multivariate polynomials in $x_1,\dots,x_i$ of degree 2 whose terms x_i^2 vanish. It suffices to prove that they are arbitrary functions from $\mathbb{Z}_q^{i-1}$ to $\mathbb{Z}_q[x_i]$ given by $(x_1,\dots,x_{i-1}) \mapsto f^{(x_1,\dots,x_{i-1})}(x_i)$ where the image polynomials are of the form $\gamma_i x_i$ for all $(x_1,\dots,x_{i-1})$. The important information is that $f_i^{(x_1,\dots,x_{i-1})}$ does not depend on $x_i,\dots,x_n$. This can be proven in a protocol with $2n$ rounds where the prover has to commit to the coefficient γ_i for $f^{(x_1,\dots,x_{i-1})}$ after he has received the challenges $x_1,\dots,x_{i-1}$ but before getting the challenges $x_i,\dots,x_n$. Then, intuitively, if $\vec{s}_i$ is not binary, the prover can not use $\gamma_i x_i$ to make Equation (1) true for all $(x_1,\dots,x_n)$ because the left-hand side contains the non-zero term $\langle\vec{\varphi}, \vec{s}_i \circ (\vec{1}-\vec{s}_i)\rangle x_i^2$ that is quadratic in x_i. He can also not use the later γ_j because they all get multiplied by later challenges x_j that the prover does not know when making the commitments. A precise analysis shows that this argument has soundness error $2n/q$ for uniformly random challenges x_i.

So our protocol will have many rounds but we do not consider this to be a problem as we are only interested in the non-interactive variant where the number of rounds has no direct impact on the performance of the proof system. The interactive variant only serves as a convenient intermediate representation that is easy to reason about. From a theoretical point of view our multi round protocol is simple in that the extraction algorithm is relatively straight-forward compared to for example Bulletproofs where a complicated tree extraction algorithm is needed.

For the second problem we do not know how to prove that $\vec{z}$ must follow the fixed form from using the approximate opening proof protocol alone. But in conjunction with the binary proof protocol it turns out to be provable. The argument proceeds along the following lines. Let $\vec{s}_i^*$ be the bound weak openings to the hashes $\vec{u}_i$ that we can extract from the approximate proof. If they are not all binary, then there is a last non-binary vector $\vec{s}_{i_0}^*$. We can write $\vec{z} - x_{i_0+1}\vec{s}_{i_0+1}^* - \cdots - x_n\vec{s}_n^* = \vec{y}^* + x_{i_0}\vec{s}_{i_0}^*$ in any accepting transcript where $\boldsymbol{A}\vec{y}^* = \vec{w} + x_1\vec{u}_1 + \cdots + x_{i_0-1}\vec{u}_{i_0-1}$. So this can be viewed as a masked opening of the single secret vector $\vec{s}_{i_0}^*$ because the left hand side is short. Then we can use the argument for the non-amortized case to argue that the prover is bound to the

$\vec{\boldsymbol{y}}^*$ in all interactions with fixed first challenges $x_1, \ldots, x_{i_0-1}$. Indeed, if in an accepting transcript $\vec{\boldsymbol{z}}' = \vec{\boldsymbol{y}}^{**} + x'_{i_0}\vec{\boldsymbol{s}}^*_{i_0} + \cdots + x'_n\vec{\boldsymbol{s}}^*_n$ with $\vec{\boldsymbol{y}}^{**} \neq \vec{\boldsymbol{y}}^{**}$, then we can compute a short Module-SIS solution

$$\bar{x}(\vec{\boldsymbol{y}}^* - \vec{\boldsymbol{y}}^{**}) = \bar{x}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}' - (x_{i_0+1} - x'_{i_0+1})\vec{\boldsymbol{s}}^*_{i_0+1} - \cdots - (x_n - x'_n)\vec{\boldsymbol{s}}^*_n) - (x_{i_0} - x'_{i_0})\bar{x}\vec{\boldsymbol{s}}^*_{i_0}$$

for $\boldsymbol{A}$, where $\bar{x}$ is a difference of two challenges such that $\bar{x}\vec{\boldsymbol{s}}^*_{i_0}$ is short. This in turn suffices to show that the prover has small success probability in the binary proof restricted to the vectors $\vec{s}_{i_0}, \ldots, \vec{s}_n$.

Given this exact amortized binary opening proof, we extend it to be also able to prove linear and product relations on the secret vectors. This already provides a sublinear-size proof system even when the size of the commitments $\vec{\boldsymbol{u}}_i$ is counted as part of the proof size. There are n hashes, each of essentially constant size. Unfortunately, this simple sublinear-size proof system is only competitive in a small regime of parameters. We achieve competitive proof sizes for larger parameters in a further protocol where we use the previous exact amortized binary opening proof as a building block to prove knowledge of a Merkle tree opening by induction over the levels of the tree when only the root hash is given (see Section 5).

So we use a Merkle tree with hashes $\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$ for $i = 1, \ldots, 2^a - 1$, where $\vec{\boldsymbol{u}}_1$ is the root hash and $\vec{\boldsymbol{u}}_{2^{a-1}}, \ldots, \vec{\boldsymbol{u}}_{2^a-1}$ are the leaves. The binary preimages $\vec{\boldsymbol{s}}_i$ are the expansions of the two children $\vec{\boldsymbol{u}}_{2i}$ and $\vec{\boldsymbol{u}}_{2i+1}$; that is, $\vec{\boldsymbol{s}}_i = \vec{\boldsymbol{s}}_{i,l} \parallel \vec{\boldsymbol{s}}_{i,r}$ and $\vec{\boldsymbol{u}}_{2i} = \boldsymbol{G}\vec{\boldsymbol{s}}_{i,l}$, $\vec{\boldsymbol{u}}_{2i+1} = \boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}$. Here $\boldsymbol{G}$ is the power-of-two gadget matrix $\boldsymbol{G} = \boldsymbol{I} \otimes (1, 2, \ldots, 2^{\lceil \log q \rceil})$, i.e. the identity matrix tensored with the two-power vector.

Now, in the protocol the prover sends an amortized masked opening of all the preimages,

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \sum_{i=1}^{2^a-1} x_i\vec{\boldsymbol{s}}_i.$$

The main idea is that all the terms $x_i\vec{\boldsymbol{s}}_i$ for $i > 1$ can be absorbed into the masking vector so that we have $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}}_0 + x_1\vec{\boldsymbol{s}}_1$. This is just a masked opening of $\vec{\boldsymbol{s}}_1$ and the verifier checks that

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}}_0 + x_1\vec{\boldsymbol{u}}_1$$

using the vector $\vec{\boldsymbol{w}}_0 = \boldsymbol{A}\vec{\boldsymbol{y}}_0 = \boldsymbol{A}\vec{\boldsymbol{y}} + \sum_{i=2}^{2^a-1} s_i\vec{\boldsymbol{u}}_i$ that he has received from the prover before the challenge x_1. Next, from this opening proof we can extract $\vec{\boldsymbol{s}}_1$. Moreover the prover also proves the linear relation

$$\vec{\boldsymbol{w}}_0 = \vec{\boldsymbol{w}}_1 + x_2\boldsymbol{G}\vec{\boldsymbol{s}}_{1,l} + x_3\boldsymbol{G}\vec{\boldsymbol{s}}_{1,r}$$

for a vector $\vec{\boldsymbol{w}}_1 = \boldsymbol{A}\vec{\boldsymbol{y}}_1 = \boldsymbol{A}\vec{\boldsymbol{y}} + \sum_{i=4}^{2^a-1} x_i\vec{\boldsymbol{u}}_i$ that he has sent before the challenges x_2 and x_3. So, this implies

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}}_1 + x_1\vec{\boldsymbol{u}}_1 + x_2\vec{\boldsymbol{u}}_2 + x_3\vec{\boldsymbol{u}}_3.$$

In other words the extracted $\vec{s}_1$ defines the hashes in the first level of the tree and there is a proof for the verification equation of an amortized opening proof for this level. So we can continue recursively and extract level by level from the prover until we have an opening for the full tree. Our protocol can be seen as a sequence of exact amortized binary opening proofs, one for each level for the tree, that use the linear proof technique to prove the verification equation for the proof for the next level. The proof also shows that all the preimages $\vec{s}_i$ are binary as this is needed for the approach to work, as explained.

We use our Merkle tree opening protocol that can also prove linear and product relations on the preimages of the leaves to prove instances of Rank-1 Constraint Satisfaction (R1CS) [BCG+13] which is an NP-complete problem (see Section 6). Recall that in the (simplified) R1CS setting, the prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that it knows a vector $\vec{s} \in \mathbb{Z}_q^k$ such that

$$(A\vec{s}) \circ (B\vec{s}) = C\vec{s} \tag{2}$$

where $A, B, C \in \mathbb{Z}_q^{k \times k}$ are public matrices and $\circ$ denotes the component-wise product. The usual way to prove such a relation is to first commit to $\vec{s}$ as well as to the vectors

$$\vec{a} = A\vec{s}, \vec{b} = B\vec{s}, \vec{c} = C\vec{s}. \tag{3}$$

Then, $\mathcal{P}$ only needs to prove the linear relations described in (3) and the multiplicative relation $\vec{a} \circ \vec{b} = \vec{c}$. This method requires us to commit to three additional vectors over $\mathbb{Z}_q$ of length k. In Section 6.1 we show a potential optimization to prove (2) by committing to only one vector in $\mathbb{Z}_q^k$.

Table 1 contains a comparison of our proof system for R1CS to Ligero. We chose a range of constraints above 2^{20} as our proof system is most effective for such large numbers of constraints. In particular, we observe that for large instances, e.g. $k \geq 2^{24}$, our system achieves 2-3 times smaller proof sizes than Ligero. The proof sizes for Ligero were directly measured by running the implementation from `https://github.com/scipr-lab/libiop`. For both proof systems we used a field size of about 128 bits and comparable soundness errors.

2 Preliminaries

2.1 Notation

Let q be an odd prime, and $\mathbb{Z}_q$ denote the ring of integers modulo q. For $r \in \mathbb{Z}$, we define $r \bmod q$ to be the unique element in the interval $[-\frac{q-1}{2}, \frac{q-1}{2}]$ that is congruent to r modulo q. We write $\vec{v} \in \mathbb{Z}_q^m$ to denote vectors over $\mathbb{Z}_q$ and matrices over $\mathbb{Z}_q$ will be written as regular capital letters M. By default, all vectors are column vectors. We write $\vec{v} \parallel \vec{w}$ for the concatenation of $\vec{v}$ and $\vec{w}$ (which is still a column vector). We write $x \xleftarrow{\$} S$ when $x \in S$ is sampled uniformly at random from the finite set S and similarly $x \xleftarrow{\$} D$ when x is sampled according to the distribution D.

		Proof Size	
Number of constraints	Soundness error	Ligero	**Our System**
2^{19}	2^{-115}	4.58 MB	**4.53 MB**
2^{20}	2^{-114}	8.35 MB	**5.22 MB**
2^{21}	2^{-113}	8.90 MB	**6.08 MB**
2^{22}	2^{-112}	16.23 MB	**7.19 MB**
2^{23}	2^{-111}	17.39 MB	**10.79 MB**
2^{24}	2^{-110}	31.83 MB	**13.21 MB**
2^{25}	2^{-109}	34.15 MB	**16.59 MB**
2^{26}	2^{-108}	62.14 MB	**21.68 MB**
2^{27}	2^{-107}	66.03 MB	**29.04 MB**
2^{28}	2^{-106}	121.90 MB	**42.42 MB**

Table 1. Comparison of proof sizes between our proof system for R1CS over $\mathbb{Z}_q$ with $q \approx 2^{128}$, and Ligero.

Let d be a power of two and denote $\mathcal{R}$ and $\mathcal{R}_q$ to be the rings $\mathbb{Z}[X]/(X^d+1)$ and $\mathbb{Z}_q[X]/(X^d+1)$, respectively. Bold lower-case letters $\boldsymbol{p}$ denote elements in $\mathcal{R}$ or $\mathcal{R}_q$ and bold lower-case letters with arrows $\vec{\boldsymbol{b}}$ represent column vectors with components in $\mathcal{R}$ or $\mathcal{R}_q$. We also use bold upper-case letters for matrices $\boldsymbol{B}$ over $\mathcal{R}$ or $\mathcal{R}_q$. The ring $\mathcal{R}_q$ is a $\mathbb{Z}_q$-module spanned by the power basis $\{1, X, \ldots, X^{d-1}\}$. The multiplication homomorphism $\boldsymbol{x} \mapsto \boldsymbol{a}\boldsymbol{x}$ for an $\boldsymbol{a} = a_0 + \cdots + a_{d-1}X^{d-1} \in \mathcal{R}_q$ is represented by the negacyclic rotation matrix

$$\mathsf{Rot}(\boldsymbol{a}) = \begin{pmatrix} a_0 & -a_{d-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{d-1} & a_{d-2} & \ldots & a_0 \end{pmatrix} \in \mathbb{Z}_q^{d \times d}.$$

This extends to $\mathcal{R}_q$-module homomorphisms given by $\boldsymbol{A} \in \mathcal{R}^{m \times n}$ in a block-wise fashion. They are represented by $\mathsf{Rot}(\boldsymbol{A}) \in \mathbb{Z}_q^{md \times nd}$.

In this paper we choose prime q such that $\mathbb{Z}_q$ contains a primitive $2d$-th root of unity $\zeta \in \mathbb{Z}_q$ but no elements whose order is a higher power of two, i.e. $q - 1 \equiv 2d \pmod{4d}$. Therefore, we have

$$X^d + 1 \equiv \prod_{j=0}^{d-1} \left(X - \zeta^{2j+1}\right) \pmod{q} \tag{4}$$

where ζ^{2j+1} ($j \in \mathbb{Z}_d$) ranges over all the d primitive $2d$-th roots of unity. We define the Number Theoretic Transform (NTT) of a polynomial $\boldsymbol{p} \in \mathcal{R}_q$ as follows:

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

We will use the property that for any polynomials $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{R}_q$, we have $\mathsf{NTT}(\boldsymbol{f}) \circ \mathsf{NTT}(\boldsymbol{g}) = \mathsf{NTT}(\boldsymbol{fg})$ where $\circ$ is the component-wise vector multiplication.

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

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

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

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

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

2.2 Module-SIS and Module-LWE Problems

We employ the computationally binding and computationally hiding commitment scheme from [BDL+18] in our protocols, and rely on the well-known Module-LWE (MLWE) and Module-SIS (MSIS) problems [LPR10, Din12, LS15, Mic02, LM06, PR06] problems to prove the security of our constructions. Both problems are defined over a ring $\mathcal{R}_q$ for a positive modulus $q \in \mathbb{Z}^+$.

Definition 1 ($\mathsf{MSIS}_{\kappa,\beta}$)**.** *In the* Module-SIS *problem with parameters $\kappa, \lambda > 0$ and $\beta < q$ a uniformly random matrix $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa \times (\kappa+\lambda)}$ is given. Then the goal is to find a vector $\vec{\boldsymbol{s}} \in \mathcal{R}_q^{\kappa+\lambda}$ such that $\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{0}}$ and $0 < \|\vec{\boldsymbol{s}}\|_2 \leq \beta$. We say that an adversary $\mathcal{A}$ has advantage ϵ in solving $\mathsf{MSIS}_{\kappa,\beta}$ if*

$$\Pr\left[\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{0}} \text{ and } 0 < \|\vec{\boldsymbol{s}}\|_2 \leq \beta \;\middle|\; \boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa \times (\kappa+\lambda)};\ \vec{\boldsymbol{s}} \leftarrow \mathcal{A}(\boldsymbol{A})\right] \geq \epsilon.$$

Definition 2 ($\mathsf{MLWE}_{\lambda,\chi}$)**.** *In the* Module-LWE *problem with parameters $\kappa, \lambda > 0$ and χ an "error" distribution over $\mathbb{Z}_q$, a pair $(\boldsymbol{A}, \vec{\boldsymbol{t}}) \in \mathcal{R}_q^{\kappa \times (\kappa+\lambda)} \times \mathcal{R}_q^{\kappa}$ is given where $\boldsymbol{A}$ is uniformly random. Then the goal is to distinguish between the two cases where either $\vec{\boldsymbol{t}}$ is given by $\vec{\boldsymbol{t}} = \boldsymbol{A}\vec{\boldsymbol{s}}$ for a secret vector $\vec{\boldsymbol{s}} \xleftarrow{\$} \chi^{(\kappa+\lambda)d}$ sampled from the error distribution, or $\vec{\boldsymbol{t}}$ is independently uniformly random. We say that an adversary $\mathcal{A}$ has advantage ϵ in distinguishing $\mathsf{MLWE}_{\lambda,\chi}$ if*

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

For our practical security estimations of these two problems against known attacks, the parameter κ in the Module-LWE problem and the parameter λ in the Module-SIS problem do not play a crucial role. Therefore, we omit then in the notations $\mathsf{MSIS}_{\kappa,\beta}$ and $\mathsf{MLWE}_{\lambda,\chi}$.

2.3 Challenge Space

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

Consider the coefficients of the polynomial $\boldsymbol{c} \bmod (X - \zeta^{2j+1})$ for $\boldsymbol{c} \leftarrow C$. Then, all coefficients follow the same distribution over $\mathbb{Z}_q$. Let us write Y for the random variable over $\mathbb{Z}_q$ that follows this distribution. Attema et al. [ALS20] give an upper bound on the maximum probability of Y.

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

$$\Pr[Y = x] \leq \frac{1}{q} + \frac{2d}{q} \sum_{j \in \mathbb{Z}_q^\times / \langle \zeta \rangle} \prod_{i=0}^{d-1} \left| \frac{1}{2} + \frac{1}{2} \cos(2\pi j y \zeta^i / q) \right|. \tag{5}$$

One observes that computing the sum on the right-hand side would take essentially $O(q)$ time. Hence, computing the upper-bound for $\Pr[Y = x]$ is infeasible for large primes q. However, based on experiments for smaller primes [3], we assume that the probability is very close to $1/q$. In fact, this process exhibits a phase-shift behaviour, where the probability very rapidly drops to values close to $1/q$ as soon as the entropy of $\boldsymbol{c}$ is slightly higher than $\log q$.

2.4 BDLOP Commitment Scheme

We use a variant of the commitment scheme from [BDL+18], which allows to commit to a vector of polynomials in $\mathcal{R}_q$ [4]. Suppose that we want to commit to $\vec{\boldsymbol{m}} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\mu)^T \in \mathcal{R}_q^\mu$. Then, in the commitment parameter generation, a uniformly random matrix $\boldsymbol{B}_0 \xleftarrow{\$} \mathcal{R}_q^{\kappa \times (\kappa+\lambda+\mu)}$ and vectors $\vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_\mu \xleftarrow{\$} \mathcal{R}_q^{\kappa+\lambda+\mu}$ are generated and output as public parameters. In practice they never have to be stored because they can be expanded from a short seed. One may choose to generate $\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_\mu$ in a more structured way as in [BDL+18] since it saves some computation.

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

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

[3] In particular, [ALS20, ENS20] computed that for $q \approx 2^{32}$, the maximum probability for each coefficient of $c \bmod X^4 - \zeta^{8j+4}$ is around $2^{-31.4}$.

[4] We provide more background on commitment schemes in Appendix A.

where $\vec{\boldsymbol{t}}_0$ forms the binding part and each $\boldsymbol{t}_i$ encodes a message polynomial $\boldsymbol{m}_i$. The commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \parallel \boldsymbol{t}_1 \parallel \cdots \parallel \boldsymbol{t}_\mu$ is computationally hiding under the $\mathsf{MLWE}_{\lambda,\chi}$ assumption and computationally binding under the $\mathsf{MSIS}_{\kappa,\beta}$ assumption for some $q > \beta > 2\sqrt{(\kappa+\lambda+\mu)d}$; see [BDL+18].

Moreover, the scheme is not only binding for the opening $(\vec{\boldsymbol{m}}, \vec{\boldsymbol{r}})$ known by the prover, but also binding with respect to a relaxed opening $(\vec{\boldsymbol{m}}^*, \bar{\boldsymbol{c}}, \vec{\boldsymbol{r}}^*)$. The relaxed opening also includes a short invertible polynomial $\bar{\boldsymbol{c}}$ and the randomness vector $\vec{\boldsymbol{r}}^*$ is longer than $\vec{\boldsymbol{r}}$. Attema et al. [ALS20] further reduce the requirements of an opening and define the notion of a weak opening (see Appendix A.4).

3 Interactive Schwartz-Zippel

The Schwartz-Zippel Lemma [Sch80, Zip79] (first proven by Ore [Ore22]) is an important tool in the construction of many zero-knowledge proof systems. It says that for a non-zero polynomial $f \in \mathbb{Z}_q[X_1, \ldots, X_n]$ of total degree d, the probability that $f(x_1, \ldots, x_n) = 0$ for independently uniformly random $x_1, \ldots, x_n \in \mathbb{Z}_q$ is at most d/q. Note that the probability does not depend on the number n of variables. This is used in zero-knowledge proof systems by committing to the coefficients $c_{\boldsymbol{\alpha}}$ of f, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ is a multi-index, and then proving

$$f(x_1, \ldots, x_n) = \sum_{|\boldsymbol{\alpha}| \leq d} c_{\boldsymbol{\alpha}} x_1^{\alpha_1} \ldots x_n^{\alpha_n}$$

for uniformly random challenges $x_1, \ldots, x_n \in \mathbb{Z}_q$ from the verifier. Then, if the coefficient commitments where made before the challenges x_i were known by the prover, it is clear that the coefficients must be independent from the x_i. So, this implies that $f = \sum_{|\boldsymbol{\alpha}| \leq d} c_{\boldsymbol{\alpha}} X_1^{\alpha_1} \ldots X_n^{\alpha_n}$ with soundness error d/q. Now, one is usually only interested in a few of the coefficients $c_{\boldsymbol{\alpha}}$, typically the n coefficients of the pure highest-degree terms divisible by X_i^d for some i. The rest are called garbage coefficients. But since the total number of coefficients, and hence commitments, is equal to $\binom{n+d}{d}$, this gets impractical already for small n and therefore the multivariate case with $n > 1$ is not often used in practical zero-knowledge proof systems.

In this section we develop a new proof technique that only needs a number of garbage commitments that is linear in n while having a modest cost of a linear loss in soundness. First, we decompose the polynomial f such that

$$f(X_1, \ldots, X_n) = f_0 + f_1(X_1) + \cdots + f_n(X_1, \ldots, X_n), \tag{6}$$

where $f_0 \in \mathbb{Z}_q$ is the constant coefficient of f and $f_i \in \mathbb{Z}_q[X_1, \ldots, X_i]$, $i \geq 1$, consist of the monomials $c_{\boldsymbol{\alpha}} X_1^{\alpha_1} \ldots X_n^{\alpha_n}$ of f with $\alpha_i \geq 1$ and $\alpha_{i+1} = \cdots = \alpha_n = 0$, i.e. the monomials that are divisible by X_i but not by any X_j for $j > i$. Next, note that every polynomial f_i can be viewed as a univariate polynomial in X_i over the ring $\mathbb{Z}_q[X_1, \ldots, X_{i-1}]$, divisible by X_i. More precisely, $f_i = f_{i,1} X_i + \cdots + f_{i,d-1} X_i^{d-1} + l_i X_i^d$ where $f_{i,j} \in \mathbb{Z}_q[X_1, \ldots, X_{i-1}]$ and $l_i \in \mathbb{Z}_q$ since f is of total degree d. Now, we are only really interested in the coefficients l_i, and it turns

out there is no need to prove that the other coefficients are actually polynomials in $X_1, \ldots, X_{i-1}$ of degree at most $d-1$. Indeed, we have the following lemma.

