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 5and the full version of the paper.).

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. 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.

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.

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

		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.

$\mathcal{R}$ 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 $\mathsf{NTT}(\boldsymbol{f}) \circ \mathsf{NTT}(\boldsymbol{g}) = \mathsf{NTT}(\boldsymbol{f}\boldsymbol{g})$ 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

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

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].

[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 the full version.

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 the full version).

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*

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

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

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

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

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

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

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

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

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

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 $\vec{\boldsymbol{s}}_i^*$, 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 $\vec{\boldsymbol{z}} = \vec{\boldsymbol{z}}_0 + \cdots + \vec{\boldsymbol{z}}_{l-1}\delta^{l-1}$. 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 $\boldsymbol{B}_0$.

The proof of Theorem 1 is contained in the full version of the paper.

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.

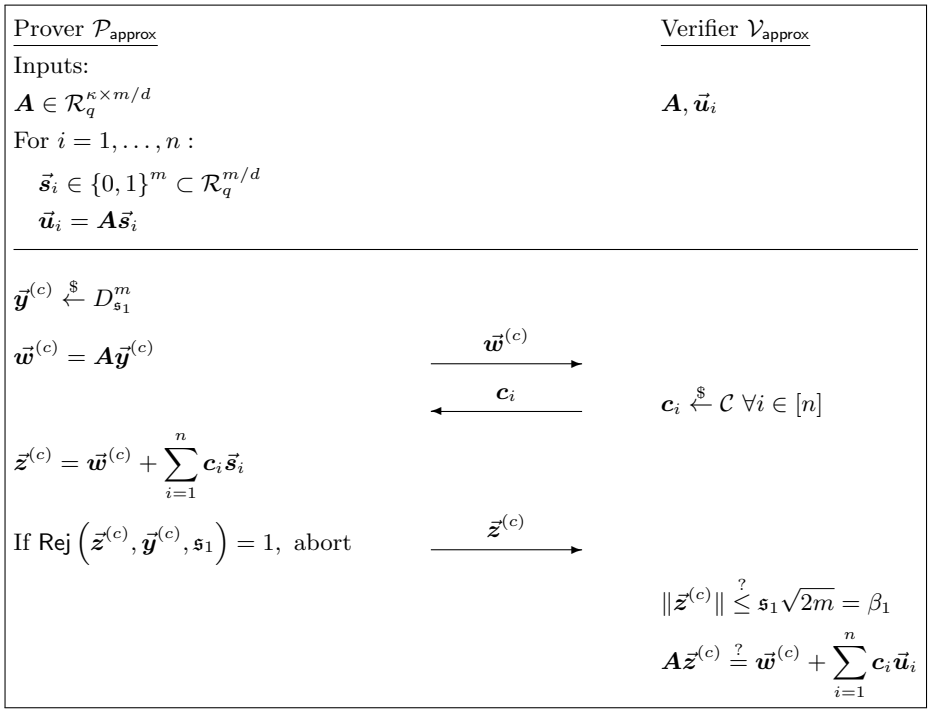

Fig. 2. Approximate amortized opening proof for lattice-based hashes. Used for bootstrapping the exact amortized proof in Figure 1.

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

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.

