Practical Lattice-Based Zero-Knowledge Proofs for Integer Relations [⋆]

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

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

Abstract. We present a novel lattice-based zero-knowledge proof system for showing that (arbitrary-sized) committed integers satisfy additive and multiplicative relationships. The proof sizes of our schemes are between two to three orders of magnitude smaller than in the lattice proof system of Libert et al. (CRYPTO 2018) for the same relations. Because the proof sizes of our protocols grow linearly in the integer length, our proofs will eventually be longer than those produced by quantum-safe succinct proof systems for general circuits (e.g. Ligero, Aurora, etc.). But for relations between reasonably-sized integers (e.g. 512-bit), our proofs still result in the smallest zero-knowledge proof system based on a quantum-safe assumption. Of equal importance, the run-time of our proof system is at least an order of magnitude faster than any other quantum-safe scheme.

1 Introduction

Zero-knowledge proofs are fast becoming widespread in practical cryptography, and recent advances in their constructions that utilize the nice algebraic properties of finite fields and pairings, has resulted in a plethora of fairly fast schemes with very short proofs. The quantum-safe literature, while also producing some recent breakthrough results, is somewhat more subdued. The most efficient known technique for creating quantum-safe zero-knowledge proofs for arbitrary circuits are the PCP / IOP proof systems (e.g. Aurora [7] and Ligero [1]) which only rely on the security of cryptographic hash functions. The proofs produced by such schemes are even sub-linear in length, being logarithmic for Aurora and square-root for Ligero. And apart from digital signatures (e.g. [13], which are essentially zero-knowledge proofs) and commitment schemes with opening proofs (e.g. [4]), there hasn't been any zero-knowledge proofs for "useful" relations that outperform the aforementioned generic proofs, at least in terms of proof size. This state of affairs is a bit peculiar, because unlike for basic primitives (e.g. encryption / signatures), making a quantum-safe mathematical hardness assumption does not seem to aid in creating shorter zero-knowledge proofs over just assuming the existence of generic collision-resistant hash functions.

The recent work of [2,15] that builds on [4] to create efficient zero-knowledge proof systems for multiplicative and $\mathbb{Z}_q$-linear relations raises the possibility that one can indeed construct lattice-based zero-knowledge proofs for useful, natural relations that are smaller than those produced by generic proofs. In this work, we design zero-knowledge proofs for arbitrary-length integer addition and multiplication, which for reasonably sized lengths (e.g. up to 512-bits) result in the shortest, and fastest, known zero-knowledge proofs based on the hardness of a quantum-safe problem.[3]

While our proof sizes are linear, they have some advantages over the generic sub-linear proof systems like Ligero and Aurora. Both of the aforementioned schemes use the Kilian / Micali [19,28] approach which entails committing to the encoding of the messages in a Merkle tree and then outputting several hundred

[⋆] This is the full version of [26] presented at ACM CCS 2020.

[3] As many other practical schemes utilizing the Fiat-Shamir framework, we show the security of our scheme in the ROM based on the hardness of a quantum-safe problem rather than the more desirable security reduction in the QROM. The recent results of [20,12,23,11], which give QROM proofs of protocols proven in the ROM under different assumptions or looser reductions, give evidence that such schemes are in fact still secure against quantum attackers. There also hasn't been an example of any natural scheme proved secure in the ROM based on a quantum-safe problem that ends up being insecure in the QROM.

paths of that tree to prove knowledge of the committed message. Even though the depth of this tree may only be logarithmic in the instance size, the fact that one needs to open several hundred paths imposes a "start-up" cost of somewhere between 50-100KB for 128-bit security. So while a linear-size proof will eventually be longer than these approaches, for small instance sizes our proofs end up being somewhat shorter.

Another dimension in which our scheme, and lattice schemes in general, outperform is efficiency. The fundamental operation used in lattice schemes over polynomial rings is polynomial multiplication, which is extremely efficient when performed using the number theoretic transform (NTT). As an example, despite lattice-based encryption schemes having public keys and ciphertexts around 1KB, which is 30X longer than those of elliptic curve schemes, they can be 5X *faster* than the most efficient instantiations of the latter; requiring under 3 micro-seconds per encryption on standard processors (c.f. [27]). While zero-knowledge proofs involve a lot more operations than encryption schemes, these constructions can still be significantly more efficient than generic proofs which require somewhere between half a second (for Ligero) and around 5 seconds (for Aurora) to prove knowledge of a generic (SHA-256) commitment.

For example, we show that proving knowledge of an additive relation between 32 (resp. 128)-bit integers results in proof sizes of 11 (resp. 25) KB (see Figure 4), while multiplicative relations have proof sizes of 15 (resp. 40) KB (see Figure 6).[4] We provide implementations of our protocols that achieve a running time of 2.4 milliseconds, on a standard laptop, for 128-bit multiplication proofs. As far as we are aware, these are the most efficient potentially quantum-safe proofs for these problems. And because the operations involved in lattice constructions have been shown to be readily ported to more constrained devices, this opens up the possibility of quantum-safe zero-knowledge proofs being used in "daily" interactions, e.g. credit card transactions, where operations should take (significantly) less time than a second.

Apart from [7,1], the only other quantum-safe solution for these problems that we're aware of is the work of [22] which also bases these operations on the hardness of lattice problems. The proofs in [22] use the Stern-proof approach, which requires repetitions of a Σ-protocol having soundness error 2/3, and so even simple proofs of commitment openings are several megabytes long. Furthermore, the multiplication proof committed to every step of the Karatsuba algorithm, which ends up requiring commitments of asymptotic length $O(k^{\log_2 3})$, for multiplication of k-bit integers. This could make the entire proof on the order of hundreds of megabytes. In contrast, our multiplication algorithm uses an in-place NTT transformation which keeps the length of the proof linear for most reasonable parameters.[5] In practice, we estimate that the size of the proofs in [22] are about three orders of magnitude larger than in the current work.

Proving that committed data satisfies additive and multiplicative relations can be useful in many real-world scenarios. As a simple example that uses both, consider an auction or a collection of stock purchase orders. The bidder/buyer places a set of orders for x_i quantity of item i at price y_i. Because the final price paid may depend on the prices of non-winning bidders, it is important to wean out invalid bids. So the bidder/buyer may also prove that he has greater than $\sum_i x_i \cdot y_i$ in his account (which has been certified by a bank). In general, because our proofs are quite efficient and fairly short when the number of operations is not too large, they can be used inside "smart contracts" (in a very broad sense) that utilize addition and multiplication.

1.1 Polynomial / Vector Commitments and Zero-Knowledge Proofs

Our results build on the recent progress from [2] and [15] on proving relations between committed messages in the lattice-based commitment scheme from [4]. The commitment scheme in [4] works over a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ and the commitment to messages $\boldsymbol{m}_i \in \mathcal{R}_q$ is of the form

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

[4] These numbers include the combination of the commitment size and the proof size.

[5] Asymptotically, our multiplication scheme does increase by a factor of $\log k$, but this logarithmic growth first appears for very large (e.g. 2048-bit) integers, which is probably not particularly relevant to practical applications.

where $\boldsymbol{B}_0, \boldsymbol{b}_i$ are public random matrices, and $\vec{\boldsymbol{r}}$ is a randomness vector chosen in the commitment procedure. It was already shown in [4] how to prove, in zero-knowledge, the knowledge of the messages and linear relations (over $\mathcal{R}_q$) of the $\boldsymbol{m}_i$. The results of [2] extended this to efficient proofs of multiplicative relations and [15] showed that one can prove linear relations over the $\mathbb{Z}_q$. In particular, if the NTT[6] of $\boldsymbol{m}_i \in \mathcal{R}_q$ is the vector $\vec{v}_i \in \mathbb{Z}_q^d$, then [2] allows one to prove relations of the form $\vec{v}_1 \circ \vec{v}_2 = \vec{v}_3 \in \mathbb{Z}_q^d$, where $\circ$ denotes component-wise multiplication. The result of [15] allows proofs of the form $\vec{v}_1 = M\vec{v}_2$ over $\mathbb{Z}_q^d$. It is also easy to see that these results can be extended to arbitrary-length vectors.

Slightly more formally, the results of [4,2,15] provide a binding and hiding vector commitment scheme (with $\mathsf{Com}, \mathsf{Open}$ operations), along with zero-knowledge proofs of knowledge of the following relations:

1. [Product Proof] Given a commitment c, prove knowledge of $\vec{v}_i \in \mathbb{Z}_q^m$ such that $\mathsf{Open}(c) = (\vec{v}_1, \vec{v}_2, \vec{v}_3) \wedge \vec{v}_1 \circ \vec{v}_2 = \vec{v}_3$.
2. [Unstructured Linear Proof] Given a commitment c and a matrix $M \in \mathbb{Z}_q^{n\times m}$, prove knowledge of $\vec{v}_1 \in \mathbb{Z}_q^m, \vec{v}_2 \in \mathbb{Z}_q^n$ such that $\mathsf{Open}(c) = (\vec{v}_1, \vec{v}_2) \wedge M\vec{v}_1 = \vec{v}_2$.

In this work, we will use this abstraction to prove additive and multiplicative relations between integers. Our proofs will generally consist of many instantiations of the product and unstructured linear proofs, and the most efficient approach for combining these proofs is not to run them separately, but rather run them simultaneously and reuse many overlapping parts. We describe, and implement, a framework for creating such proofs. Of independent interest is also our novel technique for proving "relaxed" range proofs of committed values. Unlike typical range proofs, which linearly depend on k for proving that an integer is in the range $[0, 2^k)$, we show how to create a "relaxed" range proof which proves that elements are in the range $[0, n \cdot 2^k)$ for some small value n, but without having the proof size depend on k. We believe that this technique can find applications outside of our integer relations proof. We now elaborate on our techniques in more detail.

1.2 Our Results and Techniques

The works of [2,15] instantiated the commitment scheme of [4] over polynomial rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$ with $d = 128$ and q a prime of size $\approx 2^{32}$. Inside the zero-knowledge proof, certain auxiliary terms in $\mathcal{R}_q$ need to be committed to, and so it is advantageous to take the size of this ring to be as small as possible. On the other hand, technical reasons prevent the q in the commitment scheme from being too small, and too small a d will not allow for random challenges with small coefficients to have 128 bits of entropy. Additionally, having $q < 2^{32}$ allows for more efficient operations in modern CPUs. For these reasons, it is preferable to always use the same ring (and not increase the modulus q) when working with large integers. We will now describe several solutions for integer relation proofs. The most straightforward one, described in Section 1.2, applies when the integers are very small (e.g. 16 bits). In this case, one can just perform the operations directly in the underlying ring modulo $q \approx 2^{32}$, with the only necessary check being that the sum / product do not "wrap around" modulo q. In Section 1.2 we describe an amortization technique for performing batches of 128 integer operations at the cost of almost one such operation, with the caveat being that the q in $\mathcal{R}_q$ may need to be somewhat increased.

These approaches do not work for larger integers, and we instead need to do addition and multiplication directly over the binary representations. This is the main result of the current work and an overview of the techniques is given below.

Operations on small integers. Suppose that $q \approx 2^{32}$ and and we would like to create a proof that the product of two integers is the third (i.e. $m_1 m_2 = m_3$). If the two multiplicands are smaller (in absolute value) than 2^{14}, then one can create a proof as follows: let $c = \mathsf{Com}(m_1, m_2, m_3, \vec{b}_1, \vec{b}_2)$, where $m_i \in \mathbb{Z}_q^1$ and $\vec{b}_i \in \mathbb{Z}_q^{15}$ are the binary decompositions (we can assume that it's in two's complement, but any representation can be

[6] If $X^d + 1 = (X - r_1) \cdot \ldots \cdot (X - r_d) \bmod q$, then the NTT coefficients of $\boldsymbol{m} \in \mathcal{R}_q$ are $(\boldsymbol{m} \bmod X - r_1, \ldots, \boldsymbol{m} \bmod X - r_b) = (\boldsymbol{m}(r_1) \bmod q, \ldots, \boldsymbol{m}(r_d) \bmod q)$.

supported) of m_i. Define the matrix $M \in \mathbb{Z}_q^{1\times 15}$ be $[2^0\ 2^1\ 2^2\ \ldots\ 2^{13}\ \ -2^{14}]$. We would then want to prove the following relation:

$$\vec{v}_1 \circ \vec{v}_2 = \vec{v}_3 \wedge \left(\text{for } i = 1, 2,\ \vec{b}_i \circ (\vec{1} - \vec{b}_i) = 0 \wedge M\vec{b}_i = \vec{v}_i\right).$$

Notice that because the above proves that all the coefficients of $\vec{b}_i$ are binary, it implies that $M\vec{b}_i$ is less (in magnitude) than 2^{14}, and so the product $\vec{v}_1 \circ \vec{v}_2 = \vec{v}_3$, which we prove modulo q, also holds over the integers.

Addition of arbitrary-length integers. Suppose that $a_k \ldots a_1 a_0$ is the binary representation of the integer $a = \sum_{i=0}^{k} a_i 2^i$. Then define the polynomial $\boldsymbol{a} = \sum_{i=0}^{k} a_i X^i \in \mathbb{Z}[X]$. Note that even if $a+b=c$ as integers, it's possible that $\boldsymbol{a}+\boldsymbol{b} \neq \boldsymbol{c}$ due to the carries involved in binary integer addition that isn't present in the addition of polynomials. Nevertheless, $(\boldsymbol{a}+\boldsymbol{b})(2) = \boldsymbol{c}(2)$ and so there exists a polynomial $\boldsymbol{f}$ such that

$$\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{f} \cdot (X-2) = \boldsymbol{c}. \tag{1}$$

It's not hard to see that this polynomial $\boldsymbol{f}$ consists exactly of the carries that appear when performing the schoolbook addition of $a_k \ldots a_1 a_0$ and $b_k \ldots b_1 b_0$. And because $\boldsymbol{f}$ corresponds to the carries when adding two numbers, its coefficients are binary.

If we define a matrix $M = \begin{bmatrix} -2 & 0 & \ldots & 0 \\ 1 & -2 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & -2 \\ 0 & 0 & \ldots & 1 \end{bmatrix} \in \mathbb{Z}^{(k+1)\times k}$, then $M\vec{f}$ interpreted as a polynomial over $\mathbb{Z}[X]$ is equal to $\boldsymbol{f} \cdot (X-2)$. To prove that $\vec{a}, \vec{b}, \vec{c}$ are binary vectors and $\vec{a} + \vec{b} = \vec{c}$ as binary representations of integers, one would create a commitment $\mathsf{Com}(\vec{a}, \vec{b}, \vec{c}, \vec{f})$ and prove that

$$\begin{aligned} \vec{a} \circ (\vec{1} - \vec{a}) = 0 \wedge \vec{b} \circ (\vec{1} - \vec{b}) = 0 \wedge \vec{c} \circ (\vec{1} - \vec{c}) = 0 \\ \wedge\, \vec{f} \circ (\vec{1} - \vec{f}) = 0 \wedge \vec{a} + \vec{b} + M\vec{f} = \vec{c}. \end{aligned}$$

Because all the coefficients of $\vec{a}, \vec{b}, \vec{c}, \vec{f}$ are proved to be binary, the linear relation $\vec{a} + \vec{b} + M\vec{f} = \vec{c}$, which is proved over $\mathbb{Z}_q$, also holds over $\mathbb{Z}$. The work of [18] also proved correctness of addition by proving an equality equivalent to (1), but the underlying components (i.e. the commitment and the ZK proof) are different. In particular, the very efficient multiplicative and linear proofs that our abstract framework uses are not compatible with the commitments used in [18]. We conjecture that using our framework can improve the whole MatRiCT system of [18], but it would certainly involve a lot of non-trivial details; and we leave doing it as an open problem.

Multiplication of arbitrary-length integers. Suppose that $\vec{a}, \vec{b}, \vec{c}$ are the binary decompositions of a, b, c and $ab = c$. If we interpret the binary decomposition of an integer as a polynomial (like in Section 1.2), then we obtain an equality that is analogous to (1) – in particular

$$\boldsymbol{ab} + \boldsymbol{f} \cdot (X-2) = \boldsymbol{c}, \tag{2}$$

where $\boldsymbol{f}$ is again a polynomial corresponding to the carry vector when performing the schoolbook multiplication of $a_k \ldots a_1 a_0$ and $b_k \ldots b_1 b_0$. There are, however, two important differences between (1) and (2). The first is that our abstract framework does not natively support polynomial multiplication $\boldsymbol{ab}$. The second issue is that the polynomial $\boldsymbol{f}$ corresponding to the carry is no longer binary, but has coefficient in the range between 0 and k. The naive way to give a proof that the coefficients of $\vec{f}$ are between 0 and k is to prove that $\vec{f} \circ (\vec{1} - \vec{f}) \circ \ldots \circ (\vec{k} - \vec{f}) = 0$. Such a proof is unfortunately quite costly as it is $O(k)$ times larger than just one proof. A better approach is to rewrite all the coefficients of $\vec{f}$ in binary to create a vector $\vec{f'}$ and then prove that $\vec{f'} \circ (\vec{1} - \vec{f'}) = 0$. This is better because the length of $\vec{f'}$ is a $\log k$ factor larger than the length of $\vec{f}$, and so our proof will only be a factor of $O(\log k)$ larger. But since this is still a noticeable overhead in practice, we will aim for another way to resolve this issue that only increases the proof size by a small additive factor.

Let us first explain how the multiplication $\boldsymbol{ab}$ is handled. We certainly do not want to commit to the k multiplicative "shifts" of $\boldsymbol{b}$ multiplied by a_i involved in the schoolbook multiplication as that would again increase the proof by a factor of $O(k)$. Instead, we will make use of the fact that we can prove component-wise products and linear relations in order to prove the multiplication $\boldsymbol{ab}$ using their NTT representations.[7] Let $g(X)$ be a polynomial of degree $d > 2k$ such that $g(X) = (X - r_1) \cdot \ldots \cdot (X - r_d) \bmod q$ for $r_i \in \mathbb{Z}_q$. It is easy to set up an initial q so that such a polynomial (it could even be $g(X) = X^d - 1$) exists. Then define M to be the Vandermonde matrix

$$M = \begin{bmatrix} 1 & r_1 & r_1^2 & \ldots & r_1^{d-1} \\ 1 & r_2 & r_2^2 & \ldots & r_2^{d-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & r_d & r_d^2 & \ldots & r_d^{d-1} \end{bmatrix} \in \mathbb{Z}_q^{d \times d},$$

and furthermore define $M_k \in \mathbb{Z}_q^{d \times k}$ to be the first k columns of M. Then the NTT of the polynomial $\boldsymbol{a}$, defined as $(\boldsymbol{a}(r_1), \ldots, \boldsymbol{a}(r_d))$, can be written as $M_k \vec{a}$, when $\vec{a} = [a_0\ a_1\ \ldots\ a_k]^T$ (we use M_k instead of M because $\boldsymbol{a}$ only has degree k, and so the later columns of M don't affect the NTT). Using the ring-homomorphic properties of the NTT, if $\boldsymbol{ab} = \boldsymbol{c}$ in the ring $\mathbb{Z}_q[X]/(g(X))$, then we have that $(M_k\vec{a}) \circ (M_k\vec{b}) = M\vec{c} \bmod q$. But since the degree of $\boldsymbol{a}$ and $\boldsymbol{b}$ is less than $k < d/2$, the product $\boldsymbol{ab}$ does not get reduced modulo $g(X)$, and so $\boldsymbol{ab} = \boldsymbol{c}$ holds over $\mathbb{Z}_q[X]$ as well. Thus proving this equality using our abstract framework would involve committing to $\vec{a}, \vec{b}, \vec{c}$ and $\hat{a}, \hat{b}, \hat{c} \in \mathbb{Z}_q^d$, where

$$\hat{a} = M_k\vec{a},\ \hat{b} = M_k\vec{b},\ \hat{c} = M\vec{c}. \tag{3}$$

We can then use the product proof to show that $\hat{a} \circ \hat{b} = \hat{c}$, and then the unstructured linear proof to prove the relations in (3). Proving polynomial multiplication therefore requires just three extra commitments.