Lemma 2. *Let $f\colon \mathbb{Z}_q^n \to \mathbb{Z}_q$ be a function of the form*

$$f(x_1, \ldots, x_n) = f_0 + f_1(x_1) + f_2^{(x_1)}(x_2) + \cdots + f_n^{(x_1,\ldots,x_{n-1})}(x_n),$$

where $f_0 \in \mathbb{Z}_q$, $f_1 \in \mathbb{Z}_q[X_1]$, and, for $i \geq 2$, $f_i \in (\mathbb{Z}_q[X_i])^{\mathbb{Z}_q^{i-1}}$, i.e. f_i is a function from $\mathbb{Z}_q^{i-1}$ to $\mathbb{Z}_q[X_i]$, given by $(x_1, \ldots, x_{i-1}) \mapsto f_i^{(x_1,\ldots,x_{i-1})}$. Suppose that $f_i^{(x_1,\ldots,x_{i-1})}$ is divisible by X_i (i.e. has zero constant coefficient) and of degree at most d for all $(x_1, \ldots, x_{i-1}) \in \mathbb{Z}_q^{i-1}$, $i \geq 1$. Moreover, suppose that there exists a $j \geq 1$ such that $f_j^{(x_1,\ldots,x_{j-1})} \neq 0$ for all $(x_1, \ldots, x_{j-1}) \in \mathbb{Z}_q^{j-1}$. Then, for uniformly random $(x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, the probability that $f(x_1, \ldots, x_n) = 0$ is at most $(n+1-j)d/q$. That is,

$$\Pr\left[f(x_1, \ldots, x_n) = 0\right] \leq \frac{(n+1-j)d}{q}.$$

Proof. We write $f_{\leq i}$ for the partial function

$$f_{\leq i}(x_1, \ldots, x_i) = f_0 + f_1(x_1) + f_2^{(x_1)}(x_2) + \cdots + f_i^{(x_1,\ldots,x_{i-1})}(x_i)$$

that only includes the functions up to f_i. In particular, $f_{\leq n} = f$. Then we find

$$\begin{aligned}
&\Pr\left[f(x_1, \ldots, x_n) = 0\right] \\
&= \Pr\left[f_{\leq n-1}(x_1, \ldots, x_{n-1}) = 0\right] \\
&\quad \cdot \Pr\left[f(x_1, \ldots, x_n) = 0 \,|\, f_{\leq n-1}(x_1, \ldots, x_{n-1}) = 0\right] \\
&\quad + \Pr\left[f_{\leq n-1}(x_1, \ldots, x_{n-1}) \neq 0\right] \\
&\qquad \cdot \Pr\left[f(x_1, \ldots, x_n) = 0 \,|\, f_{\leq n-1}(x_1, \ldots, x_{n-1}) \neq 0\right] \\
&\leq \Pr\left[f_{\leq n-1}(x_1, \ldots, x_{n-1}) = 0\right] \\
&\quad + \Pr\left[f(x_1, \ldots, x_n) = 0 \,|\, f_{\leq n-1}(x_1, \ldots, x_{n-1}) \neq 0\right] \\
&\leq \Pr\left[f_{\leq n-2}(x_1, \ldots, x_{n-2}) = 0\right] \\
&\quad + \Pr\left[f_{\leq n-1}(x_1, \ldots, x_{n-1}) = 0 \,|\, f_{\leq n-2}(x_1, \ldots, x_{n-2}) \neq 0\right] \\
&\quad + \Pr\left[f(x_1, \ldots, x_n) = 0 \,|\, f_{\leq n-1}(x_1, \ldots, x_{n-1}) \neq 0\right] \\
&\leq \ldots \\
&\leq \Pr\left[f_{\leq j}(x_1, \ldots, x_j) = 0\right] \\
&\quad + \Pr\left[f_{\leq j+1}(x_1, \ldots, x_{j+1}) = 0 \,|\, f_{\leq j}(x_1, \ldots, x_j) \neq 0\right] \\
&\quad + \ldots \\
&\quad + \Pr\left[f(x_1, \ldots, x_n) = 0 \,|\, f_{\leq n-1}(x_1, \ldots, x_{n-1}) \neq 0\right].
\end{aligned}$$

Consider the first probability $\Pr\left[f_{\leq j}(x_1, \ldots, x_j) = 0\right]$ after the last inequality. For every choice $(x'_1, \ldots, x'_{j-1}) \in \mathbb{Z}_q^{j-1}$, the function

$$f_{\leq j}(x'_1, \ldots, x'_{j-1}, x_j) = f_{\leq j-1}(x'_1, \ldots, x'_{j-1}) + f_j^{(x'_1,\ldots,x'_{j-1})}(x_j)$$

is a fixed univariate polynomial in x_j of degree at most d and the random variable x_j is independent from it. Moreover, we know from the assumption in the lemma that the polynomial is non-zero since f_j is non-zero and divisible by x_j; that is, f_j is never constant. Therefore,

$$\begin{aligned}
&\Pr\left[f_{\leq j}(x_1,\ldots,x_j)=0\right]\\
&=\sum_{x'_1,\ldots,x'_{j-1}\in\mathbb{Z}_q}\Pr\left[x_1=x'_1\wedge\cdots\wedge x_{j-1}=x'_{j-1}\right]\Pr\left[f_{\leq j}(x'_1,\ldots,x'_{j-1},x_j)=0\right]\\
&\leq\sum_{x'_1,\ldots,x'_{j-1}\in\mathbb{Z}_q}\left(\frac{1}{q}\right)^{j-1}\frac{d}{q}=\frac{d}{q}.
\end{aligned}$$

Similarly, for the other probabilities $\Pr\left[f_{\leq i}(x_1,\ldots,x_i)=0\,|\,f_{\leq i-1}(x_1,\ldots,x_{i-1})\neq 0\right]$ we interpret $f_{\leq i}(x_1,\ldots,x_i)$ as the evaluation of a polynomial of degree at most d at the independently uniformly random point x_i. This time we condition on the event that the constant coefficient of the polynomial, which is given by $f_{\leq i-1}(x_1,\ldots,x_{i-1})$, is non-zero. Hence,

$$\Pr\left[f_{\leq i}(x_1,\ldots,x_i)=0\,|\,f_{\leq i-1}(x_1,\ldots,x_{i-1})\neq 0\right]\leq d/q$$

for all $i=j+1,\ldots,n$. □

3.1 Making Use of Lemma 2 in Zero-Knowledge Protocols

Suppose we want to prove that the polynomial $f\in\mathbb{Z}_q[X_1,\ldots,X_n]$ of total degree d does not contain any terms divisible by X_i^d for any i; that is, f is of degree at most $d-1$ in each X_i. Then decompose f as in Equation (6), and define the functions $\mathbb{Z}_q^{i-1}\to\mathbb{Z}_q[X_i]$, $(x_1,\ldots,x_{i-1})\mapsto f_i^{(x_1,\ldots,x_{i-1})}(X_i)=f_i(x_1,\ldots,x_{i-1},X_i)$, that forget the polynomial structure of f_i in the variables $X_1,\ldots,X_i$. Now, in a multi-round protocol where the uniformly random challenges x_i are spread-out over $2n$ rounds we can commit to the $d-1$ coefficients $\gamma_{i,k}$ of $f_i^{(x_1,\ldots,x_{i-1})}(X_i)=\gamma_{i,1}X_i+\cdots+\gamma_{i,d-1}X_i^{d-1}$ immediately after seeing $x_1,\ldots,x_{i-1}$ but before knowing $x_i,\ldots,x_n$. Then we show

$$f(x_1,\ldots,x_n)-\left(\gamma_0+\sum_{k=1}^{d-1}\gamma_{1,k}x_1^k+\cdots+\sum_{k=1}^{d-1}\gamma_{n,k}x_n^k\right)=0.$$

Here we assume that we know how to prove that some element of $\mathbb{Z}_q$ is the evaluation $f(x_1,\ldots,x_n)$ of the fixed polynomial f of degree d. The fact that the commitments to the coefficients $\gamma_{i,k}$ were produced before $x_i,\ldots,x_n$ were known shows that they can only be functions of $x_1,\ldots,x_{i-1}$. So, we have effectively proven

$$g_0+g_1(x_1)+g_2^{(x_1)}(x_2)+\cdots+g_n^{(x_1,\ldots,x_{n-1})}(x_n)=0,$$

for uniformly random $(x_1,\ldots,x_n)\in\mathbb{Z}_q^n$ and functions g_i as in Lemma 2 that fulfill the requirements that they have zero constant coefficient and are of degree

at most d. Furthermore, for each $i \in \{1, \ldots, n\}$ and all $(x'_1, \ldots, x'_{i-1}) \in \mathbb{Z}_q^{i-1}$, the coefficient for X_i^d of $g_i^{(x'_1,\ldots,x'_{i-1})}$ is given by the corresponding coefficient in f. It follows that we proven f to be of degree $d-1$ in all X_i with soundness error nd/q. Note that we only needed $n(d-1)+1$ garbage commitments.

As an example, in our lattice-based protocols we let the prover ultimately send amortized masked openings $\vec{z}(x_1, \ldots, x_n) = \vec{y} + x_1\vec{s}_1 + \cdots + x_n\vec{s}_n$ of secret vectors $\vec{s}_i \in \mathbb{Z}_q^m$ with challenges $x_i \in \mathbb{Z}_q$, and we want to be able to prove that all secret vectors are binary. So, using another uniformly random challenge vector $\vec{\varphi} \in \mathbb{Z}_q^m$, we want to show that the quadratic ($d = 2$) polynomial

$$f(x_1, \ldots, x_n) = \langle \vec{z} \circ ((x_1 + \cdots + x_n)\vec{1} - \vec{z}), \vec{\varphi} \rangle \tag{7}$$

does not contain terms of the form x_i^2. Here each of the polynomials $f_i^{(x_1,\ldots,x_{i-1})}$ involves only one garbage coefficient and is of the form $f_i^{(x_1,\ldots,x_{i-1})}(X_i) = \gamma_i X_i$. So we end up only needing $n+1$ garbage commitments to the coefficients γ_i. The protocol proceeds as follows. The prover receives the challenge vector $\vec{\varphi}$ and commits to the first garbage coefficient $\gamma_0 = -\langle \vec{y} \circ \vec{y}, \vec{\varphi} \rangle$. Then, over the course of the next $2n$ rounds, the protocol alternates between the prover committing to the next garbage coefficient

$$\gamma_i = \left\langle \vec{y} \circ (1 - 2\vec{s}_i) + \sum_{j=1}^{i-1} x_j(\vec{s}_j \circ (\vec{1} - \vec{s}_i) + \vec{s}_i \circ (\vec{1} - \vec{s}_j)), \vec{\varphi} \right\rangle,$$

and the verifier sending the next challenge x_i, for $i = 1, \ldots, n$. Afterwards, the protocol is finished by proving the linear relation (in the garbage coefficients)

$$\langle \vec{z} \circ ((x_1 + \cdots + x_n)\vec{1} - \vec{z}), \vec{\varphi} \rangle - (\gamma_0 + \gamma_1 x_1 + \cdots + \gamma_n x_n) = 0. \tag{8}$$

In the PCP literature, when proving such pointwise multiplicative relations on many vectors, a different technique is used to keep the number of garbage coefficients linear in the number of vectors. Namely, instead of multivariate masked openings of degree one, univariate openings of degree n are used where the different vectors are separated as the basis coefficients with respect to a basis given by Lagrange interpolation polynomials. See [GGPR13] for details. This technique does not seem to be compatible with our lattice-based setting. Concretely, we will later need to conclude from SIS hardness that the prover is bound to the vectors in the masked opening and our approach for achieving this requires multivariate openings.

Moreover, the so-called sum check protocols for multivariate polynomials from [LFKN92, Sha92] have similarities with our protocol. These protocols also have n rounds and in each round the polynomial is reduced to a univariate polynomial.

We don't consider it a problem that our protocol has many rounds. We don't view the number of rounds to be of practical importance that needs to be optimized. The interactive variants of our protocols only serve as a convenient intermediate representation that is easy to reason about. But in practice only

the non-interactive variants will ever be used and there the number of rounds only has an indirect effect on for example the prover and verifier runtime and the soundness error but no independent relevance. If the protocol can achieve negligible soundness error and still has acceptable runtimes and proof sizes, then the number of rounds doesn't matter.

4 Exact Amortized Binary Opening Proof

The aim of this section is to present a protocol for proving knowledge of (exactly) binary preimages $\vec{\boldsymbol{s}}_i \in \{0,1\}^m \subset \mathcal{R}_q^{m/d}$ to n collision-resistant hashes $\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$. Our starting point is the approximate amortized proof that goes back to [BBC+18]. There the prover samples a short masking vector $\vec{\boldsymbol{y}}$ and commits to it by sending $\vec{\boldsymbol{w}} = \boldsymbol{A}\vec{\boldsymbol{y}}$. The verifier then sends n short challenge polynomials $\boldsymbol{c}_1, \dots, \boldsymbol{c}_n$ and the prover replies by sending the amortized masked opening $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}_1\vec{\boldsymbol{s}}_1 + \cdots + \boldsymbol{c}_n\vec{\boldsymbol{s}}_n$. The verifier accepts if $\vec{\boldsymbol{z}}$ is short and a preimage of $\vec{\boldsymbol{w}} + \boldsymbol{c}_1\vec{\boldsymbol{u}}_1 + \cdots + \boldsymbol{c}_n\vec{\boldsymbol{u}}_n$. This protocol is sound, because, for every $i = 1, \dots, n$, the prover must be able to answer two challenge tuples successfully that differ only in the one challenge $\boldsymbol{c}_i$. Then the difference of the two corresponding masked openings yields the approximate solution $\boldsymbol{A}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}') = (\boldsymbol{c}_i - \boldsymbol{c}_i')\vec{\boldsymbol{u}}_i$.

Next, we want to get rid of the perturbation factors $\bar{\boldsymbol{c}}_i = \boldsymbol{c}_i - \boldsymbol{c}_i'$. In general and for efficient parameters they are not invertible so we can not simply divide through, but it is possible to use the strategy from [ALS20] where one pieces together many extractions from potentially several parallel repetitions of the protocol in order to get so-called weak openings $\vec{\boldsymbol{s}}_i^*$ such that $\boldsymbol{A}\vec{\boldsymbol{s}}_i^* = \vec{\boldsymbol{u}}_i$ (c.f. [ALS20, Definition 4.2]). The weak openings are not necessarily short but the prover is still bound to them; see [ALS20, Lemma 4.3].

Now, to extend the proof and show that the $\vec{\boldsymbol{s}}_i^*$ are in fact binary, the amortized masked opening $\vec{\boldsymbol{z}}$ from above with polynomial challenges is not of much help. The problem is that the polynomial product effectively intermingles all the secret coefficients and then it seems inefficient to prove all the quadratic relations about individual coefficients that we need for proving that each and every coefficient is binary. Therefore, our protocol has a second stage with integer challenges $x_i \in \mathbb{Z}_q$ and masked opening

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + x_1\vec{\boldsymbol{s}}_1 + \cdots + x_n\vec{\boldsymbol{s}}_n.$$

To get as much soundness as possible, and at the same time not increase q more than necessary, we want the challenges x_i to be uniformly random modulo q. But since we are relying on MSIS hardness we can not send $\vec{\boldsymbol{z}}$ directly. Instead, we compose it from l short $\vec{\boldsymbol{z}}_j$ with short integer challenges $x_{i,j} \in \mathbb{Z}$, $j = 0, \dots, l-1$. More precisely, we set $\delta = \lceil q^{1/l} \rceil$, and $x_i \bmod q = x_{i,0} + \cdots + x_{i,l-1}\delta^{l-1}$ (non-negative standard representative), where $0 \le x_{i,j} < \delta$. Then, the prover sends the polynomial vectors

$$\vec{\boldsymbol{z}}_j = \vec{\boldsymbol{y}}_j + x_{1,j}\vec{\boldsymbol{s}}_1 + \cdots + x_{n,j}\vec{\boldsymbol{s}}_n.$$

In principle the second stage with integer challenges $x_{i,j}$ alone would allow to extract the weak openings $\vec{s}_i^*$, but we still include the first stage with polynomial challenges as it turns out that the final norm bound for which we need Module-SIS to be hard depends on the norm of the product of two challenges. Hence, when one of the challenges can be a shorter polynomial challenge, this results in a smaller Module-SIS norm bound and ultimately smaller proof sizes.

Next, for the actual binary proof we work with the composed $\vec{z} = \vec{z}_0 + \cdots + \vec{z}_{l-1}\delta^{l-1}$. We forget the polynomial structure and let $\vec{z} = \vec{y} + x_1\vec{s}_1 + \cdots + x_n\vec{s}_n \in \mathbb{Z}_q^m$ be given by the coefficient vectors that correspond to the polynomial vectors. This allows for the approach from Section 3.1 for proving that all secret coefficients are binary. Let $\vec{\varphi} \in \mathbb{Z}_q^m$ be a uniformly random challenge vector from the verifier. Eventually we need to prove Equation (8) with garbage coefficients γ_i that are from commitments produced interactively with increasing dependence on the challenges x_i as explained. We use the BDLOP commitment scheme and apply the linear proof from [ENS20], which we call the *auxiliary proof* in this protocol. Since our binary proof has a soundness error bigger than $1/q$, there is no need to apply the soundness boosting techniques for the linear proof. That is, we use the simpler proof without automorphisms. So, after the initial approximate proof, at the beginning of the second stage, the prover initializes the BDLOP commitment scheme. He samples a randomness vector $\vec{\boldsymbol{r}}^{(t)} \in \mathcal{R}_q^{\kappa_2+\lambda+\mu}$ and commits to it in the top part $\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}} \in \mathcal{R}_q^{\kappa_2}$. Here κ_2, λ, and $\mu = \lceil (n+1)/d \rceil + 1$ are the BDLOP MSIS rank, MLWE rank, and message rank, respectively. Since the prover needs to commit to only one $\mathbb{Z}_q$-element at a time and not a full $\mathcal{R}_q$-polynomial, he is going to send individual NTT coefficients of the low part of the BDLOP commitment scheme. More precisely, the prover precomputes the NTT vector $\vec{e} = \mathsf{NTT}(\boldsymbol{B}_1\vec{\boldsymbol{r}}^{(t)}) \in \mathbb{Z}_q^{\lceil (n+1)/d \rceil d}$. Then, when he wants to commit to $\gamma_i \in \mathbb{Z}_q$, he sends $\tau_i = e_i + \gamma_i$, $i = 0, \ldots, n$. In the end the verifier has the full commitment polynomial vector $\vec{\boldsymbol{t}}_2 = \mathsf{NTT}^{-1}(\vec{\tau}) = \boldsymbol{B}_2\vec{\boldsymbol{r}} + \mathsf{NTT}^{-1}(\vec{\gamma})$.

After the initialization of BDLOP, the prover samples l masking vectors $\vec{\boldsymbol{y}}_j$ for the short shares $\vec{\boldsymbol{z}}_j$ of $\vec{\boldsymbol{z}}$ and sends the commitments $\vec{\boldsymbol{w}}_j = \boldsymbol{A}\vec{\boldsymbol{y}}_j$, together with $\vec{\boldsymbol{t}}_0$. The verifier follows by sending the challenge vector $\vec{\varphi}$ for the binary proof. Next, the core subprotocol with $2n + 2$ rounds starts. Here the prover and verifier alternate between garbage commitments to the parts $f_i = \gamma_i x_i$ of the polynomial $f(x_1, \ldots, x_n)$ in Equation (7), and the challenges x_i. Finally, the prover computes the shares $\vec{\boldsymbol{z}}_j$, performs rejection sampling on them, and sends them if there was no rejection. This concludes the second stage and main part of the protocol. Finally, the protocol is finished with the auxiliary proof for Equation (8), exactly as in [ENS20].

Before we spell-out the protocol in detail in Figure 1 and then analyze its security, we mention a technical problem that we have to overcome in the security proof of the protocol. When we sketched the binary proof in Section 3.1, we assumed that $\vec{z}$ is the evaluation of a fixed polynomial in the challenges $x_1, \ldots, x_n$. In other words for the extraction this means that we must be sure that

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}}^* + x_1\vec{\boldsymbol{s}}_1^* + \cdots + x_n\vec{\boldsymbol{s}}_n^*$$

in (almost all) accepting transcripts with always the same weak openings $\vec{\boldsymbol{y}}^*$ and $\vec{\boldsymbol{s}}_i^*$. The problem is that this is harder to prove in our amortized setting. Let us recall the argument for the single-secret case with $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}}^* + x\vec{\boldsymbol{s}}^*$, which was presented in [ALS20]. If we find some accepting transcript where the masked opening $\vec{\boldsymbol{z}}'$ is given by $\vec{\boldsymbol{z}}' = \vec{\boldsymbol{y}}^{**} + x'\vec{\boldsymbol{s}}^*$ with a different $\vec{\boldsymbol{y}}^{**} \neq \vec{\boldsymbol{y}}^*$, then we know a challenge difference $\bar{x}$ such that $\bar{x}\vec{\boldsymbol{s}}^*$ is short and $\bar{x}(\vec{\boldsymbol{z}} - \vec{\boldsymbol{z}}') - (x - x')\bar{x}\vec{\boldsymbol{s}}^* = \bar{x}(\vec{\boldsymbol{y}}^* - \vec{\boldsymbol{y}}^{**}) \neq 0$ is a Module-SIS solution. This argument can not be extended to the amortized setting since we would need to multiply by many different $\bar{x}_i$ and not find a sufficiently short Module-SIS solution. But it turns out we can turn the whole argument around and proceed via the contraposition. Concretely, if one of the weak openings $\vec{\boldsymbol{s}}_i^*$ is not binary, then we must be able to find accepting transcripts with different $\vec{\boldsymbol{y}}^{**}$ that results in a SIS solution. See the proof of Theorem 1 for the details.

Theorem 1. *The protocol in Figure 1 is correct, computationally honest verifier zero-knowledge under the Module-LWE assumption and computationally knowledge-sound under the Module-SIS assumption. More precisely, let p be the maximum probability of $\boldsymbol{c} \bmod X - \zeta$ as in Lemma 1. Let ω be a bound on the ℓ_1-norm of the $\boldsymbol{c}_i$.*

Then, for correctness, unless the honest prover $\mathcal{P}$ aborts due to rejection sampling, it convinces the honest verifier $\mathcal{V}$ with overwhelming probability.

For zero-knowledge, there exists an efficient simulator $\mathcal{S}$, that, without access to the secret $\vec{\boldsymbol{s}}_i$, outputs a simulation of a non-aborting transcript of the protocol between $\mathcal{P}$ and $\mathcal{V}$ for every statement $\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$. An algorithm that can distinguish the simulation from the real transcript with advantage ε can distinguish $\mathsf{MLWE}_{\lambda,\chi}$ with advantage $\varepsilon - 2^{100}$ in the same running time.

For knowledge-soundness, there is an extractor $\mathcal{E}$ with the following properties. When given resettable black-box access to a deterministic prover $\mathcal{P}^*$ that convinces $\mathcal{V}$ with probability $\varepsilon > (2n+2)/q + p$, $\mathcal{E}$ either outputs binary preimages $\vec{\boldsymbol{s}}_i^* \in \{0,1\}^m$ for all hashes $\vec{\boldsymbol{u}}_i$, an $\mathsf{MSIS}_{\kappa,B}$ solution for $\boldsymbol{A}$ with $B = 4(\omega\beta_2 + \delta\beta_1 + n\omega\delta\sqrt{m})$, or an $\mathsf{MSIS}_{\kappa_2,8\omega\beta_3}$ solution for $\boldsymbol{B}_0$.

Remark. In the interest of simplicity, we have chosen to present the protocol for binary secret vectors only. It should be clear that the protocol can easily be adapted to prove knowledge of secret preimages that have coefficients from a larger interval, for example ternary coefficients in $\{-1, 0, 1\}$. Then the prover would send two garbage commitments before each challenge x_i.

4.1 Extending the Proof to Linear and Product Relations

In applications of our exact opening proof one usually also wants to prove linear and product relations on the preimage (coefficient) vectors $\vec{s}_i$. We now show that our protocol can easily be extended to include such relations with little additional cost.

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
Inputs:		
$\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa_2 \times (\kappa_2+\lambda+\mu)}$		$\boldsymbol{B}_0, \boldsymbol{B}_1, \vec{\boldsymbol{b}}_2$
$\boldsymbol{B}_1 \in \mathcal{R}_q^{(\mu-1) \times (\kappa_2+\lambda+\mu)}$		$\boldsymbol{A}, \vec{\boldsymbol{u}}_i$
$\vec{\boldsymbol{b}}_2 \in \mathcal{R}_q^{\kappa_2+\lambda+\mu}$		
$\boldsymbol{A} \in \mathcal{R}_q^{\kappa \times m/d}$		
For $i = 1, \ldots, n$:		
$\vec{\boldsymbol{s}}_i \in \{0,1\}^m \subset \mathcal{R}_q^{m/d}$		
$\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$		

The prover and verifier run the approximate amortized opening proof from Figure 2, $\langle \mathcal{P}_{\mathsf{approx}}(\boldsymbol{A}, \vec{\boldsymbol{s}}_i), \mathcal{V}_{\mathsf{approx}}(\boldsymbol{A}, \vec{\boldsymbol{u}}_i) \rangle$. The verifier rejects if $\mathcal{V}_{\mathsf{approx}}$ rejects.

$(\vec{\boldsymbol{r}}^{(t)}, \vec{\boldsymbol{t}}_0, \vec{e}) = \mathsf{AUXINIT}(\boldsymbol{B}_0, \boldsymbol{B}_1)$		
For $j = 0, \ldots, l-1$:		
$\vec{\boldsymbol{y}}_j \xleftarrow{\$} D_{\mathfrak{s}_2}^m$		
$\vec{\boldsymbol{w}}_j = \boldsymbol{A}\vec{\boldsymbol{y}}_j$	$\xrightarrow{\vec{\boldsymbol{t}}_0, \vec{\boldsymbol{w}}_j}$	
	$\xleftarrow{\vec{\varphi}}$	$\vec{\varphi} \xleftarrow{\$} \mathbb{Z}_q^m$
$\vec{y} = \sum_{j=0}^{l-1} \vec{y}_j \delta^j$		
$\gamma_0 = -\langle \vec{y} \circ \vec{y}, \vec{\varphi} \rangle$		
$\tau_0 = e_0 + \gamma_0$	$\xrightarrow{\tau_0}$	

The prover and verifier run the core protocol from Figure 4, $(\tau_1, x_1, \ldots, \tau_n, x_n) = \langle \mathcal{P}_{\mathsf{core}}(\vec{e}, \vec{y}, \vec{\varphi}, \vec{s}_i), \mathcal{V}_{\mathsf{core}}() \rangle$. Then they set $\vec{\tau} = (\tau_0, \ldots, \tau_n)^T$, $\vec{x} = (1, x_1, \ldots, x_n)^T$ and $\vec{\boldsymbol{t}}_1 = \mathsf{NTT}^{-1}(\vec{\tau})$, and decompose $x_i = x_{i,0} + \cdots + x_{i,l-1}\delta^{l-1}$ for $i = 1, \ldots, n$. The prover keeps the garbage coefficients γ_i from $\mathcal{P}_{\mathsf{core}}$ and sets $\vec{\gamma} = (\gamma_0, \ldots, \gamma_n)^T$.

