More Efficient Amortization of Exact Zero-Knowledge Proofs for LWE

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

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

Abstract. We propose a practical zero-knowledge proof system for proving knowledge of short solutions $\mathbf{s}, \mathbf{e}$ to linear relations $\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{u} \pmod q$ which gives the most efficient solution for two naturally-occurring classes of problems. The first is when $\mathbf{A}$ is very "tall", which corresponds to a large number of LWE instances that use the same secret $\mathbf{s}$. In this case, we show that the proof size is independent of the height of the matrix (and thus the length of the error vector $\mathbf{e}$) and rather only linearly depends on the length of $\mathbf{s}$. The second case is when $\mathbf{A}$ is of the form $\mathbf{A}' \otimes \mathbf{I}$, which corresponds to proving many LWE instances (with different secrets) that use the same samples $\mathbf{A}'$. The length of this second proof is square root in the length of $\mathbf{s}$, which corresponds to square root of the length of all the secrets. Our constructions combine recent advances in "purely" lattice-based zero-knowledge proofs with the Reed-Solomon proximity testing ideas present in some generic zero-knowledge proof systems – with the main difference is that the latter are applied directly to the lattice instances without going through intermediate problems.

Keywords. Lattices, Zero-Knowledge Proofs, LWE, Amortization

1 Introduction

Zero-knowledge proofs, in which a prover convinces a verifier of knowledge of a witness to the fact that an instance belongs to a language, are an integral cryptographic building block. For relations among values (e.g. public keys, ciphertexts, commitments, etc.) stemming from classical cryptography based on the hardness of discrete logarithm and factoring, there exist many very efficient (even succinct) zero-knowledge proofs. When it comes to proving relations between public and secret information for *lattice* primitives, however, the landscape of efficient zero-knowledge proofs is significantly less advanced. The fundamental lattice problem upon which most of lattice cryptography rests is the LWE problem, which states that it is hard to distinguish a uniformly random tuple $(\mathbf{A}, \mathbf{u})$ and $(\mathbf{A}, \mathbf{u} = \mathbf{A}\mathbf{s} + \mathbf{e})$, where the coefficients of $\mathbf{s}$ and $\mathbf{e}$ are small. The main techniques of this paper will prove knowledge of $\mathbf{s}$ and $\mathbf{e}$ with small coefficients that satisfy

$$\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{u}. \tag{1}$$

As is typical with zero-knowledge proofs, there is no one technique that is best for all scenarios. Similarly, our proofs in this paper will not be the shortest for all the parametrizations of the above equation, but they will be the most compact for some important parameter settings which intuitively correspond to simultaneously proving many LWE instances at the same time.

1.1 Prior Work

Relaxed Proofs. The most efficient proofs of equations of the form as in (1) work over some polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ rather than over $\mathbb{Z}_q$, and are able to prove knowledge of vectors of polynomials $\bar{\mathbf{s}}, \bar{\mathbf{e}}$ with small coefficients (but larger than the ones in $\mathbf{s}$ and $\mathbf{e}$) and a polynomial $\bar{c}$ with $-1/0/1$ coefficients satisfying $\mathbf{A}\bar{\mathbf{s}} + \bar{\mathbf{e}} = \bar{c}\mathbf{t}$. While not exactly proving (1), this zero-knowledge proof is enough to obtain very efficient lattice-based digital signatures (e.g. [DKL+18]) and rather efficient commitment schemes with zero-knowledge openings [BDL+18]. Another variation of the proof is just like above except there is no multiplicative factor

$\bar{c}$. Such proofs are particularly efficient in the amortized setting [BBC$^+$18] and are useful as a preprocessing step in certain multi-party protocols [BCS19] or in voting protocols where the authorities perform many simultaneous proofs [dPLNS17].