We now move on to showing how to prove that the coefficients of $\boldsymbol{f}$ in (2) are small. While we know that they are in the range between 0 and k, it is not really necessary for the prover to prove exactly this fact. All that is needed is to show that the coefficients of $\boldsymbol{f}$ are small enough so that (2) holds over $\mathbb{Z}[X]$ rather than $\mathbb{Z}_q[X]$. So rather than using a standard range proof, we will use a significantly more efficient "relaxed" range proof. The main idea for the proof is the fact[8] that for any vectors $\vec{f} \in \mathbb{Z}^k$ and $\vec{y} \in \mathbb{Z}^n$, the probability over the random choice of $R \in \{0,1\}^{n \times k}$ that the maximum coefficient of $R\vec{f} + \vec{y} \bmod q$ will be at least half of the largest coefficient of $\vec{f}$ is at least $1 - 2^{-n}$. To put it another way, if the coefficients of $R\vec{f} + \vec{y} \bmod q$ are small, then the coefficients of $\vec{f}$ must be small as well.

The above observation can be transformed into a zero-knowledge proof that $\vec{f}$ has small coefficients as follows (this idea was also recently independently used in [6]): the prover generates the masking vector $\vec{y}$, receives the challenge R from the verifier (or uses the Fiat-Shamir heuristic to generate R himself[9]) and creates the vector $\vec{z} = R\vec{f} + \vec{y} \bmod q$. He then applies the same type of rejection sampling technique used in Fiat-Shamir lattice signatures/commitment schemes to hide the dependence of $\vec{f}$ on $\vec{z}$. If the rejection sampling procedure passes, the prover outputs $\vec{z}$ (otherwise, the proof is restarted). Using the unstructured linear proof, he can also output a proof that $\vec{z} = R\vec{f} + \vec{y} \bmod q$. The verifier sees $R, \vec{z}$, and the commitment

[7] We point out that this NTT representation is not (necessarily) related to the NTT representation over the ring $\mathcal{R}_q$ discussed in Section 1.1. In particular, we now want an NTT over a ring whose degree depends on k, whereas the NTT in Section 1.1 is done over the fixed ring $\mathcal{R}_q$.

[8] Which is a slight generalization of [5, Lemma 2.3]

[9] To keep the size of the proof small, only a seed that generates R is included in the proof and R itself is generated as $R = \mathsf{hash}(\mathsf{seed})$ where hash is some public domain extension function (XOF) based on, for example, SHA-3.

$\mathsf{Com}(\vec{f}, \vec{y})$ and can check by himself that the coefficients of $\vec{z}$ are small, and then verify the unstructured linear proof. This proves that the coefficients of $\vec{f}$ are indeed at most twice the coefficients of $\vec{z}$. Because the $\vec{f}$ used by the honest prover has coefficients at most k, the ℓ_2-norm of the vector $R\vec{f}$ is $O(k^2\sqrt{n})$, and so the rejection sampling technique prescribes that the coefficients of $\vec{y}$ and of $\vec{z}$ also be $O(k^2\sqrt{n})$. For $n = 128$ and typical values of k (e.g. $k \leq 1024$), this value will be comfortably less than $q \approx 2^{32}$.

Amortized proofs. The "relaxed" range proof from above can also be used to amortize proofs that follow the idea in Section 1.2. Recall that the idea was to commit to the integers m_i as one-dimensional vectors and then prove that these integers are short via binary decomposition. Because the basic objects in the commitments and proofs in [2,15] are elements in the ring $\mathbb{Z}_q[X]/(X^{128}+1)$, a commitment to one element in $\mathbb{Z}_q$ is as expensive as a commitment to 128 elements (i.e. a commitment to an integer m is as expensive as a commitment to a 128-dimensional integer vector $\vec{m}$). But the reason that the proof in Section 1.2 does not stay equally short for messages $\vec{m}_i \in \mathbb{Z}^{128}$ (instead of $m_i \in \mathbb{Z}$) is because we also required the binary decomposition of each integer. When we only had one small integer, this wasn't an issue because this decomposition fit into one commitment, but with 128 integers, we are essentially increasing the vector length by a logarithmic factor.

The relaxed range proof gives us a different way to prove smallness of the elements in the vector $\vec{m}_i$. In particular, if one shows that $R\vec{m}_i + \vec{y}$ has small coefficients, then this implies that all the integers comprising $\vec{m}_i$ are small. But as in the previous section, the size of the coefficients that one can prove increases by a factor of $O(k^2\sqrt{n})$, where $k = n = 128$. Thus depending on the original sizes of the integers involved in the computation, one may need to increase the value of q in order to ensure that modular reduction does not occur, especially for multiplication where the factor of $O(k^2\sqrt{n})$ gets multiplied twice. Nevertheless, increasing q may be a good trade-off for being able to pack 128 proofs into one commitment and proof.

2 Preliminaries

2.1 Notation

Denote $\mathbb{Z}_p$ to be the ring of integers modulo p. Let q be an odd prime. We write $\vec{v} \in R^m$ to denote vectors over a ring R and matrices over R will be written as regular capital letters M. Define I_n to be the $n \times n$ identity matrix over $\mathbb{Z}_q$. Also, $\vec{0}_n$ and $0_{n,m}$ are the zero vector and the zero matrix in $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^{n\times m}$ respectively. For simplicity, we denote $0_n = 0_{n,n}$. By default, all vectors are column vectors. We write $\vec{v}\|\vec{w}$ for a usual concatenation of $\vec{v}$ and $\vec{w}$ (which is still a column vector). For $\vec{v}, \vec{w} \in \mathbb{Z}_q^k$, $\vec{v} \circ \vec{w}$ is the usual component-wise multiplication. 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. We write $[n]$ to denote the set $\{1, \ldots, n\}$ and also $[a, b] = \{x \in \mathbb{Z} : a \leq x \leq b\}$.

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

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

Sizes of elements. For an element $w \in \mathbb{Z}_q$, we write $\|w\|_\infty$ to mean $|w \bmod^{\pm} q|$. Define the ℓ_∞ and ℓ_2 norms for $\boldsymbol{w} = w_0 + w_1X + \ldots + w_{d-1}X^{d-1} \in \mathcal{R}$ as follows:

$$\|\boldsymbol{w}\|_\infty = \max_j \|w_j\|_\infty, \quad \|\boldsymbol{w}\| = \sqrt{\|w_0\|_\infty^2 + \ldots + \|w_{d-1}\|_\infty^2}.$$

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

$$\|\vec{\boldsymbol{w}}\|_\infty = \max_j \|\boldsymbol{w}_j\|_\infty, \quad \|\vec{\boldsymbol{w}}\| = \sqrt{\|\boldsymbol{w}_1\|^2 + \ldots + \|\boldsymbol{w}_k\|^2}.$$

For $\delta \in \mathbb{N}$, we denote $S_\delta = \{\boldsymbol{w} \in \mathcal{R} : \|\boldsymbol{w}\|_\infty \leq \gamma\}$.

Two's complement. An integer $w \in [-2^{N-1}, 2^{N-1} - 1]$ is represented in two's complement as a vector of N bits $(w_0, \ldots, w_{N-1}) \in \{0,1\}^N$ such that

$$w = -w_{N-1} 2^{N-1} + \sum_{i=0}^{N-2} w_i 2^i.$$

For readability, we define functions $\mathsf{TC}_N(w)$ and $\mathsf{sTC}_N(w)$ as:

$$\begin{aligned} &\mathsf{TC}_N(w) = (w_0, \ldots, w_{N-1}) \\ &\mathsf{sTC}_N(w) = (w_0, \ldots, w_{N-2}, -w_{N-1}). \end{aligned} \tag{4}$$

2.2 Cyclotomic Rings

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

$$X^d + 1 \equiv \prod_{j \in \mathbb{Z}_{2l}^\times} \left(X^{\frac{d}{l}} - \zeta^j \right) \pmod q. \tag{5}$$

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

Let $W_q := W_{d/l,q}$. We define the Number Theoretic Transform (NTT) of a polynomial $\boldsymbol{p} \in \mathcal{R}_q$ as follows:

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

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

2.3 Challenge Space

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

Consider the coefficients of the polynomial $\boldsymbol{c} \bmod (X^{d/l} - \zeta^j)$ for $\boldsymbol{c} \leftarrow C$. Clearly 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. [2] give an upper bound on the maximum probability of Y.

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

$$\Pr(Y = x) \leq \frac{1}{q} + \frac{2l}{q} \sum_{j \in \mathbb{Z}_q^\times / \langle \zeta \rangle} \prod_{i=0}^{l-1} \left| \frac{1}{2} + \frac{1}{2} \cos(2\pi j y \zeta^i / q) \right|. \tag{6}$$

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

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

2.4 Module-SIS and Module-LWE Problems

In our protocols, where the commitment scheme from [4] is used, the security is based on the well-known computational lattice problems, namely Module-LWE (M-LWE) and Module-SIS (M-SIS) [21]. Both problems are defined over $\mathcal{R}_q$.

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

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

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

$$\begin{aligned} \Big|\Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times\lambda};\ \vec{\boldsymbol{s}} \xleftarrow{\$} \chi^\lambda;\ \vec{\boldsymbol{e}} \xleftarrow{\$} \chi^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})\right] \\ - \Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times\lambda};\ \vec{\boldsymbol{b}} \xleftarrow{\$} \mathcal{R}_q^m;\ b \leftarrow \mathcal{A}(\boldsymbol{A}, \vec{\boldsymbol{b}})\right]\Big| \geq \epsilon. \end{aligned} \tag{7}$$

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

2.5 Probability Distributions

For sampling randomness in the commitment scheme that we use, and to define a variant of the Ring Learning with Errors problem, we need to define an error distribution χ^d on $\mathcal{R}$. In this paper we sample the coefficients of the random polynomials in the commitment scheme using the distribution χ on $\{-1, 0, 1\}$ where ± 1 both have probability $5/16$ and 0 has probability $6/16$. This distribution is chosen (rather than the more "natural" uniform one) because it is easy to sample given a random bitstring by computing $a_1 + a_2 - b_1 - b_2 \bmod 3$ with uniformly random bits a_i, b_i.

Rejection Sampling. In our zero-knowledge proof, the prover will want to output a vector $\vec{\boldsymbol{z}}$ whose distribution should be independent of a secret randomness vector $\vec{\boldsymbol{r}}$, so that $\vec{\boldsymbol{z}}$ cannot be used to gain any information on the prover's secret. During the protocol, the prover computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ where $\vec{\boldsymbol{r}}$ is the randomness used to commit to the prover's secret, $\boldsymbol{c} \xleftarrow{\$} C$ is a challenge polynomial, and $\vec{\boldsymbol{y}}$ is a "masking" vector. To remove the dependency of $\vec{\boldsymbol{z}}$ on $\vec{\boldsymbol{r}}$, we use the uniform rejection sampling technique as in [24,13].

The main advantages of having uniform rather than Gaussian sampling are twofold. Firstly, (i) it allows us to use standard compression techniques, i.e. chopping of low-order bits as in Dilithium [13] and (ii) it significantly decreases the protocol run-time due to simpler implementation.

Approximate Range Proofs. Baum and Lyubashevsky [5] showed that if $C\vec{s}$ has small coefficients, for a vector $\vec{s}$ over $\mathbb{Z}_q$ and uniformly random binary matrix C, then with high probability $\vec{s}$ must have small coefficients as well. We slightly generalise their result for $C\vec{s} + \vec{e}$ where $\vec{e}$ is an arbitrary vector over $\mathbb{Z}_q$. The proof follows almost identically as in [5].

Lemma 2.4. *Let $\vec{s} \in \mathbb{Z}_q^m$ and $\vec{e} \in \mathbb{Z}_q^n$. Then*

$$\Pr\left[\|C\vec{s} + \vec{e}\|_\infty < \frac{1}{2}\|\vec{s}\|_\infty : C \xleftarrow{\$} \{0,1\}^{n\times m}\right] \leq 2^{-n}.$$

2.6 Commitment Scheme

We recall the commitment scheme from [4] used in our constructions. Suppose that we want to commit to a message vector $\vec{\boldsymbol{m}} = (\boldsymbol{m}_1, \dots, \boldsymbol{m}_n) \in \mathcal{R}_q^n$ for $n \geq 1$ and that module ranks of κ and λ are required for M-SIS and M-LWE security, respectively. Then, in the key generation, a matrix $\boldsymbol{B}_0 \xleftarrow{\$} \mathcal{R}_q^{\kappa \times (\kappa+\lambda+n)}$ and vectors $\vec{\boldsymbol{b}}_1, \dots, \vec{\boldsymbol{b}}_n \xleftarrow{\$} \mathcal{R}_q^{\kappa+\lambda+n}$ are generated and output as public parameters. For optimization, some part of them can be set as the zero or identity matrices, but this is not important for our purposes.

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

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

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

3 Framework for Multiplicative and Linear Relations

We present a general framework for proving multiplicative and linear relations between committed messages using the techniques presented in [2,15]. Concretely, suppose that prover $\mathcal{P}$ has a vector of n messages $\vec{m} = (\vec{m}_1, \dots, \vec{m}_n) \in \mathbb{Z}_q^{nl}$. Then, $\mathcal{P}$ wants to prove the following:

1. $\forall P \in \mathsf{pp}, P(\vec{m}) = \vec{0}$ where each $P : \left(\mathbb{Z}_q^l\right)^n \to \mathbb{Z}_q^l$ in pp is a public polynomial of n variables over $\mathbb{Z}_q^l$,
2. $A\vec{m} = \vec{u}$ for $\mathsf{ulp} = (A, \vec{u}) \in \mathbb{Z}_q^{vl \times nl} \times \mathbb{Z}_q^{vl}$ [10].

For our applications, we will only consider polynomials $P \in \mathsf{pp}$ of total degree at most three. However, this can be extended to more general polynomials at the cost of larger number of garbage commitments.

We denote $\mathcal{L}_n(\mathsf{pp}, \mathsf{ulp}) \subseteq \mathbb{Z}_q^{nl}$ to be the language of messages $\vec{m}$, which satisfy all two properties above.

Proving knowledge of $\vec{m} \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$ consists of two parts. First, using the product proof argument from [2], we prove $P(\vec{m}) = \vec{0}$ for all $P \in \mathsf{pp}$. Then, we apply the unstructured linear proof from [15] to show that $A\vec{m} = \vec{u}$. Since an overwhelming majority of the techniques used is thoroughly described in previous work, we skip some technical details and refer them to [2,15]. For completeness, we sketch out main contributions of [2,15] in Appendix A.

3.1 Main Protocol

We briefly recall that we work over the cyclotomic ring $\mathcal{R}_q$ with dimension d which is a power-of-two. Prime q is selected so that polynomial $X^d + 1$ splits into polynomials of degree d/l as in (5). We will follow the "commit-and-prove" functionality [14,10]. Namely, our main protocol $\pi = (\mathsf{Com}(\vec{m}), \mathit{\Pi}_n^3(\mathsf{pp}, \mathsf{ulp}))$ can be split into two parts:

– $\mathsf{Com}_{n,3}(\vec{m})$: Prover $\mathcal{P}$ generates randomness $\vec{\boldsymbol{r}} \xleftarrow{\$} \chi^{(\lambda+\kappa+n+3)d}$ and outputs $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \dots, \boldsymbol{t}_{n+3})$ of the following form:

$$\begin{cases} \vec{\boldsymbol{t}}_0 &= \boldsymbol{B}_0 \vec{\boldsymbol{r}}, \\ \boldsymbol{t}_i &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}\left(\vec{m}_i\right) \text{ for } i \in [n]. \end{cases} \tag{8}$$

– A protocol $\mathit{\Pi}_n^3(\mathsf{pp}, \mathsf{ulp})$ defined in Figure 1 for proving that $\vec{m}$ belongs to $\mathcal{L}_{n,n'}(\mathsf{pp}, \mathsf{ulp})$. It combines protocols shown in Appendix A.3 and A.4. We assume the verifier $\mathcal{V}$ already has commtiments $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \dots, \boldsymbol{t}_{n+3}$.

[10] Actually, A can have arbitrary number of rows but then we would have to pad rows with zeroes in order to get a multiple of l.

Soundness error of this protocol is around $q^{-d/l}$ if we assume M-SIS is hard. If parameters are chosen such that $q^{-d/l}$ is not negligible then we describe in Appendix A how to increase soundness using Galois automorphisms.

We briefly describe the protocol from Figure 1. Prover $\mathcal{P}$ starts by sampling a vector of small polynomials $\vec{\boldsymbol{y}}$ and sending $\vec{\boldsymbol{w}} = \boldsymbol{B}\vec{\boldsymbol{y}}$. Furthermore, the prover samples a random polynomial $\boldsymbol{g}$ with the first d/l coefficients equal to 0. Then, it sends a commitment $\boldsymbol{t}_{n+1}$ to $\boldsymbol{g}$ as a part of the unstructured linear proof (ULP) as well as $\vec{\boldsymbol{w}}$.

Given challenges $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi \in \mathcal{R}_q$ from the verifier $\mathcal{V}$, prover $\mathcal{P}$ computes garbage commitments $\boldsymbol{t}_1', \boldsymbol{t}_2'$ along with a polynomial $\boldsymbol{v}$ as central components of the product proof (PP). The prover is also given a challenge $\vec{\gamma} \in W_q^{vl}$. Then, $\mathcal{P}$ calculates $\boldsymbol{w}', \boldsymbol{h}$ as a part of ULP. Next, the prover returns $(\boldsymbol{t}_1', \boldsymbol{t}_2', \boldsymbol{v}, \boldsymbol{w}', \boldsymbol{h})$.

In the final rounds, $\mathcal{V}$ samples a challenge polynomial from C and sends it to $\mathcal{P}$. The prover then computes $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and applies rejection sampling. If it does not abort, then $\mathcal{P}$ outputs $\vec{\boldsymbol{z}}$.

As far as the verification equations are concerned, the verifier checks that $\vec{\boldsymbol{z}}$ consists of only small polynomials and $\boldsymbol{B}\vec{\boldsymbol{z}}_0 \stackrel{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$. Then, Lines 03-06 and 07-10 in Figure 2 correspond to PP and ULP respectively.