For $j = 0, \ldots, l-1$:		
$\vec{\boldsymbol{z}}_j = \vec{\boldsymbol{y}}_j + \sum_{i=1}^{n} x_{i,j}\vec{\boldsymbol{s}}_i$		
If $\mathsf{Rej}((\vec{\boldsymbol{z}}_j), (\vec{\boldsymbol{y}}_j), \mathfrak{s}_2) = 1$, abort	$\xrightarrow{\vec{\boldsymbol{z}}_j}$	
		$\mathsf{VERIFY}(\boldsymbol{A}, \vec{\boldsymbol{u}}_i, \vec{\boldsymbol{w}}_j, x_{i,j}, \vec{\boldsymbol{z}}_j)$

The prover and verifier run the auxiliary linear proof from Figure 5, $\langle \mathcal{P}_{\mathsf{aux}}(\boldsymbol{B}, \vec{\boldsymbol{r}}^{(t)}, \vec{\boldsymbol{t}}, \vec{\gamma}, \vec{x}), \mathcal{V}_{\mathsf{aux}}(\boldsymbol{B}, \vec{\boldsymbol{t}}, p, \vec{x}) \rangle$, where $p = \langle \vec{z} \circ ((x_1 + \cdots + x_n)\vec{1} - \vec{z}), \vec{\varphi} \rangle$ for $\vec{z} = \sum_{j=0}^{l-1} \vec{z}_j \delta^j$. The verifier accepts if $\mathcal{V}_{\mathsf{aux}}$ accepts.

Fig. 1. Exact amortized opening proof for lattice-based hashes.

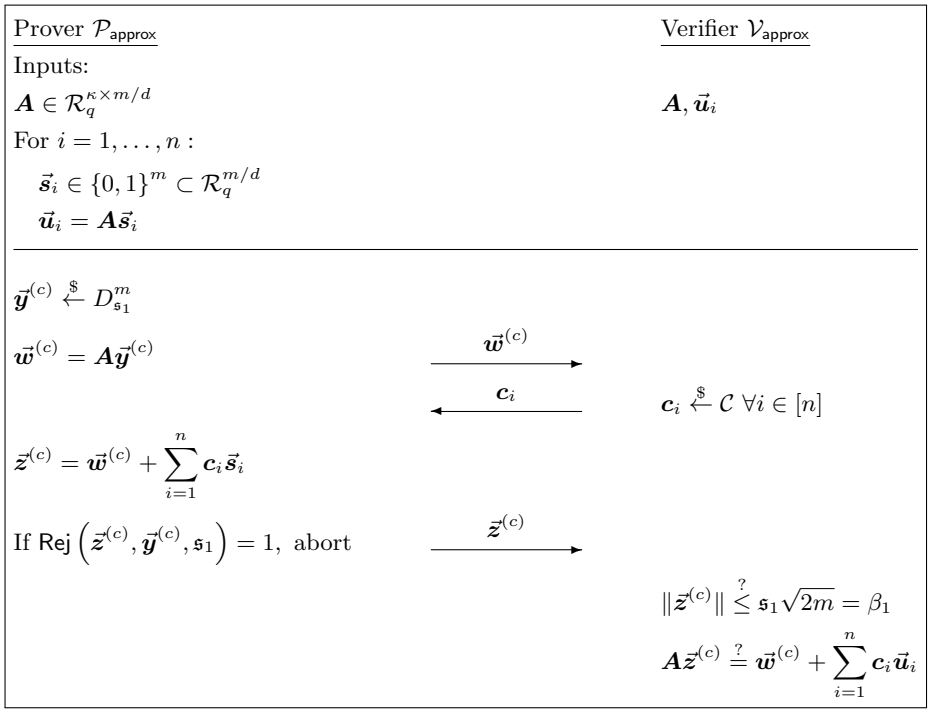


Fig. 2. Approximate amortized opening proof for lattice-based hashes. Used for bootstrapping the exact amortized proof in Figure 1.

$\underline{\mathsf{VERIFY}(\boldsymbol{A}, \vec{\boldsymbol{u}}_i, \vec{\boldsymbol{w}}_j, x_{i,j}, \vec{\boldsymbol{z}}_j)}$

01 For $j = 0, \ldots, l-1$:
02 $\quad \|\vec{\boldsymbol{z}}_j\| \overset{?}{\leq} \mathfrak{s}_2\sqrt{2m} = \beta_2$
03 $\quad \boldsymbol{A}\vec{\boldsymbol{z}}_j \overset{?}{=} \vec{\boldsymbol{w}}_j + \sum_{i=1}^{n} x_{i,j}\vec{\boldsymbol{u}}_i$

$\underline{\mathsf{AUXINIT}(\boldsymbol{B}_0, \boldsymbol{B}_1)}$

01 $\vec{\boldsymbol{r}}^{(t)} \xleftarrow{\$} \chi^{(\kappa_2+\lambda+\mu)d}$
02 $\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}}^{(t)}$
03 $\vec{e} = \mathsf{NTT}(\boldsymbol{B}_1\vec{\boldsymbol{r}}^{(t)})$
04 return $(\vec{\boldsymbol{r}}^{(t)}, \vec{\boldsymbol{t}}_0, \vec{e})$

$\underline{\mathsf{VERIFYAUX}(\boldsymbol{B}, \vec{\boldsymbol{t}}, \vec{\boldsymbol{w}}^{(t)}, \boldsymbol{c}^{(t)}, \vec{\boldsymbol{z}}^{(t)}, \theta, \boldsymbol{h}, \boldsymbol{v}, p, \vec{x})}$

01 $h_0 \overset{?}{=} 0$
02 $\|\vec{\boldsymbol{z}}^{(t)}\| \overset{?}{\leq} \mathfrak{s}_3\sqrt{2(\kappa_2+\lambda+\mu)d} = \beta_3$
03 $\boldsymbol{B}_0\vec{\boldsymbol{z}}^{(t)} \overset{?}{=} \vec{\boldsymbol{w}}^{(t)} + \boldsymbol{c}^{(t)}\vec{\boldsymbol{t}}_0$
04 $\langle\vec{\boldsymbol{b}}_2, \vec{\boldsymbol{z}}^{(t)}\rangle + \langle\mathsf{NTT}^{-1}(d\theta\vec{x}), \boldsymbol{B}_1\vec{\boldsymbol{z}}^{(t)}\rangle$
$\quad \overset{?}{=} \boldsymbol{v} + \boldsymbol{c}^{(t)}(\boldsymbol{t}_2 + \langle\mathsf{NTT}^{-1}(d\theta\vec{x}), \vec{\boldsymbol{t}}_1\rangle - \theta p - \boldsymbol{h})$

Fig. 3. Helper functions $\mathsf{VERIFY}()$, $\mathsf{AUXINIT}()$ and $\mathsf{VERIFYAUX}()$ used by exact amortized opening proof in Figure 1. They check the verification equations, initialize the auxiliary commitment, and check the verification equations of the auxiliary linear proof, respectively.

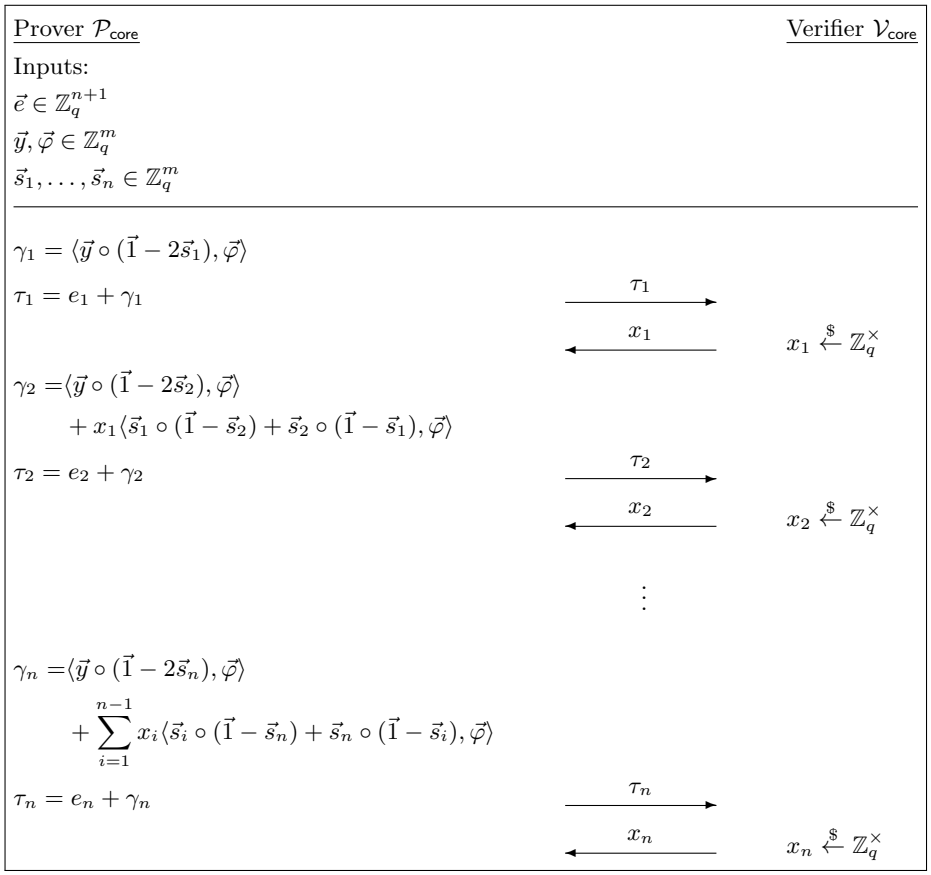


Fig. 4. Core protocol for exact amortized opening proof in Figure 1

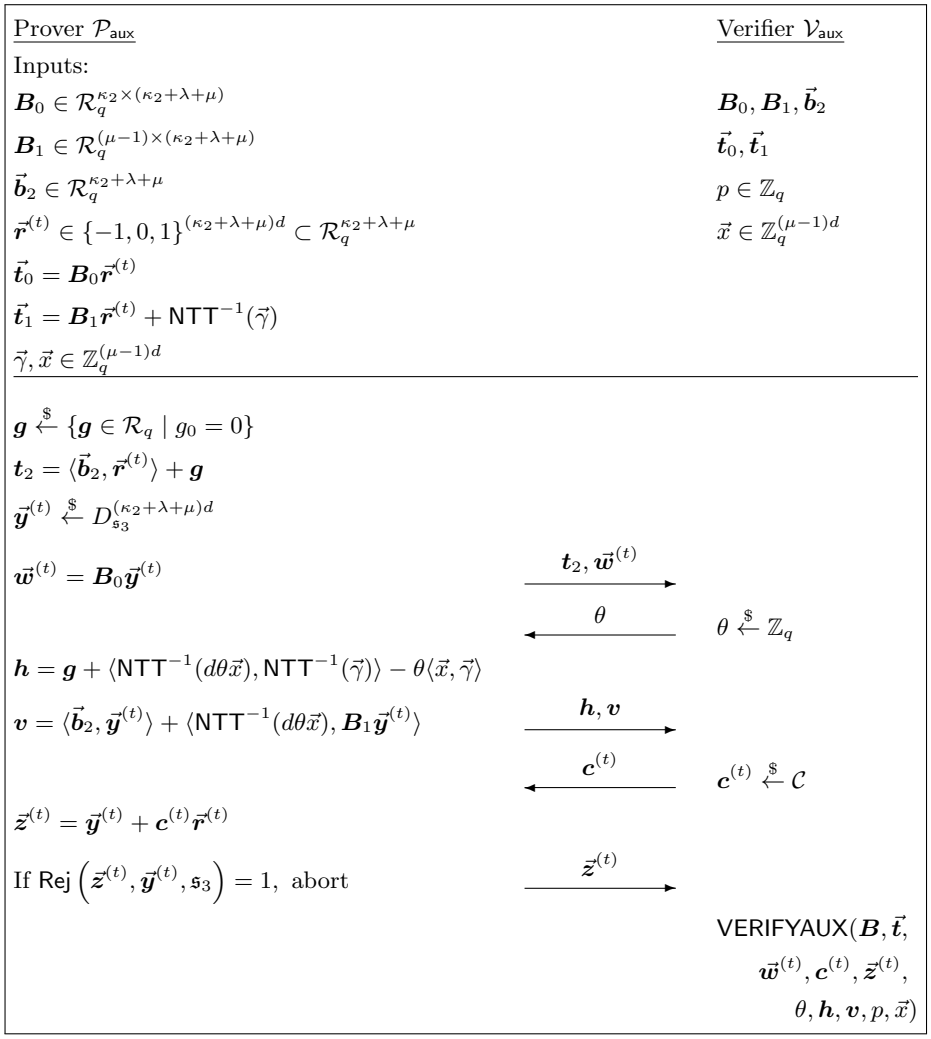


Fig. 5. Auxiliary linear proof needed in our exact amortized opening proof in Figure 1 and in the tree opening proof in Figures 6 and 7 .

Linear relations. Let $\vec{s} = \vec{s}_1 \parallel \cdots \parallel \vec{s}_n$ be the concatenation of all the binary $\vec{s}_i$ and $M = (M_1, \ldots, M_n) \in \mathbb{Z}_q^{\nu \times nm}$ with $M_i \in \mathbb{Z}_q^{\nu \times m}$ be a public matrix. Now suppose in full generality that we want to prove the linear equation

$$M\vec{s} = M_1\vec{s}_1 + \cdots + M_n\vec{s}_n = \vec{v}$$

for some public vector $\vec{v} \in \mathcal{R}_q^\nu$. So this is an "unstructured" linear equation not necessarily compatible with the polynomial structure. As usual, the equation can be proven by probabilistically reducing it to a scalar product first. So we prove

$$\langle M\vec{s} - \vec{v}, \vec{\psi}\rangle = \langle \vec{s}, M^T\vec{\psi}\rangle - \langle \vec{v}, \vec{\psi}\rangle = \sum_{i=1}^{n} \langle \vec{s}_i, M_i^T\vec{\psi}\rangle - \langle \vec{v}, \vec{\psi}\rangle = 0$$

for a uniformly random challenge vector $\vec{\psi} \in \mathbb{Z}_q^\nu$ that is given to the prover after the hashes $\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$ are known.

Now, we use a very similar approach to the one from Section 3.1. Concretely, let $\vec{\rho} = x_1^{-1}\vec{\rho}_1 + \cdots + x_n^{-1}\vec{\rho}_n$ where $\vec{\rho}_i = M_i^T\vec{\psi}$. Then we want to show that in the multivariate quadratic polynomial

$$f_{\mathsf{lin}}(x_1, \ldots, x_n) = \langle \vec{z}, \vec{\rho}\rangle - \langle \vec{v}, \vec{\psi}\rangle$$

the constant coefficient vanishes. More precisely, we want to prove the relation

$$\langle \vec{z}, \vec{\rho}\rangle - \langle \vec{v}, \vec{\psi}\rangle - \sum_{i=1}^{n} (\gamma_{2i-1}x_i^{-1} + \gamma_{2i}x_i) = 0$$

with garbage coefficients

$$\gamma_{2i-1}^{(\mathsf{lin})} = \langle \vec{y}, \vec{\rho}_i\rangle + \sum_{i=1}^{i-1} x_j\langle \vec{s}_j, \vec{\rho}_i\rangle,$$
$$\gamma_{2i}^{(\mathsf{lin})} = \sum_{j=1}^{i-1} x_j^{-1}\langle \vec{s}_i, \vec{\rho}_j\rangle.$$

We can share the garbage commitments between the linear and binary proofs by simply adding f_{lin} to f from Equation (7). That is, we finally prove

$$\langle \vec{z} \circ ((x_1 + \cdots + x_n)\vec{1} - \vec{z}), \vec{\varphi}\rangle + \langle \vec{z}, \vec{\rho}\rangle - \langle \vec{v}, \vec{\psi}\rangle$$
$$- \left(\gamma_0 + \sum_{i=1}^{n} (\gamma_{2i-1}x_i^{-1} + \gamma_{2i}x_i)\right) = 0.$$

This is sufficient although the equation now contains the constant garbage coefficient γ_0 so that it is not immediately clear why the contribution from the linear proof to the constant term vanishes. The reason is that the prover can commit to $\gamma_0 = -\langle \vec{y} \circ \vec{y}, \vec{\varphi}\rangle$ before the challenge $\vec{\psi}$ is known. Then, if the linear equation $M\vec{s} = \vec{v}$ were false, there would be a uniformly random contribution to the constant term that is independent from γ_0.

Product relations. By product relations we mean multiplicative relations of the form $s_1 s_2 = s_3$ between coefficients s_1, s_2, s_3 of the secret vectors $\vec{s}_i$. For simplicity we restrict to the case where the coefficients s_1, s_2, s_3 are from the same vector $\vec{s}_i$ and the relation holds in all vectors $\vec{s}_i$. More precisely, we consider relations $s_{i,j_1} s_{i,j_2} = s_{i,j_3}$ for a triple $(j_1, j_2, j_3) \in \{0, \ldots, m-1\}^3$ and all i. This is sufficient for many applications by packing the $\vec{s}_i$ in a suitable manner. For example, if we want to hash three binary vectors $\vec{a}, \vec{b}, \vec{c} \in \{0,1\}^{kn}$ for some $k \geq 1$, and prove that $\vec{a} \circ \vec{b} = \vec{c}$, then we write $\vec{a} = \vec{a}_1 \parallel \cdots \parallel \vec{a}_k$ with $\vec{a}_i \in \{0,1\}^n$, and let $\vec{s}_i$ be the columns of the matrix with rows $\vec{a}_i^T, \vec{b}_i^T, \vec{c}_i^T$,

$$\left(\vec{s}_1 \ldots \vec{s}_n\right) = \left(\vec{a}_1 \cdots \vec{a}_k \; \vec{b}_1 \cdots \vec{b}_k \; \vec{c}_1 \cdots \vec{c}_k\right)^T .$$

Now to prove the above relation we need to show that $s_{i,j} s_{i,j+k} = s_{i,j+2k}$ for all $i = 1, \ldots, n$ and $j = 0, \ldots, k-1$. Note that such relations are only a very slight generalisation of the relations $s_{i,j}(1 - s_{i,j}) = 0$ that we already prove in the binary proof. More general product relations are possible, but they come with a cost of more garbage commitments.

In the protocol, for every product relation $s_{i,j_1} s_{i,j_2} = s_{i,j_3}$ we add the polynomial

$$f_{\mathsf{prod}}(x_1, \ldots, x_n) = (z_{j_1} z_{j_2} - (x_1 + \cdots + x_n) z_{j_3})\theta$$

for a uniformly random challenge $\theta \in \mathbb{Z}_q$ to the previous $f + f_{\mathsf{lin}}$ that we prove. Similarly as f from the binary proof, the polynomial f_{prod} is a quadratic polynomial that has no terms divisible by x_i^2 if the product relation is true.

4.2 Proof Size

We study the size of the proof that is output by the non-interactive version of the protocol in this section. The non-interactive version is obtained by applying the Fiat-Shamir transform. We handle the slightly more general case where the secret vectors $\vec{\boldsymbol{s}}_i$ are not necessarily binary but have coefficients in the range $\{-\lfloor b/2 \rfloor, \ldots, \lfloor (b-1)/2 \rfloor\}$. Then there are $(b-1)n + 1$ garbage coefficients. The masking vector commitments $\vec{\boldsymbol{w}}^{(c)}$, $\vec{\boldsymbol{w}}_j$ and $\vec{\boldsymbol{w}}^{(t)}$ do not need to be included in the proof since they can be computed from the verification equations and then verified with the random oracle when the challenges are included in the proof. For $\vec{\boldsymbol{w}}^{(c)}$ and $\vec{\boldsymbol{w}}^{(t)}$ this is always efficient. Whether it is also efficient for the $\vec{\boldsymbol{w}}_j$ depends on n. For large n the cost of the n challenges x_i becomes bigger than the cost of the $\vec{\boldsymbol{w}}_j$. The polynomial $\boldsymbol{v}$ in the auxiliary proof does not need to be transmitted either. Hence a complete proof amounts to the objects $\vec{\boldsymbol{c}}, \vec{\boldsymbol{z}}^{(c)}$ for the approximate amortized proof; $\vec{\boldsymbol{t}}_0, \vec{\varphi}, \vec{\boldsymbol{t}}_1, \vec{x}, \vec{\boldsymbol{z}}_j$ for the main part; and $\boldsymbol{t}_2, \theta, \boldsymbol{h}, \boldsymbol{c}^{(t)}, \vec{\boldsymbol{z}}^{(t)}$ for the auxiliary proof. The actual size of the challenges as (vectors of) polynomials or $\mathbb{Z}_q$-integers does not contribute to the proof size since they can be expanded from small seeds by using a PRG. For the security level we are aiming for, 16 bytes suffice for each challenge seed. The full-size elements $\vec{\boldsymbol{t}}_0, \vec{\boldsymbol{t}}_1, \boldsymbol{t}_2, \boldsymbol{h}$ have a total size of $(\kappa_2 + \mu + 1)d\lceil \log q \rceil = (\kappa_2 + \lceil ((b-1)n + 1)/d \rceil + 2)d\lceil \log q \rceil$ bits. Next, the short vectors $\vec{\boldsymbol{z}}^{(c)}, \vec{\boldsymbol{z}}_j, \vec{\boldsymbol{z}}^{(t)}$

have size $m \log 12\mathfrak{s}_1 + lm \log 12\mathfrak{s}_2 + (\kappa_2 + \lambda + \mu)d \log 12\mathfrak{s}_3$ bits. Here we assume that the coefficients of the short vectors are bounded by $6\mathfrak{s}_i$ in absolute value, which can be ensured by the prover. Finally, the challenges $\vec{\boldsymbol{c}}, \vec{\varphi}, \vec{x}, \theta, \boldsymbol{c}^{(t)}$ need $4 + n$ seeds of total size $128(4 + n)$ bits.

We now compute the required standard deviations $\mathfrak{s}_1, \mathfrak{s}_2, \mathfrak{s}_3$ for the Gaussian masking vectors $\vec{\boldsymbol{y}}^{(c)}$, $\vec{\boldsymbol{y}}_j$ and $\vec{\boldsymbol{y}}^{(t)}$. So, we need to bound the ℓ_2 norms of the secrets vectors $\boldsymbol{c}_1\vec{\boldsymbol{s}}_1 + \cdots + \boldsymbol{c}_n\vec{\boldsymbol{s}}_n$, $x_{1,j}\vec{\boldsymbol{s}}_1 + \cdots + x_{n,j}\vec{\boldsymbol{s}}_n$, and $\boldsymbol{c}^{(t)}\vec{\boldsymbol{r}}$. For the rejection sampling we use the improved algorithm from [LNS21a] that leaks one bit of information about the secret. In usual applications of the proof system the prover will only ever compute one or at most very few proofs about a particular set of hashes $\vec{\boldsymbol{u}}_i$. We assume that the challenge polynomial distribution $\mathcal{C}$ for $\boldsymbol{c}_i$ and $\boldsymbol{c}^{(t)}$ is such that the polynomial coefficients are independently identically distributed in $\{-1, 0, 1\}$ with probabilities $1/4, 1/2, 1/4$, respectively. So the challenge polynomials have $3d/2$ bits of entropy. In particular, for ring rank $d = 128$ and fully splitting q of length around 128 bits, the NTT coefficients of $\boldsymbol{c}_i$ will have maximum probability p close to $1/q$. Then, a coefficient of a polynomial in $\boldsymbol{c}_i\vec{\boldsymbol{s}}_i$ is the weighted sum of d independent coefficients of $\boldsymbol{c}_i$, where the weights are given by the coefficients of the corresponding polynomial in $\vec{\boldsymbol{s}}_i$ (up to signs). Moreover, a coefficient of $\boldsymbol{c}_1\vec{\boldsymbol{s}}_1 + \cdots + \boldsymbol{c}_n\vec{\boldsymbol{s}}_n$ is the sum of dn such coefficients. Write S_n for this random variable. Its distribution is centered and has standard deviation $\mathfrak{s}_n \leq \lfloor b/2 \rfloor \sqrt{dn/2}$. By the central limit theorem, the distribution of the standardization $\frac{S_n}{\mathfrak{s}_n}$ converges to the standard normal distribution for $n \to \infty$. This is also true for the random variable S'_n that is distributed according to the discrete Gaussian Distribution $D_{\mathfrak{s}_n}$ with the same standard deviation as S_n. So, for all $x \in \mathbb{Z}$,

$$\lim_{n\to\infty} |\Pr[S_n \leq x\mathfrak{s}_n] - \Pr[S'_n \leq x\mathfrak{s}_n]| = 0,$$

and $D_{\mathfrak{s}_n}$ is a good model for the distribution of the coefficients of $\boldsymbol{c}_1\vec{\boldsymbol{s}}_1 + \cdots + \boldsymbol{c}_n\vec{\boldsymbol{s}}_n$. By the tail bound from Lemma 4, a coefficient is smaller than $14\lfloor b/2 \rfloor \sqrt{dn/2}$ in absolute value with probability bigger than $1 - 2^{-140}$. Then, using the union bound we conclude that no coefficient is bigger than that. Therefore, we have

$$\|\boldsymbol{c}_1\vec{\boldsymbol{s}}_1 + \cdots + \boldsymbol{c}_n\vec{\boldsymbol{s}}_n\|_2 \leq 14 \left\lfloor \frac{b}{2} \right\rfloor \sqrt{\frac{dmn}{2}} = \mathfrak{s}_1,$$

and similarly,

$$\|x_{1,j}\vec{\boldsymbol{s}}_1 + \cdots + x_{n,j}\vec{\boldsymbol{s}}_n\|_2 \leq 14 \left\lfloor \frac{b}{2} \right\rfloor \sqrt{\frac{(\delta^2 - 1)dmn}{12}} = \mathfrak{s}_2.$$

In the second inequality we have used that the discrete uniform distribution on $[-\delta/2, \delta/2 - 1]$ has standard deviation $\sqrt{(\delta^2 - 1)/12}$. For $\boldsymbol{c}^{(t)}\vec{\boldsymbol{r}}^{(t)}$ we make use of the fact that also $\vec{\boldsymbol{r}}^{(t)}$ is random with polynomial coefficients distributed according to the centered binomial distribution χ_2 modulo 3. It follows that every coefficient has standard deviation $\sqrt{5d/16}$, and, again by the tail and

union bounds, no coefficient is bigger than $14\sqrt{5d/16}$ with large probability. So,

$$\left\|\boldsymbol{c}^{(t)}\vec{\boldsymbol{r}}^{(t)}\right\|_2 \leq 14\sqrt{5d^2(\kappa_2+\lambda+\mu)/16} = \mathfrak{s}_3.$$

Example As an example we compute concrete sizes for proving $n = 1024$ hashes $\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$ of binary vectors $\vec{\boldsymbol{s}}_i$ of length $m = 2048$ over the ring $\mathcal{R}_q$ of rank $d = 128$ modulo a 128-bit fully-splitting prime q. We choose $l = 4$ so that $\delta \approx 2^{32}$. For the Module-SIS rank κ_2 and the Module-LWE rank λ of the BDLOP commitments scheme we use $\kappa_2 = 2$ and $\lambda = 32$. Then $\mathsf{MSIS}_{\kappa_2,8d\beta_3}$ has a classical Core-SVP cost of 2^{100} when using the BDGL16 sieve, and $\mathsf{MLWE}_{\lambda,\chi_2}$ has a classical Core-SVP cost of 2^{108}. The height κ of $\boldsymbol{A}$, i.e. the hash rank for the $\vec{\boldsymbol{u}}_i$, does not influence the proof size of our protocol, but we need Module-SIS to be hard for vectors of length $B = 4(d\beta_2 + \delta\beta_1/2 + dn\delta b\sqrt{m}/4)$. This is for example the case with $\kappa = 7$, where $\mathsf{MSIS}_{\kappa,B}$ has classical Core-SVP cost of 2^{213}. With these parameters we find that the proof size as explained above is 108.5 kilobytes. This translates to an amortized size of 108.6 bytes per equation.