While the above relaxations of the relation in (1) have found some useful applications, especially when they are used in standalone protocols (e.g. [EZS$^+$19,EKS$^+$20]), they are not very useful for proving relations in schemes for which the parameters have been optimally set according to some external constraints. Because the proofs only show knowledge of larger $\bar{\mathbf{s}}$ and $\bar{\mathbf{e}}$ than the ones used by an honest prover, it necessitates increasing the sizes of other parameters (like q and n, m) in order to obtain the same security level. Using relaxed proofs would require us to increase the parameters in these schemes solely for the purpose of being compatible with the (possibly seldom-used) zero-knowledge proofs. In order to avoid this inefficiency, it is necessary to have a proof which proves that the secrets are in exactly the same range as is used by the honest prover.

Exact proofs. One technique for getting short and exact proofs of (1) was given in [dPLS19] where the idea is to first convert (1) to an equivalent (statistically-hiding) discrete-logarithm relation, and then prove knowledge of exponents corresponding to the coefficients of $\mathbf{s}, \mathbf{e}$ using Bulletproofs [BCC$^+$16,BBB$^+$18]. The resulting proof is just a few kilobytes, but has the shortcomings of being (very) slow and not fully quantum-safe. In particular, both the prover and the verifier are required to perform on the order of hundreds of thousands of exponentiations, even for fairly modest sizes of the parameters in (1), which requires a few dozen seconds and does not scale well for larger instances. In terms of quantum-security, while the proof is statistical zero-knowledge, soundness is only based on the hardness of the discrete logarithm problem.

Another strategy uses information-theoretic proof systems such as PCPs, interactive PCPs [KR08,BCS16], or interactive oracle proofs (IOPs) [BCS16,RRR16]. In these proof systems, the verifier does not read the prover's messages in their entirety, but rather makes a sublinear number of queries to individual message positions for verification. Given a suitable PCP or IOP, one can convert it into a sublinear-sized cryptographic argument by following the approach of Kilian [Kil92] and Micali [Mic00]. In the resulting argument, the prover commits to each of their proof messages using a Merkle tree, and opens individual message positions using Merkle paths. Thus, security is based solely on collision resistant hash functions, which are quantum-safe.

Various prior works (e.g. [BCR$^+$19,AHIV17,BCG$^+$17]) produce such arguments by designing IOPs for the R1CS or circuit satisfiability problems, which are NP complete. To prove other relations, one must first convert them into suitable problem instances. Unfortunately, directly applying these arguments to lattice relations (e.g. via conversion to R1CS) can be extremely resource-intensive. For example, [BCOS20] reported that constructing a group signature using the arguments from [BCR$^+$19] resulted in rather short outputs (of about 100KB), but they were not able to sign due to the (cloud) PC running out of memory. Thus, to better use this strategy, one should design PCPs and IOPs which target (1) directly. The work of [BCG$^+$17] gives one way of doing this, by giving IOPs which simulate the behaviour of zero-knowledge arguments that use homomorphic commitments. More precisely, [BCG$^+$17] shows how to compile 'ideal linear commitment' (ILC) protocols, a type of information-theoretic protocol, into IOP protocols, by encoding each of the prover's messages using an error-correcting code. Taking the ILC-to-IOP and IOP-to-argument transformations together, gives an 'encode-then-hash' method for committing to messages which acts essentially like a homomorphic commitment scheme.

Our approach can be seen as a combination of algebraic techniques from previous lattice-based works, which design zero-knowledge arguments for (1) directly based on homomorphic, lattice-based commitments, but replacing the lattice-based commitments with the 'encode-then-hash' commitment scheme implicit in [BCG$^+$17]. The main difference between our work and IOP constructions such as [BCG$^+$17,BCR$^+$19,AHIV17,BCG$^+$17] is that we do not require a (possibly costly) reduction to the intermediate R1CS problem, instead taking inspiration from lattice-based protocols (e.g. [BLS19]) which handle (1) with only a small number of commitments.