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

Inputs:

$\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa\times(\lambda+\kappa+n+3)}; \vec{\boldsymbol{b}}_1 \ldots, \vec{\boldsymbol{b}}_{n+3} \in \mathcal{R}_q^{\lambda+\kappa+n+3}$ | $\boldsymbol{B}_0; \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{n+3}$

$\vec{\boldsymbol{r}} \in \{-1,0,1\}^{(\lambda+\kappa+n+3)d} \subset \mathcal{R}_q^{\lambda+\kappa+n+3}$ | $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_n$

$\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}}$

$\boldsymbol{m}_i = \mathsf{NTT}^{-1}(\vec{m}_i), \boldsymbol{t}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \boldsymbol{m}_i$ for $i \in [n]$

$\vec{\boldsymbol{y}} \xleftarrow{\$} S_{\delta_1-1}^{(\lambda+\kappa+n+3)}$ and set $\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$

$\boldsymbol{g} \xleftarrow{\$} \{\boldsymbol{f} \in \mathcal{R}_q : f_0 = \ldots = f_{d/l-1} = 0\}$

$\boldsymbol{t}_{n+1} = \langle \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}$ $\quad \xrightarrow{\boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}}} \quad$ $\vec{\gamma} = (\vec{\gamma}_1, \ldots, \vec{\gamma}_v) \xleftarrow{\$} W_q^{vl}$

$\xleftarrow{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi, \vec{\gamma}} \quad$ $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi \xleftarrow{\$} \mathcal{R}_q$

For $t = 1, \ldots, \xi$:

$$\boldsymbol{\psi}_{t,0} := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell} \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle$$

$$\boldsymbol{\psi}_{t,1} := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell} \left(\langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_\ell + \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_j + \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i \right) + \sum_{i\le j} \boldsymbol{\eta}_{t,i,j} \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle$$

$$\boldsymbol{\psi}_{t,2} := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell} \left(\langle \vec{\boldsymbol{b}}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i\boldsymbol{m}_j + \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i\boldsymbol{m}_\ell + \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_j\boldsymbol{m}_\ell \right) + \sum_{i\le j} \boldsymbol{\eta}_{t,i,j} \left(\boldsymbol{m}_i \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}} \rangle + \boldsymbol{m}_j \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle \right) + \sum_{i=1}^{n} \boldsymbol{\nu}_{t,i} \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle$$

$$\boldsymbol{t}_1' = \langle \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{r}} \rangle - \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,2},\ \boldsymbol{t}_2' = \langle \vec{\boldsymbol{b}}_{n+2}, \vec{\boldsymbol{r}} \rangle - \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,1} + \langle \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{y}} \rangle, \boldsymbol{v} = \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,0} - \langle \vec{\boldsymbol{b}}_{n+2}, \vec{\boldsymbol{y}} \rangle$$

For $j \in [n] : \boldsymbol{p}_j = \mathsf{NTT}^{-1}\left(\sum_{i=1}^{v} A_{i,j}^T \vec{\gamma}_i \right)$

$$\boldsymbol{w}' = \langle \sum_{j=1}^{n} \boldsymbol{p}_j \vec{\boldsymbol{b}}_j + \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{y}} \rangle, \boldsymbol{h} = \sum_{j=1}^{n} \boldsymbol{m}_j\boldsymbol{p}_j - \frac{\langle \vec{u}, \vec{\gamma} \rangle}{l} + \boldsymbol{g} \quad \xrightarrow{\boldsymbol{t}_1', \boldsymbol{t}_2', \boldsymbol{v}, \boldsymbol{w}', \boldsymbol{h}}$$

$\xleftarrow{\boldsymbol{c}} \quad$ $\boldsymbol{c} \xleftarrow{\$} C$

$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$

If $\|\vec{\boldsymbol{z}}\|_\infty \ge \delta_1 - \beta_1$, abort $\quad \xrightarrow{\vec{\boldsymbol{z}}} \quad$ $\mathsf{Ver}(\boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}}, \boldsymbol{\alpha}_i, \vec{\gamma}, \boldsymbol{t}_1', \boldsymbol{t}_2', \boldsymbol{v}, \boldsymbol{w}', \boldsymbol{h}, \boldsymbol{c}, \vec{\boldsymbol{z}})$

Fig. 1. Protocol $\Pi_n^3(\mathsf{pp}, \mathsf{ulp})$ for proving $\vec{m} = (\vec{m}_1, \ldots, \vec{m}_n) \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$. We denote polynomials in $\mathsf{pp} = \{P_1, \ldots, P_\xi\}$ as in (33) and $\mathsf{ulp} = (A, \vec{u})$. Also, we partition the matrix A as in (43). Verification equations Ver are defined in Figure 2. Parameters δ_1, β_1 are used for uniform rejection sampling.

```
$\mathsf{Ver}(\boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}}, \boldsymbol{\alpha}_i, \vec{\gamma}, \boldsymbol{t}'_1, \boldsymbol{t}'_2, \boldsymbol{v}, \boldsymbol{w}', \boldsymbol{h}, \boldsymbol{c}, \vec{\boldsymbol{z}})$
$\|\vec{\boldsymbol{z}}\|_\infty \overset{?}{<} \delta_1 - \beta_1$
$\boldsymbol{B}_0\vec{\boldsymbol{z}} \overset{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$
For $i \in [n]$ and $j \in [2]$ :
    $\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_i, \boldsymbol{f}'_j = \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{z}} \rangle - \boldsymbol{t}'_j$
$-\sum_{t=1}^{\xi} \sum_{i \le j \le \ell} \boldsymbol{\alpha}_t \boldsymbol{\mu}_{t,i,j,\ell} \boldsymbol{f}_i \boldsymbol{f}_j \boldsymbol{f}_\ell + \boldsymbol{c} \sum_{t=1}^{\xi} \sum_{i \le j} \boldsymbol{\alpha}_t \boldsymbol{\eta}_{t,i,j} \boldsymbol{f}_i \boldsymbol{f}_j$
    $-\boldsymbol{c}^2 \sum_{t=1}^{\xi} \sum_{i=1}^{n} \boldsymbol{\alpha}_t \boldsymbol{\nu}_{t,i} \boldsymbol{f}_i + \sum_{t=1}^{\xi} \boldsymbol{c}^3 \boldsymbol{\rho}_t \overset{?}{=} -\boldsymbol{c}^2 \boldsymbol{f}'_1 + \boldsymbol{c}\boldsymbol{f}'_2 - \boldsymbol{v}$
For $i = 0, \ldots, d/l - 1 : h_i \overset{?}{=} 0$
For $j \in [n] : \boldsymbol{p}_j = \mathsf{NTT}^{-1}\left(\sum_{i=1}^{v} A_{i,j}^T \vec{\gamma}_i\right)$
$\boldsymbol{t}_f = \sum_{i=1}^{n} \boldsymbol{t}_i \boldsymbol{p}_i - \frac{\langle \vec{u}, \vec{\gamma} \rangle}{l}$
$\langle \sum_{i=1}^{n} \boldsymbol{p}_i \vec{\boldsymbol{b}}_i + \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{z}} \rangle \overset{?}{=} \boldsymbol{w}' + \boldsymbol{c}(\boldsymbol{t}_f + \boldsymbol{t}_{n+1} - \boldsymbol{h})$.
```

Fig. 2. Verification equations for Figure 1.

Security Analysis. The protocol $\pi = (\mathsf{Com}(\vec{m}), \Pi_n^3(\mathsf{pp}, \mathsf{ulp}))$ for proving $\vec{m} \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$ is an honest-verifier zero-knowledge proof of knowledge under the hardness of M-SIS and M-LWE. Indeed, correctness and zero-knowledge follow identically as in [2, Theorem 5.1] and [15, Theorem 4.1].

For soundness, we describe the main difference from the previous works. Define an extractor $\mathcal{E}$ as in [15, Theorem 4.1]. Unless $\mathcal{E}$ finds a $\mathsf{MSIS}_{\kappa,8d\beta}$ solution, where $\beta = \delta_1 - \beta_1 - 1$, it computes vectors $\vec{\boldsymbol{y}}^*$ and $\vec{\boldsymbol{r}}^*$ such that for every accepting transcript with fixed first message $(\boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}})$, $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}}^* + \boldsymbol{c}\vec{\boldsymbol{r}}^*$. The next step would be to set messages $\boldsymbol{m}_i^* \in \mathcal{R}_q$ which satisfy:

$$\boldsymbol{t}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}_i^* \text{ for } i \in [n].$$

Let $\vec{m}_i^* = \mathsf{NTT}(\boldsymbol{m}_i^*) \in W_q^l$. Recall that we additionally need to prove $\vec{m}_1^*, \ldots, \vec{m}_n^* \in \mathbb{Z}_q^l$. If $\mathcal{R}_q$ splits completely, i.e. $l = d$, then $W_q = \mathbb{Z}_q$ so we are done. Otherwise, an additional argument is needed. This will depend on various applications (e.g. Sections 4 and 5) and we leave this out of scope of the proof here. Lastly, proving that $\vec{m}_1^*, \ldots, \vec{m}_n^*$ satisfy relations w.r.t. $(\mathsf{pp}, \mathsf{ulp})$ follows identically as in the proof of [15, Theorem 4.1].

Proof size. We look at the size of the non-interactive proof outputs created by the protocol π. We observe that for the non-interactive proof $\boldsymbol{w}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ need not to be included in the output as they are uniquely determined by the remaining components. Additionally, the challenges can be generated from a small seed of 256 bits, which itself is generated as the hash of some components. Therefore, the contribution of the challenges to the total proof length is extremely small and thus we neglect it.

As "full-sized" elements of $\mathcal{R}_q$, we have $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{n+1}, \boldsymbol{t}'_1, \boldsymbol{t}'_2$, and $\boldsymbol{h}$ (in fact, $\boldsymbol{h}$ is missing d/l coefficients, but that is a negligible consideration). Therefore, we have in total $n + \kappa + 4$ full-sized elements of $\mathcal{R}_q$, which altogether costs $(n + \kappa + 4)d \log q$ bits.

The only remaining part is $\vec{\boldsymbol{z}}_i$'s. Due to rejection sampling, each coefficient of $\vec{\boldsymbol{z}}_i$ can be bounded in absolute value by δ_1. If we then take into account the total number of coefficients in $\vec{\boldsymbol{z}}_i$ and an additional sign bit for each coefficient, then we get

$$(\lambda + \kappa + n + 3) \cdot d \cdot \log(2\delta_1)$$

bits of communication required for all $\vec{\boldsymbol{z}}_i$'s together. Also, we find a bound β_1 on the infinity norm of $\boldsymbol{c}\vec{\boldsymbol{r}}$. It is easy to see that no coefficient of the product $\boldsymbol{c}\vec{\boldsymbol{r}}$ can exceed d. Thus, we can set $\beta_1 = d$. Hence, in order to obtain the expected number of repetitions M for the rejection sampling, one would choose δ_1 which satisfies:

$$1/M = \left(\frac{2(\delta_1 - \beta_1) - 1}{2\delta_1 - 1}\right)^{(\lambda+\kappa+n+3)d} \approx e^{-(\lambda+\kappa+n+3)d\beta_1/\delta_1}.$$

In conclusion, the overall proof length is about

$$(n+\kappa+4)d\log q \;+\; (\lambda+\kappa+n+3)\cdot d\cdot\log(2\delta_1) \quad \text{bits.} \tag{9}$$

We can apply well-known proof length optimizations such as the standard compression techniques as in [13] or bounding $\beta_1 = \|\boldsymbol{c}\vec{\boldsymbol{r}}\|_\infty$ more cleverly. We provide more details in Appendix A.5.

Commitment generation inside $\boldsymbol{\pi}$. We remark that the stand-alone protocol in Fig. 1, i.e. without $\mathsf{Com}_{n,3}(\vec{m})$, is not honest-verifier zero-knowledge. The reason is that the randomness vector $\vec{\boldsymbol{r}}$ is fixed before the protocol begins and thus the argument that $\boldsymbol{t}_{n+1}, \boldsymbol{t}'_1, \boldsymbol{t}'_2$ are indistinguishable from random is not valid. In order to make the protocol in Fig. 1 zero-knowledge, one would generate additional randomness vector $\vec{\boldsymbol{r}}'$ and send $\vec{\boldsymbol{\tau}}_0 = \boldsymbol{B}'_0\vec{\boldsymbol{r}}'$ where $\boldsymbol{B}'_0$ is a public matrix. Additionally, one would sample $\vec{\boldsymbol{y}}'$ similarly as $\vec{\boldsymbol{y}}'$ and output $\vec{\boldsymbol{w}}' = \boldsymbol{B}'_0\vec{\boldsymbol{y}}'$. Then, the prover would run the protocol in Fig. 1, i.e. the product and linear proof parts, but with $\vec{\boldsymbol{r}}'$ and $\vec{\boldsymbol{y}}'$ instead of $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{y}}$ respectively. At the end, $\mathcal{P}$ sends masked openings to both $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{r}}'$:

$$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}} \text{ and } \vec{\boldsymbol{z}}' = \vec{\boldsymbol{y}}' + \boldsymbol{c}'\vec{\boldsymbol{r}}'$$

for challenges $\boldsymbol{c}, \boldsymbol{c}' \overset{\$}{\leftarrow} C$. With this approach, $\boldsymbol{t}_{n+1}, \boldsymbol{t}'_1, \boldsymbol{t}'_2$ are indistinguishable from random under the M-LWE assumption since $\vec{\boldsymbol{r}}'$ was indeed generated by the prover. However, introducing additional randomness significantly increases the proof size.

3.2 Polynomials of Total Degree 2

Recall that our framework allows to include in pp multivariate polynomials of degree at most three. However, it is easy to see that when pp contains no polynomials of degree three, then only one garbage commitment $\boldsymbol{t}'_1$ is enough to prove multiplicative relations [2] (by reasoning identically as in Section A.3). Hence, the total proof size is now equal to

$$(n+\kappa+3)d\log q \;+\; (\lambda+\kappa+n+2)\cdot d\cdot\log(2\delta_1) \quad \text{bits,} \tag{10}$$

for $\mathfrak{s}$ defined as in (9). We will denote this modified protocol as $\pi = (\mathsf{Com}_{n,2}(\vec{m}), \Pi_n^2(\mathsf{pp}, \mathsf{ulp}))$.

4 Integer Addition

In this section we provide an efficient zero-knowledge proof for integer addition for committed messages. Specifically, given commitments to integers a, b, c (depending on the application, some of these values can be given out in the clear), we want to prove that $a + b = c$. In order to consider both positive and negative values, we use the two's complement representation (see Section 2.1).

4.1 Two's Complement

Suppose $a, b, c \in [-2^{N-1}, 2^{N-1} - 1]$ and we want to prove $a + b = c$. Then, a can be represented in two's complement as N bits $a_0, \ldots, a_{N-1} \in \{0, 1\}$ which satisfy $a = -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i$. Similarly, we write

$b = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$ and $c = -c_{N-1}2^{N-1} + \sum_{i=0}^{N-2} c_i 2^i$. Then, by the two's complement addition algorithm we obtain the following system of equations:

$$\begin{cases} f_0 + \quad a_0 + \quad b_0 = \quad c_0 + 2f_1 \\ f_1 + \quad a_1 + \quad b_1 = \quad c_1 + 2f_2 \\ \quad\vdots \\ f_{N-2} + a_{N-2} + b_{N-2} = c_{N-2} + 2f_{N-1} \\ f_{N-1} + a_{N-1} + b_{N-1} = c_{N-1} + 2f_{N-1}. \end{cases}$$

for a carry vector $\vec{f} = (f_0, \ldots, f_{N-1}) \in \{0,1\}^N$ so that $f_0 = 0$. We remark that two's complement addition implies $a + b = c$ only if the last two carry bits are identical (i.e. f_{N-1}).

Since all values in the system of equations above are small, we can equivalently consider it modulo q and we rewrite it in a vector notation:

$$\vec{f} + \vec{a} + \vec{b} \equiv \vec{c} + 2J\vec{f} \pmod{q} \tag{11}$$

where $\vec{a} = \mathsf{TC}_N(a), \vec{b} = \mathsf{TC}_N(b), \vec{c} = \mathsf{TC}_N(c)$ and

$$J = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{Z}_q^{N \times N}. \tag{12}$$

One observes that the matrix $I_N - 2J$ is upper-triangular and $\det(I_N - 2J) = -1$. Thus, it is also invertible.

Now we transform the problem of proving addition in terms of our framework.

Small integers. Let us assume that $N = l$, i.e. we are interested in proving addition of small l-bit numbers. First, we prove that $\vec{m} = (\vec{a}, \vec{b}, \vec{c}, \vec{f})$ is a binary vector and $d_0 = 0$. These conditions can be captured by defining a set of pp multivariate quadratic polynomials as follows. Define

$$F_a(\vec{m}) = \vec{a} \circ \vec{a} - \vec{a}$$

and observe that $F_a(\vec{m}) = \vec{0}$ if and only if $\vec{a}$ binary values (similarly for F_b, F_c). Note that we also need to ensure that $f_0 = 0$, i.e. $\vec{f}$ is a well-formed carry vector. This can be done by defining a slightly different polynomial

$$F_f(\vec{m}) = \vec{f} \circ \vec{f} - \vec{\chi} \circ \vec{f},$$

where $\vec{\chi} := (0, 1, \ldots, 1) \in \mathbb{Z}_q^l$. Then, $\vec{f} \in \{0,1\}^l$ and $f_0 = 0$ is equivalent to $F_f(\vec{m}) = \vec{0}$. Finally, set

$$\mathsf{pp} = \{F_a, F_b, F_c, F_f\} \tag{13}$$

On the other hand, Equation 11 is equivalent to $A\vec{m} = \vec{0}$ where

$$A := \begin{pmatrix} I_N & I_N & -I_N & (I_N - 2J) \end{pmatrix} \in \mathbb{Z}_q^{N \times 4N}. \tag{14}$$

Then set $\mathsf{ulp} = (A, \vec{0})$. Eventually, we transform the problem of proving integer addition into an instance of our framework, i.e.

$$a, b, c \in [-2^{N-1}, 2^{N-1} - 1] \wedge \; a + b = c \Leftrightarrow \vec{m} \in \mathcal{L}_4(\mathsf{pp}, \mathsf{ulp}). \tag{15}$$

We provide a simple protocol for proving addition of small integers in Figure 3.

Prover $\mathcal{P}$	Verifier $\mathcal{V}$
Inputs:	
$a, b, c \in [-2^{N-1}, 2^{N-1}-1]$ such that $a+b=c$;	$\boldsymbol{B}_0; \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_6$
$\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa \times (\lambda+\kappa+6)}; \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_6 \in \mathcal{R}_q^{\lambda+\kappa+6}$	
$\vec{a} = \mathsf{TC}_l(a), \vec{b} = \mathsf{TC}_l(b), \vec{c} = \mathsf{TC}_l(c)$	
$\vec{f} = (I-2J)^{-1}(\vec{c} - \vec{a} - \vec{b}) \in \mathbb{Z}_q^l$	
Run $\mathsf{Com}_{4,2}(\vec{a}\|\|\vec{b}\|\|\vec{c}\|\|\vec{f})$	
Run $\mathit{\Pi}_4^2(\mathsf{pp}, \mathsf{ulp})$	

Fig. 3. Simple proof of integer addition where $N = l$. Here, pp and ulp are defined in (13) and (14) respectively. We also use the two's complement decomposition TC_l as defined in (4). Protocols $\mathsf{Com}_{n,2}(\vec{m})$ and $\mathit{\Pi}_4^2(\mathsf{pp}, \mathsf{ulp})$ are described in Section 3.2 for $n = 4$.