One application of our amortized exact proof system is for proving statement about the plaintexts in FHE ciphertexts. The FHE ciphertexts have a purpose outside of the proof system and therefore their size does not count towards the proof size. Moreover, they can not be compressed because otherwise one could decrypt them anymore. Our proof system now allows to proof many such ciphertexts with a small amortized cost.

Proof. Correctness. We need to argue that the equations checked by the verifier are almost always true, except with negligible probability. From the sampling of the masking vectors $\vec{\boldsymbol{y}}^{(c)}$, $\vec{\boldsymbol{y}}_j$, and $\vec{\boldsymbol{y}}^{(t)}$ as discrete Gaussians with standard deviations $\mathfrak{s}_1$, $\mathfrak{s}_2$, and $\mathfrak{s}_3$, respectively, and Lemma 3 it follows that $\vec{\boldsymbol{z}}^{(c)}$, $\vec{\boldsymbol{z}}_j$, $\vec{\boldsymbol{z}}^{(t)}$ are again distributed as discrete Gaussians in non-aborting transcripts with the same standard deviations. Hence, the probability that the ℓ_2-norm $\left\|\vec{\boldsymbol{z}}^{(c)}\right\|_2$ is at most $\beta_1 = \mathfrak{s}_1\sqrt{2m}$ as checked by the verifier, is overwhelming by Lemma 4. The same holds for the norm checks of the other masked openings. Next, the equations $\boldsymbol{A}\vec{\boldsymbol{z}}^{(c)} = \vec{\boldsymbol{w}}^{(c)} + \sum_i \boldsymbol{c}_i\vec{\boldsymbol{u}}_i$, $\boldsymbol{A}\vec{\boldsymbol{z}}_j = \vec{\boldsymbol{w}}_j + \sum_i x_{i,j}\vec{\boldsymbol{u}}_i$, and $\boldsymbol{B}_0\vec{\boldsymbol{z}}^{(t)} = \vec{\boldsymbol{w}}^{(t)} + \boldsymbol{c}^{(t)}\vec{\boldsymbol{t}}_0$ are immediately seen to be always true. We also find

$$h_0 = g_0 + \langle\theta\vec{x},\vec{\gamma}\rangle - \langle\theta\vec{x},\vec{\gamma}\rangle = 0.$$

Finally, the last verification equation in $\mathsf{VERIFYAUX}()$ from the auxiliary proof remains. By substituting the expressions for $\vec{\boldsymbol{z}}^{(c)}$, $\boldsymbol{v}$, $\vec{\boldsymbol{t}}_1$, $\boldsymbol{t}_2$ and $\boldsymbol{h}$ as computed by the honest prover we obtain

$$\begin{aligned}
&\langle\vec{\boldsymbol{b}}_2,\vec{\boldsymbol{y}}^{(t)}\rangle + \boldsymbol{c}^{(t)}\langle\vec{\boldsymbol{b}}_2,\vec{\boldsymbol{r}}^{(t)}\rangle + \langle\mathsf{NTT}^{-1}(d\theta\vec{x}),\boldsymbol{B}_1\vec{\boldsymbol{y}}^{(t)}\rangle + \boldsymbol{c}^{(t)}\langle\mathsf{NTT}^{-1}(d\theta\vec{x}),\boldsymbol{B}_1\vec{\boldsymbol{r}}^{(t)}\rangle \\
&= \langle\vec{\boldsymbol{b}}_2,\vec{\boldsymbol{y}}^{(t)}\rangle + \langle\mathsf{NTT}^{-1}(d\theta\vec{x}),\boldsymbol{B}_1\vec{\boldsymbol{y}}^{(t)}\rangle + \boldsymbol{c}^{(t)}(\langle\vec{\boldsymbol{b}}_2,\vec{\boldsymbol{r}}^{(t)}\rangle + \boldsymbol{g}) \\
&\quad + \boldsymbol{c}^{(t)}(\langle\mathsf{NTT}^{-1}(d\theta\vec{x}),\boldsymbol{B}_1\vec{\boldsymbol{r}}^{(t)}\rangle + \langle\mathsf{NTT}^{-1}(d\theta\vec{x}),\mathsf{NTT}^{-1}(\vec{\gamma})\rangle - \theta p) \\
&\quad - \boldsymbol{c}^{(t)}(\boldsymbol{g} + \langle\mathsf{NTT}^{-1}(d\theta\vec{x}),\mathsf{NTT}^{-1}(\vec{\gamma})\rangle - \langle\theta\vec{x},\vec{\gamma}\rangle).
\end{aligned}$$

Cancelling all equal terms shows that this equation is implied by

$$\begin{aligned}
&p - \langle \vec{x}, \vec{\gamma} \rangle \\
&= \langle \vec{z} \circ ((x_1 + \cdots + x_n)\vec{1} - \vec{z}, \vec{\varphi} \rangle - \langle \vec{x}, \vec{\gamma} \rangle \\
&= -\langle \vec{y} \circ \vec{y}, \vec{\varphi} \rangle \\
&\quad + \sum_{i=1}^{n} x_i \left(\langle \vec{y} \circ (\vec{1} - 2\vec{s}_i), \vec{\varphi} \rangle + \sum_{j=1}^{i-1} x_j \langle \vec{s}_i \circ (\vec{1} - \vec{s}_j) + \vec{s}_j \circ (\vec{1} - \vec{s}_i), \vec{\varphi} \rangle \right) \\
&\quad - \left(\gamma_0 + \sum_{i=1}^{n} x_i \gamma_i \right) \\
&= 0.
\end{aligned}$$

It follows from the construction of the garbage coefficients γ_i that this equation is indeed correct.

Zero-Knowledge. The proof of the zero-knowledge property is standard for this type of protocol. In the real execution the masked openings $\vec{\boldsymbol{z}}^{(c)}$, $\vec{\boldsymbol{z}}_j$, $\vec{\boldsymbol{z}}^{(t)}$ are distributed as discrete Gaussians. So, when we replace them by independently sampled discrete Gaussian vectors and compute $\vec{\boldsymbol{w}}^{(c)}$, $\vec{\boldsymbol{w}}_j$, $\vec{\boldsymbol{w}}^{(t)}$, $\boldsymbol{v}$ such that the verification equations are still all correct, then we get a transcript that is statistically close to the real transcript with statistical distance 2^{-100} by Lemma 3. Now, the randomness vector $\vec{\boldsymbol{r}}^{(t)}$ for the BDLOP commitment scheme is only used anymore in the MLWE vectors $\boldsymbol{B}_0\vec{\boldsymbol{r}}^{(t)}$, $\boldsymbol{B}_1\vec{\boldsymbol{r}}^{(t)}$, and $\langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}}^{(t)} \rangle$. Hence, when we replace these by uniformly random vectors, we obtain a transcript that can be distinguished from the previous transcript with advantage ε only by an adversary that can distinguish $\mathsf{MLWE}_{\lambda,\chi}$ with the same advantage. Next, we can replace the vectors $\vec{\boldsymbol{t}}_0$, $\vec{\boldsymbol{t}}_1$, and $\boldsymbol{t}_2$ by independently uniformly random vectors without changing the distribution of the transcript. Now, the only element that still depends on the secret vectors $\vec{\boldsymbol{s}}_i$ is the polynomial $\boldsymbol{h}$. But $\boldsymbol{h}$ contains the additive uniformly random polynomial $\boldsymbol{g}$ with constant zero coefficient. And $\boldsymbol{g}$ is now only used in the computation of $\boldsymbol{h}$ since we have already replaced $\boldsymbol{t}_2$. Therefore, we can also replace $\boldsymbol{h}$ by a uniformly random polynomial with zero constant coefficient without changing the transcript distribution. We have thus arrived at a simulatable transcript as required in the theorem statement.

Knowledge-Soundness. The first goal for the extractor is to obtain weak openings $\vec{\boldsymbol{s}}_i^*$ for all the hashes $\vec{\boldsymbol{u}}_i$. This is achieved by extracting the approximate amortized opening proof at the beginning of the protocol. Here one NTT coefficient $\vec{\boldsymbol{s}}_i^* \bmod \mathfrak{p}_j$ of one opening $\vec{\boldsymbol{s}}_i^*$ is extracted at a time, where $i \in \{1, \ldots, n\}$ and $\mathfrak{p}_j \subset \mathcal{R}$, $j \in \{1, \ldots, d\}$, is one of the prime ideals dividing q in $\mathcal{R}$. The extractor repeatedly interacts with $\mathcal{P}^*$ where he freshly samples all challenges in each interaction until he hits an accepting interaction. Let $\boldsymbol{c}_i$ and $\vec{\boldsymbol{z}}^{(c)}$ be the i-th polynomial challenge and the corresponding masked opening in this interaction, respectively. Next, the extractor interacts again with the prover, but this time he sends the same challenges as in the previous accepting interaction, except for $\boldsymbol{c}_i$, which is replaced by a freshly sampled challenge $\boldsymbol{c}_i'$ under the constraint that

$\boldsymbol{c}_i' \not\equiv \boldsymbol{c}_i \pmod{\mathfrak{p}_j}$. The extractor repeats until he either hits a second accepting interaction with masked opening $\vec{\boldsymbol{z}}^{(c')}$, or the upper runtime limit $2/(\varepsilon/2-p)$ is reached, in which case the extractor aborts. We compute the expected runtime and probability for obtaining two accepting transcripts. The first accepting hit takes expected time $1/\varepsilon$. Then, with probability at least $1/2$, the challenges lie on a heavy row for $\boldsymbol{c}_i$ so that the acceptance probability when keeping all challenges fixed except $\boldsymbol{c}_i$ is at least $\varepsilon/2$. The constraint $\boldsymbol{c}_i' \not\equiv \boldsymbol{c} \pmod{\mathfrak{p}_j}$ reduces the bound on the acceptance probability to $\varepsilon/2 - p$. Now, the probability that a process with expected runtime $1/(\varepsilon/2-p)$ runs longer than twice this amount is at most $1/2$ by Markov's inequality. Hence, when the first accepting interaction lies on a heavy row for $\boldsymbol{c}_i$, then with probability at least $1/2$ we obtain an accepting interaction in the second stage before aborting due to the time limit. Overall, we have that with probability at least $1/4$ we obtain two accepting interactions in expected time at most $1/\varepsilon + 2/(\varepsilon/2-p) < 3/(\varepsilon/2-p)$. By restarting the whole process when it aborts, we conclude that in expected time $12/(\varepsilon/2-p)$ we get the two accepting interactions.

Next, we have from the verification equations in the approximate amortized opening proof in Figure 2,

$$\boldsymbol{A}\left(\vec{\boldsymbol{z}}^{(c)} - \vec{\boldsymbol{z}}^{(c')}\right) = (\boldsymbol{c}_i - \boldsymbol{c}_i')\vec{\boldsymbol{u}}_i.$$

Since $\boldsymbol{c}_i - \boldsymbol{c}_i' \not\equiv 0 \pmod{\mathfrak{p}_j}$, we can divide by the challenge difference in the residue field modulo $\mathfrak{p}_j$ and obtain $\vec{\boldsymbol{s}}_{i,j}^*$ such that $\boldsymbol{A}\vec{\boldsymbol{s}}_{i,j}^* \equiv \vec{\boldsymbol{u}}_i \pmod{\mathfrak{p}_j}$. By performing the strategy for all prime ideals $\mathfrak{p}_j$, we can lift the local solutions $\vec{\boldsymbol{s}}_{i,j}^*$ to a global solution $\boldsymbol{A}\vec{\boldsymbol{s}}_i^* = \vec{\boldsymbol{u}}_i$ over $\mathcal{R}_q$. In the process we obtain $\bar{\boldsymbol{c}}_j$ and $\bar{\vec{\boldsymbol{z}}}_j$ for all j such that $\bar{\boldsymbol{c}}_j \notin \mathfrak{p}_j$ and $\bar{\boldsymbol{c}}_j\vec{\boldsymbol{s}}_i^* \equiv \bar{\vec{\boldsymbol{z}}}_j \pmod{\mathfrak{p}_j}$. The last equality must even hold over $\mathcal{R}_q$, $\bar{\boldsymbol{c}}_j\vec{\boldsymbol{s}}_i^* = \bar{\vec{\boldsymbol{z}}}_j$, and hence $\vec{\boldsymbol{s}}_i^*$ is a weak opening for $\vec{\boldsymbol{u}}_i$. For otherwise we have the Module-SIS solution $\bar{\boldsymbol{c}}_{j'}\bar{\vec{\boldsymbol{z}}}_j - \bar{\boldsymbol{c}}_j\bar{\vec{\boldsymbol{z}}}_{j'}$ for $\boldsymbol{A}$ of length $8\omega\beta_1$ for some j'.

In the same way we obtain weak openings $\vec{\boldsymbol{s}}_i^*$ for all $\vec{\boldsymbol{u}}_i$. If they are all binary then we are done. So assume this is not the case and that $\vec{\boldsymbol{s}}_{i_0}^*$ is the last non-binary vector, i.e. $\vec{\boldsymbol{s}}_{i_0}^* \notin \{0,1\}^m$ and $\vec{\boldsymbol{s}}_i^* \in \{0,1\}^m$ for all $i > i_0$. For the following discussion we are only interested in the main part of the protocol, i.e. the protocol in Figure 1 without the initial approximate proof and the trailing auxiliary linear proof. The acceptance probability for the main protocol, i.e. the probability that the main verification equations in $\mathsf{VERIFY}()$ are correct, is at least ε. We partition the challenge tuple of the main protocol into two pieces $X_0 = (x_1, \ldots, x_{i_0-1})$ and $X_1 = (\vec{\varphi}, x_{i_0}, \ldots, x_n)$. There exists a choice of the challenges in X_0 such that the acceptance probability over the remaining challenges X_1 is at least ε. We fix this choice. Then, for each choice of the challenges in X_1 we can define $\vec{\boldsymbol{y}}_j^* \in \mathcal{R}_q^{m/d}$, $j = 0, \ldots, l-1$, such that

$$\vec{\boldsymbol{z}}_j = \vec{\boldsymbol{y}}_j^* + x_{i_0,j}\vec{\boldsymbol{s}}_{i_0}^* + \cdots + x_{n,j}\vec{\boldsymbol{s}}_n^* \tag{9}$$

in the corresponding interaction. If the interaction is accepting, then, by construction,

$$\boldsymbol{A}\vec{\boldsymbol{y}}_j^* = \vec{\boldsymbol{w}}_j + x_{1,j}\vec{\boldsymbol{u}}_1 + \cdots + x_{i_0-1,j}\vec{\boldsymbol{u}}_{i_0-1}.$$

The $\vec{\boldsymbol{y}}_j^*$ must be the same for all accepting interactions. To the contrary, assume without loss of generality that there are two challenge tuples $(\varphi, x_{i_0}, \ldots, x_n)$ and $(\varphi', x'_{i_0}, \ldots, x'_n)$ with different $\vec{\boldsymbol{y}}_0^*$ and $\vec{\boldsymbol{y}}_0^{*\prime}$, respectively. The vectors $\vec{\boldsymbol{y}}_0^*$ and $\vec{\boldsymbol{y}}_0^{*\prime}$ must differ modulo at least one of the prime ideals, say $\mathfrak{p}_j$. But then, from the extraction of the weak openings for the hashes, we know $\bar{\boldsymbol{c}}_j$ such that $\bar{\boldsymbol{c}}_j \notin \mathfrak{p}_j$ and $\bar{\boldsymbol{c}}_j \vec{\boldsymbol{s}}_{i_0}^*$ is short; that is, of length at most $2\beta_1$. Thus,

$$\begin{aligned}
&\boldsymbol{A}(\bar{\boldsymbol{c}}_j \vec{\boldsymbol{z}}_0 - x_{i_0,0} \bar{\boldsymbol{c}}_j \vec{\boldsymbol{s}}_{i_0}^* - \bar{\boldsymbol{c}}_j \sum_{i=i_0+1}^{n} x_{i,0} \vec{\boldsymbol{s}}_i^*) \\
&= \bar{\boldsymbol{c}}_j \boldsymbol{A} \vec{\boldsymbol{y}}_0^* = \bar{\boldsymbol{c}}_j \boldsymbol{A} \vec{\boldsymbol{y}}_0^{*\prime} \\
&= \boldsymbol{A}(\bar{\boldsymbol{c}}_j \vec{\boldsymbol{z}}'_0 - x'_{i_0,1} \bar{\boldsymbol{c}}_j \vec{\boldsymbol{s}}_{i_0}^* - \bar{\boldsymbol{c}}_j \sum_{i=i_0+1}^{n} x'_{i,1} \vec{\boldsymbol{s}}_i^*)
\end{aligned}$$

gives a Module-SIS solution for A of length less than $B = 2(2\omega\beta_2 + 2\delta\beta_1 + 2n\omega\delta\sqrt{m})$. Here we have used that all $\vec{\boldsymbol{s}}_i^*$ are binary and hence of length at most $\sqrt{m}$ for $i > i_0$.

For completeness, we sketch a simple concrete treatment of the previous argument. With probability at least $1/2$, a completely random accepting interaction (random over all challenges in X_0 and X_1) is such that the challenges in X_0 define a "heavy row" for the challenges in X_1. This means that the acceptance probability over the challenges in X_1 when fixing X_0 is at least $\varepsilon/2$. So, it is easy for the extractor to find such a heavy row. Now, assume that with another probability of at least $1/2$, a heavy row is such that the largest fraction of accepting interactions with matching $\vec{\boldsymbol{y}}_j^*$ make up for less than $1/2$ of all the accepting interactions on the heavy row. Then, with probability at least $1/8$ two random accepting interactions with equal X_0 would have different $\vec{\boldsymbol{y}}_j^*$ and lead to an MSIS solution as above. So, this cannot be and therefore with probability at least $1/2$, a heavy row is such that at least $1/2$ of the accepting interactions on the row result in the same masking vectors $\vec{\boldsymbol{y}}_j^*$. Hence, it is easy for the extractor to find challenges X_0 such that the interactions with fixed X_0 are accepting and share the same $\vec{\boldsymbol{y}}_j^*$ with probability at least $\varepsilon/4$ over the challenges in X_1. We can increase the latter probability by also increasing the runtime needed by the extractor to find the challenges X_0. But since we don't care about the extractor runtime in the theorem statement we can assume the limiting probability ε.

So, our conclusion above makes sense that we can fix challenges X_0 such that interactions are accepting and the $\vec{\boldsymbol{z}}_j$ are given by Equation (9) for fixed $\vec{\boldsymbol{y}}_j^*$ with probability at least ε. For the integer coefficient vector $\vec{z} = \sum_{j=0}^{l-1} \vec{z}_j \delta^j$ this translates to that it is given by

$$\vec{z} = \vec{y}^* + x_{i_0} \vec{s}_{i_0}^* + \cdots + x_n \vec{s}_n^*$$

in those interactions, with fixed $\vec{y}^*, \vec{s}_i^*$. Next, in the auxiliary linear proof at the end of the protocol the relation

$$\langle \vec{z} \circ ((x_1 + \cdots + x_n)\vec{1} - \vec{z}), \vec{\varphi} \rangle = \gamma_0^* + \gamma_1^* x_1 + \cdots + \gamma_n^* x_n$$

is proven. Here the coefficients γ_i^* are the (extracted) garbage coefficients inside the BDLOP commitment. They are such that γ_i is independent from $x_i, \dots, x_n$ since it was committed to before the prover knew $x_i, \dots, x_n$. By plugging in the expression for $\vec{z}$ we can now decompose the relation into

$$f_0 + f_{i_0} x_{i_0} + \langle \vec{s}_{i_0}^* \circ (\vec{1} - \vec{s}_{i_0}^*), \vec{\varphi} \rangle x_{i_0}^2 + f_{i_0+1}^{(x_{i_0})} x_{i_0+1} + \cdots + f_n^{(x_{i_0}, \dots, x_{n-1})} x_n = 0$$

where

$$\begin{aligned} f_0 &= \langle \vec{y}^* \circ ((x_1 + \cdots + x_{i_0-1})\vec{1} - \vec{y}^*), \vec{\varphi} \rangle \\ &\quad - (\gamma_0^* + x_1\gamma_1^* + \cdots + x_{i_0-1}\gamma_{i_0-1}^*) \\ f_i^{(x_{i_0}, \dots, x_{i-1})} &= \langle \vec{y}^* \circ (\vec{1} - \vec{s}_i^*), \vec{\varphi} \rangle + \langle \vec{s}_i^* \circ ((x_1 + \cdots + x_{i_0-1})\vec{1} - \vec{y}^*), \vec{\varphi} \rangle \\ &\quad + \sum_{j=i_0}^{i-1} x_j \langle \vec{s}_j^* \circ (\vec{1} - \vec{s}_i^*) + (\vec{1} - \vec{s}_j^*) \circ \vec{s}_i^*, \vec{\varphi} \rangle - \gamma_i^* \end{aligned}$$

So we see that the coefficients $f_i^{(x_{i_0}, \dots, x_{i-1})}$ depend on $x_1, \dots, x_{i-1}$ but not on $x_i, \dots, x_n$. Moreover, with probability $(1 - 1/q)$ for uniformly random $\vec{\varphi}$, the term $\langle \vec{s}_{i_0}^* \circ (\vec{1} - \vec{s}_{i_0}^*), \vec{\varphi} \rangle$ is non-zero since $\vec{s}_{i_0}^*$ is not binary. Therefore, Lemma 2 says that the equation is only true with probability at most $1/q + 2n/q < \varepsilon$. Hence, with probability at least $\varepsilon - (2n+1)/q$, the prover must be able to succeed in the auxiliary linear proof although the relation is not true. Now, since this probability is bigger than the soundness error of the linear proof, $\varepsilon - (2n+1)/q > 1/q + p$, [ENS20, Theorem 3.1] says that in this case we can extract an $\mathsf{MSIS}_{\kappa_2, 8\omega\beta_3}$ solution for $\boldsymbol{B}_0$. □

5 Induction

In many applications the public input hashes $\vec{\boldsymbol{u}}_i$ to our exact binary opening proof from Section 4 are in fact produced as part of a larger zero-knowledge proof system and their size counts towards the proof size. In the opening proof the two dominating terms in the proof size are of order $n \log q$ for the garbage commitments, and $m \log q$ for the masked openings, for a total of mn secret coefficients. On the other hand, the hashes $\vec{\boldsymbol{u}}_i$ are of size $n\kappa d \log q$. So we see that their size is very significant for the overall bandwidth efficiency. In fact, the hashes are about two orders of magnitude larger than their proof and it would be good if we did not need to transmit all the $\vec{\boldsymbol{u}}_i$. In this section we show how this can in fact be achieved by hashing them up in a Merkle hash tree and using our opening proof as a building block to prove by induction an opening to the hash tree when only the root hash is given.