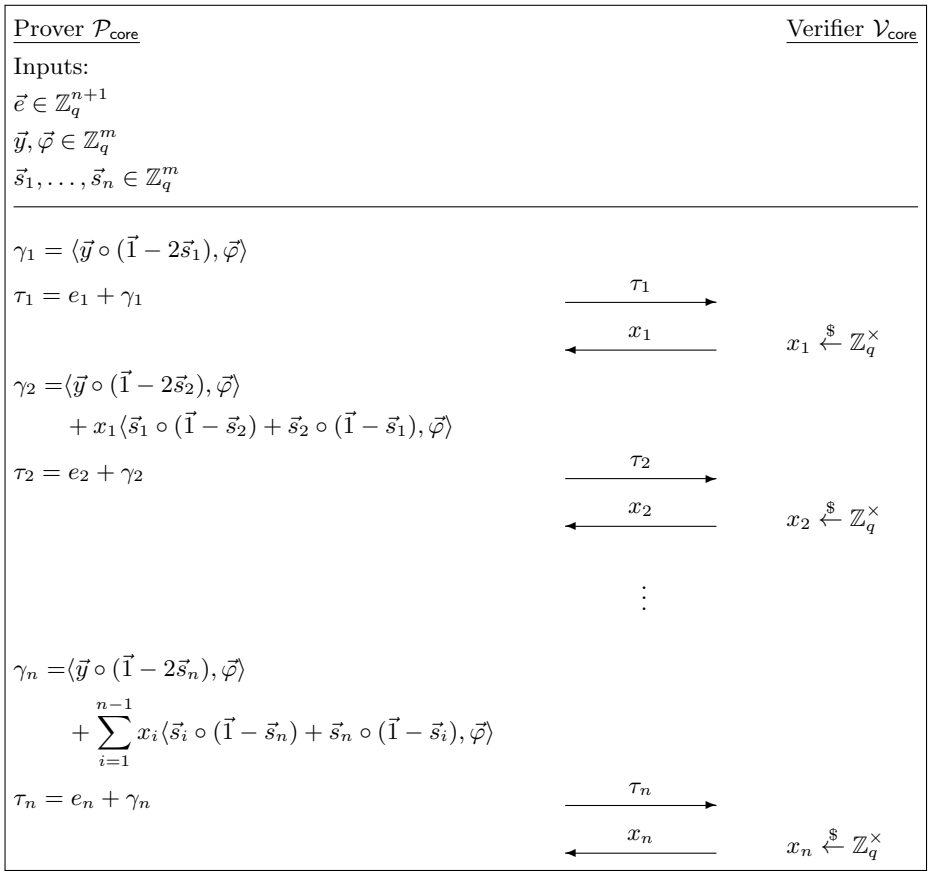

Fig. 4. Core protocol for exact amortized opening proof in Figure 1

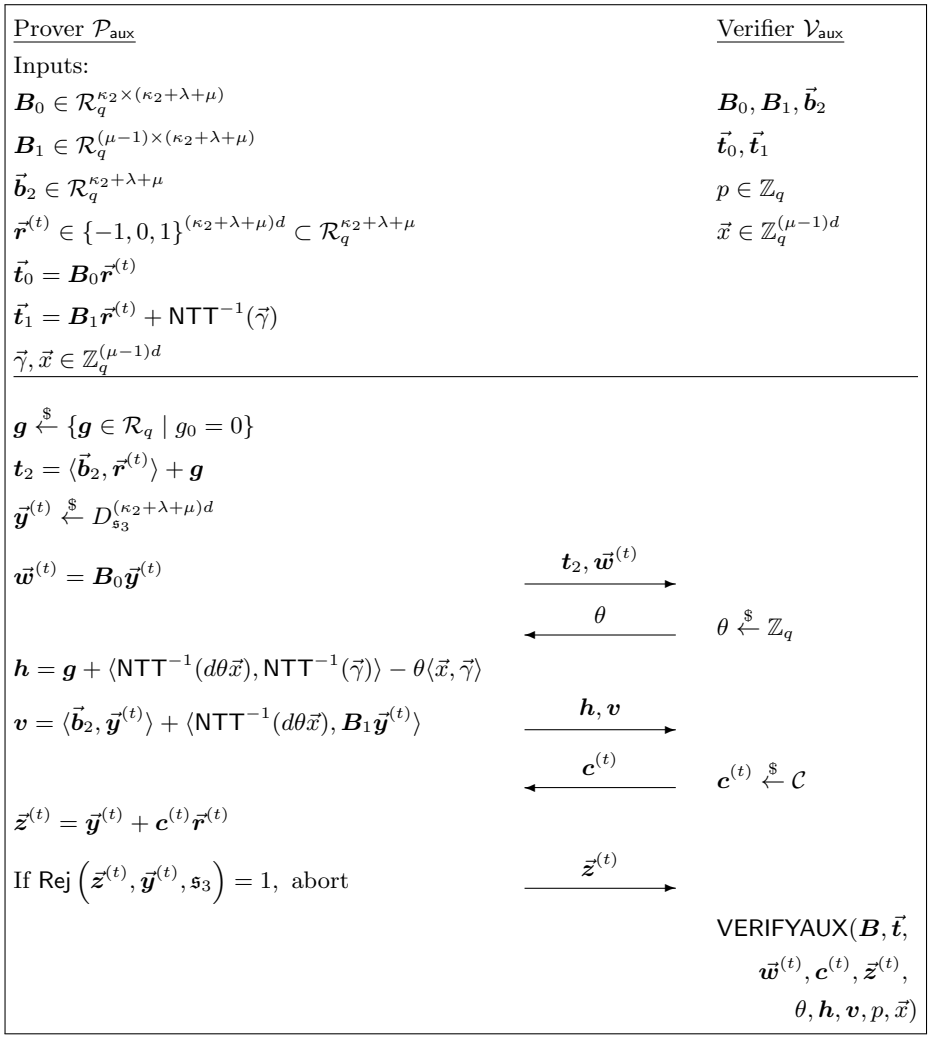

Fig. 5. Auxiliary linear proof needed in our exact amortized opening proof in Figure 1 and in the tree opening proof.

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 $\vec{c}, \vec{\varphi}, \vec{x}, \theta, \boldsymbol{c}^{(t)}$ 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, a coefficient is smaller than 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.

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

$$\begin{aligned} \vec{\boldsymbol{u}}_i = \boldsymbol{A}\vec{\boldsymbol{s}}_i, \quad \vec{\boldsymbol{s}}_i = \begin{pmatrix} \vec{\boldsymbol{s}}_{i,l} \\ \vec{\boldsymbol{s}}_{i,r} \end{pmatrix} \in \{0,1\}^m \subset \mathcal{R}_q^{m/d}, \\ \boldsymbol{G}\vec{\boldsymbol{s}}_{i,l} = \vec{\boldsymbol{u}}_{2i}, \\ \boldsymbol{G}\vec{\boldsymbol{s}}_{i,r} = \vec{\boldsymbol{u}}_{2i+1} \end{aligned} \tag{9}$$

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 $0 \leq k \leq a - 1$.

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 $x_i\vec{\boldsymbol{s}}_i$, $i \geq 2^{k+1}$, 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,

$$\sum_{i=2^k}^{2^{k+1}-1} (x_{2i}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,l} + x_{2i+1}\boldsymbol{G}\vec{\boldsymbol{s}}_{i,r}) = \vec{\boldsymbol{w}}_k - \vec{\boldsymbol{w}}_{k+1}. \tag{10}$$

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 (9), and that

$$\boldsymbol{A}\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}}_{k'} + \sum_{i=1}^{2^{k'+1}-1} x_i \vec{\boldsymbol{u}}_i \tag{11}$$

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 (11) 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 (11) 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 (10) 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 $\boldsymbol{c}_{2^k}, \ldots, \vec{\boldsymbol{c}}_{2^{k+1}-1}$. 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 $\vec{\boldsymbol{w}}_0^{(c)} + \boldsymbol{c}_1 \vec{\boldsymbol{u}}_1$.

We defer the specification of the protocol and its security analysis to the full version of the paper. There we also describe how to apply it for proving R1CS, and discuss the comparison of the protocol to Ligero.

References

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed ς-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 $(k_{1,}$ đots, $k_{\mu})$-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.

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.

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.

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.

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.

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.