Security Analysis. Correctness and zero-knowledge of the protocol follow directly from the discussion above and previous works [15, Theorem 4.1].

Let us focus on soundness of our protocol. From Section 3.1 we know that there is an efficient extractor $\mathcal{E}$ which obtains

$$\vec{m}^* = (\vec{a}^*, \vec{b}^*, \vec{c}^*, \vec{f}^*) \in \mathcal{L}_4(\mathsf{pp}, \mathsf{ulp}).$$

Indeed, the way we defined pp in (13) assures that messages $\vec{a}^*, \vec{b}^*, \vec{c}^*$ and $\vec{f}^*$ are binary and thus, over $\mathbb{Z}_q$ as well. Then, $\mathcal{E}$ can simply compute $a^* = -a^*_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i^* 2^i$ and similarly for b^*, c^*. Hence, by (15) we have $a^* + b^* = c^*$.

Large integers. Suppose that one is interested in proving addition of large N-bit integers, i.e. $N = \gamma l$ for some $\gamma > 1$. The approach is almost identical as before. However, since $N > l$, we need to partition the vector $\vec{a} = (\vec{a}_0, \ldots, \vec{a}_{\gamma-1})$ where each $\vec{a}_i \in \mathbb{Z}_q^l$ (similarly for $\vec{b}, \vec{c}, \vec{f}$). Then, we send commitments $\boldsymbol{t}_{a,i}, \ldots, \boldsymbol{t}_{f,i}$ to $\vec{a}_i, \ldots, \vec{f}_i$ respectively where $i = 0, \ldots, \gamma - 1$.

In order to prove that $\vec{m} = (\vec{a}, \vec{b}, \vec{c}, \vec{f}) \in \mathbb{Z}_q^{4\gamma l}$ is a binary vector, we need to define multivariate polynomials $F_{a,i}(\vec{m}) = \vec{a}_i \circ \vec{a}_i - \vec{a}_i$ as well as $F_{b,i}, F_{c,i}$ for $i \in [\gamma]$. Recall that we still need to show $f_0 = 0$. Hence, we define $F_{f,0}(\vec{m}) = \vec{f}_0 \circ \vec{f}_0 - \vec{\chi} \circ \vec{f}_0$ and $F_{f,i}(\vec{m}) = \vec{f}_i \circ \vec{f}_i - \vec{f}_i$ for $i > 0$. Finally, set

$$\mathsf{pp} = \{(F_{a,i}), (F_{b,i}), (F_{c,i}), (F_{f,i}) : i = 0, \ldots, \gamma - 1\}.$$

We still define the matrix A as in (14) and set $\mathsf{ulp} = (A, \vec{0})$. To sum up, we reduced the problem of proving addition of N-bit integers to an instance of our framework:

$$a, b, c \in [-2^{N-1}, 2^{N-1}-1] \wedge\ a + b = c \Leftrightarrow \vec{m} \in \mathcal{L}_{4\gamma}(\mathsf{pp}, \mathsf{ulp}). \tag{16}$$

Proof size. For small values $N \leq l$, we will run the protocol from Figure 1. Then, by taking calculations from Section 3.2, we obtain the following total proof size:

$$(7 + \kappa) d \log q \ +\ (\lambda + \kappa + 6) \cdot d \cdot \log(2\delta_1) \quad \text{bits.} \tag{17}$$

For proving addition of large integers, we apply the protocol from Figure 8 which uses automorphisms. Suppose that $N = \gamma l$ some γ. Then, we know from Appendix A.5 and Section 3.2 that the proof size of $\mathit{\Pi}_{4\gamma}^2(\mathsf{pp}, \mathsf{ulp})$ is $(2d/l + 1) d \log q \ +\ k(\lambda + \kappa + 4\gamma + d/l + 1) d \cdot \log(2\delta_1)$ bits. Hence, by including sizes of the commitments $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_{a,i}, \ldots, \boldsymbol{t}_{f,i}$ sent by the prover in $\mathsf{Com}_{4\gamma,2}(\vec{m})$, the total proof size can be bounded by:

$$(2d/l + 4\gamma + \kappa + 1) d \log q \ +\ k(\lambda + \kappa + 4\gamma + d/l + 1) d \cdot \log(2\delta_1) \ \text{ bits.}$$

N	l	k	κ	λ	δ_1	δ_2	D	proof size
32	32	1	11	10	2^{17}	$(q-1)/2^{14}$	11	11.0KB
128	128	4	10	10	2^{18}	$(q-1)/2^{13}$	14	24.8KB
512	128	4	10	10	2^{18}	$(q-1)/2^{13}$	14	44.7KB

Fig. 4. Proof size comparison for proving integer addition $a+b=c$ for $a,b,c \in [-2^{N-1}, 2^{N-1}-1]$. In each scenario, we pick $q \approx 2^{30}$ and $d = 128$. Here, parameters δ_1, δ_2, D are used for the commitment compression (see Section A.5).

Table 4 illustrates proof sizes for concrete parameters. We remark that we already include standard optimization techniques described in Appendix A.5 and A.6. As in prior works (cf. [17,16,8,15,2]), we measure the hardness of these problems in terms of root Hermite factor δ. Concretely, we aim for $\delta \approx 1.0045$ in order to obtain around 128-bit security. For each instantiation in Figure 4 we select $q \approx 2^{30}$ and appropriate parameters λ, κ such that the root Hermite factor (for both SIS and LWE) $\delta \approx 1.0045$.

In Appendix B we describe how to apply our protocol in the context of range proofs.

5 Integer Multiplication

We present how to prove knowledge of integers a, b, c such that $ab = c$. Unlike in [22], we do not follow the schoolbook or Karatsuba algorithms. Instead, we make use of the properties of FFT multiplication. Concretely, let us write $a, b \in [-2^{N-1}, 2^{N-1}-1]$ and $c \in [-2^{2N-1}, 2^{2N-1}-1]$ in two's complement representation, i.e. $a = -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i, b = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$ and $c = -c_N 2^N + \sum_{i=0}^{N-1} c_i 2^i$. Now, define

$$\boldsymbol{a}(X) = a_0 + a_1 X + \cdots + a_{N-2}X^{N-2} - a_{N-1}X^{N-1} \in \mathbb{Z}[X]$$

and similarly for $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{Z}[X]$. Clearly, the coefficient vector of $\boldsymbol{a}$ is $\mathsf{sTC}_N(a)$. Now, observe that $\boldsymbol{a}(2)\boldsymbol{b}(2) - \boldsymbol{c}(2) = 0$. Hence, there exists a polynomial $\boldsymbol{f}$ of degree at most $2(N-1)-1$ which satisfies:

$$\boldsymbol{a}(X)\boldsymbol{b}(X) - \boldsymbol{c}(X) = (X-2)\boldsymbol{f}(X). \tag{18}$$

Lemma C.1 says that $\|\boldsymbol{f}\|_\infty \leq N+1$. Our goal will be to prove (18).

Let $R_{q,2l} = \mathbb{Z}_q[X]/(X^{2l}-1)$. Note that each $\zeta^i \in \mathbb{Z}_q$ is a root of the polynomial $X^{2l}-1$ modulo q since ζ was defined in Section 2 as the primitive $2l$-th root of unity. Therefore, the ring $R_{q,2l}$ splits completely. One can define the NTT operation over $R_{q,2l}$ as follows. Concretely, for a polynomial $\boldsymbol{f} \in R_{q,2l}$ with a coefficient vector $\vec{f}$, the NTT of $\boldsymbol{f}$ is the vector $\vec{f'} \in \mathbb{Z}_q^{2l}$ such that $V\vec{f} = \vec{f'}$ where V is the Vandermonde matrix

$$V = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \zeta & \cdots & \zeta^{2l-1} \\ 1 & \zeta^2 & \cdots & \zeta^{2(2l-1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \zeta^{2l-1} & \cdots & \zeta^{(2l-1)(2l-1)} \end{pmatrix} \in \mathbb{Z}_q^{2l \times 2l}.$$

We will use the homomorphic property of NTTs, i.e. for any two polynomials $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h} \in R_{q,2l}$ with coefficient vectors $\vec{f}, \vec{g}, \vec{h}$ respectively, we have

$$\boldsymbol{h} = \boldsymbol{f}\boldsymbol{g} \iff V\vec{h} = (V\vec{f}) \circ (V\vec{g}).$$

Clearly, V is invertible. Also, let us partition $V = (V_1 || V_2)$ where each $V_i \in \mathbb{Z}_q^{2l \times l}$.

Suppose that $l = N$. The key idea to prove (18) is to treat $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{f}$ as polynomials in $R_{q,2l}$ and equivalently consider (18) over $R_{q,2l}$ as long as N is much smaller than q. In order to make sure there is no modular reduction, we have to later prove that $\boldsymbol{f}$ has relatively small coefficients. Observe that proving (18) over $R_{q,l}$

is equivalent to the same equation over $\mathbb{Z}_q[X]$ as long as we make sure that polynomials $\boldsymbol{a}, \boldsymbol{b}$ have degrees less then l, i.e. no overflow modulo $X^{2l} - 1$ occurs.

Let $\vec{f} = (\vec{f_1}||\vec{f_2}) \in \mathbb{Z}_q^{2l}$ be the coefficient vector of $\boldsymbol{f}$. At the beginning, prover $\mathcal{P}$ generates vectors $\vec{e}_1, \ldots, \vec{e}_{d/l} \xleftarrow{\$} S^l_{\delta_3 - 1}$ from a uniform distribution. Then, $\mathcal{P}$ sends commitments to $\vec{f_i}, \vec{e_i}$. The verifier generates a uniformly random matrix $B = (B_1||B_2)$, where each $B_i \in \{0,1\}^{d \times l}$ and sends to $\mathcal{P}$. Next, the prover applies rejection sampling on

$$\vec{g} = B\vec{f} + \vec{e} = B_1\vec{f_1} + B_2\vec{f_2} + \vec{e} \in \mathbb{Z}_q^d, \tag{19}$$

where $\vec{e} = (\vec{e}_1|| \ldots ||\vec{e}_{d/l})$, and outputs $\vec{g}$. Eventually, $\mathcal{V}$ checks whether $\vec{g}$ is small. Furthermore, since B and $\vec{f}'$ are known to the verifier, proving $\vec{g} = B\vec{f} + \vec{e}$ is an instance of an unstructured linear proof (Appendix A.4). Eventually, by Lemma 2.4, this implies that $\vec{f}$ is relatively small as well.

Going back to our integers a, b, c, denote $\vec{a} = \mathsf{sTC}_l(a)$ (see (4)) and $(\vec{a}'_1||\vec{a}'_2) = \vec{a}' = V_1\vec{a} \in \mathbb{Z}_q^{2l}$ where each $\vec{a}'_i \in \mathbb{Z}_q^l$ (and the same for $\vec{b}$). Also, set $(\vec{c}_1||\vec{c}_2) = \mathsf{sTC}_{2l}(c), (\vec{c}'_1||\vec{c}'_2) = V\vec{c}$ and $(\vec{f}'_1||\vec{f}'_2) = V\vec{f}$. Furthermore, we let

$$\vec{m} = (\vec{a}, \vec{a}'_1, \vec{a}'_2, \vec{b}, \vec{b}'_1, \vec{b}'_2, \vec{c}_1, \vec{c}_2, \vec{c}'_1, \vec{c}'_2, \vec{f}_1, \vec{f}_2, \vec{f}'_1, \vec{f}'_2, \vec{e}_1, \ldots, \vec{e}_{d/l}).$$

Note that $\vec{m} \in \mathbb{Z}_q^{(14+d/l)l}$. Then, (18) is equivalent to $G_1(\vec{m}) = G_2(\vec{m}) = \vec{0}$ where

$$G_i(\vec{m}) := \vec{a}'_i \circ \vec{b}'_i - \vec{c}'_i - \vec{\gamma}_i \circ \vec{f}'_i \text{ for } i = 1, 2$$

and $(\vec{\gamma}_1||\vec{\gamma}_2) = (1-2, \zeta - 2, \zeta^2 - 2, \ldots, \zeta^{2l-1} - 2)$ is the NTT of $X - 2$.

We need to show that $\vec{a}, \vec{b}, \vec{c}_2 \in \{0,1\}^{l-1} \times \{-1, 0\}$ and $\vec{c}_1$ is binary. We do that by defining $F_a(\vec{m}) = \vec{a} \circ \vec{a} - \vec{\chi} \circ \vec{a}$ where $\vec{\chi} = (1, \ldots, 1, -1) \in \mathbb{Z}_q^l$ (and similarly for F_b, F_{c_1}, F_{c_2}). Then, we set:

$$\mathsf{pp} := \{G_1, G_2, F_a, F_b, F_{c_1}, F_{c_2}\}. \tag{20}$$

Next, we have to prove the relation between $\vec{a}$ and $\vec{a}'$, i.e. $\vec{a}' = V_1\vec{a}$ and similarly for $\vec{b}, \vec{c}$. For $\vec{f}$, however, we will show the inverse i.e. $V^{-1}\vec{f}' = \vec{f}$. This will be crucial for the soundness argument. We can combine all these linear relations between polynomial and NTT coefficients above, along with (19) into one equation $A\vec{m} = \vec{u}$ where $A \in \mathbb{Z}_q^{(5+d/l)l \times (14+d/l)l}$ and $\vec{u} \in \mathbb{Z}_q^{(5+d/l)l}$ are defined as

$$A = \begin{pmatrix} V_1 & -I_{2l} & 0_l & 0_{2l} & 0_{2l} & 0_{2l} & 0_{2l} & 0_{2l} & 0_{2l,d} \\ 0_{2l,l} & 0_{2l} & V_1 & -I_{2l} & 0_{2l} & 0_{2l} & 0_{2l} & 0_{2l} & 0_{2l,d} \\ 0_{2l,l} & 0_{2l} & 0_{2l,l} & 0_{2l} & V & -I_{2l} & 0_{2l} & 0_{2l} & 0_{2l,d} \\ 0_{2l,l} & 0_{2l} & 0_{2l,l} & 0_{2l} & 0_{2l} & 0_{2l} & -I_{2l} & V^{-1} & 0_{2l,d} \\ 0_{d,l} & 0_{d,2l} & 0_{d,l} & 0_{d,2l} & 0_{d,2l} & 0_{d,2l} & B & 0_{d,2l} & I_d \end{pmatrix} \tag{21}$$

and $\vec{u} = (\vec{0}_{5l}||\vec{g})$. Let $\mathsf{ulp} = (A, \vec{u})$. Finally, we use our framework from Section 3 to prove that $\vec{m} \in \mathcal{L}_{14+d/l}\,(\mathsf{pp}, \mathsf{ulp})$.

5.1 Main Protocol

Figure 5 describes a simple protocol for proving integer multiplication. We briefly analyse security of the protocol. First, correctness and zero-knowledge follow straightforwardly from the discussion above and previous works [15, Theorem 4.1].

Soundness arguments works as follows. From Section 3.1 and the discussion above, there is an efficient extractor $\mathcal{E}$ which extracts openings of the commitments $\boldsymbol{t}_i$:

$$\vec{m}^* = (\vec{a}, \vec{a}'_1, \vec{a}'_2, \vec{b}, \vec{b}'_1, \vec{b}'_2, \vec{c}_1, \vec{c}_2, \vec{c}'_1, \vec{c}'_2, \vec{f}_1, \vec{f}_2, \vec{f}'_1, \vec{f}'_2, \vec{e}_1, \ldots, \vec{e}_{d/l}).$$

We have to prove that $\vec{m}^*$ is a vector over $\mathbb{Z}_q$. By definition of pp and ulp, we know that $\vec{a}, \vec{a}'_i$ satisfy $\vec{a} \in \mathbb{Z}_q^l$ and $V\vec{a} = (\vec{a}'_1||\vec{a}'_2)$. Since V and $\vec{a}$ are over $\mathbb{Z}_q$, we conclude that $\vec{a}',\vec{a}'_2$ are also over $\mathbb{Z}_q$. We argue similarly for

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

Inputs:

$a, b \in [-2^{N-1}, 2^{N-1}-1]$ and $c \in [-2^{2N-1}, 2^{2N-1}-1]$ such that $c = ab$ | $\boldsymbol{B}_0; \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{16+d/l}$

$\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa \times (\lambda+\kappa+16+d/l)}; \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{16+d/l} \in \mathcal{R}_q^{\lambda+\kappa+16+d/l}$

$\vec{e}_1, \ldots, \vec{e}_{d/l} \xleftarrow{\$} [-\delta_1'+1, \delta_1'-1]^l$

$\vec{e} = (\vec{e}_1, \ldots, \vec{e}_{d/l})$

$\vec{a} = \mathsf{sTC}_l(a), \vec{b} = \mathsf{sTC}_l(b), \vec{c} = \mathsf{sTC}_{2l}(c)$

$\boldsymbol{a} = \sum_{i=0}^{l-1} a_i X^i, \boldsymbol{b} = \sum_{i=0}^{l-1} b_i X^i, \boldsymbol{c} = \sum_{i=0}^{l-1} c_i X^i$

$\boldsymbol{f} = (\boldsymbol{ab} - \boldsymbol{c})/(X-2) \in \mathbb{Z}[X], \vec{f}_i = (f_{il}, \ldots, f_{il+l-1})$ for $i = 1, 2$

$\begin{pmatrix} \vec{a}_1' \\ \vec{a}_2' \end{pmatrix} = V_1\vec{a}, \begin{pmatrix} \vec{b}_1' \\ \vec{b}_2' \end{pmatrix} = V_1\vec{b}, \begin{pmatrix} \vec{c}_1' \\ \vec{c}_2' \end{pmatrix} = V_1\vec{c}, \begin{pmatrix} \vec{f}_1' \\ \vec{f}_2' \end{pmatrix} = V_1 f_1 + V_2 f_2$

Run $\mathsf{Com}_{14+d/l,2}\left(\vec{a}||\vec{a}_1'||\vec{a}_2'||\vec{b}||\vec{b}_1'||\vec{b}_2'||\vec{c}||\vec{c}_1'||\vec{c}_2'||\vec{f}_1||\vec{f}_2||\vec{f}_1'||\vec{f}_2'||\vec{e}\right)$

$\xleftarrow{B_1, B_2}$ | $B_1, B_2 \xleftarrow{\$} \{0,1\}^{d \times l} \subset \mathbb{Z}_q^{l \times l}$

$\vec{g} = B_1\vec{f}_1 + B_2\vec{f}_2 + \vec{e}$

If $\|\vec{g}\|_\infty \geq \delta_1' - \beta_1'$, abort $\xrightarrow{\vec{g}}$

Run $\Pi^2_{14+d/l}(\mathsf{pp}, \mathsf{ulp})$ | $\|\vec{g}\|_\infty \overset{?}{<} q/12$

| Check ver. eq. for $\Pi^2_{14+d/l}(\mathsf{pp}, \mathsf{ulp})$

Fig. 5. Proof of integer multiplication where $N = l$. Here, pp and ulp are defined in (20) and (21) respectively. We use the signed two's complement decomposition sTC_l from (4). Protocols $\mathsf{Com}_{14+d/l,2}(\vec{m})$ and $\Pi^2_{14+d/l}(\mathsf{pp}, \mathsf{ulp})$ are described in Section 3.2 for $n = 14 + d/l$.

other extracted messages $\vec{b}, \vec{b}_i', \vec{c}_i, \vec{c}_i'$. Next, we know from the definition of pp that $\vec{a}_i' \circ \vec{b}_i' - \vec{c}_i' - \vec{\gamma}_i \circ \vec{f}_i' = \vec{0}$ for $i = 1, 2$. Since all the entries in $\vec{\gamma}_i \in \mathbb{Z}_q^l$ are non-zero, we conclude that $\vec{f}_i' \in \mathbb{Z}_q^l$. Furthermore, we know that $V^{-1}(\vec{f}_1'||\vec{f}_2') = (\vec{f}_1||\vec{f}_2)$. Therefore, $\vec{f}_1, \vec{f}_2 \in \mathbb{Z}_q^l$ since both V^{-1} and $\vec{f}_i'$ are over $\mathbb{Z}_q$. This also directly implies that all $\vec{e}_i \in \mathbb{Z}_q^l$

Essentially, $\mathcal{E}$ finds binary (apart from the last coefficient which is in $\{-1, 0\}$) polynomials $\boldsymbol{a}^*, \boldsymbol{b}^*, \boldsymbol{c}^* \in R_{q,2l}$, where the former two are of degree at most $l-1$, and a polynomial $\boldsymbol{f}^* \in R_{q,2l}$ of degree at most $2l-1$ such that $\boldsymbol{a}^*\boldsymbol{b}^* - \boldsymbol{c}^* = (X-2)\boldsymbol{f}^*$ over $R_{q,2l}$. We argue that the absolute values of coefficients of $(X-2)\boldsymbol{f}^*$ are strictly less than $q/2$. This would imply that this equation also holds over $\mathbb{Z}[X]$ and thus $\boldsymbol{a}^*(2)\boldsymbol{b}^*(2) = \boldsymbol{c}^*(2)$.