Tree Construction. In our lattice-based hash tree, the hash input vector for an inner node consists of the binary expansions of the hash output vectors from the two children of the node. So the number of input bits m of the hash function

must be twice the number of output bits, i.e. $m = 2\kappa d\lceil \log q \rceil$. Then we define the *gadget matrix*

$$\boldsymbol{G} = \boldsymbol{I}_\kappa \otimes \left(1\ 2\ \cdots\ 2^{\lceil \log q \rceil - 1}\right) \in \mathcal{R}_q^{\kappa \times \kappa \lceil \log q \rceil}$$

that we use to reconstruct the hashes from their binary expansions. Now, the hash tree is constructed as follows. Let a be the depth of the tree. Then, the inner nodes are given by

$$\begin{aligned} \vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i, \quad \vec{\boldsymbol{s}}_i = \begin{pmatrix} \vec{\boldsymbol{s}}_{i,l} \\ \vec{\boldsymbol{s}}_{i,r} \end{pmatrix} \in \{0,1\}^m \subset \mathcal{R}_q^{m/d}, \\ \boldsymbol{G}\vec{\boldsymbol{s}}_{i,l} = \vec{\boldsymbol{u}}_{2i}, \\ \boldsymbol{G}\vec{\boldsymbol{s}}_{i,r} = \vec{\boldsymbol{u}}_{2i+1} \end{aligned} \tag{10}$$

for $i = 1, \ldots, 2^{a-1} - 1$. In particular $\vec{\boldsymbol{u}}_1$ is the root of the tree. The leafs are $\vec{\boldsymbol{u}}_{2^{a-1}+j} = \boldsymbol{A}\vec{\boldsymbol{s}}_{2^{a-1}+j}$ for $j = 0, \ldots, 2^{a-1} - 1$. More generally, the nodes $\vec{\boldsymbol{u}}_{2^k}, \ldots, \vec{\boldsymbol{u}}_{2^{k+1}-1}$ form level k of the tree, where $0 \le k \le a - 1$.

Proof by Induction. So we have a total of $n = 2^a - 1$ binary vectors $\vec{\boldsymbol{s}}_i$ that recursively hash to $\vec{\boldsymbol{u}}_1$ and that we want to prove knowledge of. Our protocol is easiest to understand as a sequence $\pi_{a-1}, \pi_{a-2}, \ldots, \pi_0$ of $a = \lceil \log n \rceil$ subproofs that are essentially instances of our binary opening proof from Section 4. There is one subproof for each level of the tree in the order from the leaves to the root, and the subproofs are indexed by the corresponding level. More precisely, π_k proves knowledge of the level-k binary vectors $\vec{\boldsymbol{s}}_{2^k}, \ldots, \vec{\boldsymbol{s}}_{2^{k+1}-1}$.

All the π_k share one amortized masked opening of all the vectors $\vec{\boldsymbol{s}}_i$. Hence, in the very end the prover sends

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \sum_{i=1}^{2^a-1} x_i \vec{\boldsymbol{s}}_i.$$

Actually, the prover sends the short shares $\vec{\boldsymbol{z}}_j = \vec{\boldsymbol{y}}_j + \sum_i x_{i,j}\vec{\boldsymbol{s}}_i$ that compose to $\vec{\boldsymbol{z}}$ but we explain the protocol in terms of the single vector $\vec{\boldsymbol{z}}$ as this simplifies the presentation. The 2^k challenges $x_{2^k}, \ldots, x_{2^{k+1}-1}$ for the level-k binary vectors are from the subproof π_k. Therefore and because of the reverse ordering of the subproofs, at the beginning of π_k the prover knows all the challenges $x_{2^{k+1}}, \ldots, x_{2^a-1}$ from deeper levels. We can thus absorb the terms $x_i\vec{\boldsymbol{s}}_i$, $i \ge 2^{k+1}$, in $\vec{\boldsymbol{z}}$ into the masking vector and use

$$\vec{\boldsymbol{y}}_k = \vec{\boldsymbol{y}} + \sum_{i=2^{k+1}}^{2^a-1} x_i \vec{\boldsymbol{s}}_i$$

as the masking vector in π_k. So unlike in isolated instances of the binary opening proof, π_k inherits the mask from previous parts of the overall protocol instead of sampling a fresh mask. The prover then sends the commitment $\vec{\boldsymbol{w}}_k = \boldsymbol{A}\vec{\boldsymbol{y}}_k$

(composed from $\vec{\boldsymbol{w}}_{k,j} = \boldsymbol{A}\vec{\boldsymbol{y}}_{k,j}$). Next, he engages in the 2^{k+1}-round interaction where he produces the garbage commitments and receives the challenges x_{2^k+j} for proving exactly as before that the vectors $\vec{\boldsymbol{s}}_{2^k+j}$ are binary. Furthermore, the verifier only knows the root hash $\vec{\boldsymbol{u}}_1$ and can check the verification equation

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}}_0 + x_1\vec{\boldsymbol{u}}_1.$$

for the last subproof π_0 at the end of the protocol. So, to connect the subproofs with each other and prove the verification equations for the π_k, $k \geq 1$, the prover proves the following linear relations,

$$\sum_{i=2^k}^{2^{k+1}-1} (x_{2i}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,l} + x_{2i+1}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}) = \vec{\boldsymbol{w}}_k - \vec{\boldsymbol{w}}_{k+1}. \tag{11}$$

The challenges x_{2i}, x_{2i+1}, and the vectors $\vec{\boldsymbol{w}}_k$, $\vec{\boldsymbol{w}}_{k+1}$ are known by both the prover and the verifier at the start of π_k so this relation can be proven with the linear proof technique from Section 4.1. Concretely, for each $k = 0, \ldots, a-2$ let $\vec{\psi}_k \in \mathbb{Z}_q^{\kappa d}$ be a challenge and define

$$\vec{\rho}_i = \mathsf{Rot}\begin{pmatrix} x_{2i}\boldsymbol{G}^\dagger \\ x_{2i+1}\boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_k$$

for all $i = 2^k, \ldots, 2^{k+1}$ with the multiplication matrix $\mathsf{Rot}(\boldsymbol{G}^\dagger)$ associated to the conjugate transpose of the polynomial matrix $\boldsymbol{G}$. Then in π_k the prover commits to the garbage coefficients

$$\gamma_{2i-1}^{(\mathsf{lin})} = \left\langle \vec{y}_k + \sum_{j=i+1}^{2^{k+1}-1} x_j\vec{s}_j, \vec{\rho}_i \right\rangle,$$
$$\gamma_{2i}^{(\mathsf{lin})} = \left\langle \vec{s}_i, \sum_{j=i+1}^{2^{a-1}-1} x_j^{-1}\vec{\rho}_j \right\rangle$$

for $i = 2^k, \ldots, 2^{k+1} - 1$. Finally, the following linear relation is proven in the auxiliary proof at the end of the protocol,

$$\left\langle \vec{z}, \sum_{i=1}^{2^{a-1}-1} x_i^{-1}\vec{\rho}_i \right\rangle - \sum_{k=0}^{a-2} \left\langle \vec{w}_k - \vec{w}_{k+1}, \vec{\psi}_k \right\rangle$$
$$= \sum_{i=1}^{2^{a-1}-1} \left(x_i^{-1}\gamma_{2i-1}^{(\mathsf{lin})} + x_i\gamma_{2i}^{(\mathsf{lin})} \right).$$

We now explain at a high level why this protocol suffices for proving the hash tree. For $0 \leq k \leq a-2$, consider the statement S_k that the prover knows binary vectors $\vec{\boldsymbol{s}}_1, \ldots, \vec{\boldsymbol{s}}_{2^k-1}$ and corresponding $\vec{\boldsymbol{u}}_1, \ldots, \vec{\boldsymbol{u}}_{2^{k+1}-1}$ as in Equation (10), and that

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}}_{k'} + \sum_{i=1}^{2^{k'+1}-1} x_i\vec{\boldsymbol{u}}_i \tag{12}$$

is true for all $0 \leq k' \leq k$ in (almost) all accepting interactions. The statement is trivially true for $k = 0$ because the list of known vectors is empty in this case and (12) is directly checked by the verifier.

Now, we argue that the subproof π_k proves the statement S_{k+1} if S_k holds true. We rewrite (12) and have

$$\boldsymbol{A}\left(\vec{\boldsymbol{z}} - \sum_{i=1}^{2^k-1} x_i \vec{\boldsymbol{s}}_i\right) = \vec{\boldsymbol{w}}_k + \sum_{i=2^k}^{2^{k+1}-1} x_i \vec{\boldsymbol{u}}_i.$$

Here the preimage on the left hand side is short since $\vec{\boldsymbol{z}}$ is short and all the $\vec{\boldsymbol{s}}_i$ are binary. So, for every accepting transcript we can compute a short vector $\vec{\boldsymbol{z}}_k = \vec{\boldsymbol{z}} - \sum_{i=1}^{2^k-1} x_i \vec{\boldsymbol{s}}_i$ that fulfills the main verification equation for the binary opening proof π_k for level k. Conceptually this means any prover for the protocol in this section can be converted to a prover for the level-k hashes exactly as in Section 4. Therefore we can use the extractor for our exact opening proof from Section 4 and compute binary preimages $\vec{\boldsymbol{s}}_{2^k}, \ldots, \vec{\boldsymbol{s}}_{2^{k+1}-1}$ for the hashes $\vec{\boldsymbol{u}}_{2^k}, \ldots, \vec{\boldsymbol{u}}_{2^{k+1}-1}$. Moreover the newly extracted binary preimages define the level-$(k+1)$ hashes $\vec{\boldsymbol{u}}_{2^{k+1}}, \ldots, \vec{\boldsymbol{u}}_{2^{k+2}-1}$, and from the linear proof for (11) included in π_k it follows that

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}}_k + \sum_{i=1}^{2^{k+1}-1} x_i \boldsymbol{u}_i = \vec{\boldsymbol{w}}_{k+1} + \sum_{i=1}^{2^{k+2}-1} x_i \vec{\boldsymbol{u}}_i.$$

Therefore we have established that statement S_{k+1} is true.

It then follows by induction that the statement S_{a-1} is true. And a very similar argument for the last-level proof π_{a-1}, just without the linear proof connecting to a previous level, shows that the prover also knows preimages for the tree leaves, which completes the proof of the full hash tree.

Note that there is no problem with zero-knowledge associated with sending all the $\vec{\boldsymbol{w}}_k$ since they differ from $\vec{\boldsymbol{w}}_{a-1} = \boldsymbol{A}\vec{\boldsymbol{y}}$ only by terms of the form $x_i \vec{\boldsymbol{u}}_i$ that we would send in the clear if we directly used the proof from Section 4. Finally note that the size of the $\vec{\boldsymbol{w}}_k$ is small — we have effectively traded the $n+1 = 2^a$ uniformly random vectors $\vec{\boldsymbol{u}}_i$, $\vec{\boldsymbol{w}}$ for the only $a+1$ vectors $\vec{\boldsymbol{u}}_1$ and $\vec{\boldsymbol{w}}_k$.

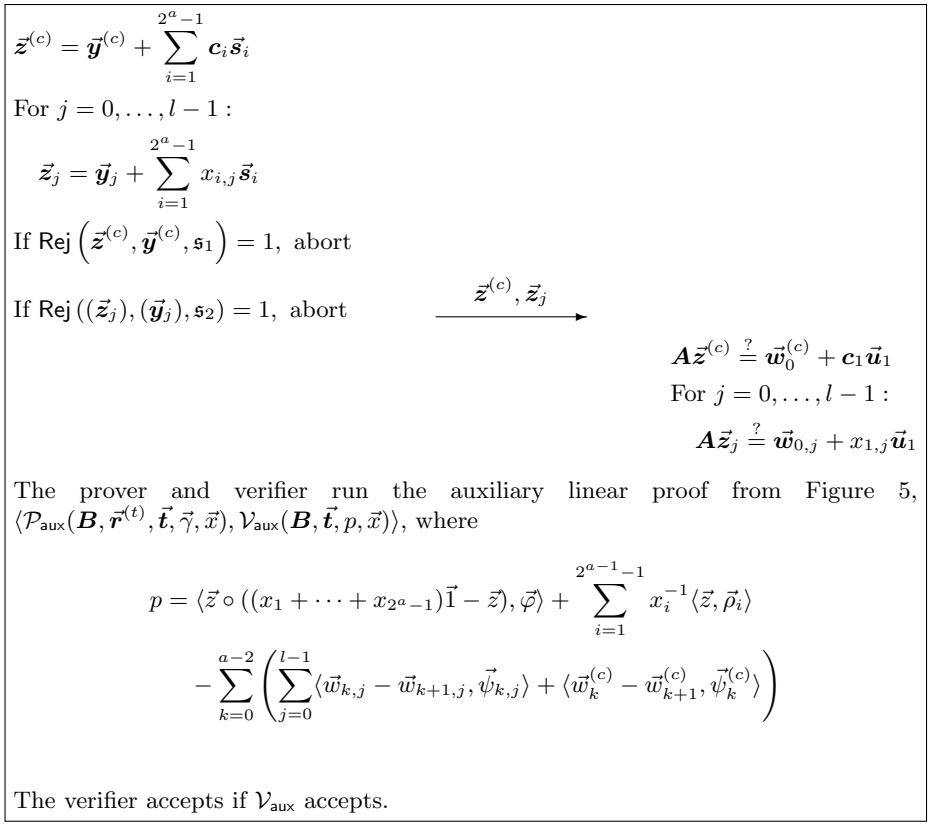
As before we want to use the approximate amortized opening proof with polynomial challenges to bootstrap our protocol in order to benefit from smaller SIS norm bounds. Therefore, the prover also samples an additional masking vector $\vec{\boldsymbol{y}}^{(c)}$ at the beginning of the protocol. Then, in each subproof π_k, he first sends $\vec{\boldsymbol{w}}_k^{(c)} = \boldsymbol{A}\vec{\boldsymbol{y}}^{(c)} + \sum_{i=2^{k+1}}^{2^a-1} \boldsymbol{c}_i \vec{\boldsymbol{u}}_i$, and then receives the next challenge polynomials $\boldsymbol{c}_{2^k}, \ldots, \boldsymbol{c}_{2^{k+1}-1}$. Finally, at the end of the protocol the prover sends $\vec{\boldsymbol{z}}^{(c)} = \vec{\boldsymbol{y}}^{(c)} + \sum_{i=1}^{2^a-1} \boldsymbol{c}_i \vec{\boldsymbol{s}}_i$. The verifier checks that $\vec{\boldsymbol{z}}^{(c)}$ is short and a preimage of $\vec{\boldsymbol{w}}_0^{(c)} + \boldsymbol{c}_1 \vec{\boldsymbol{u}}_1$.

Theorem 2. *The protocol in Figures 6 and 7 is correct, computational honest verifier zero-knowledge under the Module-LWE assumption and computationally knowledge-sound under the Module-SIS assumption. More precisely, let p be the maximum probability of $\boldsymbol{c} \bmod X - \zeta$ as in Lemma 1.*

Prover $\mathcal{P}_{\mathsf{tree}}$		Verifier $\mathcal{V}_{\mathsf{tree}}$
Inputs:		
$\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa_2 \times (\kappa_2+\lambda+\mu)}$		$\boldsymbol{B}_0, \boldsymbol{B}_1, \vec{\boldsymbol{b}}_2$
$\boldsymbol{B}_1 \in \mathcal{R}_q^{(\mu-1) \times (\kappa_2+\lambda+\mu)}$		$\boldsymbol{A}, \vec{\boldsymbol{u}}_1$
$\vec{\boldsymbol{b}}_2 \in \mathcal{R}_q^{\kappa_2+\lambda+\mu}$		
$\boldsymbol{A} \in \mathcal{R}_q^{\kappa \times m/d}$		
For $i = 1, \ldots, 2^a - 1$:		
$\vec{\boldsymbol{s}}_i = \begin{pmatrix} \vec{\boldsymbol{s}}_{i,l} \\ \vec{\boldsymbol{s}}_{i,r} \end{pmatrix} \in \{0,1\}^m \subset \mathcal{R}_q^{m/d}$		
$\vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i$		
$\vec{\boldsymbol{u}}_{2i} = \boldsymbol{G}\vec{\boldsymbol{s}}_{i,l}$		
$\vec{\boldsymbol{u}}_{2i+1} = \boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}$		
$(\vec{\boldsymbol{r}}^{(t)}, \vec{\boldsymbol{t}}_0, \vec{e}) = \mathsf{AUXINIT}(\boldsymbol{B}_0, \boldsymbol{B}_1)$	$\xrightarrow{\vec{\boldsymbol{t}}_0}$	
	$\xleftarrow{\vec{\varphi}}$	$\vec{\varphi} \xleftarrow{\$} \mathbb{Z}_q^m$
$\vec{\boldsymbol{y}}^{(c)} \xleftarrow{\$} D_{\mathfrak{s}_1}^m$		
For $j = 0, \ldots, l-1$:		
$\vec{\boldsymbol{y}}_j \xleftarrow{\$} D_{\mathfrak{s}_2}^m$		
$\vec{y} = \sum_{j=0}^{l-1} \vec{y}_j \delta^j$		
$\gamma_0 = -\langle \vec{y} \circ \vec{y}, \vec{\varphi} \rangle$		
$\tau_0 = e_0 + \gamma_0$	$\xrightarrow{\tau_0}$	

The prover and verifier run the recursive subproof protocol from Figures 8 and 9, $\langle \mathcal{P}_{\mathsf{subprf}}(a-1, \boldsymbol{A}, \vec{\boldsymbol{s}}_i, \vec{\boldsymbol{y}}^{(c)}, \vec{\boldsymbol{y}}_j, \vec{e}, \vec{\varphi}), \mathcal{V}_{\mathsf{subprf}}(a-1) \rangle$. They receive the transcript with elements $\vec{\boldsymbol{w}}_k^{(c)}$, $\vec{\boldsymbol{w}}_{k,j}$, $\vec{\psi}_k^{(c)}$, $\vec{\psi}_{k,j}$, $\boldsymbol{c}_i$, τ_{2i-1}, τ_{2i}, x_i. Then they set $\vec{\tau} = (\tau_0, \ldots, \tau_{2^{a+1}-2})^T$, $\vec{x} = (1, x_1^{-1}, x_1, \ldots, x_{2^a-1}^{-1}, x_{2^a-1})^T$ and $\vec{\boldsymbol{t}}_1 = \mathsf{NTT}^{-1}(\vec{\tau})$, and decompose $x_i = x_{i,0} + \cdots + x_{i,l-1}\delta^{l-1}$ for $i = 1, \ldots, 2^a - 1$. Moreover, the prover keeps the garbage coefficients $\gamma_{2i-1}, \gamma_{2i}$ from $\mathcal{P}_{\mathsf{subprf}}$ and sets $\vec{\gamma} = (\gamma_0, \ldots, \gamma_{2^{a+1}-2})^T$.

Fig. 6. Exact opening proof for lattice-based Merkle tree, part I. The BDLOP message rank μ is given by $\mu = (2^{a+1} - 1)d + 1$



The prover and verifier run the auxiliary linear proof from Figure 5, $\langle \mathcal{P}_{\mathsf{aux}}(\boldsymbol{B}, \vec{r}^{(t)}, \vec{t}, \vec{\gamma}, \vec{x}), \mathcal{V}_{\mathsf{aux}}(\boldsymbol{B}, \vec{t}, p, \vec{x}) \rangle$, where

$$p = \langle \vec{z} \circ ((x_1 + \cdots + x_{2^a-1})\vec{1} - \vec{z}), \vec{\varphi} \rangle + \sum_{i=1}^{2^{a-1}-1} x_i^{-1} \langle \vec{z}, \vec{\rho}_i \rangle - \sum_{k=0}^{a-2} \left(\sum_{j=0}^{l-1} \langle \vec{w}_{k,j} - \vec{w}_{k+1,j}, \vec{\psi}_{k,j} \rangle + \langle \vec{w}_k^{(c)} - \vec{w}_{k+1}^{(c)}, \vec{\psi}_k^{(c)} \rangle \right)$$

The verifier accepts if $\mathcal{V}_{\mathsf{aux}}$ accepts.

Fig. 7. Exact opening proof for lattice-based Merkle tree, part II.

Prover $\mathcal{P}_{\mathsf{subprf}}$		Verifier $\mathcal{V}_{\mathsf{subprf}}$
Inputs:		
$k \geq 0$		k
$\boldsymbol{A} \in \mathcal{R}_q^{\kappa \times m/d}$		
For $i = 1, \ldots, 2^{k+1} - 1$:		
$\quad \vec{\boldsymbol{s}}_i = \begin{pmatrix} \vec{\boldsymbol{s}}_{i,l} \\ \vec{\boldsymbol{s}}_{i,r} \end{pmatrix} \in \{0,1\}^m \subset \mathcal{R}_q^{m/d}$		
$\vec{\boldsymbol{y}}_k^{(c)} \in \mathcal{R}_q^{m/d}$		
For $j = 0, \ldots, l-1$:		
$\quad \vec{\boldsymbol{y}}_{k,j} \in \mathcal{R}_q^{m/d}$		
$\vec{e} \in \mathbb{Z}_q^{2^{a+1}-1}$		
$\vec{\varphi} \in \mathbb{Z}_q^m$		
$\vec{\rho}_{2^{k+1}}, \ldots, \vec{\rho}_{2^a-1}$		
$\boldsymbol{c}_{2^{k+1}}, \ldots, \boldsymbol{c}_{2^a-1} \in \mathcal{C}$		
$x_{2^{k+1}}, \ldots, x_{2^a-1} \in \mathbb{Z}_q$		
$\vec{\boldsymbol{w}}_k^{(c)} = \boldsymbol{A}\vec{\boldsymbol{y}}_k^{(c)}$		
For $j = 0, \ldots, l-1$:		
$\quad \vec{\boldsymbol{w}}_{k,j} = \boldsymbol{A}\vec{\boldsymbol{y}}_{k,j}$	$\xrightarrow{\vec{\boldsymbol{w}}_k^{(c)}, \vec{\boldsymbol{w}}_{k,j}}$	
		$\vec{\psi}_k^{(c)} \xleftarrow{\$} \mathbb{Z}_q^{\kappa d}$
		For $j = 0, \ldots, l-1$:
		$\quad \vec{\psi}_{k,j} \xleftarrow{\$} \mathbb{Z}_q^{\kappa d}$
		For $i = 2^k, \ldots, 2^{k+1} - 1$:
	$\xleftarrow{\vec{\psi}_k^{(c)}, \vec{\psi}_{k,j}, \boldsymbol{c}_i}$	$\quad \boldsymbol{c}_i \xleftarrow{\$} \mathcal{C}$
For $i = 2^k, \ldots, 2^{k+1} - 1$:		
$\quad$ If $k < a - 1$:		
$\quad\quad \vec{\rho}_i = \sum_{j=0}^{l-1} \mathsf{Rot}\begin{pmatrix} x_{2i,j}\boldsymbol{G}^\dagger \\ x_{2i+1,j}\boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_{k,j}$		
$\quad\quad\quad + \mathsf{Rot}\begin{pmatrix} \boldsymbol{c}_{2i}^{\sigma^{-1}}\boldsymbol{G}^\dagger \\ \boldsymbol{c}_{2i+1}^{\sigma^{-1}}\boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_k^{(c)}$		
$\quad$ Else $\vec{\rho}_i = 0$		

Fig. 8. Subprotocol for the Merkle tree proof in Figures 6 and 7, part I.

The prover and verifier run the core protocol from Figure 10,

$$(\tau_{2i-1}, \tau_{2i}, x_i)_{2^k \le i \le 2^{k+1}-1} = \langle \mathcal{P}_{\mathsf{core}}(k, \vec{e}, \vec{y}_k, \vec{\varphi}, (\vec{s}_i)_{2^k \le i \le 2^{k+1}-1}, (\vec{\rho}_i)_{i \ge 2^k}, (x_i)_{i \ge 2^{k+1}}), \mathcal{V}_{\mathsf{core}}(k) \rangle.$$

Then they decompose $x_i = x_{i,0} + \cdots + x_{i,l-1}\delta^{l-1}$ for $i = 2^k, \ldots, 2^{k+1} - 1$.

If $k > 0$, then the prover and verifier recursively run

$$\left\langle \mathcal{P}_{\mathsf{subprf}}\left(k-1, \boldsymbol{A}, (\vec{\boldsymbol{s}}_i)_{i \le 2^k-1}, \vec{\boldsymbol{y}}^{(c)}_{k-1}, \vec{\boldsymbol{y}}_{k-1,j}, \vec{e}, \vec{\varphi}, (\vec{\rho}_i)_{i \ge 2^k}, (x_i)_{i \ge 2^k}, (\boldsymbol{c}_i)_{i \ge 2^k}\right), \mathcal{V}_{\mathsf{subprf}}(k-1) \right\rangle,$$

where $\vec{\boldsymbol{y}}^{(c)}_{k-1} = \vec{\boldsymbol{y}}^{(c)}_k + \sum_{i=2^k}^{2^{k+1}-1} \vec{\boldsymbol{c}}_i \vec{\boldsymbol{s}}_i$ and $\vec{\boldsymbol{y}}_{k-1,j} = \vec{\boldsymbol{y}}_{k,j} + \sum_{i=2^k}^{2^{k+1}-1} x_{i,j} \vec{\boldsymbol{s}}_i$.

Fig. 9. Subprotocol for the Merkle tree proof in Figures 6 and 7, part II.

Then, for correctness, unless the honest prover $\mathcal{P}$ aborts due to the rejection sampling, it convinces the honest verifier $\mathcal{V}$ with overwhelming probability.

For zero-knowledge, there exists an efficient simulator $\mathcal{S}$, that, when having access to all the hashes $\vec{\boldsymbol{u}}_i$ ($i \in \{1, \ldots, n\}$) in the Merkle tree but not the preimages $\vec{\boldsymbol{s}}_{2^{a-1}}, \ldots, \vec{\boldsymbol{s}}_{2^a-1}$ for the leaves, outputs a simulation of a non-aborting transcript of the protocol between $\mathcal{P}$ and $\mathcal{V}$. An adversary who can distinguish the simulation from the real transcript with advantage ε can distinguish $\mathsf{MLWE}_{\lambda,\chi}$ with advantage $\varepsilon - 2^{100}$.