Until recently, the shortest fully quantum-safe proofs for proving the exact version of (1) have been direct adaptations [KTX08,LNSW13] of Stern's original protocol [Ste93] for proving knowledge of low-weight codewords over $\mathbb{Z}_2$. The work of [Beu20] used a cut-and-choose approach to leverage the larger field size in (1)

Size of Secret Set	Proof Size (KB)	Prover Time (sec)	Verifier Time (sec)
4	209	0.78	0.06
8	224	1.12	0.06
16	254	1.77	0.06
32	314	3.13	0.07
64	434	6.03	0.08
128	674	12.97	0.10
256	1154	31.27	0.15

Table 1. Proof sizes and single-core running times (implemented in C++ using NTL [Sho], running on a Skylake processor) for the proof system in Figure 1 when $\mathbf{A} \in \mathcal{R}_q^{64\times 1}$ (i.e. 64 Ring-LWE instances) where $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ for $q \approx 2^{60}$ and $d = 2048$. The secret vectors (over $\mathcal{R}_q$) $\mathbf{s}, \mathbf{e}$ each have coefficients coming from a set of a size specified in the first column rather than coming from the set $\{-1, 0, 1\}$ as specified in Figure 1, so the protocol has to be adjusted as described in the caption of that Figure. The soundness error is around 2^{-57}, so one may need to repeat the proof twice to achieve cryptographic soundness. This will roughly double the proof size and running time. The size of the commitment/ciphertext $\mathbf{u}$ is 960 KB. The proof sizes would be the same if $\mathbf{A}$ were an unstructured matrix in $\mathbb{Z}_q^{n\times m}$ where $n = 64 \cdot 2048$ and $m = 2048$. The running times, however, would be higher because one can no longer use NTT for performing fast multiplications $\mathbf{As}$. For the same parameters, the proof size per 2048×2048 dimensional instance (of which there are 64 in our example proof) from [ENS20] is 45 KB when the size of the secret set is 3.

to obtain a more efficient generalization of Stern's protocol. Using very different techniques, the works of [BLS19,YAZ$^+$19] achieved a slightly shorter proof for (1) when the secret coefficients are chosen from the set $\{-1, 0, 1\}$. For $n = m = 1024$, and $q = 2^{32}$, the proofs (with 2^{-128} soundness error) are around 400 KB long. Further building on these results, the currently shortest proof is a little under 50KB [ENS20]. For direct comparisons with the results in this paper, we also computed the parameter sizes using the techniques in [ENS20] for $n = m = 2048, q \approx 2^{60}$, and 2^{-57} soundness. The proof size for these parameters is around 45KB.

1.2 Our results.

In this work we give *exact proofs* for several scenarios in which one needs to prove many LWE instances. The first case is proving (1) for the case where $\mathbf{A}$ is a tall matrix. In our running example, we'll have $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ where $m = 2048$ and $n = 64 \cdot 2048$. This can be seen as 64 LWE instances where the dimensions of $\mathbf{A}$ are a square 2048×2048. A simple example where this comes up in practice is when one encrypts a long message using a symmetric LWE encryption scheme (e.g. for FHE applications) or if encrypting a message to different public keys using the same randomness (as in e.g. [PVW08,KKPP20])[3].

In Table 1.2, we give the proof size and running time for our problem instance, where the dependence is on the size of the set from which the coefficients of $\mathbf{s}$ and $\mathbf{e}$ are chosen. In the example from [ENS20] mentioned at the end of the last section, these were chosen from the set $\{-1, 0, 1\}$ of size 3. Thus from the first line of Table 1.2, we see that the amortized proof is about 3 KB per instance, which is more than an order of magnitude improvement in size.

The second scenario for which we provide improved proofs is for the case when we have many equations as in (1) with the same public randomness $\mathbf{A}$ but different $\mathbf{s}$ and $\mathbf{e}$. In a way, it complements our first result in which the LWE instances had the same secret, but different public randomness. In this scenario, we give a proof that is square root in the size of the secret, and it produces proofs that are several times smaller than the non-amortized version. We did not implement this scheme, but as it uses essentially the same operations as the one in our first scenario, the running times should be comparable in practice.