As discussed earlier, $\mathcal{E}$ is able to extract vectors $\vec{f}_i \in \mathbb{Z}_q^l$, $\vec{e} \in Z_q^d$ such that $B_0\vec{f}_0 + B_1\vec{f}_1 + \vec{e} = \vec{g}$ for challenge matrices B_0, B_1 and a response $\vec{g}$ from a deterministic prover $\mathcal{P}^*$. We know that $\|\vec{g}\|_\infty < q/12$. Then, since $B = (B_1||B_2) \in \mathbb{Z}_q^{d \times 2l}$ is a matrix of random binary values, by Lemma 2.4 we get that the maximum coefficient of $(\vec{f}_1^*||\vec{f}_2^*)$ is strictly smaller that $q/6$, i.e. $\|\boldsymbol{f}^*\|_\infty < q/6$, with probability at least $1 - 2^d$. Hence, $\|(X-2)\boldsymbol{f}^*\|_\infty < q/2$ with an overwhelming probability.

Proof size. First, consider the case when $q^{-d/l}$ is negligible. By taking calculations from Section 3.2 and accounting for one extra response $\vec{g} \in \mathbb{Z}_q^d$ from $\mathcal{P}$, we obtain the following total proof size:

$$(17 + d/l + \kappa)d\log q \;+\; (\lambda + \kappa + 16 + d/l) \cdot d \cdot \log(2\delta_1) + d \cdot \log(2\delta_1')\text{ bits.} \tag{22}$$

Now, we need to compute β', i.e. a bound on $\|B_1\vec{f}_1 + B_2\vec{f}_2\|_\infty$. From Lemma C.1 we can give a naive bound $\beta_1 \leq 2l(N+1)$. We also need to make sure that $\delta' < q/12$ but since we will usually select $q \approx 2^{30}$, this is not a problem. Concretely, we use $\delta' = 2^{26}$. Thus, the expected number of repetitions M for the rejection sampling for $\vec{g}$ satisfies:

$$1/M = \left(\frac{2(\delta'_1 - \beta'_1) - 1}{2\delta'_1 - 1}\right)^d \approx e^{-d\beta'_1/\delta'_1}.$$

When $q^{-d/l}$ is not negligible, we apply the protocol from Figure 8. By taking calculations from Appendix A.5 and Section 3.2, we obtain the following proof size:

$$(15 + 3d/l + \kappa)d \log q \;+\; k(\lambda + \kappa + 15 + 2d/l)d \cdot \log(2\delta_1) + d \cdot \log(2\delta'_1) \text{ bits}.$$

We provide concrete proof sizes for specific values of N in Figure 6. Similarly as in Section 4.1, we select $q \approx 2^{30}$ and appropriate parameters λ, κ such that the root Hermite factor (for both SIS and LWE) is around $\delta \approx 1.0045$. When calculating the sizes, we already implement small optimizations described in Section C.2 along with improvements to the main framework in Appendix A.5 and A.6.

Additionally, one could consider picking a different base (e.g. 3 instead of 2) to cover larger ranges for a, b, c. In this case, one would apply the protocol from Figure 1 with polynomials of degree three.

N	l	k	κ	λ	δ_1	δ_2	D	proof size
32	32	1	11	10	2^{17}	$(q-1)/2^{14}$	11	14.5KB
128	128	4	10	10	2^{18}	$(q-1)/2^{13}$	14	40.2KB
512	128	4	10	10	2^{18}	$(q-1)/2^{13}$	14	99.8KB

Fig. 6. Proof size comparison for proving integer multiplication $ab = c$ for $a, b \in [-2^{N-1}, 2^{N-1} - 1]$. In each scenario, we pick $q \approx 2^{30}$ and $d = 128$. Here, parameters δ_1, δ_2, D are used for the commitment compression (see Section A.5).

In Section C we describe small tricks improvements to the multiplication protocol which slightly improve the proof size.

6 Implementation

We provide an open-source constant-time single-threaded implementation of our integer addition and multiplication proof systems for integers of length $N = 128$. The software can be obtained from

`github.com/gregorseiler/irelzk`.

It borrows heavily from the implementation techniques that have been used to speed up the Kyber [9], Dilithium [13] and NTTRU [27] lattice-based schemes. The software is optimized for x86 CPUs supporting the AVX2 instruction set and especially all of the polynomial arithmetic over $\mathcal{R}_q$ is fully vectorized. Most parts of the AVX2 code are written using C intrinsics, while the AVX2 NTT is implemented in assembly language. As in many lattice-based construction based on the Module-LWE and Module-SIS problems, among the most time-consuming tasks are the expansion of the commitment matrices $\boldsymbol{B}_0$ and $\vec{\boldsymbol{b}}_i$ and polynomial multiplication.

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

For fast polynomial multiplication, whenever possible, we keep polynomials in the NTT basis representation to save NTT operations, and also sample uniformly random polynomials directly in the NTT basis.

Specifically, uniform polynomials that are sent as part of the proofs are sent in the NTT basis. Our code also includes fast in-place implementations of the Galois automorphisms that operate in the NTT basis.

The AVX2 optimized NTT implementation operates on dense vectors of 8 32-bit coefficients. It uses the modified Montgomery reduction algorithm from [29] and [27] to reduce intermediate 64 bit products. This allows to handle dense vector registers more efficiently and saves multiplication instructions that lie on the critical path. To our knowledge, it is the first time the technique has been used for 32-bit arithmetic. At the core of the code lies a vectorized interleaved butterfly implementation that processes 32 coefficients in 4 vector registers at a time. The instruction for parallel Montgomery and single-word reductions in a butterfly operation are very carefully scheduled, taking into account the front-end decoding throughput from the L1 cache and the back-end execution resources. A full NTT execution runs in 540 cycles on an Intel Skylake core.

For the hash function that is needed in the Fiat-Shamir transform to obtain all the challenge polynomials, we use SHAKE128. We list the runtimes of our implementation in Figure 7.

	Addition	Multiplication
Proving	1.09 ms	2.39 ms
Verifying	0.20 ms	0.41 ms

Fig. 7. Timings of our implementation of the integer addition and multiplication proof system for integers of length $N = 128$ on a single core of an Intel Skylake CPU running at 3.5 GHz. The given numbers are the medians of 500 executions each.

Acknowledgements

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

References

1. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM Conference on Computer and Communications Security*, pages 2087–2104. ACM, 2017.
2. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 470–499. Springer, 2020.
3. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.
4. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, pages 368–385, 2018.
5. Carsten Baum and Vadim Lyubashevsky. Simple amortized proofs of shortness for linear relations over polynomial rings. *IACR Cryptology ePrint Archive*, 2017:759, 2017.
6. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In *Public Key Cryptography (1)*, volume 12110 of *Lecture Notes in Computer Science*, pages 495–526. Springer, 2020.
7. 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.

8. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 176–202. Springer, 2019.
9. Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P*, pages 353–367, 2018.
10. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503. ACM, 2002.
11. Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. *CoRR*, abs/2003.05207, 2020.
12. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 356–383. Springer, 2019.
13. 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.
14. Alex Escala and Jens Groth. Fine-tuning groth-sahai proofs. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 630–649. Springer, 2014.
15. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. *IACR Cryptol. ePrint Arch.*, 2020:518, 2020. `https://eprint.iacr.org/2020/518`.
16. Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 115–146. Springer, 2019.
17. Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 67–88. Springer, 2019.
18. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *ACM Conference on Computer and Communications Security*, pages 567–584. ACM, 2019.
19. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992.
20. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, 2018.
21. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.
22. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 700–732. Springer, 2018.
23. Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 326–355. Springer, 2019.
24. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.
25. Vadim Lyubashevsky. Basic lattice cryptography: Encryption and fiat-shamir signatures. 2019.
26. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. *IACR Cryptology ePrint Archive*, 2020. To appear at ACM CCS 2020.
27. Vadim Lyubashevsky and Gregor Seiler. NTTRU: truly fast NTRU using NTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):180–201, 2019.
28. Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
29. Gregor Seiler. Faster AVX2 optimized NTT multiplication for ring-lwe lattice cryptography. *IACR Cryptology ePrint Archive*, 2018:39, 2018. http://eprint.iacr.org/2018/039.

A Additional Background

As already mentioned in Section 3, the key ingredients for proving $\vec{m} \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$ are the new efficient product proof [2] and unstructured linear proof [15] protocols. For completeness, we sketch out main ideas used in the previous works [2,15].

A.1 Automorphisms

In this section we work over the cyclotomic ring $\mathcal{R}_q$ with dimension d which is a power-of-two. Prime q is selected so that polynomial $X^d + 1$ splits into polynomials of degree d/l as in (5).

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

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

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

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

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

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

A.2 Weak Opening

Attema et al. [2] introduce the notion of a weak opening that is produced by the opening proof. The commitment scheme in [4] is still binding with respect to this notion if M-SIS is hard.

Definition A.1. *A* weak opening *for the commitment $\vec{\boldsymbol{t}} = (\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)^T$ consists of l polynomials $\bar{\boldsymbol{c}}_i \in \mathcal{R}_q$, a randomness vector $\vec{\boldsymbol{r}}^*$ over $\mathcal{R}_q$ and a message $\boldsymbol{m}^* \in \mathcal{R}_q$ such that*

$$\begin{aligned}
&\left\|\bar{\boldsymbol{c}}_i\right\|_1 \leq 2d \text{ and } \bar{\boldsymbol{c}}_i \bmod (X^{d/l} - \zeta^{2i+1}) \neq 0 \text{ for all } 0 \leq i \leq l-1, \\
&\|\bar{\boldsymbol{c}}_i \vec{\boldsymbol{r}}^*\|_\infty \leq 2\beta \text{ for all } 0 \leq i \leq l-1, \\
&\boldsymbol{B}_0 \vec{\boldsymbol{r}}^* = \vec{\boldsymbol{t}}_0, \\
&\langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^* \rangle + \boldsymbol{m}^* = \boldsymbol{t}_1.
\end{aligned}$$

A.3 Product Proof

Simple case. For readability, we first consider the case when we only need to prove knowledge of $\vec{m}$ so that $P(\vec{m}) = \vec{0}$ for a single $P \in \mathsf{pp}$. As mentioned before, we assume P is a multivariate polynomial of total degree at most 3. Hence, we can write a general formula for P:

$$\begin{aligned}
P(\vec{x}_1, \ldots, \vec{x}_n) := \sum_{i \leq j \leq \ell} \vec{\mu}_{i,j,\ell} \circ (\vec{x}_i \circ \vec{x}_j \circ \vec{x}_\ell) + \sum_{i \leq j} \vec{\eta}_{i,j} \circ (\vec{x}_i \circ \vec{x}_j) \\
+ \sum_{i=1}^{n} \vec{\nu}_i \circ \vec{x}_i + \vec{\rho}
\end{aligned} \tag{23}$$

Clearly, $P(\vec{m}_1, \ldots, \vec{m}_n) = \vec{0}$ is equivalent to

$$\begin{aligned} P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) := \sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \boldsymbol{m}_i \boldsymbol{m}_j \boldsymbol{m}_\ell + \sum_{i \leq j} \boldsymbol{\eta}_{i,j} \boldsymbol{m}_i \boldsymbol{m}_j \\ + \sum_{i=1}^{n} \boldsymbol{\nu}_i \boldsymbol{m}_i + \boldsymbol{\rho} = 0 \end{aligned} \tag{24}$$

where $\boldsymbol{m}_i = \mathsf{NTT}^{-1}(\vec{m}_i)$ and $\boldsymbol{\mu}_{i,j,\ell} = \mathsf{NTT}^{-1}(\vec{\mu}_{i,j,\ell})$ (and similarly for $\boldsymbol{\eta}_{i,j}, \boldsymbol{\nu}_i, \boldsymbol{\rho}$).

We follow the strategy from [2]. Let $\vec{\boldsymbol{y}} \xleftarrow{\$} S_{\delta_1 - 1}^{(\lambda+\kappa+n+d/l+2)d}$ and $\boldsymbol{c} \xleftarrow{\$} C$. For $i \in [n]$, define $\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle - \boldsymbol{c}\boldsymbol{m}_i$. Then, we can expand the sum $\sum_{i \leq j \leq \ell} -\boldsymbol{\mu}_{i,j,\ell} \boldsymbol{f}_i \boldsymbol{f}_j \boldsymbol{f}_\ell$ as:

$$\begin{aligned} &\boldsymbol{c}^3 \left(\sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \boldsymbol{m}_i \boldsymbol{m}_j \boldsymbol{m}_\ell \right) - \left(\sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \right) \\ &- \boldsymbol{c}^2 \left(\sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \left(\langle \vec{\boldsymbol{b}}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i \boldsymbol{m}_j + \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i \boldsymbol{m}_\ell + \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_j \boldsymbol{m}_\ell \right) \right) \\ &+ \boldsymbol{c} \left(\sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \left(\langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_\ell + \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_j + \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i \right) \right). \end{aligned} \tag{25}$$

Similarly, we have:

$$\begin{aligned} \boldsymbol{c} \sum_{i \leq j} \boldsymbol{\eta}_{i,j} \boldsymbol{f}_i \boldsymbol{f}_j =& \boldsymbol{c}^3 \left(\sum_{i \leq j} \boldsymbol{\eta}_{i,j} \boldsymbol{m}_i \boldsymbol{m}_j \right) + \boldsymbol{c} \left(\sum_{i \leq j} \boldsymbol{\eta}_{i,j} \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \right) \\ &- \boldsymbol{c}^2 \left(\sum_{i \leq j} \boldsymbol{\eta}_{i,j} \left(\boldsymbol{m}_i \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}} \rangle + \boldsymbol{m}_j \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle \right) \right), \end{aligned} \tag{26}$$

and

$$-\boldsymbol{c}^2 \sum_{i=1}^{n} \boldsymbol{\nu}_i \boldsymbol{f}_i = \boldsymbol{c}^3 \left(\sum_{i=1}^{n} \boldsymbol{\nu}_i \boldsymbol{m}_i \right) - \boldsymbol{c}^2 \left(\sum_{i=1}^{n} \boldsymbol{\nu}_i \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle \right). \tag{27}$$

Then, we observe that:

$$\begin{aligned} - \sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \boldsymbol{f}_i \boldsymbol{f}_j \boldsymbol{f}_\ell + \boldsymbol{c} \sum_{i \leq j} \boldsymbol{\eta}_{i,j} \boldsymbol{f}_i \boldsymbol{f}_j - \boldsymbol{c}^2 \sum_{i=1}^{n} \boldsymbol{\nu}_i \boldsymbol{f}_i + \boldsymbol{c}^3 \boldsymbol{\rho} \\ = \boldsymbol{c}^3 P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) - \boldsymbol{c}^2 \boldsymbol{\psi}_2 + \boldsymbol{c} \boldsymbol{\psi}_1 - \boldsymbol{\psi}_0, \end{aligned} \tag{28}$$

where $\boldsymbol{\psi}_0, \boldsymbol{\psi}_1, \boldsymbol{\psi}_2$ are defined as follows:

$$\boldsymbol{\psi}_0 := \sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle,$$

$$\begin{aligned} \boldsymbol{\psi}_1 := &\sum_{i \leq j \leq \ell} \boldsymbol{\mu}_{i,j,\ell} \left(\langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_\ell + \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_j + \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}} \rangle \boldsymbol{m}_i \right) \\ &+ \sum_{i \leq j} \boldsymbol{\eta}_{i,j} \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}} \rangle \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}} \rangle, \end{aligned} \tag{29}$$

and

$$\begin{aligned}\boldsymbol{\psi}_2 := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{i,j,\ell}\left(\langle \vec{\boldsymbol{b}}_\ell, \vec{\boldsymbol{y}}\rangle \boldsymbol{m}_i\boldsymbol{m}_j + \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}\rangle \boldsymbol{m}_i\boldsymbol{m}_\ell + \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle \boldsymbol{m}_j\boldsymbol{m}_\ell\right) \\ + \sum_{i\le j} \boldsymbol{\eta}_{i,j}\left(\boldsymbol{m}_i\langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}\rangle + \boldsymbol{m}_j\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle\right) + \sum_{i=1}^{n} \boldsymbol{\nu}_i \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle.\end{aligned} \tag{30}$$

We are ready to describe a simple protocol for proving $P(\vec{m}) = \vec{0}$. Prover $\mathcal{P}$ starts by sampling a vector $\vec{\boldsymbol{y}} \xleftarrow{\$} S_{\delta_1-1}^{(\lambda+\kappa+n+d/l+2)d}$ of small polynomials and setting $\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$. Also, the prover computes

$$\begin{cases} \boldsymbol{t}_1' &= \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{r}}\rangle - \boldsymbol{\psi}_2, \\ \boldsymbol{t}_2' &= \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{r}}\rangle - \boldsymbol{\psi}_1 - \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{y}}\rangle. \end{cases} \tag{31}$$

Finally, it sets $\boldsymbol{v} = \boldsymbol{\psi}_0 - \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{y}}\rangle$ and sends $(\vec{\boldsymbol{w}}, \boldsymbol{t}_1', \boldsymbol{t}_2', \boldsymbol{v})$ to the verifier $\mathcal{V}$.