For knowledge-soundness, there is an extractor $\mathcal{E}$ with the following properties. When given resettable black-box access to a deterministic prover $\mathcal{P}^*$ that convinces $\mathcal{V}$ with probability $\varepsilon > 6n/q + \lceil \log n \rceil p$, $\mathcal{E}$ either outputs a binary tree opening $\vec{\boldsymbol{s}}^*_i \in \{0,1\}^m$ ($i \in \{1, \ldots, n\}$) as in Equation (10), an $\mathsf{MSIS}_{\kappa,B}$ solution for $\boldsymbol{A}$ with $B = 4(\omega\beta_2 + \delta\beta_1 + n\omega\delta\sqrt{m})$, or an $\mathsf{MSIS}_{\kappa_2, 8\omega\beta_3}$ solution for $\boldsymbol{B}_0$.

Proof. Correctness. The correctness of the protocol follows from the discussion in this Section and tedious analysis of the verification equations for the messages sent by the honest prover.

Zero-Knowledge. The proof of the zero-knowledge property is similar to the proof of Theorem 1. One transforms the real computation of a transcript in a series of steps until one arrives at a transcript that is indistinguishable from the real transcript and efficiently computable without knowing the leaf preimages $\vec{\boldsymbol{s}}_i$. In every step, when changing the distribution of some of the messages, one updates the messages $\vec{\boldsymbol{w}}^{(c)}_0$, $\vec{\boldsymbol{w}}_{0,j}$ and $\vec{\boldsymbol{w}}^{(t)}$ that are determined by all the other messages and the verification equations in real transcripts. Moreover, one also updates the messages $\vec{\boldsymbol{w}}^{(c)}_k$ and $\vec{\boldsymbol{w}}_{k,k}$ for $k \ge 1$ that are determined by $\vec{\boldsymbol{w}}^{(c)}_0$ and $\vec{\boldsymbol{w}}_{0,j}$, respectively, and the public tree nodes $\vec{\boldsymbol{u}}_i$. First one replaces all the masked openings $\vec{\boldsymbol{z}}^{(c)}$, $\vec{\boldsymbol{z}}_i$ and $\vec{\boldsymbol{z}}^{(t)}$ with independently sampled Gaussian vectors. Then one can swap the commitments $\vec{\boldsymbol{t}}_0$, $\vec{\boldsymbol{t}}_1$, $\boldsymbol{t}_2$ for uniformly random vectors by

Prover $\mathcal{P}_{\mathsf{core}}$		Verifier $\mathcal{V}_{\mathsf{core}}$
Inputs:		
$k \leq 0$		k
$\vec{e} \in \mathbb{Z}_q^{2^{a+1}-1}$		
$\vec{y}_k, \vec{\varphi} \in \mathbb{Z}_q^m$		
$\vec{s}_{2^k}, \ldots, \vec{s}_{2^{k+1}-1} \in \mathbb{Z}_q^m$		
$\vec{\rho}_{2^k}, \ldots, \vec{\rho}_{2^a-1} \in \mathbb{Z}_q^m$		
$x_{2^{k+1}}, \ldots, x_{2^a-1} \in \mathbb{Z}_q$		
$\gamma_{2^{k+2}-2} = \langle \vec{y}_k \circ (\vec{1} - \vec{s}_{2^{k+1}-1}), \vec{\varphi} \rangle$ $+ \langle \vec{s}_{2^{k+1}-1} \circ ((x_{2^{k+1}} + \cdots + x_{2^a-1})\vec{1} - \vec{y}_k), \vec{\varphi} \rangle$ $+ \langle \vec{s}_{2^{k+1}-1}, x_{2^{k+1}}^{-1}\vec{\rho}_{2^{k+1}} + \cdots + x_{2^a-1}^{-1}\vec{\rho}_{2^a-1} \rangle$ $\gamma_{2^{k+2}-3} = \langle \vec{y}_k, \vec{\rho}_{2^{k+1}-1} \rangle$ $\tau_{2^{k+2}-2} = e_{2^{k+2}-2} + \gamma_{2^{k+2}-2}$ $\tau_{2^{k+2}-3} = e_{2^{k+2}-3} + \gamma_{2^{k+2}-3}$	$\xrightarrow{\tau_{2^{k+2}-2},\, \tau_{2^{k+2}-3}}$ $\xleftarrow{x_{2^{k+1}-1}}$	$x_{2^{k+1}-1} \xleftarrow{\$} \mathbb{Z}_q^\times$
$\gamma_{2^{k+2}-4} = \langle \vec{y}_k \circ (\vec{1} - \vec{s}_{2^{k+1}-2}), \vec{\varphi} \rangle$ $+ \langle \vec{s}_{2^{k+1}-2} \circ ((x_{2^{k+1}} + \cdots + x_{2^a-1})\vec{1} - \vec{y}_k), \vec{\varphi} \rangle$ $+ x_{2^{k+1}-1}\langle \vec{s}_{2^{k+1}-1} + \vec{s}_{2^{k+1}-2} - 2\vec{s}_{2^{k+1}-1} \circ \vec{s}_{2^{k+1}-2}, \vec{\varphi} \rangle$ $+ \langle \vec{s}_{2^{k+1}-2}, x_{2^{k+1}-1}^{-1}\rho_{2^{k+1}-1} + \cdots + x_{2^a-1}^{-1}\vec{\rho}_{2^a-1} \rangle$ $\gamma_{2^{k+2}-5} = \langle \vec{y}_k, \vec{\rho}_{2^{k+1}-2} \rangle + x_{2^{k+1}-1}\langle \vec{s}_{2^{k+1}-1}, \vec{\rho}_{2^{k+1}-2} \rangle$ $\tau_{2^{k+2}-4} = e_{2^{k+2}-4} + \gamma_{2^{k+2}-4}$ $\tau_{2^{k+2}-5} = e_{2^{k+2}-5} + \gamma_{2^{k+2}-5}$	$\xrightarrow{\tau_{2^{k+2}-4},\, \tau_{2^{k+2}-5}}$ $\xleftarrow{x_{2^{k+1}-2}}$	$x_{2^{k+1}-2} \xleftarrow{\$} \mathbb{Z}_q^\times$
	$\vdots$	
$\gamma_{2^{k+1}} = \langle \vec{y}_k \circ (\vec{1} - \vec{s}_{2^k}), \vec{\varphi} \rangle$ $+ \langle \vec{s}_{2^k} \circ ((x_{2^{k+1}} + \cdots + x_{2^a})\vec{1} - \vec{y}_k), \vec{\varphi} \rangle$ $+ \sum_{i=2^k+1}^{2^{k+1}-1} x_i \langle \vec{s}_i \circ (\vec{1} - \vec{s}_{2^k}) + \vec{s}_{2^k} \circ (\vec{1} - \vec{s}_i), \vec{\varphi} \rangle$ $+ \sum_{i=2^k+1}^{2^a-1} x_i^{-1} \langle \vec{s}_{2^k}, \vec{\rho}_i \rangle$ $\gamma_{2^{k+1}-1} = \langle \vec{y}_k, \vec{\rho}_{2^k} \rangle + \sum_{i=2^k+1}^{2^{k+1}-1} x_i \langle \vec{s}_i, \rho_{2^k}^{\rightarrow} \rangle$ $\tau_{2^{k+1}} = e_{2^{k+1}} + \gamma_{2^{k+1}}$ $\tau_{2^{k+1}-1} = e_{2^{k+1}-1} + \gamma_{2^{k+1}-1}$	$\xrightarrow{\tau_{2^{k+1}},\, \tau_{2^{k+1}-1}}$ $\xleftarrow{x_{2^k}}$	$x_{2^k} \xleftarrow{\$} \mathbb{Z}_q^\times$

Fig. 10. Core protocol for Merkle tree opening proof in Figures 6 and 7.

relying on the Module-LWE assumption. Finally, it remains to replace $\boldsymbol{h}$ which is uniformly random with $\boldsymbol{h}_0 = 0$ since it contains the additive polynomial $\boldsymbol{g}$ that is used nowhere else anymore.

Knowledge-Soundness. We will repeatedly need to condition on events where subsets of the challenges have fixed values. To this end, we first introduce some notation. Denote the total challenge space of the protocol by $\mathcal{X}$; that is, $\mathcal{X}$ is the product of all the individual challenge spaces. The probability distributions on the individual challenge spaces induce the product distribution on $\mathcal{X}$. There is a probability preserving one-to-one correspondence between challenge tuples $\mathsf{X} \in \mathcal{X}$ and transcripts tr of interactions between $\mathcal{P}^*$ and the honest verifier $\mathcal{V}$ since $\mathcal{P}^*$ is deterministic.

For arguing about the polynomial equations that lie at the heart of the binary and linear subproofs, we need to be able to open the garbage commitments τ_i. The prover sends the top part $\vec{\boldsymbol{t}}_0$ of the BDLOP commitment scheme in his first message. Hence, $\vec{\boldsymbol{t}}_0$ is independent from all the challenges and the same in all interactions with $\mathcal{P}^*$. We can thus extract the auxiliary linear proof at the end of the protocol once, which includes an approximate opening proof for $\vec{\boldsymbol{t}}_0$. We obtain a weak opening $\vec{\boldsymbol{r}}^{(t)*}$ for $\vec{\boldsymbol{t}}_0$ that we can use to open the garbage commitments in any transcript. Concretely, we compute $\vec{e}^* = \mathsf{NTT}(\boldsymbol{B}\vec{\boldsymbol{r}}^{(t)*})$, and then $\gamma_i^* = \tau_i - e_i^*$ as the extraction of the commitment τ_i in some transcript. With the opened commitments we can now write the statement of the auxiliary proof in the clear:

$$\begin{aligned}
&\langle \vec{z} \circ ((x_1 + \cdots + x_{2^a-1})\vec{1} - \vec{z}), \vec{\varphi} \rangle + \sum_{i=1}^{2^a-1} x_i^{-1} \langle \vec{z}, \vec{\rho}_i \rangle \\
&\quad - \sum_{k=0}^{a-2} \left(\sum_{j=0}^{l-1} \langle \vec{w}_{k,j} - \vec{w}_{k+1,j}, \vec{\psi}_{k,j} \rangle + \langle \vec{w}_k^{(c)} - \vec{w}_{k+1}^{(c)}, \vec{\psi}_k^{(c)} \rangle \right) \qquad (13) \\
&= \gamma_0^* + \sum_{i=1}^{2^a-1} (x_i^{-1} \gamma_{2i-1}^* + x_i \gamma_{2i}^*),
\end{aligned}$$

where, for $k = 0, \ldots, a-2$, $i = 2^k, \ldots, 2^{k+1} - 1$,

$$\vec{\rho}_i = \sum_{j=0}^{l-1} \mathsf{Rot} \begin{pmatrix} x_{2i,j} \boldsymbol{G}^\dagger \\ x_{2i+1,j} \boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_{k,j} + \mathsf{Rot} \begin{pmatrix} \boldsymbol{c}_{2i}^{\sigma_{-1}} \boldsymbol{G}^\dagger \\ \boldsymbol{c}_{2i+1}^{\sigma_{-1}} \boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_k^{(c)}.$$

The soundness error of the auxiliary proof is $1/q + p$ (c.f. [ENS20, Theorem 3.1]). This implies that the probability that an interaction between $\mathcal{P}^*$ and $\mathcal{V}$ is accepting and (13) turns out to be false can be at most $1/q + p$. Assume the contrary. Decompose the challenge space $\mathcal{X}$ into two parts, $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_{\mathsf{aux}}$, where $\mathcal{X}_{\mathsf{aux}}$ is the space of the challenges $\theta, \boldsymbol{c}^{(t)}$ for the auxiliary proof. All the quantities in Equation (13) are determined by the challenges from $\mathcal{X}_0$ and hence before the auxiliary proof challenges from $\mathcal{X}_{\mathsf{aux}}$ are sent in the protocol. Consider a fixed $\mathsf{X}_0 \in \mathcal{X}_0$ such that (13) is false. If the success probability of $\mathcal{P}^*$ over the remaining challenges $\mathsf{X}_{\mathsf{aux}} \xleftarrow{\$} \mathcal{X}_{\mathsf{aux}}$ is bigger than $1/q + p$, then, by sending the fixed

values in X_0 for the challenges outside of the auxiliary proof, one can convert $\mathcal{P}^*$ to a prover of a false statement in the auxiliary proof with sufficient success probability so that the extractor for the auxiliary proof can be used. This would either directly give an $\mathsf{MSIS}_{\kappa_2,8\omega\beta_3}$ solution for $\boldsymbol{B}_0$, or different openings $\gamma_i^{**} \neq \gamma_i^*$ to the garbage commitments that in turn immediately result in a Module-SIS solution for $\boldsymbol{B}_0$. Hence, this cannot be and the probability must be at most $1/q+p$. Therefore, also the probability that any interaction is accepting and (13) is false can be at most $1/q + p$.

Now, suppose that for a $k \in \{0, \ldots, a-1\}$, we already know binary vectors $\vec{\boldsymbol{s}}_i^* \in \{0,1\}^m$ as in Equation (10) for all $i = 1, \ldots, 2^k - 1$. This defines the hashes $\vec{\boldsymbol{u}}_2^*, \ldots, \vec{\boldsymbol{u}}_{2^{k+1}-1}^*$ by $\vec{\boldsymbol{u}}_{2i}^* = \boldsymbol{G}\vec{\boldsymbol{s}}_{i,l}^*$ and $\vec{\boldsymbol{u}}_{2i+1}^* = \boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}^*$. Then call an interaction with $\mathcal{P}^*$ k-accepting if it is accepting and such that, for all $0 \leq k' \leq k$,

$$\begin{aligned} \boldsymbol{A}\vec{\boldsymbol{z}}_j &= \vec{\boldsymbol{w}}_{k',j} + \sum_{i=1}^{2^{k'+1}-1} x_{i,j}\vec{\boldsymbol{u}}_i^*, \\ \boldsymbol{A}\vec{\boldsymbol{z}}^{(c)} &= \vec{\boldsymbol{w}}_{k'}^{(c)} + \sum_{i=1}^{2^{k'+1}-1} \boldsymbol{c}_i\vec{\boldsymbol{u}}_i^*. \end{aligned} \tag{14}$$

Clearly, k-accepting implies for all $0 \leq k' < k$,

$$\begin{aligned} &\sum_{j=0}^{l-1} \langle \vec{w}_{k',j} - \vec{w}_{k'+1,j}, \vec{\psi}_{k',j}\rangle + \langle \vec{w}_{k'}^{(c)} - \vec{w}_{k'+1}^{(c)}, \vec{\psi}_{k'}^{(c)}\rangle \\ &= \sum_{j=0}^{l-1} \sum_{i=2^{k'+1}}^{2^{k'+2}-1} \langle x_{i,j}\vec{u}_i^*, \vec{\psi}_{k',j}\rangle + \sum_{i=2^{k'+1}}^{2^{k'+2}-1} \langle \mathsf{Rot}(\boldsymbol{c}_i)\vec{u}_i^*, \vec{\psi}_{k'}^{(c)}\rangle \\ &= \sum_{j=0}^{l-1} \sum_{i=2^{k'}}^{2^{k'+1}-1} \left\langle \mathsf{Rot}(x_{2i,j}\boldsymbol{G}r)\vec{s}_{i,l}^* + \mathsf{Rot}(x_{2i+1,j}\boldsymbol{G})\vec{s}_{i,r}^*, \vec{\psi}_{k',j}\right\rangle \\ &\quad + \sum_{i=2^{k'}}^{2^{k'+1}-1} \left\langle \mathsf{Rot}(\boldsymbol{c}_{2i}\boldsymbol{G})\vec{s}_{i,l}^* + \mathsf{Rot}(\boldsymbol{c}_{2i+1}\boldsymbol{G})\vec{s}_{i,r}^*, \vec{\psi}_{k'}^{(c)}\right\rangle \\ &= \sum_{i=2^{k'}}^{2^{k'+1}-1} \left\langle \vec{s}_i^*, \sum_{j=0}^{l-1} \mathsf{Rot}\begin{pmatrix} x_{2i,j}\boldsymbol{G}^\dagger \\ x_{2i+1,j}\boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_{k',j} + \mathsf{Rot}\begin{pmatrix} \boldsymbol{c}_{2i}^{\sigma_{-1}}\boldsymbol{G}^\dagger \\ \boldsymbol{c}_{2i+1}^{\sigma_{-1}}\boldsymbol{G}^\dagger \end{pmatrix} \vec{\psi}_{k'}^{(c)} \right\rangle \\ &= \sum_{i=2^{k'}}^{2^{k'+1}-1} \langle \vec{s}_i^*, \vec{\rho}_i\rangle. \end{aligned} \tag{15}$$

Next, denote by S_k the statement that we have extracted $\vec{\boldsymbol{s}}_1^*, \ldots, \vec{\boldsymbol{s}}_{2^k-1}^*$ as above and that an interaction is k-accepting with probability $\varepsilon_k \geq \varepsilon - 3(2^{k+1}-2)/q - kp$. Clearly, S_0 is true by the assumption in the theorem. We now show that if S_k is true, then we can extract the next level of the tree and obtain vectors $\vec{\boldsymbol{s}}_{2^k}^*, \ldots, \vec{\boldsymbol{s}}_{2^{k+1}-1}^*$ such that S_{k+1} is true.

Since $\varepsilon_k > p$ and it is easy to decide if the verification equations (14) for $k' = k$ are fulfilled, we can extract weak openings $\vec{\boldsymbol{s}}_i^*$ to the hashes $\vec{\boldsymbol{u}}_i^*$ for $i = 2^k, \ldots, 2^{k+1} - 1$. This works as in the proof of Theorem 1 by extracting the underlying approximate amortized opening proof with polynomial challenges $\boldsymbol{c}_i$. Concretely, we then have $\boldsymbol{A}\vec{\boldsymbol{s}}_i^* = \vec{\boldsymbol{u}}_i$ together with $\bar{\boldsymbol{c}}_{i,j} \in \bar{\mathcal{C}}$ such that $\bar{\boldsymbol{c}}_{i,j} \notin \mathfrak{p}_j$ and $\bar{\boldsymbol{c}}_{i,j}\vec{\boldsymbol{s}}_i^*$ is of length at most $2\beta_1$.

Now, we need to show that the newly extracted $\vec{\boldsymbol{s}}_i^*$ must all be binary and that interactions are $(k+1)$-accepting with probability at least $\varepsilon_k - 3 \cdot 2^{k+1}/q - p$. Towards an indirect proof of the binary property suppose that $\vec{\boldsymbol{s}}_{i_0}^*$ is the last non-binary vector, meaning that $i_0 \in \{2^k, \ldots, 2^{k+1} - 1\}$ is the smallest index such that $\vec{\boldsymbol{s}}_{i_0}^*$ is not binary (recall that the challenges x_i are ordered in reverse order in the protocol). Then, for uniformly random $\vec{\varphi} \in \mathcal{R}_q^m$, the scalar product $\langle \vec{s}_{i_0}^* \circ (\vec{1} - \vec{s}_{i_0}^*), \vec{\varphi} \rangle$ only vanishes with probability $1/q$. Hence, an interaction is k-accepting and the scalar product non-zero with probability at least $\varepsilon_k - 1/q$.

Decompose $\mathcal{X}$ into three parts now; $\mathcal{X} = \mathcal{X}_0 \times \mathbb{Z}_q^{i_0} \times \mathcal{X}_{\mathsf{aux}}$, where $\mathbb{Z}_q^{i_0}$ is the space for the integer challenges $x_1, \ldots, x_{i_0}$, and $\mathcal{X}_{\mathsf{aux}}$ for the auxiliary proof challenges $\theta, \boldsymbol{c}^{(t)}$. Let tr be a transcript with challenge tuple $\mathsf{X} = (\mathsf{X}_0, x_1, \ldots, x_{i_0}, \theta, \boldsymbol{c}^{(t)})$ such that tr is k-accepting and the above scalar product $\langle \vec{s}_{i_0}^* \circ (\vec{1} - \vec{s}_{i_0}^*), \vec{\varphi} \rangle$ non-zero for the $\vec{\varphi}$ in X_0.

For all $j \in \{0, \ldots, l-1\}$, define the masking vectors $\vec{\boldsymbol{y}}_j^* \in \mathcal{R}_q^\kappa$ such that $\vec{\boldsymbol{z}}_j = \vec{\boldsymbol{y}}_j^* + x_{1,j}\vec{\boldsymbol{s}}_1^* + \cdots + x_{i_0,j}\vec{\boldsymbol{s}}_{i_0}^*$ in tr. From the k-acceptance of tr it follows that

$$\boldsymbol{A}\vec{\boldsymbol{y}}_j^* = \vec{\boldsymbol{w}}_{k,j} + \sum_{i=i_0+1}^{2^{k+1}-1} x_{i,j}\vec{\boldsymbol{u}}_i^*.$$

We do not know yet that the $\vec{\boldsymbol{y}}_j^*$ are short. In fact, the $\vec{\boldsymbol{z}}_j$ are short and $\vec{\boldsymbol{s}}_1^*, \ldots, \vec{\boldsymbol{s}}_{i_0-1}^*$ are binary, but $\vec{\boldsymbol{s}}_{i_0}^*$ is not binary (by assumption) and we do not even know whether it is necessarily short. On the other hand, we know that $\vec{\boldsymbol{s}}_{i_0}^*$ is a weak opening for $\vec{\boldsymbol{u}}_{i_0}^*$. That is, for each prime ideal $\mathfrak{p}$ dividing q in $\mathcal{R}$ we have a short polynomial $\bar{\boldsymbol{c}}$ such that $\bar{\boldsymbol{c}} \notin \mathfrak{p}$ and $\bar{\boldsymbol{c}}\vec{\boldsymbol{s}}_{i_0}^*$ is of length at most $2\beta_1$. This allows us to extend the $\vec{\boldsymbol{y}}_j^*$ to weak openings of the above right hand sides. Indeed, $\bar{\boldsymbol{c}}\vec{\boldsymbol{y}}_j^* = \bar{\boldsymbol{c}}\vec{\boldsymbol{z}}_j - x_{i_0,j}\bar{\boldsymbol{c}}\vec{\boldsymbol{s}}_{i_0}^* - \bar{\boldsymbol{c}}\sum_{i=1}^{i_0-1} x_{i,j}\vec{\boldsymbol{s}}_i^*$ is of length at most $2\omega\beta_2 + 2\delta\beta_1 + 2(2^{k+1} - 1)\omega\delta\sqrt{m} \leq B/2$.

Next, the challenges $x_{i_0+1}, \ldots, x_{2^{k+1}-1}$ belong to the first part X_0 of the challenge tuple X. The masking vector commitments $\vec{\boldsymbol{w}}_{k,j}$ only depend on challenges belonging to X_0. Therefore, the right hand sides of the previous displayed equations are completely determined by X_0 before any of the remaining challenges are sent. So the prover is bound to the extracted $\vec{\boldsymbol{y}}^*$ in all interactions where the first challenges are given by X_0. It then follows that the vector $\vec{z} = \vec{z}_0 + \cdots + \vec{z}_{l-1}\delta^{l-1}$ in all such interactions must be given by

$$\vec{z} = \vec{y}^* + \sum_{i=1}^{i_0} x_i\vec{s}_i^*, \tag{16}$$

where $\vec{y}^* = \vec{y}_0^* + \cdots + \vec{y}_{l-1}^*\delta^{l-1}$.