[3] Incorporating a message vector $\mathbf{m}$ in (1) simply involves rewriting $\mathbf{e} = \mathbf{e}' + \lfloor q/2 \rceil \cdot \mathbf{m}$ where $\mathbf{e}'$ is the LWE error

1.3 Technical Overview

As mentioned previously, the strategy employed in our basic protocol uses ideas from the lattice-based schemes of [BLS19,YAZ$^+$19] in combination with the 'encode-then-hash' commitment scheme implicit in [BCG$^+$17]. This latter building block of the proof is an additively-homomorphic commitment scheme $\mathtt{Com}$ committing to vectors in $\mathbb{Z}_q^m$ and possessing an efficient ZKPoK of the committed value. That is, if $S = \mathtt{Com}(\mathbf{s})$ and $T = \mathtt{Com}(\mathbf{t})$, then for any $c \in \mathbb{Z}_q$, $\mathbf{s} + \mathbf{t}c = \mathtt{Open}(S + Tc)$. Let us, for now, just take this scheme as a black box. As an additional piece of notation, we define $\mathbf{1}$ to be a vector all of whose coefficients are 1 and for two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^m$, we define $\mathbf{v} \circ \mathbf{w}$ to be the component-wise product of $\mathbf{v}$ and $\mathbf{w}$. We will now give a simplified version of the protocol in Figure 1 where the prover is trying to convince the verifier that $\mathbf{s}, \mathbf{e} \in \{0,1\}^m$.

The prover starts out by choosing a uniformly-random masking vector $\mathbf{t} \in \mathbb{Z}_q^m$ and creating a commitment $T = \mathtt{Com}(\mathbf{t})$ and $S = \mathtt{Com}(\mathbf{s})$. At the end of the protocol, the prover will eventually send the polynomial $\mathbf{t}X + \mathbf{s}$ evaluated at the challenge $X = x \in \mathbb{Z}_q$. If we write $\mathbf{f} = \mathbf{t}X + \mathbf{s}$, then

$$\mathbf{f} \circ (\mathbf{f} - \mathbf{1}) = (\mathbf{t} \circ \mathbf{t})X^2 + \mathbf{t} \circ (2\mathbf{s} - \mathbf{1})X + \mathbf{s} \circ (\mathbf{s} - \mathbf{1}) \tag{2}$$

If all the coefficients in $\mathbf{s}$ are 0/1, then the constant term will be 0; and so the equation $\frac{1}{X} \cdot \mathbf{f} \circ (\mathbf{f} - \mathbf{1})$ will be the linear equation $\mathbf{v}_1 X + \mathbf{v}_0$ where $\mathbf{v}_1 = \mathbf{t} \circ \mathbf{t}$ and $\mathbf{v}_0 = \mathbf{t} \circ (2\mathbf{s} - \mathbf{1})$. The prover creates commitments $V_1 = \mathtt{Com}(\mathbf{v}_1)$ and $V_0 = \mathtt{Com}(\mathbf{v}_0)$.

The prover also defines

$$\mathbf{d} = \mathbf{u} - \mathbf{A}\mathbf{f} = \mathbf{u} - \mathbf{A}\mathbf{s} - \mathbf{A}\mathbf{t}X = \mathbf{e} - \mathbf{A}\mathbf{t}X \tag{3}$$

and similarly observes that

$$\mathbf{d} \circ (\mathbf{d} - \mathbf{1}) = (\mathbf{A}\mathbf{t}) \circ (\mathbf{A}\mathbf{t})X^2 + (\mathbf{A}\mathbf{t}) \circ (2\mathbf{e} + 1)X + \mathbf{e} \circ (\mathbf{e} - \mathbf{1}), \tag{4}$$

and therefore $\frac{1}{X} \cdot \mathbf{d} \circ (\mathbf{d} - \mathbf{1})$ will also be a linear equation if and only if all the coefficients of $\mathbf{e}$ are 0/1. The prover similarly creates commitments $W_1 = \mathtt{Com}((\mathbf{A}\mathbf{t}) \circ (\mathbf{A}\mathbf{t}))$ and $W_0 = \mathtt{Com}((\mathbf{A}\mathbf{t}) \circ (2\mathbf{e} + \mathbf{1}))$.