After receiving a challenge $\boldsymbol{c} \xleftarrow{\$} C$ from $\mathcal{V}$, the prover computes a masked opening $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ and applies standard rejection sampling. If it does not abort, $\mathcal{P}$ sends $\vec{\boldsymbol{z}}$ to $\mathcal{V}$.

Now, $\mathcal{V}$ checks that $\vec{\boldsymbol{z}}$ consists of small polynomials and $\boldsymbol{B}_0\vec{\boldsymbol{z}} \overset{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$ over $\mathcal{R}_q$. Next, $\mathcal{V}$ sets $\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}}\rangle - \boldsymbol{c}\boldsymbol{t}_i$ for $i \in [n]$ and also

$$\boldsymbol{f}_i' = \langle \vec{\boldsymbol{b}}_{n+i}, \vec{\boldsymbol{z}}\rangle - \boldsymbol{c}\boldsymbol{t}_i' \text{ for } i = 1, 2.$$

Then, the verifier checks whether:

$$\begin{aligned}- \sum_{i\le j\le \ell} \boldsymbol{\mu}_{i,j,\ell}\boldsymbol{f}_i\boldsymbol{f}_j\boldsymbol{f}_\ell + \boldsymbol{c}\sum_{i\le j} \boldsymbol{\eta}_{i,j}\boldsymbol{f}_i\boldsymbol{f}_j - \boldsymbol{c}^2\sum_{i=1}^{n} \boldsymbol{\nu}_i\boldsymbol{f}_i + \boldsymbol{c}^3\boldsymbol{\rho} \\ \overset{?}{=} -\boldsymbol{c}\boldsymbol{f}_1' + \boldsymbol{f}_2' - \boldsymbol{v}.\end{aligned} \tag{32}$$

We briefly discuss correctness of the protocol. First, note that values $\boldsymbol{f}_i$ are the same as the ones defined above, since

$$\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}}\rangle - \boldsymbol{c}\boldsymbol{t}_i = \boldsymbol{c}\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}\rangle + \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle - \boldsymbol{c}\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}\rangle - \boldsymbol{c}\boldsymbol{m}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle - \boldsymbol{c}\boldsymbol{m}_i.$$

Therefore, Equation 32 follows directly from (28) and the fact that $P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) = 0$.

We also point out that by Lemma 2.1, soundness error of this protocol is about $q^{-d/l}$ assuming that M-SIS is hard. We show in Appendix A.3 how to boost soundness using Galois automorphisms.

Multiple polynomials. We now describe how to prove knowledge of $\vec{m}$ such that $P(\vec{m}) = \vec{0}$ for all $P \in \mathsf{pp}$. Let $\mathsf{pp} = \{P_1, \ldots, P_\xi\}$ where $\xi = |\mathsf{pp}|$. Similarly as above, we can write an explicit formula for each polynomial P_t over $\mathcal{R}_q$:

$$\begin{aligned}P_t(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell}\boldsymbol{x}_i\boldsymbol{x}_j\boldsymbol{x}_\ell + \sum_{i\le j} \boldsymbol{\eta}_{t,i,j}\boldsymbol{x}_i\boldsymbol{x}_j \\ + \sum_{i=1}^{n} \boldsymbol{\nu}_{t,i}\boldsymbol{x}_i + \boldsymbol{\rho}_t.\end{aligned} \tag{33}$$

Clearly, for each P_t, one can also write down (28) with potentially different $\boldsymbol{\psi}_{t,0}, \boldsymbol{\psi}_{t,1}, \boldsymbol{\psi}_{t,2} \in \mathcal{R}_q$. We now linear-combine ξ such equations. Namely, for any $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi \in \mathcal{R}_q$ we have:

$$\begin{aligned} &-\sum_{t=1}^{\xi}\sum_{i\le j\le \ell} \boldsymbol{\alpha}_t \boldsymbol{\mu}_{t,i,j,\ell} \boldsymbol{f}_i \boldsymbol{f}_j \boldsymbol{f}_\ell + \boldsymbol{c}\sum_{t=1}^{\xi}\sum_{i\le j} \boldsymbol{\alpha}_t \boldsymbol{\eta}_{t,i,j} \boldsymbol{f}_i \boldsymbol{f}_j \\ &- \boldsymbol{c}^2 \sum_{t=1}^{\xi}\sum_{i=1}^{n} \boldsymbol{\alpha}_t \boldsymbol{\nu}_{t,i} \boldsymbol{f}_i + \boldsymbol{c}^3 \sum_{t=1}^{\xi} \boldsymbol{\rho}_t \\ &= \boldsymbol{c}^3 \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t P_t(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) - \boldsymbol{c}^2 \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,2} + \boldsymbol{c}\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,1} - \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,0}. \end{aligned} \tag{34}$$

The main idea here is to first let the verifier pick uniformly random $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi \in \mathcal{R}_q$, and then prove that $Q(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) = 0$, where

$$Q := \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t P_t, \tag{35}$$

using the protocol from Section A.3. Concretely, prover $\mathcal{P}$ starts by sampling $\vec{\boldsymbol{y}} \xleftarrow{\$} S_{\delta_1-1}^{(\lambda+\kappa+n+d/l+2)d}$ and sending $\vec{\boldsymbol{w}} = \boldsymbol{B}_0 \vec{\boldsymbol{y}}$. After receiving challenges $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi \xleftarrow{\$} \mathcal{R}_q$ from the verifier, $\mathcal{P}$ computes

$$\begin{cases} \boldsymbol{t}_1' &= \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{r}} \rangle - \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,2}, \\ \boldsymbol{t}_2' &= \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{r}} \rangle - \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,1} - \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{y}} \rangle. \end{cases} \tag{36}$$

Finally, it sets

$$\boldsymbol{v} = \sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,0} - \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{y}} \rangle$$

and sends $(\boldsymbol{t}_1', \boldsymbol{t}_2', \boldsymbol{v})$ to the verifier.

After receiving a challenge $\boldsymbol{c} \xleftarrow{\$} C$ from $\mathcal{V}$, the prover sends a masked opening $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ unless the rejection sampling aborts.

Similarly as before, $\mathcal{V}$ checks that $\vec{\boldsymbol{z}}$ consists of small polynomials and $\boldsymbol{B}_0 \vec{\boldsymbol{z}} \stackrel{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$. Next, $\mathcal{V}$ sets $\boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_i$ for $i \in [n]$ and also

$$\boldsymbol{f}_i' = \langle \vec{\boldsymbol{b}}_{n+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_i' \text{ for } i = 1, 2.$$

Eventually, the verifier checks whether:

$$\begin{aligned} &-\sum_{t=1}^{\xi}\sum_{i\le j\le \ell} \boldsymbol{\alpha}_t \boldsymbol{\mu}_{t,i,j,\ell} \boldsymbol{f}_i \boldsymbol{f}_j \boldsymbol{f}_\ell + \boldsymbol{c}\sum_{t=1}^{\xi}\sum_{i\le j} \boldsymbol{\alpha}_t \boldsymbol{\eta}_{t,i,j} \boldsymbol{f}_i \boldsymbol{f}_j \\ &\quad - \boldsymbol{c}^2 \sum_{t=1}^{\xi}\sum_{i=1}^{n} \boldsymbol{\alpha}_t \boldsymbol{\nu}_{t,i} \boldsymbol{f}_i + \sum_{t=1}^{\xi} \boldsymbol{c}^3 \boldsymbol{\rho}_t \stackrel{?}{=} -\boldsymbol{c}\boldsymbol{f}_1' + \boldsymbol{f}_2' - \boldsymbol{v}. \end{aligned} \tag{37}$$

As discussed in [2], the amortization technique does not decrease the soundness thanks to a nice property of the Schwartz-Zippel lemma. Intuitively, as soon as one of the relations is false, then the linear combination of all of the relations will be uniformly random, and this will be detected with overwhelming probability.

Boosting soundness. We recall that the two previous protocols shown in Section A.3 have soundness error around $q^{-d/l}$ (ignoring the term related to solving M-SIS). In order to decrease this probability further, we apply Galois automorphisms. Let $\sigma = \sigma_{2l/k+1}$ be the automorphism of order kd/l such that $q^{-kd/l}$ is negligible. Then, by linear-combining all permutations $\sigma^{\iota}(Q(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n))$ with independent challenge coefifcients $\boldsymbol{\phi}_\iota \in \mathcal{R}_q$, we want to prove $Q'(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) = 0$ where

$$Q' := \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota}(Q(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n)). \tag{38}$$

In order to do so, we follow the strategy from Section A.3.

First, let $\vec{\boldsymbol{y}}_0, \ldots, \vec{\boldsymbol{y}}_{k-1} \xleftarrow{\$} S_{\delta_1-1}^{(\lambda+\kappa+n+d/l+2)d}$ and $\boldsymbol{c} \xleftarrow{\$} C$. This time, we define $\boldsymbol{f}_j^{(\iota)} := \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}_\iota \rangle - \sigma^\iota(\boldsymbol{c})\boldsymbol{m}_j$ for $j \in [n]$ and $\iota \in \{0, \ldots, k-1\}$. Then, similarly as in (34), we have:

$$\begin{aligned}
&\sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \sum_{i \le j \le \ell} -\boldsymbol{\alpha}_t \boldsymbol{\mu}_{t,i,j,\ell} \boldsymbol{f}_i^{(\iota)} \boldsymbol{f}_j^{(\iota)} \boldsymbol{f}_\ell^{(\iota)} + \sigma^\iota(\boldsymbol{c}) \sum_{t=1}^{\xi} \sum_{i \le j} \boldsymbol{\alpha}_t \boldsymbol{\eta}_{t,i,j} \boldsymbol{f}_i^{(\iota)} \boldsymbol{f}_j^{(\iota)} \right) \\
&+ \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sigma^\iota(\boldsymbol{c}^2) \sum_{t=1}^{\xi} \sum_{i=1}^{n} -\boldsymbol{\alpha}_t \boldsymbol{\nu}_{t,i} \boldsymbol{f}_i^{(\iota)} + \sigma^\iota(\boldsymbol{c}^3) \sum_{t=1}^{\xi} \boldsymbol{\rho}_t \right) \\
&= \boldsymbol{c}^3 Q'(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) - \boldsymbol{c}^2 \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,2} \right) \\
&+ \boldsymbol{c} \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,1} \right) - \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,0} \right).
\end{aligned} \tag{39}$$

We can now describe the full protocol. To begin with, $\mathcal{P}$ samples $\vec{\boldsymbol{y}}_0, \ldots, \vec{\boldsymbol{y}}_{k-1} \xleftarrow{\$} S_{\delta_1-1}^{(\lambda+\kappa+n+d/l+2)d}$ and sends $\vec{\boldsymbol{w}}_\iota = \boldsymbol{B}_0 \vec{\boldsymbol{y}}_\iota$ for $\iota \in \{0, \ldots, k-1\}$. After receiving challenges $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi \xleftarrow{\$} \mathcal{R}_q$ and $\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{k-1} \xleftarrow{\$} \mathcal{R}_q$ from $\mathcal{V}$, the prover computes

$$\begin{cases} \boldsymbol{t}_1' &= \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{r}} \rangle - \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,2} \right), \\ \boldsymbol{t}_2' &= \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{r}} \rangle - \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,1} \right) + \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{y}} \rangle. \end{cases} \tag{40}$$

and sets

$$\boldsymbol{v} = \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \boldsymbol{\alpha}_t \boldsymbol{\psi}_{t,0} \right) - \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{y}} \rangle.$$

Then, $\mathcal{P}$ sends $(\boldsymbol{t}_1', \boldsymbol{t}_2', \boldsymbol{v})$ to the verifier.

After receiving a challenge $\boldsymbol{c} \xleftarrow{\$} C$ from $\mathcal{V}$, the prover sends k masked openings $\vec{\boldsymbol{z}}_\iota = \vec{\boldsymbol{y}}_\iota + \sigma^\iota(\boldsymbol{c})\vec{\boldsymbol{r}}$ for $\iota = 0, 1, \ldots, k-1$, unless one of the k rejection sampling algorithms aborts.

Similarly as before, $\mathcal{V}$ checks that each $\vec{\boldsymbol{z}}_\iota$ consists of small polynomials and $\boldsymbol{B}_0 \vec{\boldsymbol{z}}_\iota \stackrel{?}{=} \vec{\boldsymbol{w}}_\iota + \sigma^\iota(\boldsymbol{c})\vec{\boldsymbol{t}}_0$. Next, $\mathcal{V}$ sets

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

and also

$$\boldsymbol{f}_i' = \langle \vec{\boldsymbol{b}}_{n+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_i' \text{ for } i = 1, 2.$$

Then, the verifier checks whether:

$$\begin{aligned}
&\sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sum_{t=1}^{\xi} \sum_{i \le j \le \ell} -\boldsymbol{\alpha}_t \boldsymbol{\mu}_{t,i,j,\ell} \boldsymbol{f}_i^{(\iota)} \boldsymbol{f}_j^{(\iota)} \boldsymbol{f}_\ell^{(\iota)} + \sigma^\iota(\boldsymbol{c}) \sum_{t=1}^{\xi} \sum_{i \le j} \boldsymbol{\alpha}_t \boldsymbol{\eta}_{t,i,j} \boldsymbol{f}_i^{(\iota)} \boldsymbol{f}_j^{(\iota)} \right) \\
&+ \sum_{\iota=0}^{k-1} \boldsymbol{\phi}_\iota \sigma^{-\iota} \left(\sigma^\iota(\boldsymbol{c}^2) \sum_{t=1}^{\xi} \sum_{i=1}^{n} -\boldsymbol{\alpha}_t \boldsymbol{\nu}_{t,i} \boldsymbol{f}_i^{(\iota)} + \sigma^\iota(\boldsymbol{c}^3) \sum_{t=1}^{\xi} \boldsymbol{\rho}_t \right) \stackrel{?}{=} -\boldsymbol{c}^2 \boldsymbol{f}_1' + \boldsymbol{c}\boldsymbol{f}_2' - \boldsymbol{v}.
\end{aligned} \tag{41}$$

Attema et al. show that the success probability of a cheating prover is now reduced to at most $\varepsilon = \left(\frac{3}{q^{d/l}}\right)^k$.

A.4 Unstructured Linear Proof

Let $\mathsf{ulp} = (A, \vec{u})$. In order to prove knowledge of $\vec{m} \in \mathbb{Z}_q^{nl}$ such that $A\vec{m} = \vec{u}$, we show that

$$\langle A\vec{m} - \vec{u}, \vec{\gamma} \rangle = 0 \tag{42}$$

for a uniformly random challenge vector $\vec{\gamma} = (\vec{\gamma}_1, \ldots, \vec{\gamma}_v) \in W_q^{vl}$. We follow the approach presented in [15]. First, let us rewrite A as:

$$A = \begin{pmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ldots & \vdots \\ A_{v,1} & \cdots & A_{v,n} \end{pmatrix} \tag{43}$$

where $A_{i,j} \in Z_q^{l \times l}$. Then, we have:

$$\begin{aligned} \langle A\vec{m}, \vec{\gamma} \rangle &= \sum_{i=1}^{v} \sum_{j=1}^{n} \langle A_{i,j}\vec{m}_j, \vec{\gamma}_i \rangle = \sum_{j=1}^{n} \sum_{i=1}^{v} \langle \vec{m}_j, A_{i,j}^T \vec{\gamma}_i \rangle \\ &= \sum_{j=1}^{n} \sum_{\iota=0}^{l-1} \left(\mathsf{NTT}(\boldsymbol{m}_j) \circ \mathsf{NTT}(\boldsymbol{p}_j)\right)_\iota = \sum_{j=1}^{n} \sum_{\iota=0}^{l-1} \mathsf{NTT}(\boldsymbol{m}_j \boldsymbol{p}_j)_\iota \\ &= \sum_{\iota=0}^{l-1} \mathsf{NTT}\left(\sum_{j=1}^{n} \boldsymbol{m}_j \boldsymbol{p}_j\right)_\iota \end{aligned}$$

where $\boldsymbol{p}_j := \mathsf{NTT}^{-1}\left(\sum_{i=1}^{v} A_{i,j}^T \vec{\gamma}_i\right)$. Hence, (42) is equivalent to $\sum_{\iota=0}^{l-1} \mathsf{NTT}(\boldsymbol{f})_\iota = 0$ for

$$\boldsymbol{f} = \sum_{j=1}^{n} \boldsymbol{m}_j \boldsymbol{p}_j - \frac{\langle \vec{u}, \vec{\gamma} \rangle}{l}.$$

Note that the verifier can compute a commitment to $\boldsymbol{f}$ by calculating:

$$\boldsymbol{t}_f := \sum_{i=1}^{n} \boldsymbol{t}_i \boldsymbol{p}_i - \frac{\langle \vec{u}, \vec{\gamma} \rangle}{l} = \langle \sum_{i=1}^{n} \boldsymbol{p}_i \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \boldsymbol{f}.$$

Next, we use the following lemma which was first introduced in [15].

Lemma A.2. *Let $\boldsymbol{f} \in \mathcal{R}_q$. Then, $\sum_{j=0}^{l-1} \mathsf{NTT}(f)_j = l \sum_{i=0}^{d/l-1} f_i X_i$.*

From Lemma A.2 we know that if $A\vec{m} = \vec{u}$ then the first d/l coefficients of $\boldsymbol{f}$ are equal to zero. Hence, the idea is to first commit to a uniformly random polynomial $\boldsymbol{g}$ which also has the first d/l coefficients zeroes and then later reveal $\boldsymbol{h} := \boldsymbol{f} + \boldsymbol{g}$. The verifier then can manually check whether the first d/l coefficients of $\boldsymbol{h}$ are indeed equal to zero.

Concretely, the protocol is as follows. Prover $\mathcal{P}$ starts by generating a vector of small polynomials $\vec{\boldsymbol{y}}$ and sending $\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$ as well as a commitment $\boldsymbol{t}_{n+1} = \langle \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}$ to $\boldsymbol{g}$. After getting a challenge $\vec{\gamma} \in W_q^{vl}$, the prover computes $\boldsymbol{f}$ as above and sends $\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g}$. Also, $\mathcal{P}$ sends

$$\boldsymbol{w}' = \langle \sum_{i=1}^{n} \boldsymbol{p}_i \vec{\boldsymbol{b}}_i + \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{y}} \rangle.$$

Next, the verifier selects $\boldsymbol{c} \xleftarrow{\$} C$. Finally, $\mathcal{P}$ outputs $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$ after applying rejection sampling. Then, $\mathcal{V}$ checks that (i) $\vec{\boldsymbol{z}}$ is small, (ii) $\boldsymbol{B}_0\vec{\boldsymbol{z}} = \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$, (iii) the first d/l coefficients of $\boldsymbol{h}$ are all zeroes and lastly:

$$\langle \sum_{i=1}^{n} \boldsymbol{p}_i \vec{\boldsymbol{b}}_i + \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{z}} \rangle = \boldsymbol{w}' + \boldsymbol{c}(\boldsymbol{t}_f + \boldsymbol{t}_{n+1} - \boldsymbol{h}).$$

We observe that when $A\vec{m} \neq \vec{u}$ then the probability that $\langle A\vec{m} - \vec{u}, \vec{\gamma} \rangle = 0$ is equal to $1/|W_q| = q^{-d/l}$. We show in Section A.4 how to decrease this error further if $q^{-d/l}$ is not negligible.