By substituting (15) and (16) into (13) we find

$$
\begin{aligned}
&\Bigg(\langle \vec{y}^* \circ ((x_{i_0+1} + \cdots + x_{2^a-1})\vec{1} - \vec{y}^*), \vec{\varphi}\rangle \\
&\quad + \langle \vec{y}^*, x_{i_0+1}^{-1}\vec{\rho}_{i_0+1} + \cdots + x_{2^a-1}^{-1}\vec{\rho}_{2^a-1}\rangle + \sum_{i=2^k}^{i_0} \langle \vec{s}_i^*, \vec{\rho}_i\rangle\Bigg) \\
&+ \sum_{i=1}^{i_0}\Bigg[x_i^{-1}\Bigg(\langle \vec{y}^*, \vec{\rho}_i\rangle + \sum_{j=i+1}^{i_0} x_j\langle \vec{s}_j^*, \vec{\rho}_i\rangle\Bigg) \\
&\qquad + x_i\Bigg(\langle \vec{y}^* \circ (\vec{1} - \vec{s}_i^*), \vec{\varphi}\rangle + \langle \vec{s}_i^* \circ ((x_{i_0+1} + \cdots + x_{2^a-1})\vec{1} - \vec{y}^*), \vec{\varphi}\rangle \\
&\qquad\quad + \sum_{j=i+1}^{i_0}\big(x_j\langle \vec{s}_j^* \circ (\vec{1} - \vec{s}_i^*), \vec{\varphi}\rangle + x_j\langle \vec{s}_i^* \circ (\vec{1} - \vec{s}_j^*), \vec{\varphi}\rangle\big) \\
&\qquad\quad + \langle \vec{s}_i^*, x_{i_0+1}^{-1}\vec{\rho}_{i_0+1} + \cdots + x_{2^a-1}^{-1}\vec{\rho}_{2^a-1}\rangle \\
&\qquad\quad + \sum_{j=i+1}^{i_0} x_j^{-1}\langle \vec{s}_i^*, \vec{\rho}_j\rangle\Bigg) \\
&\qquad + x_i^2\langle \vec{s}_i^* \circ (\vec{1} - \vec{s}_i^*), \vec{\varphi}\rangle\Bigg] \\
&- \sum_{k'=k}^{a-2}\left(\sum_{j=0}^{l-1}\langle \vec{w}_{k'j} - \vec{w}_{k'+1,j}, \vec{\psi}_{k',j}\rangle + \langle \vec{w}_{k'}^{(c)} - \vec{w}_{k'+1}^{(c)}, \vec{\psi}_{k'}^{(c)}\rangle\right) \\
&= \left(\gamma_0^* + \sum_{i=i_0+1}^{2^a-1} x_i^{-1}\gamma_{2i-1}^* + x_i\gamma_{2i}^*)\right) + \sum_{i=1}^{i_0}(x_i^{-1}\gamma_{2i-1}^* + x_i\gamma_{2i}^*).
\end{aligned}
\tag{17}
$$

In this equation, the following quantities are determined before any of the challenges $x_1, \ldots, x_{i_0}$ are sent in the protocol and hence they only depend on challenges in X_0: The extracted vectors $\vec{y}^*$ and $\vec{s}_i^*$ for all i; the challenges $\vec{\varphi}$ and $\vec{\psi}_{k',j}, \vec{\psi}_{k'}^{(c)}$ for $k' = k, \ldots, a-2$; ρ_i for $i = 2^k, \ldots, 2^a - 1$; the extracted commitment messages γ_0^* and $\gamma_{2i-1}^*, \gamma_{2i}^*$ for $i = i_0 + 1, \ldots, 2^a - 1$; and the masking

vector commitments $\vec{w}_{k',j}, \vec{w}_{k'}^{(c)}$ for $k' = k, \ldots, a-1$. Therefore,

$$\begin{aligned} f_0 =& \langle \vec{y}^* \circ ((x_{i_0+1} + \cdots + x_{2^a-1})\vec{1} - \vec{y}^*), \vec{\varphi} \rangle \\ &+ \langle \vec{y}^*, x_{i_0+1}^{-1}\vec{\rho}_{i_0+1} + \cdots + x_{2^a-1}^{-1}\vec{\rho}_{2^a-1} \rangle + \sum_{i=2^k}^{i_0} \langle \vec{s}_i^*, \vec{\rho}_i \rangle \\ &- \sum_{k'=k}^{a-2} \left(\sum_{j=0}^{l-1} \langle \vec{w}_{k',j} - \vec{w}_{k'+1,j}, \vec{\psi}_{k',j} \rangle + \langle \vec{w}_{k'}^{(c)} - \vec{w}_{k'+1}^{(c)}, \vec{\psi}_{k'}^{(c)} \rangle \right) \\ &- \left(\gamma_0^* + \sum_{i=i_0+1}^{2^a-1} (x_i^{-1}\gamma_{2i-1}^* + x_i\gamma_{2i}^*) \right) \end{aligned}$$

is independent from all the challenges $x_1, \ldots, x_{i_0}$. Then,

$$\begin{aligned} f_{i,-1}^{(x_{i+1},\ldots,x_{i_0})} &= \langle \vec{y}^*, \vec{\rho}_i \rangle + \sum_{j=i+1}^{i_0} x_j \langle \vec{s}_j^*, \vec{\rho}_i \rangle - \gamma_{2i-1}^* \\ f_{i,1}^{(x_{i+1},\ldots,x_{i_0})} &= \langle \vec{y}^* \circ (\vec{1} - \vec{\boldsymbol{s}}_i^*), \vec{\varphi} \rangle \\ &\quad + \langle \vec{s}_i^* \circ ((x_{i_0+1} + \cdots + x_{2^a-1})\vec{1} - \vec{y}^*), \vec{\varphi} \rangle \\ &\quad + \langle \vec{s}_i^*, x_{i_0+1}^{-1}\vec{\rho}_{i_0+1} + \cdots + x_{2^a-1}^{-1}\vec{\rho}_{2^a-1} \rangle \\ &\quad + \sum_{j=i+1}^{i_0} \left(x_j \langle \vec{s}_j^* \circ (\vec{1} - \vec{s}_i^*), \vec{\varphi} \rangle + x_j \langle \vec{s}_i^* \circ (\vec{1} - \vec{s}_j^*), \vec{\varphi} \rangle + x_j^{-1} \langle \vec{s}_i^*, \vec{\rho}_j \rangle \right) \\ &\quad - \gamma_{2i}^* \end{aligned}$$

are functions of $x_{i+1}, \ldots, x_{i_0}$ but independent from $x_1, \ldots, x_i$. Using these functions we can now write Equation 17 as

$$f_0 + \sum_{i=1}^{i_0} \left(f_{i,-1}^{(x_{i+1},\ldots,x_{i_0})} x_i^{-1} + f_{i,1}^{(x_{i+1},\ldots,x_{i_0})} x_i \right) + \langle \vec{s}_{i_0}^* \circ (\vec{1} - \vec{s}_{i_0}^*), \vec{\varphi} \rangle x_{i_0}^2 = 0.$$

Now, we bound the probability over $(x_1, \ldots, x_{i_0}, \theta, \boldsymbol{c}^{(t)}) \xleftarrow{\$} \mathbb{Z}_q^{i_0} \times \mathcal{X}_{\mathsf{aux}}$ that an interaction is k-accepting when the first challenges are given by X_0. The previous displayed equation is of the form of Lemma 2 (with reverse ordering of the variables). It then follows from Lemma 2 that (17) is only true with probability at most $3(2^{k+1}-1)/q$. If Equation (17) is false, then the prover must cheat in the auxiliary proof in order to let the interaction be k-accepting. We have explained at the beginning of the proof that the prover can only achieve this with probability at most $1/q + p$. Therefore, we see that the k-accepting success probability is bounded by $3(2^{k+1}-1)/q + (1/q + p)$ when the first challenges are X_0. But since X_0 is from an arbitrary transcript tr with probability $\varepsilon_k - 1/q$, it follows that $\varepsilon_k < 3 \cdot 2^{k+1}/q + p$, which is a contradiction to the assumption $\varepsilon_k > \varepsilon - 3(2^{k+1}-2)/q - kp > 6n/q + \lceil \log n \rceil p - 3(2^{k+1}-2)/q - kp > 3 \cdot 2^{k+1}/q - p$. Hence, all the extracted $\vec{\boldsymbol{s}}_i^*$ must be binary.

Finally, we bound the probability that an interaction is k-accepting but not $(k+1)$-accepting when $k \leq a-2$. In an interaction that is k-accepting but not $(k+1)$-accepting,

$$\begin{aligned}\vec{\boldsymbol{w}}_{k,j} &\neq \vec{\boldsymbol{w}}_{k+1,j} + \sum_{i=2^{k+1}}^{2^{k+2}-1} x_{i,j}\vec{\boldsymbol{u}}_i^* \\ &= \vec{\boldsymbol{w}}_{k+1,j} + \sum_{i=2^k}^{2^{k+1}-1} (x_{2i,j}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,l}^* + x_{2i+1,j}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}^*)\end{aligned}$$

for some $j \in \{0,\ldots,l-1\}$, or

$$\begin{aligned}\vec{\boldsymbol{w}}_k^{(c)} &\neq \vec{\boldsymbol{w}}_{k+1}^{(c)} + \sum_{i=2^{k+1}}^{2^{k+2}-1} \boldsymbol{c}_i\vec{\boldsymbol{u}}_i^* \\ &= \vec{\boldsymbol{w}}_{k+1}^{(c)} + \sum_{i=2^k}^{2^{k+1}-1} (\boldsymbol{c}_{2i}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,l}^* + \boldsymbol{c}_{2i+1}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}^*).\end{aligned}$$

Decompose $\mathcal{X}$ again differently as $\mathcal{X} = \mathcal{X}_0 \times (\mathbb{Z}_q^{\kappa d})^{l+1} \times \mathbb{Z}_q^{2^{k+1}-1} \times \mathcal{X}_{\mathsf{aux}}$, where $(\mathbb{Z}_q^{\kappa d})^{l+1}$ is the space for the challenges $\vec{\psi}_{k,j}$ and $\vec{\psi}_k^{(c)}$, and $\mathbb{Z}_q^{i_0}$, $\mathcal{X}_{\mathsf{aux}}$ are as before with $i_0 = 2^{k+1}-1$. Let tr be an arbitrary k-accepting but not $(k+1)$-accepting transcript with challenge tuple $(\mathsf{X}_0, \vec{\psi}_{k,j}, \vec{\psi}_k^{(c)}, x_1, \ldots, x_{2^{k+1}-1}, \theta, \boldsymbol{c}^{(t)})$. As before, the right hand sides for the masking vectors $\vec{\boldsymbol{y}}_j^*$ are completely determined by X_0 and the prover is bound to these masking vectors in any interaction with first challenges X_0. We can also still write the equation proven by the auxiliary proof as

$$f_0 + \sum_{i=1}^{2^{k+1}-1} \left(f_{i,-1}^{(x_{i+1},\ldots,x_{2^{k+1}-1})} x_i^{-1} + f_{i,1}^{(x_{i+1},\ldots,x_{2^{k+1}-1})} x_i \right) = 0$$

with the functions f_i from above for $i_0 = 2^{k+1}-1$. But now f_0 contains the term

$$\sum_{i=2^k}^{2^{k+1}-1} \langle \vec{\boldsymbol{s}}_i^*, \vec{\rho}_i \rangle - \left(\sum_{j=0}^{l-1} \langle \vec{w}_{k+1,j} - \vec{w}_{k,j}, \vec{\psi}_{k,j} \rangle + \langle \vec{w}_{k+1}^{(c)} - \vec{w}_k^{(c)}, \vec{\psi}_k^{(c)} \rangle \right)$$

that vanishes only with probability $1/q$; c.f. Equation (15). Moreover, all the other terms in f_0 are determined by X_0 and so f_0 is non-zero with probability $(1-1/q)$. In this case we can follow the same chain of arguments from above. Lemma 2 says that the equation for the auxiliary proof is only true with probability at most $3(2^{k+1}-1)/q$, and if it is not then the prover can only win with probability at most $1/q + p$. Therefore, we conclude that the probability that an arbitrary interaction is k-accepting but not $(k+1)$-accepting is at most

$3 \cdot 2^{k+1}/q + p$. It follows that

$$\begin{aligned}\epsilon_k &= \Pr[k\text{-acc}] \\ &= \Pr[k\text{-acc} \wedge \neg(k+1)\text{-acc}] + \Pr[k\text{-acc} \wedge (k+1)\text{-acc}] \\ &= \Pr[k\text{-acc} \wedge \neg(k+1)\text{-acc}] + \Pr[(k+1)\text{-acc}] \\ &< 3 \cdot 2^{k+1}/q + p + \varepsilon_{k+1}.\end{aligned}$$

Thus, an interaction is $(k+1)$-accepting with probability $\varepsilon_{k+1} > \varepsilon_k - 3 \cdot 2^{k+1}/q - p > \varepsilon - 3(2^{k+2} - 2)/q - (k+1)p$. □

5.1 Proof Size

As before we analyze the proof size for the more general case of secret coefficients in the interval $\{-\lfloor b/2 \rfloor, \ldots, \lfloor (b-1)/2 \rfloor\}$ for a base $b \geq 2$. A proof from the non-interactive variant of the protocol in this section consists of the following elements:

- $\vec{\boldsymbol{t}}_0$, $\vec{\boldsymbol{t}}_1$, $\boldsymbol{t}_2$, $\boldsymbol{h}$,
- $\vec{\boldsymbol{w}}_k^{(c)}$, $\vec{\boldsymbol{w}}_{k,j}$ for $k = 1, \ldots, a-1$,
- $\vec{\psi}_{0,j}$, $\vec{\psi}_0^{(c)}$, $\boldsymbol{c}_1$, x_1, θ, $\boldsymbol{c}^{(t)}$,
- $\vec{\boldsymbol{z}}^{(c)}$, $\vec{\boldsymbol{z}}_j$, $\vec{\boldsymbol{z}}^{(t)}$.

Only the masking vector commitments $\vec{\boldsymbol{w}}_{0,j}$ and $\vec{\boldsymbol{w}}_0^{(c)}$ can be omitted from the proof since the verification equations for all the other vectors are only proven in zero-knowledge and hence can not be used by the verifier to compute the commitments. The challenges $\vec{\psi}_{0,j}$, $\vec{\psi}_0^{(c)}$ and $\boldsymbol{c}_1$ are sent together in the interactive protocol and hence they require only one seed. The combined size of all the elements amounts to

$$\begin{aligned}S = {} & (\kappa_2 + \mu + 1)d\lceil \log q \rceil \\ & + (l+1)(\lceil \log n \rceil - 1)\kappa d \log q \\ & + 4 \cdot 128 \\ & + m \log(12\mathfrak{s}_1) + lm \log(12\mathfrak{s}_2) + (\kappa_2 + \lambda + \mu)d \log(12\mathfrak{s}_3)\end{aligned}$$

bits, where $\mu = \left\lceil \frac{bn+1}{d} \right\rceil + 1$. It remains to compute the standard deviations $\mathfrak{s}_i$. They are precisely as discussed in Section 4.2.

Examples We use the following simple approximation to the proof size to guide our choice of parameters:

$$S \approx m \left(\log q + \frac{l+1}{2} \log(b^2 mn) \right) + nb \log q + \kappa(l+1)d \log n \log q.$$

The hash input length m needs to be such that $m \log b \geq 2\kappa d \log q$ so that two hash outputs can fit, and it is has to be a multiple of d since the hash function

is defined as an $\mathcal{R}_q$-module homomorphism. So we take

$$m = \left\lceil \frac{2\kappa \log q}{\log b} \right\rceil d.$$

For $q \approx 2^{128}$ and the MSIS bound B that appears in our protocol the hash output length κd needs to be at least 2^{10} and hence we find that our protocol requires a fairly large m between about 2^{16} and 2^{20}. Therefore, from the above approximation of the proof size we see that the proof size is governed by the term $m \log q$, which will always be bigger than 2^{23}. So, the proof size will be bigger than one Megabyte. The number of secret coefficients that are hashed in the leaves of the tree is equal to $m(n+1)/2$ and the witness size given by $m\frac{(n+1)}{2} \log b$ bits. An optimal proof size with respect to the witness size is reached when $bn \approx m$ and the terms $m \log q$ and $nb \log q$ in the approximate proof size are roughly balanced. At the same time we see that it is advantageous to choose a small number l of masked opening shares so that the remaining terms in the proof size do not contribute much. It then follows that in first approximation the ratio of the proof size and the square root of the witness size is given by $2nb \log q / \sqrt{n^2 b \log b/2} = 2\sqrt{2b/\log b} \log q$. In other words, asymptotically and for $b = 2$ it holds that $S = (512 + o(1))\sqrt{S_w}$ where S_w is the size of the witness.

We present a selection of parameters in Table 2 that span a wide range of witness sizes. In all parameter sets the prime modulus is in the order of 2^{128} and the rank of the ring $\mathcal{R}$ is $d = 128$. We have always chosen the hash output rank κ, the BDLOP MSIS rank κ_2, and the BDLOP MLWE rank λ such that the corresponding problems $\mathsf{MSIS}_{\kappa,B}$, $\mathsf{MSIS}_{\kappa_2,8d\beta_3}$, and $\mathsf{MLWE}_{\lambda,\chi}$, respectively, have classical Core-SVP hardness of about 100 bits with respect to the BDLG16 sieve [BDGL16]. As usual this only includes the cost for one call to the SVP solver and ignores many additional costs, which are particularly large in our case because of the relatively large lattice dimensions. Moreover, it does not take into account the enormous memory requirements from the sieving algorithm. We are thus confident that the actual computational hardness is sufficient for 128 bits of security. For the BDLOP ranks $\kappa_2 = 3$ and $\lambda = 40$ are sufficient for all the parameter sets. The required κ mainly depends on l. The BDLOP MSIS rank can be much smaller than κ because only a polynomial challenge is used in the masked opening $\vec{z}^{(t)}$ and hence the MSIS norm $8d\beta_3$ is much smaller than B that is determined by the lengths of the masked openings $\vec{z}_j$ with large integer challenges of up to 64 bits.

6 Application to R1CS

The proof system from Section 5 allows to prove large commitment openings with acceptable costs in practice. The system therefore becomes interesting for the application of proving arithmetic circuits over $\mathbb{Z}_q$. This means, given a circuit with addition and multiplication gates over $\mathbb{Z}_q$, we want to be able to prove knowledge of an input vector that leads to a known output. We use rank-one

κ	m	n	l	b	B	Witness Size	Proof Size
16	$2^{16.60}$	$2^{10}-1$	3	39	$2^{80.45}$	32.00 MB	4.53 MB
16	$2^{16.79}$	$2^{11}-1$	3	25	$2^{80.48}$	64.07 MB	5.22 MB
16	$2^{16.97}$	$2^{12}-1$	3	27	$2^{80.58}$	128.12 MB	6.08 MB
16	$2^{17.21}$	$2^{13}-1$	3	11	$2^{80.65}$	256.21 MB	7.19 MB
25	$2^{17.98}$	$2^{14}-1$	2	9	$2^{102.66}$	800.01 MB	10.79 MB
25	$2^{18.15}$	$2^{15}-1$	2	7	$2^{102.93}$	1600.19 MB	13.21 MB
25	$2^{18.43}$	$2^{16}-1$	2	5	$2^{103.13}$	3200.78 MB	16.59 MB
25	$2^{18.64}$	$2^{17}-1$	2	4	$2^{103.86}$	6400.00 MB	21.68 MB
25	$2^{18.98}$	$2^{18}-1$	2	3	$2^{103.71}$	12800.16 MB	29.04 MB
26	$2^{19.64}$	$2^{19}-1$	2	2	$2^{104.94}$	26624.00 MB	42.42 MB

Table 2. Example parameter choices for our exact Merkle tree proof system. All examples use prime modulus $q \approx 2^{128}$, ring rank $d = 128$, BDLOP MSIS rank $\kappa_2 = 3$, and BDLOP MLWE rank $\lambda = 40$. The witness size is the size of the preimages to the leaves of the tree.

constraint satisfaction (R1CS) as a powerful intermediate representation. On one hand, R1CS is straight-forward to prove with our proof system, and, on the other hand, it easily represents arbitrary circuits. Moreover, there are advanced toolchains available that allow to compile (subsets of) standard programming languages into instances of R1CS [BCG+13]. An instance of R1CS is given by three matrices $A, B, C \in \mathbb{Z}_q^{k \times k}$ and a solution vector $\vec{s} \in \mathbb{Z}_q^k$ such that

$$(A\vec{s}) \circ (B\vec{s}) = C\vec{s}. \tag{18}$$

In order to prove knowledge of $\vec{s}$ one commits to the four vectors $\vec{s}$, $\vec{x} = A\vec{s}$, $\vec{y} = B\vec{s}$ and $\vec{z} = C\vec{s}$. Then one proves that $\vec{x}, \vec{y}, \vec{z}$ are indeed the images of $\vec{s}$ with the linear proof technique and that $\vec{x} \circ \vec{y} = \vec{z}$ with the product proof. This is easy to do in our proof system without additional cost as described in Section 4.1.

For this application the leaf hashes $\vec{u}_{2^{a-1}}, \ldots, \vec{u}_{2^a-1}$ have to be hiding commitments. We make them hiding under the Module-LWE assumption by letting the first λ polynomials in the preimage vectors $\vec{s}_{2^{a-1}+j}$ be uniformly random vectors $\vec{r}_j$ in $\{-\lfloor b/2 \rfloor, \ldots, \lfloor (b-1)/2 \rfloor\}^{\lambda d}$, where λ is chosen such that Module-LWE for the given uniform error distribution and module rank λ is hard.

We can only commit to short vectors and thus expand $\vec{s}$, $\vec{x}$, $\vec{y}$, $\vec{z}$ in base b. Then recall that the number of secret coefficients of our proof system is given by $m(n+1)/2$. It follows that for given m and n we can prove an R1CS instance over $\mathbb{Z}_q$ with k constraints if

$$4 \left\lceil \frac{k \lceil \log_b(q) \rceil}{(n+1)/2} \right\rceil \leq m - \lambda d.$$

Then, let $R \in \mathbb{Z}_q^{\lambda d \times (n+1)/2}$ be the matrix with columns $\vec{r}_0, \ldots, \vec{r}_{2^{a-1}-1}$. Let $\vec{s}' \in \mathbb{Z}_q^{k\lceil \log_b(q) \rceil}$ be the base-b expansion of $\vec{s}$, split $\vec{s}'$ into vectors of length $(n+1)/2$, $\vec{s}' = \vec{s}'_1 \parallel \cdots \parallel \vec{s}'_{(m-\lambda d)/4}$ and let $S \in \mathbb{Z}_q^{(m-\lambda d)/4 \times (n+1)/2}$ be the matrix with

columns $\vec{s}'_j$. Do the same for $\vec{x}$, $\vec{y}$ and $\vec{z}$. Now the coefficient vectors of the $\vec{s}_{2^{a-1}+j}$ are given by the columns of the matrix

$$\left(\vec{s}_{2^{a-1}} \cdots \vec{s}_{2^a-1}\right) = \begin{pmatrix} R \\ S \\ X \\ Y \\ Z \end{pmatrix}.$$

Proving the linear relations for $\vec{x}, \vec{y}, \vec{z}$ is straight forward. For proving the product relation $\vec{x} \circ \vec{y} = \vec{z}$ note that the masked openings $\vec{z}_i$ include masked openings to the expanded $\vec{x}', \vec{y}', \vec{z}'$. The verifier can use them to reconstruct masked openings for $\vec{x}, \vec{y}, \vec{z}$. Then he can construct the quadratic polynomial that contains the coefficient relations in the terms $x^2_{2^{a-1}+j}$ as in Section 4.1. By multiplying the polynomial by $(x_{2^{a-1}} + \cdots + x_{2^a-1})^{b-2}$ we can increase the degree so that we can combine this with the polynomial for the shortness proof of degree b and do not need additional garbage commitments.

Example. As an example consider the parameter set in the third-to-last row of Table 2 with $b = 4$. Here $\lambda = 38$ is sufficient so that the leaf hashes are hiding with MLWE Core-SVP hardness more than 100 bits. We then find that these parameters allow to prove an R1CS instance with $2^{26.62}$ constraints.

6.1 Reducing the Number of Commitments

We observe that Equation 18 can proven by committing to only one more vector in $\mathbb{Z}_q^k$. The approach is similar as in [ENS20]. Namely, the prover $\mathcal{P}$ sends a commitment to $\vec{s}$ and obtains a challenge vector $\vec{\gamma} = (\gamma_1, \ldots, \gamma_k) \xleftarrow{\$} \mathbb{Z}_q^k$ from the verifier. Next, $\mathcal{P}$ proves that

$$\langle (A\vec{s}) \circ (B\vec{s}) - C\vec{s}, \vec{\gamma} \rangle = 0.$$

The key idea is that the inner product can be equivalently written as

$$\begin{aligned} \langle (A\vec{s}) \circ (B\vec{s}) - C\vec{s}, \vec{\gamma} \rangle &= \langle (A\vec{s}) \circ (B\vec{s}), \vec{\gamma} \rangle - \langle C\vec{s}, \vec{\gamma} \rangle \\ &= \langle A\vec{s}, \vec{\gamma} \circ (B\vec{s}) \rangle - \langle C\vec{s}, \vec{\gamma} \rangle \\ &= \langle A\vec{s}, \Gamma B\vec{s} \rangle - \langle C\vec{s}, \vec{\gamma} \rangle \\ &= \langle \vec{s}, A^T \Gamma B\vec{s} \rangle - \langle \vec{s}, C^T \vec{\gamma} \rangle \end{aligned}$$

where matrix Γ is defined as

$$\Gamma = \begin{pmatrix} \gamma_1 & 0 & \cdots & 0 \\ 0 & \gamma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_k \end{pmatrix} \in \mathbb{Z}_q^{k \times k}.$$

Thus, the prover commits to $\vec{x} := A^T \Gamma B \vec{s} \in \mathbb{Z}_q^k$ and proves that (i) $\vec{x}$ was correctly formed (linear proof) and more importantly (ii) $\langle \vec{s}, \vec{x} \rangle - \langle \vec{s}, C^T \vec{\gamma} \rangle = 0$. We believe that with current techniques (i.e. similarly as above and in Section 4.1), one can prove (ii) by committing to small number of additional garbage terms but we leave the concrete analysis for future work. This approach comes with an negligible increase in the soundness error by an additional factor of $1/q$.

In summary, we would reduce the number of committed messages from $4k$ to $2k$ elements in $\mathbb{Z}_q$. Hence, for given m and n, we can prove an R1CS instance over $\mathbb{Z}_q$ with k constrains if

$$2 \left\lceil \frac{k\lceil \log_b(q) \rceil}{(n+1)/2} \right\rceil \leq m - \lambda d.$$

This implies that we can prove an R1CS instance with essentially two times more constrains than before.

6.2 Comparison to Ligero

We compare the performance of our proof system for R1CS in terms of proof size to the Ligero system [AHIV17]. We picked Ligero over for example Ligero++ [BFH+20] or Aurora [BCR+19] since Ligero also exhibits square root sized proofs, whereas the other two scale (poly-)logarithmically. Our lattice-based proof system does not yet achieve the proof sizes of the best non-lattice systems. But there has been tremendeous improvements in the last few years and the comparison with Ligero serves to show that lattice-based systems are not very far behind anymore. Next to the proof size, the running time and memory requirements of the prover algorithm are very important characteristics. Here Ligero has an advantage over Aurora as it is more than 10 times faster. We expect our proof system to be very competitive in terms of running time due to the underlying fast arithmetic. Lattice cryptography is generally known to be very fast — often an order of magnitue faster than classical or hash-based cryptoraphy [LS19].

Furthermore, the enormous memory requirements of the PCP-type systems like Ligero or Aurora render them unpractical for the numbers of constraints that are important in practice. For example Boschini et. al. [BCOS20] could not run the signing algorithm of their otherwise very compact quantum-safe group signature scheme that they have designed using Aurora as a building-block because of the memory requirements. And they were even using large-memory instances from the Google Cloud for their experiments. This is especially problematic since for privacy-preserving protocols to be used in practice, the prover would often need to be run on constrained devices, possibly down to smart cards or TPM chips. In our experiments we found that Aurora needs more than 67 Gigabytes of RAM and a proof computation time of around 80 minutes for 2^{22} constraints on an AMD ZEN CPU core. Ligero can be used for up to 2^{25} constraints with 67 Gigabytes of RAM. We expect our system to be very attractive with respect to memory requirements as well. But we leave a

demonstration of these performance characteristics and an implementation to future work.

Table 1 contains a comparison of our proof system for R1CS to Ligero. We chose a range of constraints above 2^{20} as our proof system is most effective for such large numbers of constraints. The proof sizes for Ligero were directly measured by running the implementation from `https://github.com/scipr-lab/libiop`. For both proof systems we used a field size of about 128 bits and comparable soundness errors.

References

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed $\varsigma $-protocol theory for lattices. In *CRYPTO (2)*, volume 12826 of *Lecture Notes in Computer Science*, pages 549–579. Springer, 2021.

AF21. Thomas Attema and Serge Fehr. Parallel repetition of $(\mathrm{k}_1, đots, \mathrm{k}_\mu)$-special-sound multi-round interactive proofs. *IACR Cryptol. ePrint Arch.*, page 1259, 2021.

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

AL21. Martin R. Albrecht and Russell W. F. Lai. Subtractive sets over cyclotomic rings - limits of schnorr-like arguments over lattices. In *CRYPTO (2)*, volume 12826 of *Lecture Notes in Computer Science*, pages 519–548. Springer, 2021.

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.

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

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society, 2018.

BBC+18. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 669–699. Springer, 2018.

BCG+13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

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

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

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, pages 10–24. SIAM, 2016.

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385. Springer, 2018.

BFH+20. Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In *CCS*, pages 2025–2038. ACM, 2020.

BLNS20. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-pcp approach to succinct quantum-safe zero-knowledge. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 441–469. Springer, 2020.

BMRS21. Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *CRYPTO (4)*, volume 12828 of *Lecture Notes in Computer Science*, pages 92–122. Springer, 2021.

Din12. Jintai Ding. New cryptographic constructions using generalized learning with errors problem. *IACR Cryptol. ePrint Arch.*, page 387, 2012.

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.

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 259–288. Springer, 2020.

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

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

EZS+19. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *CCS*, pages 567–584. ACM, 2019.

FHK+18. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST's post-quantum cryptography standardization process*, 36, 2018.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, 2013.

GMNO18. Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-snarks from square span programs. In *CCS*, pages 556–573. ACM, 2018.

Gro11. Jens Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 2011.

ISW21. Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zksnarks from lattices. In *CCS*, pages 212–234. ACM, 2021.

LFKN92. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

LM06. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2006.

LNPS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, Maxime Plançon, and Gregor Seiler. Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In *ASIACRYPT (4)*, volume 13093 of *Lecture Notes in Computer Science*, pages 218–248. Springer, 2021.

LNS20. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *CCS*, pages 1051–1070. ACM, 2020.

LNS21a. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *Public Key Cryptography (1)*, volume 12710 of *Lecture Notes in Computer Science*, pages 215–241. Springer, 2021.

LNS21b. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions. In *CRYPTO (2)*, volume 12826 of *Lecture Notes in Computer Science*, pages 611–640. Springer, 2021.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

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

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

Lyu08. Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2008.

Lyu09. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.

Mic02. Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *FOCS*, pages 356–365. IEEE Computer Society, 2002.

Ore22. Øystein Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer, 2006.
Sch80. Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
Sha92. Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.
WYKW21. Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE, 2021.
YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *CCS*, pages 2986–3001. ACM, 2021.
Zip79. Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

A Additional Background

A.1 Error Distribution, Discrete Gaussians and Rejection Sampling

For sampling randomness in the commitment scheme that we use, and to define a particular variant of the Module-LWE problem, we need to specify the error distribution χ on $\mathbb{Z}_q$. We define χ as the computationally simple centered binomial distribution on $\{-1, 0, 1\}$ where ± 1 both have probability $5/16$ and 0 has probability $6/16$. This distribution is chosen (rather than the more natural uniform one) because it is easy to sample given a random bit-string by computing $a_1 + a_2 - b_1 - b_2 \bmod 3$ with uniformly random bits a_i, b_i.

Rejection Sampling In our zero-knowledge proofs, the prover wants 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, $\boldsymbol{c} \in \{-1, 0, 1\}^d$ is a ternary challenge polynomial, and $\vec{\boldsymbol{y}}$ is a "masking vector". To remove the dependency of $\vec{\boldsymbol{z}}$ on $\vec{\boldsymbol{r}}$, we use the rejection sampling technique by Lyubashevsky [Lyu08, Lyu09, Lyu12]. In the two variants of this technique the masking vector is either sampled uniformly from some bounded region or using a discrete Gaussian distribution. We focus on the latter approach.

We first define the discrete Gaussian distribution and then state the rejection sampling algorithm in Figure 11, which plays a central role in Lemma 3.

Definition 3. *The* discrete Gaussian distribution *on* $\mathbb{Z}^n$ *centered around* $\vec{v} \in \mathbb{Z}^n$ *with standard deviation* $\mathfrak{s} > 0$ *is given by*

$$D^n_{\vec{v},\mathfrak{s}}(\vec{z}) = \frac{e^{-\|\vec{z}-\vec{v}\|_2^2/2\mathfrak{s}^2}}{\sum_{\vec{z}' \in \mathbb{Z}^n} e^{-\|\vec{z}'\|_2^2/2\mathfrak{s}^2}}.$$

When it is centered around $\vec{0} \in \mathbb{Z}^n$ *we write* $D^n_{\mathfrak{s}} = D^n_{\vec{0},\mathfrak{s}}$

Lemma 3 (Rejection Sampling). *Let* $V \subseteq \mathbb{Z}^n$ *be a set of vectors with norm at most* T *and* $\rho\colon V \to [0,1]$ *be a probability distribution. Also, write* $\mathfrak{s} = 5T$ *and* $M = \exp(12/5 + 1/50) \approx 11.25$*. Now, sample* $\vec{v} \xleftarrow{\$} \rho$ *and* $\vec{y} \xleftarrow{\$} D^n_{\mathfrak{s}}$*, set* $\vec{z} = \vec{y} + \vec{v}$*, and run* $b \leftarrow \mathsf{Rej}(\vec{z}, \vec{v}, \mathfrak{s})$*. Then, the probability that* $b = 0$ *is at least* $(1 - 2^{-100})/M$ *and the distribution of* $(\vec{v}, \vec{z})$*, conditioned on* $b = 0$*, is within statistical distance of* $2^{-100}/M$ *of the product distribution* $\rho \times D^n_{\mathfrak{s}}$*.*

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

Fig. 11. Rejection sampling algorithm from [Lyu12].

We will also use the following tail bound, which follows from [Ban93, Lemma 1.5(i)].

Lemma 4. *Let* $\vec{z} \xleftarrow{\$} D^n_{\mathfrak{s}}$*. Then*

$$\Pr\left[\|\vec{z}\|_2 < \mathfrak{s}\sqrt{2n}\right] > 1 - 2^{-\log(e/2)n/2} > 1 - 2^{-n/8}.$$

A.2 Commitment Scheme

In the usual definition, a commitment scheme consists of a triple of algorithms $(\mathsf{ComGen}, \mathsf{Com}, \mathsf{Open})$ which work as follows. $\mathsf{ComGen}\colon \emptyset \to \{0,1\}^*$ is a probabilistic algorithm that produces the public parameters $p \xleftarrow{\$} \mathsf{ComGen}()$ for the commitment scheme. $\mathsf{Com}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, $(c, r) \xleftarrow{\$} \mathsf{Com}(p, m)$ is a probabilistic algorithm that takes the public parameters p and a message m as input and produces a commitment c and some randomness r used to compute and open c. $\mathsf{Open}\colon \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, $b \leftarrow \mathsf{Open}(p, c, m, r)$ is a deterministic algorithm that takes the public parameters p, a commitment c, a message m, and randomness r as input, and produces a bit b as output.

A commitment scheme should be *hiding* and *binding*.

Definition 4 (Hiding). *A commitment scheme* $(\mathsf{ComGen}, \mathsf{Com}, \mathsf{Open})$ *is* hiding *under the assumption that a problem* P *is hard if for every adversary* $\mathcal{A}$ *and*

$\varepsilon \geq 0$ *such that*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c} p \xleftarrow{\$} \mathsf{ComGen}();\ (m_0, m_1) \xleftarrow{\$} \mathcal{A}(p) \\ b \xleftarrow{\$} \{0,1\};\ (c, r) \xleftarrow{\$} \mathsf{Com}(p, m_b) \\ b' \xleftarrow{\$} \mathcal{A}(p, c) \end{array}\right] = \frac{1}{2}(1 + \varepsilon)$$

there exists an adversary $\mathcal{A}'$ *with about the same running time than* $\mathcal{A}$ *that solves* P *with advantage* ε*.*

Definition 5 (Binding). *A commitment scheme* $(\mathsf{ComGen}, \mathsf{Com}, \mathsf{Open})$ *is* binding *under the assumption that a problem* P *is hard if for every adversary* $\mathcal{A}$ *and* $\varepsilon \geq 0$ *such that*

$$\Pr\left[\begin{array}{c} m \neq m' \text{ and} \\ 1 = \mathsf{Open}(p, c, m, r) = \mathsf{Open}(p, c, m', r') \end{array} \;\middle|\; \begin{array}{c} p \xleftarrow{\$} \mathsf{ComGen}(); \\ (m, m', r, r', c) \xleftarrow{\$} \mathcal{A}(p) \end{array}\right] = \varepsilon$$

there exists an adversary $\mathcal{A}'$ *that has about the same running time than* $\mathcal{A}$ *and solves* P *with advantage* ε*.*

A.3 Zero-Knowledge Proofs of Knowledge

Let $f\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and $t \in \{0,1\}^*$. We want to prove knowledge of a $w \in \{0,1\}^*$ such that $f(t, w) = 1$. The bit-string t is called the *statement* and w the *witness*. For example, f can be such that $f(t, w) = 1$ if and only if w encodes a binary vector $\vec{s} \in \{0,1\}^n$ such that $A\vec{s} = \vec{u}$ for a matrix $A \in \mathbb{Z}_q^{m \times n}$ and $\vec{u} \in \mathbb{Z}_q^m$ encoded by t.

Now, an *interactive proof system* for f consists of a pair $(\mathcal{P}, \mathcal{V})$ of efficient probabilistic interactive algorithms. The algorithm $\mathcal{P}$ is called the prover and gets as inputs a statement $t \in \{0,1\}^*$ and a witness $w \in \{0,1\}^*$. The algorithm $\mathcal{V}$ is the verifier and only gets t. The verifier outputs a bit $b \in \{0,1\}$ as the last message of the interaction. Intuitively, if the prover knows a witness for t, for example $f(t, w) = 1$, then he shall always make the verifier output $b = 1$, i.e. make the verifier accept (correctness), but if the prover does not know such a witness he should not be able to do so (knowledge-soundness). We also want our protocols be such that the prover does not reveal any additional information about his witness (zero-knowledge). We write $\mathsf{tr} \xleftarrow{\$} \langle \mathcal{P}(t, w), \mathcal{V}(t) \rangle$ for the transcript tr of the messages exchanged in a random experiment where $\mathcal{P}$ interacts with $\mathcal{V}$.

Definition 6. *An interactive proof system* $(\mathcal{P}, \mathcal{V})$ *for* $f\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ *is* correct *if for all* $t, w \in \{0,1\}^*$ *such that* $f(t, w) = 1$*, the verifier accepts with overwhelming probability in the interaction* $\langle \mathcal{P}(t, w), \mathcal{V}(t) \rangle$*.*

Definition 7. *Let* $(\mathcal{P}, \mathcal{V})$ *be an interactive proof system for the predicate* f*. A* simulator *is an efficient probabilistic algorithm that gets a statement but not a witness and outputs a transcript of an interaction between* $\mathcal{P}$ *and* $\mathcal{V}$*.*

Then, $(\mathcal{P}, \mathcal{V})$ *is* statistically honest-verifier zero-knowledge *if there exists a simulator* $\mathcal{S}$ *such that for all* $t, w \in \{0,1\}^*$ *with* $f(t,w) = 1$ *the simulated transcript* $\mathsf{tr} \xleftarrow{\$} \mathcal{S}(t)$ *is statistically close to the real transcript* $\mathsf{tr}' \xleftarrow{\$} \langle \mathcal{P}(t,w), \mathcal{V}(t) \rangle$.

On the other hand, $(\mathcal{P}, \mathcal{V})$ *is* computationally honest-verifier zero-knowledge *under the assumption that a problem* P *is hard if there exists a simulator* $\mathcal{S}$ *such that for all* $t, w \in \{0,1\}^*$ *with* $f(t,w) = 1$ *and every distinguisher* $\mathcal{A}$ *that is able to distinguish* $\mathsf{tr} \xleftarrow{\$} \mathcal{S}(t)$ *from* $\mathsf{tr}' \xleftarrow{\$} \langle \mathcal{P}(t,w), \mathcal{V}(t) \rangle$ *with advantage* ε*,*

$$\Big| \Pr\left[b = 1 \,|\, \mathsf{tr} \leftarrow \mathcal{S}(t);\, b \leftarrow \mathcal{A}(\mathsf{tr})\right] - \Pr\left[b = 1 \,|\, \mathsf{tr} \leftarrow \langle \mathcal{P}(t,w), \mathcal{V}(t) \rangle;\, b \leftarrow \mathcal{A}(\mathsf{tr})\right] \Big| \geq \varepsilon,$$

there exists a solver $\mathcal{A}'$ *that has about the same running time than* $\mathcal{A}$ *and is able to solve* P *with advantage* ε*.*

Definition 8. *Let* $(\mathcal{P}, \mathcal{V})$ *be an interactive proof system for the predicate* f*. An* extractor *is an algorithm that has resettable black-box access to a prover* $\mathcal{P}^*$*.*

Then, $(\mathcal{P}, \mathcal{V})$ *is* computationally knowledge-sound *with soundness error* $0 \leq \varepsilon_0 < 1$ *under the assumption that a problem* P *is hard if there exists an extractor* $\mathcal{E}$ *such that for every* $t \in \{0,1\}^*$ *and every deterministic prover* $\mathcal{P}^*$ *that makes* $\mathcal{V}$ *accept with probability* $\varepsilon > \varepsilon_0$ *in* $\langle \mathcal{P}^*(t), \mathcal{V}(t) \rangle$*,* $\mathcal{E}^{\mathcal{P}^*(t)}$ *outputs a witness* w *for* t *or a solution to* P *in expected runtime at most* $\frac{k}{\varepsilon - \varepsilon_0}$ *for some small* $k > 0$ *where running* $\mathcal{P}^*$ *once takes unit time.*

Several of our protocols are for predicates where it is easy to find a witness for all statements. These protocols are used as building blocks for our linear-size proof system. The reason why the protocols are still interesting is that they can be viewed as commit-and-proof protocolsThis means that the prover sends a commitment to the witness in his first message. Then the statement is extended to also include the commitment and the proof not only shows that the prover knows a witness for the original statement, but that he knows a commitment opening that is a witness.

In all of our protocols the verifier messages are simply uniformly random. Hence, our protocols can be made non-interactive with the Fiat-Shamir transform [FS86]. We view the interactive variants of the protocols only as intermediate representations that are easy to reason about but that have no practical relevance. As a result, we do not consider the number of rounds an important quantity and make no effort in optimizing it.

A.4 Proving Linear and Multiplicative Relations

In this paper we will use techniques developed in [ALS20, ENS20] to prove certain linear and multiplicative relations. For completeness, we will briefly recall the protocols to prove such relations and refer to [ALS20, ENS20] for more details.

Opening proof [ALS20]. The key component of proving linear and multiplicative relations is the proof of knowledge of a committed message. Concretely, the prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$, which has a BDLOP commitment $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$, that $\mathcal{P}$ knows a short randomness vector $\vec{\boldsymbol{r}}$ over $\mathcal{R}_q$ and a message $\boldsymbol{m} \in \mathcal{R}_q$ such that $\boldsymbol{B}_0\vec{\boldsymbol{r}} = \vec{\boldsymbol{t}}_0$ and $\langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}} \rangle + \boldsymbol{m} = \boldsymbol{t}_1$.

The proof goes as follows. Prover $\mathcal{P}$ starts by sampling a vector $\vec{\boldsymbol{y}}$ from a discrete Gaussian and computing $\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$. Then, it sends $\vec{\boldsymbol{w}}$ to the verifier. After receiving the challenge $\boldsymbol{c} \xleftarrow{\$} \mathcal{C}$ from $\mathcal{V}$, the prover computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and applies rejection sampling. If it does not abort, $\mathcal{P}$ sends $\vec{\boldsymbol{z}}$. Finally, the verifier checks that coefficients of $\vec{\boldsymbol{z}}$ are small and $\boldsymbol{B}_0\vec{\boldsymbol{z}} \stackrel{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$.

Attema et al. [ALS20] show that one can efficiently extract a *weak opening* of $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ defined below.

Definition 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 BDLOP commitment scheme is still binding with respect to weak openings under the Module-SIS assumption.

Product proof [ALS20]. For simplicity, we will prove that a committed polynomial $\boldsymbol{m}$ has binary NTT coefficients, i.e. $\mathsf{NTT}(\boldsymbol{m}) \in \{0,1\}^d \subset \mathbb{Z}_q^d$. This is equivalent to proving $\boldsymbol{m}(\boldsymbol{m}-1) = 0$ over $\mathcal{R}_q$. Then, one can easily generalise the protocol to prove arbitrary multiplicative relations, e.g. $\boldsymbol{m}_1\boldsymbol{m}_2 = \boldsymbol{m}_3$ for polynomials $\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3 \in \mathcal{R}_q$.

Let $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ be the BDLOP commitment to $\boldsymbol{m}$, i.e. $\boldsymbol{B}_0\vec{\boldsymbol{r}} = \vec{\boldsymbol{t}}_0$ and $\langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}} \rangle + \boldsymbol{m} = \boldsymbol{t}_1$ for a short randomness vector $\vec{\boldsymbol{r}}$. Let us adapt the notation from the opening proof above and set $\boldsymbol{f} := \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_1$. Note that $\boldsymbol{f}$ can be computed by the verifier. Since $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ we have that $\boldsymbol{f} = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}} \rangle - \boldsymbol{c}\boldsymbol{m}$. Thus,

$$\boldsymbol{f}(\boldsymbol{f} + \boldsymbol{c}) = \boldsymbol{v}_0 + \boldsymbol{v}_1\boldsymbol{c} + \boldsymbol{m}(\boldsymbol{m}-1)\boldsymbol{c}^2$$

for $\boldsymbol{v}_0 := \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}} \rangle^2$ and $\boldsymbol{v}_1 = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}} \rangle(1 - 2\boldsymbol{m})$. Hence, the goal is to prove that the quadratic coefficient of $\boldsymbol{f}(\boldsymbol{f}-1)$ vanishes. We proceed as follows.

The prover starts by generating vectors $\vec{\boldsymbol{r}}, \vec{\boldsymbol{y}}$ and the BDLOP commitment $\vec{\boldsymbol{t}} = (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \boldsymbol{t}_2)$ where

$$
\begin{cases}
\boldsymbol{B}_0\vec{\boldsymbol{r}} = \vec{\boldsymbol{t}}_0\\
\vec{\boldsymbol{b}}_1^T\vec{\boldsymbol{r}} + \boldsymbol{m} = \boldsymbol{t}_1\\
\vec{\boldsymbol{b}}_2^T\vec{\boldsymbol{r}} + \boldsymbol{v}_1 = \boldsymbol{t}_2.
\end{cases}
$$

As in the opening proof, it computes $\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$. Then, it outputs $(\boldsymbol{t}, \vec{\boldsymbol{w}}, \boldsymbol{v}_0')$ where $\boldsymbol{v}_0' = \boldsymbol{v}_0 + \vec{\boldsymbol{a}}_2^T\vec{\boldsymbol{y}}$. After getting a challenge $\boldsymbol{c} \xleftarrow{\$} \mathcal{C}$ from the verifier, $\mathcal{P}$ computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + c\vec{\boldsymbol{r}}$ and applies rejection sampling. If it does not fail, the prover outputs $\vec{\boldsymbol{z}}$. Then, $\mathcal{V}$ checks that $\vec{\boldsymbol{z}}$ has small coefficients and $\boldsymbol{B}_0\vec{\boldsymbol{z}} \stackrel{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$. Next, it computes

$$\boldsymbol{f} = \langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_1 \text{ and } \boldsymbol{f}_2 = \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_2.$$

Eventually, $\mathcal{V}$ checks whether:

$$\boldsymbol{f}(\boldsymbol{f} + \boldsymbol{c}) + \boldsymbol{f}_2 \stackrel{?}{=} \boldsymbol{v}_0'.$$

As described in [ALS20], proving additional quadratic relations does not affect the proof size.

Linear proof [ENS20]. Another building block, which will be used in our protocols, is the proof of a linear relation. Namely, suppose the prover wants to prove knowledge of a vector $\vec{s} \in \mathbb{Z}_q^m$ which satisfies:

$$A\vec{s} = \vec{u}$$

for a public matrix $A \in \mathbb{Z}_q^{n \times m}$ and vector $\vec{u} \in \mathbb{Z}_q^n$. For simplicity, we assume that $n = m = d$. The idea is to ask the verifier for a random challenge $\vec{\gamma} \xleftarrow{\$} \mathbb{Z}_q^d$ and then prove that

$$\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle = 0.$$

Now,

$$\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle = \langle A\vec{s}, \vec{\gamma} \rangle - \langle \vec{u}, \vec{\gamma} \rangle = \langle \vec{s}, A^T\vec{\gamma} \rangle - \langle \vec{u}, \vec{\gamma} \rangle = \sum_{i=0}^{l-1} f_i$$

where $\vec{f} := \vec{s} \circ (A^T\vec{\gamma}) - \vec{u} \circ \vec{\gamma}$. In other words, the sum of coefficients of $\vec{f}$ is equal to $\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle$. Consequently, Esgin et al. [ENS20, Lemma 2.1] observe that the constant coefficient of $\check{\boldsymbol{f}} := \mathsf{NTT}^{-1}(\vec{f}) \in \mathcal{R}_q$ satisfies:

$$\check{\boldsymbol{f}}_0 = \frac{\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle}{d}.$$

Hence, the goal is to prove that the constant of $\check{\boldsymbol{f}}$ is zero.

To this end, let $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ be a BDLOP commitment to $\boldsymbol{s}$ defined below:

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

Note that given a commitment to s, the verifier can manually construct a commitment to $\check{\boldsymbol{f}}$ as follows:

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

In order to prove $\check{f}_0 = 0$, at the beginning $\mathcal{P}$ commits to a uniformly random polynomial $\boldsymbol{g} \in \mathcal{R}_q$ that also has the constant coefficient equal to zero, i.e. $\boldsymbol{t}_2 = \langle \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}$. After getting a challenge $\vec{\gamma}$ from the verifier, $\mathcal{P}$ computes $\boldsymbol{h} = \boldsymbol{g} + \check{\boldsymbol{f}}$ and sends $\boldsymbol{h}$. Then, $\mathcal{V}$ checks whether $h_0 = 0$. Finally, we need to prove that the equation $\boldsymbol{h} = \boldsymbol{g} + \check{\boldsymbol{f}}$ holds. We do it by proving that $\boldsymbol{t}_2 + \boldsymbol{t}_f - \boldsymbol{h}$ is a commitment to zero.

We are now ready to sketch out the protocol. First, the prover generates $\vec{\boldsymbol{r}}, \vec{\boldsymbol{y}}$ as before and computes $\vec{\boldsymbol{t}} = (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \boldsymbol{t}_2)$ defined above. As usual, it computes $\vec{\boldsymbol{w}} = \boldsymbol{B}_0 \vec{\boldsymbol{y}}$. Then, it sends $(\vec{\boldsymbol{t}}, \vec{\boldsymbol{w}})$ to the verifier. Next, given a challenge $\vec{\gamma} \xleftarrow{\$} \mathbb{Z}_q^d$ from $\mathcal{V}$, the prover computes $\boldsymbol{h} := \boldsymbol{g} + \check{\boldsymbol{f}}$ and $\boldsymbol{w}'$ defined as

$$\boldsymbol{w}' := \langle \vec{\boldsymbol{b}}_2 + \mathsf{NTT}^{-1}(A^T \vec{\gamma}) \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{y}} \rangle.$$

Then, $\mathcal{P}$ outputs $(\boldsymbol{h}, \boldsymbol{w}')$. Further, $\mathcal{V}$ outputs the challenge $\boldsymbol{c} \xleftarrow{\$} \mathcal{C}$. Eventually, $\mathcal{P}$ calculates $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and runs rejection sampling as before. After obtaining $\vec{\boldsymbol{z}}$, the verifier checks that coefficients of $\vec{\boldsymbol{z}}$ are small, $\boldsymbol{B}_0 \vec{\boldsymbol{z}} \stackrel{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$ as in the opening proof. Then, it also checks whether the first coefficient of $\boldsymbol{h}$ is indeed zero as well as

$$\langle \vec{\boldsymbol{b}}_2 + \mathsf{NTT}^{-1}(A^T \vec{\gamma}) \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{z}} \rangle \stackrel{?}{=} \boldsymbol{w}' + \boldsymbol{c}\,(\boldsymbol{t}_2 + \boldsymbol{t}_f - \boldsymbol{h})\,.$$

This protocol extends naturally when the length of vectors $\vec{s}$ and $\vec{u}$ are a multiple of d. Finally, we highlight that having additional linear relations on $\vec{s}$ does not affect the total proof size.