Boosting soundness. The protocol for proving unstructured linear relations in Section A.4 has soundness error $q^{-d/l}$. If this is not negligible then we introduce k independent challenges $\vec{\gamma}_1, \ldots, \vec{\gamma}_k \in W_q^{vl}$ and show that

$$\langle A\vec{m} - \vec{u}, \vec{\gamma}_\iota \rangle = 0 \text{ for } \iota = 0, \ldots, k-1.$$

Thus, we reduce the probability of cheating to $q^{-kd/l}$.

For each $\vec{\gamma}_i$, define $\boldsymbol{f}_i$ and its commitment $\boldsymbol{t}_{f_i}$ similarly as in Section A.4. We show that the first d/l coefficients of $\boldsymbol{f}_i$, i.e. $f_{i,0}, \ldots, f_{i,d/l-1}$ are equal to zero. We do that by showing simultaneously $f_{0,j-1} = \ldots = f_{k-1,j-1} = 0$ for $j \in [d/l]$.

Esgin et al. observe that the polynomial

$$\bar{\boldsymbol{f}}_j := \frac{1}{k} \sum_{\mu=0}^{k-1} X^{\mu d/l} \sum_{\nu=0}^{k-1} \sigma^\nu \left(X^{-j+1} \boldsymbol{f}_\mu \right) \in \mathcal{R}_q$$

has the following property : the $\mu d/l$-th coefficient of $\bar{\boldsymbol{f}}_{j-1}$ is equal to the $(j-1)$-th coefficient of $\boldsymbol{f}_\mu$ (which should be equal to zero). We remark that the verifier can compute a commitment to $\bar{\boldsymbol{f}}_{j-1}$ by calculating:

$$\begin{aligned} \boldsymbol{\tau}_j &= \frac{1}{k} \sum_{\mu=0}^{k-1} X^{\mu d/l} \sum_{\nu=0}^{k-1} \sigma^\nu \left(X^{-j+1} \boldsymbol{t}_{f_\mu} \right) \\ &= \frac{1}{k} \sum_{\mu=0}^{k-1} X^{id/l} \sum_{\nu=0}^{k-1} \sigma^\nu \left(X^{-j+1} \langle \sum_{\eta=1}^{n} \boldsymbol{p}_{\mu,\eta} \vec{\boldsymbol{b}}_\eta, \vec{\boldsymbol{r}} \rangle \right) + \bar{\boldsymbol{f}}_j. \end{aligned} \tag{44}$$

The protocol now looks as follows. Prover $\mathcal{P}$ starts by generating d/l uniformly random polynomials $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{d/l}$ satisfying $g_{j,0} = g_{j,d/l} = \ldots = g_{j,(k-1)d/l} = 0$ and computing commitments $\boldsymbol{t}_{n+j} = \langle \vec{\boldsymbol{b}}_{n+j}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}_j$ for $j \in [d/l]$. Also it samples vectors $\vec{\boldsymbol{y}}_0, \ldots, \vec{\boldsymbol{y}}_{k-1}$ of short polynomials that are going to be used to mask $\vec{\boldsymbol{r}}$ k times with challenges of the form $\sigma^i(\boldsymbol{c})$. Furthermore, $\mathcal{P}$ computes $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$. The prover sends $\boldsymbol{t}_{g_j}$ and $\vec{\boldsymbol{w}}_i$ to $\mathcal{V}$.

Next, the verifier selects uniformly random vectors $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1} \in W_q^{vl}$ and sends them to $\mathcal{P}$. Then, the prover computes $\bar{\boldsymbol{f}}_j$ for each $j \in \mathbb{Z}_{d/l}$. By construction, each $\mu d/l$-th coefficient of $\bar{\boldsymbol{f}}_j$, where $\mu = 0, \ldots, k-1$, is equal to 0. Note that $\mathcal{V}$ can compute a commitment $\boldsymbol{\tau}_j$ to $\bar{\boldsymbol{f}}_j$ as explained above. Now the prover sets $\boldsymbol{h}_j = \bar{\boldsymbol{f}}_j + \boldsymbol{g}_j$ and computes for $i = 0, \ldots, k-1$ and $j = 1, \ldots, d/l$:

$$\begin{aligned} \boldsymbol{w}'_{i,j} = &\sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(X^{-j+1} \langle \sum_{\eta=1}^{n} \boldsymbol{p}_{\mu,\eta} \vec{\boldsymbol{b}}_\eta, \vec{\boldsymbol{y}}_{i-\nu \bmod k} \rangle \right) \\ &+ \langle \vec{\boldsymbol{b}}_{n+j}, \vec{\boldsymbol{y}}_i \rangle. \end{aligned}$$

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

Verifier $\mathcal{V}$ first checks if for all $i = 0, \ldots, k-1$,

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

Then, $\mathcal{V}$ checks that $h_{j,0}, \ldots, h_{j,k-1}$ are all equal to zero and computes $\boldsymbol{\tau}_j$ as in (44) for $j \in \mathbb{Z}_{d/l}$. Finally, the verifier checks whether for all $i = 0, \ldots, k-1$ and $j = 1, \ldots, d/l$,

$$\sum_{\mu=0}^{k-1} \frac{1}{k} X^{\mu} \sum_{\nu=0}^{k-1} \sigma^{\nu} \left(X^{-j+1} \langle \sum_{\eta=1}^{n} \mathbf{p}_{\mu,\eta} \vec{\boldsymbol{b}}_{\eta}, \vec{\boldsymbol{z}}_{i-\nu \bmod k} \rangle \right) + \langle \vec{\boldsymbol{b}}_{n+j}, \vec{\boldsymbol{z}}_i \rangle = \boldsymbol{w}'_{i,j} + \sigma^i(\boldsymbol{c})(\boldsymbol{\tau}_j + \boldsymbol{t}_{n+j} - \boldsymbol{h}_j) \tag{45}$$

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

A.5 Main Protocol

We present a protocol for proving that $\vec{m}$ belongs to $\mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$. As in Section 3.1, the prover $\mathcal{P}$ starts by running $\mathsf{Com}_{n,3}(\vec{m})$. Then, $\mathcal{P}$ runs the protocol in Fig. 8. It applies techniques shown in Sections A.3 and A.4. Soundness error of the protocol is around $q^{-kd/l}$.

We skip the security analysis since it is almost identical to the ones described in [15, Theorem 4.1] and [2, Theorem 5.1]. However, we note that depending on the applications, one has to make sure that it is possible to extract messages $\vec{m}_1, \ldots, \vec{m}_n$ from the language $\mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$ which are over $\mathbb{Z}_q$ and not just W_q. This additional argument, however, is not needed in the case $l = d$ since there we automatically have $\mathbb{Z}_q = W_q$.

Proof size. We now look at the size of the non-interactive proof outputs created by the protocol in Figure 8 similarly as in Section 3.1. First, note that for the non-interactive proof $\vec{\boldsymbol{w}}_i$'s, $\boldsymbol{v}$ and $\boldsymbol{w}'_{ij}s$ need not be included in the output as they are uniquely determined by the remaining components. Further, the challenges can be generated from a small seed of 256 bits, which itself is generated as the hash of some components. Therefore, the contribution of the challenges to the total proof length is extremely small and thus we neglect it.

As "full-sized" elements of $\mathcal{R}_q$, we have $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{n+d/l}, \boldsymbol{t}'_1, \boldsymbol{t}'_2$, and $\boldsymbol{h}_j$ (in fact,all $\boldsymbol{h}_j$ are missing k coefficients, but that is a negligible consideration). Therefore, we have in total $\kappa + n + 2d/l + 2$ full-sized elements of $\mathcal{R}_q$, which altogether costs $(\kappa + n + 2d/l + 2)d \log q$ bits.

Now, the only remaining part is $\vec{\boldsymbol{z}}_i$'s. Due to rejection sampling, each coefficient of $\vec{\boldsymbol{z}}_i$ can be bounded in absolute value by δ_1. If we then take into account the total number of coefficients in $\vec{\boldsymbol{z}}_i$ and an additional sign bit for each coefficient, then we get

$$k \cdot ((\lambda + \kappa + n + d/l + 2) \cdot d) \cdot \log(2\delta_1)$$

bits of communication required for all $\vec{\boldsymbol{z}}_i$'s together.

We set β_1 to be the bound on the infinity norm of the concatenated vector $(\boldsymbol{c}_0\vec{\boldsymbol{r}}, \ldots, \boldsymbol{c}_{k-1}\vec{\boldsymbol{r}})$. Note that $\|\vec{\boldsymbol{r}}\|_\infty = 1$ and $\|\boldsymbol{c}_i\|_1 \leq d$ implies that $\|\boldsymbol{c}\vec{\boldsymbol{r}}\|_\infty \leq d$ for $i = 0, \ldots, k-1$. Therefore, we have the following theoretical bound $\beta_1 = d$. Hence, in order to obtain the expected number of repetitions M for the rejection sampling, one would choose δ_1 which satisfies:

$$1/M = \left(\frac{2(\delta_1 - \beta_1) - 1}{2\delta_1 - 1} \right)^{(\lambda+\kappa+n+d/l+2)kd} \approx e^{-(\lambda+\kappa+n+d/l+2)kd\beta_1/\delta_1}.$$

In conclusion, the overall proof length is about

$$(\kappa + n + 2d/l + 2)d \log q \; + \; k(\lambda + \kappa + n + d/l + 2)d \cdot \log(2\delta_1) \quad \text{bits.} \tag{46}$$

An important advantage of our proof system is that the proof length (i.e., the communication size) is *independent* of the height, m, of the matrix A. Furthermore, one observes that when d/l increases then there are more "full-sized" elements of $\mathcal{R}_q$ and dimensions of the commitment scheme gets larger. On the other hand, since k decreases (for a fixed q), there are less vectors $\vec{\boldsymbol{z}}_i$ of small polynomials. Hence, we need to select parameters such that one part does not dominate the other.

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

Inputs:

$\boldsymbol{B}_0 \in \mathcal{R}_q^{\kappa\times(\lambda+\kappa+n+d/l+2)}; \vec{\boldsymbol{b}}_1 \ldots, \vec{\boldsymbol{b}}_{n+d/l} \in \mathcal{R}_q^{\lambda+\kappa+n+d/l+2}$ | $\boldsymbol{B}_0; \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{n+d/l+2}$

$\vec{\boldsymbol{r}} \in \{-1,0,1\}^{(\lambda+\kappa+n+d/l+2)d} \subset \mathcal{R}_q^{\lambda+\kappa+n+d/l+2}$ | $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{n+d/l}$

$\vec{m} = (\vec{m}_1, \ldots, \vec{m}_n) \in \mathbb{Z}_q^{nl}$

$\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}}$

$\boldsymbol{m}_i = \mathsf{NTT}^{-1}(\vec{m}_i), \boldsymbol{t}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}\rangle + \mathsf{NTT}^{-1}(\vec{m}_i)$ for $i = 1, \ldots, n$

$\vec{\boldsymbol{y}}_0, \ldots, \vec{\boldsymbol{y}}_{k-1} \xleftarrow{\$} S_{\delta_1-1}^{(\lambda+\kappa+n+d/l+2)d}$

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

  $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$

$\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{d/l} \xleftarrow{\$} \{\boldsymbol{f} \in \mathcal{R}_q : f_0 = f_{d/l} = \ldots = f_{(k-1)d/l} = 0\}$

For $i \in [d/l]$: $\boldsymbol{t}_{n+i} = \langle \vec{\boldsymbol{b}}_{n+i}, \vec{\boldsymbol{r}}\rangle + \boldsymbol{g}_i$ $\quad \xrightarrow{\boldsymbol{t}_{n+i}, \vec{\boldsymbol{w}}_i} \quad$ $\vec{\gamma}_0, \ldots, \vec{\gamma}_{k-1} \xleftarrow{\$} W_q^{vl}$

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

  $(\vec{\gamma}_{i,1}, \ldots, \vec{\gamma}_{i,v}) = \vec{\gamma}_i$

$\xleftarrow{\boldsymbol{\alpha}_i, \boldsymbol{\phi}_i, \vec{\gamma}_i} \quad \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\xi, \boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{k-1} \xleftarrow{\$} \mathcal{R}_q$

For $t = 1, \ldots, \xi$:

$$\boldsymbol{\psi}_{t,0} := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell}\langle \boldsymbol{b}_i, \vec{\boldsymbol{y}}\rangle\langle \boldsymbol{b}_j, \vec{\boldsymbol{y}}\rangle\langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}}\rangle$$

$$\boldsymbol{\psi}_{t,1} := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell}\left(\langle \boldsymbol{b}_i, \vec{\boldsymbol{y}}\rangle\langle \boldsymbol{b}_j, \vec{\boldsymbol{y}}\rangle\boldsymbol{m}_\ell + \langle \boldsymbol{b}_i, \vec{\boldsymbol{y}}\rangle\langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}}\rangle\boldsymbol{m}_j + \langle \boldsymbol{b}_j, \vec{\boldsymbol{y}}\rangle\langle \boldsymbol{b}_\ell, \vec{\boldsymbol{y}}\rangle\boldsymbol{m}_i\right) + \sum_{i\le j}\boldsymbol{\eta}_{t,i,j}\langle \boldsymbol{b}_i, \vec{\boldsymbol{y}}\rangle\langle \boldsymbol{b}_j, \vec{\boldsymbol{y}}\rangle,$$

$$\boldsymbol{\psi}_{t,2} := \sum_{i\le j\le \ell} \boldsymbol{\mu}_{t,i,j,\ell}\left(\langle \vec{\boldsymbol{b}}_\ell, \vec{\boldsymbol{y}}\rangle\boldsymbol{m}_i\boldsymbol{m}_j + \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}\rangle\boldsymbol{m}_i\boldsymbol{m}_\ell + \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle\boldsymbol{m}_j\boldsymbol{m}_\ell\right)$$
$$+ \sum_{i\le j}\boldsymbol{\eta}_{t,i,j}\left(\boldsymbol{m}_i\langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}\rangle + \boldsymbol{m}_j\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle\right) + \sum_{i=1}^{n}\boldsymbol{\nu}_{t,i}\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}}\rangle.$$

$$\boldsymbol{t}'_1 = \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{r}}\rangle - \sum_{\iota=0}^{k-1}\boldsymbol{\phi}_\iota\sigma^{-\iota}\left(\sum_{t=1}^{\xi}\boldsymbol{\alpha}_t\boldsymbol{\psi}_{t,2}\right),\ \boldsymbol{t}'_2 = \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{r}}\rangle - \sum_{\iota=0}^{k-1}\boldsymbol{\phi}_\iota\sigma^{-\iota}\left(\sum_{t=1}^{\xi}\boldsymbol{\alpha}_t\boldsymbol{\psi}_{t,1}\right) + \langle \vec{\boldsymbol{b}}_{n+d/l+1}, \vec{\boldsymbol{y}}\rangle$$

$$\boldsymbol{v} = \sum_{\iota=0}^{k-1}\boldsymbol{\phi}_\iota\sigma^{-\iota}\left(\sum_{t=1}^{\xi}\boldsymbol{\alpha}_t\boldsymbol{\psi}_{t,2}\right) - \langle \vec{\boldsymbol{b}}_{n+d/l+2}, \vec{\boldsymbol{y}}\rangle$$

For $\ell = 0, \ldots, k-1, j \in [n]$:

$$\boldsymbol{p}_{\ell,j} = \mathsf{NTT}^{-1}\left(\sum_{i=1}^{v} A_{i,j}^T\vec{\gamma}_{\ell,i}\right)$$

For $i = 0, \ldots, k-1, j \in [d/l]$:

$$\boldsymbol{w}'_{i,j} = \sum_{\mu=0}^{k-1}\frac{1}{k}X^\mu\sum_{\nu=0}^{k-1}\sigma^\nu\left(X^{-j+1}\langle\sum_{\eta=1}^{n}\boldsymbol{p}_{\mu,\eta}\vec{\boldsymbol{b}}_\eta, \vec{\boldsymbol{y}}_{i-\nu \bmod k}\rangle\right) + \langle \vec{\boldsymbol{b}}_{n+j}, \vec{\boldsymbol{y}}_i\rangle$$

For $j \in [d/l]$: $\boldsymbol{h}_j = \frac{1}{k}\sum_{\mu=0}^{k-1}X^\mu\sum_{\nu=0}^{k-1}\sigma^\nu\left(X^{-j+1}\left(\sum_{\eta=1}^{n}\boldsymbol{m}_j\boldsymbol{p}_{\mu,\eta} - \frac{\langle \vec{u}, \vec{\gamma}_\mu\rangle}{l}\right)\right) + \boldsymbol{g}_j$ $\quad \xrightarrow{\boldsymbol{t}'_1, \boldsymbol{t}'_2, \boldsymbol{v}, \boldsymbol{w}'_{i,j}, \boldsymbol{h}_j}$

$\xleftarrow{\boldsymbol{c}} \quad \boldsymbol{c} \xleftarrow{\$} C$

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

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

  If $\|\vec{\boldsymbol{z}}_i\|_\infty \ge \delta_1 - \beta_1$, abort

$\xrightarrow{\vec{\boldsymbol{z}}_i}$

$\mathsf{Ver}(\boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}}_i, \boldsymbol{\alpha}_i, \boldsymbol{\psi}_i, \vec{\gamma}_i, \boldsymbol{t}'_1, \boldsymbol{t}'_2,$
$\boldsymbol{v}, \boldsymbol{w}'_{i,j}, \boldsymbol{h}_j, \boldsymbol{c}, \vec{\boldsymbol{z}}_i)$

Fig. 8. Protocol $\Pi_n^3(\mathsf{pp}, \mathsf{ulp})$ for proving $\vec{m} \in \mathcal{L}_n(\mathsf{pp}, \mathsf{ulp})$ when $\mathcal{R}_q$ splits completely. Denote polynomials in $\mathsf{pp} = \{P_1, \ldots, P_\xi\}$ as in (33). Also, we partition the matrix A as in (43). Verification equations Ver are defined in Figure 9.

```
$\underline{\mathsf{Ver}(\boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}}_i, \boldsymbol{\alpha}_i, \boldsymbol{\psi}_i, \vec{\gamma}_i, \boldsymbol{t}'_1, \boldsymbol{t}'_2, \boldsymbol{w}'_{i,j}, \boldsymbol{h}_j, \boldsymbol{c}, \vec{\boldsymbol{z}}'_i)}$
For $i = 0, \ldots, k-1$:
    $\|\vec{\boldsymbol{z}}_i\|_\infty \overset{?}{<} \delta_1 - \beta_1$
    $\boldsymbol{B}_0 \vec{\boldsymbol{z}}_i \overset{?}{=} \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0$
For $i = 0, \ldots, k-1, \ell \in [n]$ and $j = 1, 2$ :
    $\boldsymbol{f}_\ell^{(i)} = \langle \vec{\boldsymbol{b}}_\ell, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_\ell, \boldsymbol{f}'_j = \langle \vec{\boldsymbol{b}}_{n+j}, \vec{\boldsymbol{z}}_0 \rangle - \boldsymbol{c}\boldsymbol{t}'_j$
Check (41)
For $i = 0, \ldots, k-1$ :
    For $j \in [d/l] : h_{j,id/l} \overset{?}{=} 0$
    For $j \in [n] : \boldsymbol{p}_{i,j} = \mathsf{NTT}^{-1}\left(\sum_{\ell=1}^{v} A_{\ell,j}^T \vec{\gamma}_{i,\ell}\right)$
For $j \in [d/l]$ :
    $\boldsymbol{\tau}_j = \frac{1}{k}\sum_{\mu=0}^{k-1} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(X^{-j+1}\left(\sum_{\eta=1}^{n} \boldsymbol{t}_\mu \boldsymbol{p}_{\mu,\eta} - \frac{\langle \vec{u}, \vec{\gamma}_\mu \rangle}{l}\right)\right)$
For $i = 0, \ldots, k-1$ and $j \in [d/l]$:
    $\sum_{\mu=0}^{k-1} \frac{1}{k} X^\mu \sum_{\nu=0}^{k-1} \sigma^\nu \left(X^{-j+1} \langle \sum_{\eta=1}^{n} \boldsymbol{p}_{\mu,\eta} \vec{\boldsymbol{b}}_\eta, \vec{\boldsymbol{z}}_{i-\nu \bmod k} \rangle\right)$
            $+ \langle \vec{\boldsymbol{b}}_{n+j}, \vec{\boldsymbol{z}}_i \rangle \overset{?}{=} \boldsymbol{w}'_{i,j} + \sigma^i(\boldsymbol{c})(\boldsymbol{\tau}_j + \boldsymbol{t}_{n+j} - \boldsymbol{h}_j)$
```

Fig. 9. Verification equations for Figure 8.

Suppose that we do not use the new opening proof technique from Appendix A.6. Then, we employ the following calculation of a maximum absolute coefficient in $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ as in [15]. For example, set $d = 128$. Now in this case, a coefficient of $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ is the sum of 128 coefficients with i.i.d. $P(-1) = P(1) = 5/32$ and $P(0) = 22/32$.[11] If we calculate the convolution of this distribution, we find that a coefficient is bigger than 78 in absolute value with probability less than 2^{-114}. Hence, by a union bound the probability that any of the coefficients in $\left(\sigma^0(\boldsymbol{c})\vec{\boldsymbol{r}}, \ldots, \sigma^{k-1}(\boldsymbol{c})\vec{\boldsymbol{r}}\right)$ is bigger than 78 will still be negligibly small for $k \leq 4$. Therefore, we can set $\beta_1 = 78$ instead (when $d = 128$).

High/Low Order Bits. In order to reduce the size of the commitment as well as responses $\vec{\boldsymbol{z}}_i$, we apply the compression techniques introduced in [3] for the responses and [13] for the commitment.[12] The high level intuition for both ideas is that it is sometimes not necessary to prove knowledge of an $\vec{\boldsymbol{r}}$ satisfying $\boldsymbol{B}_0\vec{\boldsymbol{r}} = c \cdot \vec{\boldsymbol{t}}_0$; and proving that $\boldsymbol{B}_0\vec{\boldsymbol{r}} \approx c \cdot \vec{\boldsymbol{t}}_0$ is enough (where $\approx$ means that the difference in the ℓ_∞ norms is small).

If one of the elements in $\boldsymbol{B}_0$ is the identity matrix, then one does not need to transmit the masked version of the part in $\vec{\boldsymbol{r}}$ (call it $\vec{\boldsymbol{r}}'$) that is multiplied by it because the product will have small norm. Even though one doesn't transmit this part, one still needs to be careful and do a rejection sampling on it to assure zero-knowledge. Since nothing about $\vec{\boldsymbol{r}}'$ is output, in order to do the simulation, it's important that $\vec{\boldsymbol{r}}'$ does not have any effect on the transmitted part. The main idea is then to make sure that when an integer $z \in \mathbb{Z}_q$ is uniquely decomposed into high-order and low-order components as $2\delta_2 z_1 + z_0$ where $-\delta_2 < z_0 < \delta_2$, adding $\vec{\boldsymbol{r}}'$ should never affect the high-order part of the sum; and then only transmit these high-order bits. The idea for doing this efficiently, and also making it compatible with the non-interactive version of the protocol, was the main result of [3]. In [13], it was shown that one can write

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

and then only transmit $\vec{\boldsymbol{t}}_{0,1}$ as the public key / commitment to the verifier. Because both c and $c \cdot \vec{\boldsymbol{t}}_{0,0}$ have small norms, the effect of $c \cdot \vec{\boldsymbol{t}}_{0,0}$ on the $c \cdot \vec{\boldsymbol{t}}_{0,1}$ part of $c\vec{\boldsymbol{t}}_0$ is only through the carry bits, which can be sent as a "hint vector". We point the reader to [13,25] for a full description of these techniques.

A.6 Improved Opening Proof

We remark that protocol in Figure 8 makes use of the new opening proof introduced in [2]. Concretely, in the final round the prover $\mathcal{P}$ outputs

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

Recall that the parameter β_1 used in rejection sampling is the bound on the infinity norm of secrets we want to mask,i.e.

$$\left(\sigma^0(\boldsymbol{c})\vec{\boldsymbol{r}}, \ldots, \sigma^{k-1}(\boldsymbol{c})\vec{\boldsymbol{r}}\right).$$

In this subsection we show how to reduce β_1 by a factor of k (if the upper-bound on $\|\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}\|_\infty$ is computed naively). Thus, (i) vectors $\vec{\boldsymbol{z}}_i$ get shorter and (ii) the SIS dimension κ decreases. We remark that this improvement is only useful when the polynomial $X^d + 1$ splits completely, hence assume from now on that $l = d$.

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

[12] The aforementioned works were constructing digital signatures. For commitment schemes, the "responses" are analogous to the signature and the "commitment" is the public key.

Let us write $\boldsymbol{c} = \boldsymbol{c}_0 + \boldsymbol{c}_1 X + \ldots + \boldsymbol{c}_{k-1}X^{k-1}$ where

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

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

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

The new opening proof protocol is presented as follows. Prover $\mathcal{P}$ samples coefficients $\vec{\boldsymbol{y}}'_0, \ldots, \vec{\boldsymbol{y}}'_{k-1}$ from $S_{\delta_1 - 1}$ as before. Then, the prover sends $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0 \vec{\boldsymbol{y}}'_i$. Next, it sets

$$\vec{\boldsymbol{y}}_i = \boldsymbol{B}_0 \left(\sum_{j=0}^{k-1} \sigma^i(X^j)\vec{\boldsymbol{y}}'_j \right)$$

and follows the protocol above until the last round. At the end $\mathcal{P}$ outputs:

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

Since each $\boldsymbol{c}_i$ has only d/k non-zero coefficients, we manage to decrease β_1 possibly by a factor of k (in practice the improvement is smaller if one upper-bounds $\|\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}\|$ more cleverly, e.g. as in Section A.5).

After receiving vectors $\vec{\boldsymbol{z}}'_j$, $\mathcal{V}$ first checks whether

$$\boldsymbol{B}_0 \vec{\boldsymbol{z}}_j \stackrel{?}{=} \vec{\boldsymbol{w}}_j + \sigma^j(\boldsymbol{c})\vec{\boldsymbol{t}}_0.$$

for $j = 0, \ldots, k-1$. Then, it computes $\vec{\boldsymbol{z}}_i = \sum_{j=0}^{k-1} \sigma^i(X^j)\vec{\boldsymbol{z}}'_j$ for $i = 0, \ldots, k-1$. The main observation is that by setting $\vec{\boldsymbol{z}}_i$ and $\vec{\boldsymbol{y}}_i$ as above, all the other verification equations stay the same.

We highlight that this technique is considered an improvement only when $l = d$. It can be generalised to cases where $l < d$ but then the prover would send kd/l vectors $\vec{\boldsymbol{z}}'_i$ which is considerably worse than sending k slightly larger vectors $\vec{\boldsymbol{z}}_i$ as in Figure 8.

B Range Proof

Suppose we want to prove knowledge of an integer x which satisfies $x \in [a, b]$, i.e. $a \leq x \leq b$ for publicly known integers a, b. Attema et al. [2] consider a special type of a range proof $x \in [0, 2^N)$. Recall that the idea is to decompose $x = \sum_{i=0}^{N-1} x_i 2^i$ into N bits and set $\vec{x} = (x_0, \ldots, x_{N-1})$ (this idea was already considered in previous works e.g. [16]). Then, they show that $\vec{x}$ is a binary vector using their product proof. For 32-bit range, Attema et al. obtain proofs of size approximately 5.9KB.

In this subsection we consider range proofs for arbitrary ranges $[a, b] \subseteq [-2^{N-1}, 2^{N-1} - 1]$. We consider two cases.

Case 1 : $b - a \leq 2^{N-1} - 1$. First, note that $a \leq x \leq b$ if and only if there exist non-negative integers $y, z \leq 2^{N-1} - 1$ such that:

$$a + y = x \text{ and } x + z = b.$$

Since $a, x, y, z, b \in [-2^{N-1}, 2^{N-1} - 1]$, we informally apply two proofs of integer addition from Section 4.1. There are, however, a few small issues with this approach. First, integers a and b are known, hence we do not need to send commitments to them. Furthermore, we have to prove that y and z are non-negative. This can be done as follows. Let $(y_0, \ldots, y_{N-1}) = \mathsf{TC}_N(y)$ be the two's complement bit representation of y (and similarly for $\vec{a}, \vec{b}, \vec{z}$). Then, $y \geq 0$ if and only if $y_{N-1} = 0$. Therefore, we can simply define additional polynomials $P_y, P_z \in \mathsf{pp}$ which will check that $y_{N-1} = z_{N-1} = 0$.

Overall, prover $\mathcal{P}$ commits to three main vectors $\vec{x}, \vec{y}, \vec{z}$ (recall that $\vec{a}, \vec{b}$ are known) as well as 2 "carry" vectors $\vec{f}, \vec{f}'$. Similarly as in (11), the following holds:

$$\begin{cases} \vec{f} + \vec{a} + \vec{y} = \vec{x} + 2J\vec{f} \\ \vec{f}' + \vec{x} + \vec{z} = \vec{b} + 2J\vec{f}' \end{cases} \tag{47}$$

We recall that matrix J is defined in (12).

Let $\vec{m} = (\vec{x}, \vec{y}, \vec{z}, \vec{f}, \vec{f}')$. We prove that $\vec{m} \in \{0,1\}^{5N}$ and $f_0 = f'_0 = y_{N-1} = z_{N-1} = 0$ by defining a set pp of multivariate polynomials similarly as in Section 4.1. These relations, along with (47), can be captured in one single equation $A\vec{m} = \vec{u}$ where

$$A = \begin{pmatrix} -I_N & I_N & 0_N & (I_N - 2J) & 0_N \\ I_N & 0_N & I_N & 0_N & (I_N - 2J) \end{pmatrix} \text{ and } \vec{u} = \begin{pmatrix} -\vec{a} \\ \vec{b} \end{pmatrix}. \tag{48}$$

Let $\mathsf{ulp} = (A, \vec{u})$. Thus, we reduced the problem of proving $x \in [a,b]$ to showing that $\vec{m} \in \mathcal{L}_{5\gamma}(\mathsf{pp}, \mathsf{ulp})$ where $\gamma = N/l$.

Case 2: $b - a \geq 2^{N-1}$. The key observation is that $a \leq x \leq b$ if and only if there exist integers $y, z \in [-2^{N-1}, 2^{N-1} - 1]$ such that

$$(a + 2^{N-1}) + y = x \text{ and } x + z = b - 2^{N-1}.$$

Also, $a + 2^{N-1}, b - 2^{N-1} \in [-2^{N-1}, 2^{N-1} - 1]$ since $b - a \geq 2^{N-1}$. Therefore, we can combine two proofs of integer addition similarly as in the first case. The only differences here are: (i) we do not need to check whether y and z are positive, (ii) public vectors $\vec{a}$ and $\vec{b}$ now represent $a + 2^{N-1}$ and $b - 2^{N-1}$ respectively. Consequently, the proof size becomes identical as in Case 1.

N	l	k	κ	λ	δ_1	δ_2	D	proof size
32	32	1	11	10	2^{17}	$(q-1)/2^{14}$	11	11.8KB
128	128	4	10	10	2^{18}	$(q-1)/2^{13}$	14	26.4KB
512	128	4	10	10	2^{18}	$(q-1)/2^{13}$	14	51.3KB

Fig. 10. Proof size comparison for proving $x \in [a,b] \subseteq [-2^{N-1}, 2^{N-1} - 1]$. In each scenario, we pick $q \approx 2^{30}$ and $d = 128$. Here, parameters δ_1, δ_2, D are used for the commitment compression (see Section A.5).

C More on Integer Multiplication

C.1 Proof of Lemma 5

Lemma C.1. *Let $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{Z}[X]$ be polynomials with coefficients in $\{-1, 0, 1\}$ such that $\deg(\boldsymbol{a}), \deg(\boldsymbol{b}) < N$ and $\deg(\boldsymbol{c}) < 2N$. Suppose there exists a polynomial $\boldsymbol{f}$ of degree at most $2N - 1$ which satisfies (18). Then, for each coefficient f_k of $\boldsymbol{f}$ corresponding to X^k, $|f_k| \leq N + 1$.*

Proof. We first show $f_0 \in \{-1, 0, 1\}$. Consider Equation 18 for $X = 0$. Then, we have $a_0 b_0 - c_0 = -2f_0$. Since $-2 \leq a_0 b_0 - c_0 \leq 2$, we get $|f_0| \leq 1$.

In general, by considering the k-th coefficient of $\boldsymbol{ab} - \boldsymbol{c}$ and $(X-2)\boldsymbol{f}$ for $k > 0$, we have the following equality:

$$\sum_{0 \le i,j < N \text{ s.t. } i+j=k} a_i b_j - c_k = f_{k-1} - 2f_k.$$

Hence, we get $|f_{k-1} - 2f_k| \le N+1$. In particular, by the triangle inequality:

$$|f_k| \le \frac{|f_{k-1} - 2f_k| + |f_{k-1}|}{2} \le \frac{N+1}{2} + \frac{|f_{k-1}|}{2}.$$

Thus, $|f_1| \le N/2 + 1$. Then, one can show by induction that

$$|f_k| \le (N+1)(1/2 + 1/4 + 1/8 + ... + 1/2^k) + 1/2^k < (N+1) + 1/2^k$$

for $k \ge 1$. Since $f_k \in \mathbb{Z}$, we have $|f_k| \le N+1$.

C.2 Various Optimisations

Reducing the numbers of $\vec{e}_i$. The relaxed range proof for $\vec{f}$ requires us to introduce vectors $\vec{e}_1, \ldots, \vec{e}_{d/l} \in \mathbb{Z}_q^l$ to obtain negligible probability in Lemma 2.4. Notably, we need d/l commitments to represent $\vec{e}$. Here, we show how to decrease the number of commitments by slightly modifying the relaxed range proof.

Concretely, suppose the prover samples coefficients of $\vec{e} \in W_q^l$ uniformly at random from $[-\delta_1' + 1, \delta_1' - 1]$ and sends its commitment $\boldsymbol{t}_{15} = \langle \vec{\boldsymbol{b}}_{15}, \vec{\boldsymbol{r}} \rangle + \mathsf{NTT}^{-1}(\vec{e})$ to $\mathcal{V}$. Then, the verifier generates a matrix $B \in W_q^{l \times 2l}$ of uniformly random binary polynomials in W_q and sends it to $\mathcal{P}$. Finally, the prover computes $\vec{g} = B\vec{f} + \vec{e} \in W_q^l$ and outputs $\vec{g}$ unless the rejection sampling aborts. The key observation here is that Lemma 2.4 can be also applied in this setting. Namely, the probability that $\|B\vec{f} + \vec{e}\|_\infty < \|\vec{f}\|_\infty$ is still at most $1/2^d$.

At the end, $\mathcal{P}$ has to show that $B\vec{f} + \vec{e} = \vec{g}$ where $B, \vec{g}$ are known to the verifier. This is clearly outside of our framework since we only consider messages and their linear relations over $\mathbb{Z}_q$. However, we observe that the techniques described in Appendix A.4 can be easily extended to prove such relations as long as $\vec{f}$ is over $\mathbb{Z}_q$.

With this modification, we reduce the number of commitments by $d/l - 1$. We already include this improvement when calculating proof sizes in Figure 6.

Fully-splitting ring. In the protocol described in Figure 5 we need to commit to $14 + d/l$ messages. We show how to reduce this number in the fully-splitting case, i.e. $l = d$ and $W_q = \mathbb{Z}_q$. Thus, we do not have to argue about vectors being over integers.

An alternative way to prove (18) is via a linear proof. Let us commit to $\vec{h} = \vec{a}' \cdot \vec{b}'$. Later we prove, via a product proof, that $\vec{h}$ is well-formed. Then, $V^{-1}\vec{h}$ is indeed a coefficient vector of the polynomial $\boldsymbol{ab}$. Therefore, Equation (18) can be equivalently written as:

$$V^{-1}\vec{h} = \vec{c} + J_2\vec{f}$$

where

$$J_2 = \begin{pmatrix} -2 & 0 & 0 \cdots & 0 & 1 \\ 1 & -2 & 0 \cdots & 0 & 0 \\ \vdots & \vdots & \vdots \; \vdots & \vdots & \\ 0 & 0 & 0 \cdots & -2 & 0 \\ 0 & 0 & 0 \cdots & 1 & -2 \end{pmatrix} \in \mathbb{Z}_q^{2l \times 2l}$$

represents multiplication by a factor $X - 2$ over $R_{q,2l}$.

The advantage of this approach is that we do not have to commit to $\vec{c}'$ and $\vec{f}'$, and hence reduce the number of messages to $12 + d/l$. We apply this in our implementation.

Large integers. Suppose that $N = \gamma d$ where $\gamma > 1$. Let us select q such that $X^{2N} + 1$ splits completely modulo q, i.e. $4N|q-1$. Then, we also have $2d|q-1$ and hence $l = d$.

The idea is to follow the argument in Section 5 over the ring $R = \mathbb{Z}_q[X]/(X^{2N}+1)$ instead of $R_{q,2l}$. In this setting, we would need γ and 2γ commitments to represent $\vec{a} \in \mathbb{Z}_q^N$ and $\vec{a}' = V_1\vec{a} \in \mathbb{Z}_q^{2N}$ respectively. By doing similar analysis for $\vec{b}, \vec{c}, \vec{f}$ and accounting for the optimization from Section C.2, the total number of commitments the prover has to send is $12\gamma + 1$.