We now begin the description of the interactive protocol. The prover sends the commitments S, T, V_0, V_1, W_0, W_1, and the verifier picks a uniformly-random challenge $x \in \mathbb{Z}_q \setminus \{0\}$. The prover responds with $\mathbf{f} = \mathbf{t}x + \mathbf{s}$ and zero-knowledge proofs of knowledge of the committed values in S, T, V_0, V_1, W_0, W_1, and zero-knowledge proofs that

$$\mathbf{f} = \mathtt{Open}(Tx + S) \tag{5}$$

$$\frac{1}{x} \cdot (\mathbf{f}) \circ (\mathbf{f} - \mathbf{1}) = \mathtt{Open}(V_1 x + V_0) \tag{6}$$

$$\frac{1}{x} \cdot (\mathbf{u} - \mathbf{A}\mathbf{f}) \circ (\mathbf{u} - \mathbf{A}\mathbf{f} - \mathbf{1}) = \mathtt{Open}(W_1 x + W_0) \tag{7}$$

The first proof implies knowledge of some $\mathbf{s}$, $\mathbf{t}$ satisfying $\mathbf{f} = \mathbf{t}x + \mathbf{s}$. Via a Schwartz-Zippel argument, the second proof, together with (2), implies that $\mathbf{s}$ has 0/1 coefficients. Similarly, the third proof and (3) imply that $\mathbf{u} - \mathbf{A}\mathbf{s}$ has 0/1 coefficients. Since the verifier has $\mathbf{f}$ and $\mathbf{u}$, he can verify all three proofs (as well as the proofs of knowledge of the committed values) and conclude that the prover knows $\mathbf{s}, \mathbf{e}$ with 0/1 coefficients such that $\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{u}$. The soundness error of this proof is approximately $1/q$, and so if q is not very large, the proof needs to be repeated several times for soundness amplification.

The Commitment Scheme. We will use the 'encode-then-hash' commitment scheme resulting from combining the two transformations from ILC-to-IOP and from IOP-to-arguments given in [BCG$^+$17]. This can be viewed as an interactive commitment scheme (which can be made non-interactive using the Fiat-Shamir transform) in which the vectors to which we would like to commit, appended with some randomness, are first encoded using a linear error-correcting code (a Reed-Solomon code). If the length of the codeword is L, then the prover creates L hashes where the input to the i^{th} hash are all the elements in the i^{th} position of all codewords. These hashes are then put into a Merkle tree and the root is transmitted as the commitment. Proving the knowledge of committed values is done via a "cut-and-choose" approach where the verifier sends a random set of size τ and asks the prover to open L in those τ positions. If τ is smaller than the number of randomness positions, then revealing τ positions information-theoretically hides the message and so the commitment scheme is statistically-hiding.

The verifier can then check whether (5) is satisfied restricted to the τ positions opened by the prover, and due to the fact that $\mathbf{f}$ is given in the clear, one can conclude using a cut-and-choose argument and some properties of Reed-Solomon codes that the values committed in T and S are indeed close to valid codewords (and thus can be decoded to some $\mathbf{s}$ and $\mathbf{t}$), and $\mathbf{f}$ is really the linear combination $\mathbf{t}x+\mathbf{s}$. The same arguments are used to show that (6) and (7) are satisfied.

Amortizing when the public randomness is fixed. An expensive part of our protocol is proving that the correct positions in the Merkle tree have been opened. Each of the τ positions that are opened require the prover to give a path to the root of the tree, which consists of $\log L$ hash function outputs – if one uses SHA-256, then one needs to $32\tau \log L$ bytes for this part of the proof. In general, τ is a small multiple of the security parameter (in practice, ≈ 512) while L is a small multiple of the length of the vectors that are being committed to. In the scheme implemented in Table 1.2, $L \approx 2^{19}$, and therefore just the tree opening would require close to 300kB. One can optimize this by having multiple roots, (which lowers the number of levels), but this is still the most expensive part of the proof when the size of the secret set is not too large.

A significant saving can be achieved when the matrix $\mathbf{A}$ is the same for all the equations $\mathbf{A}\mathbf{s}_j + \mathbf{e}_j = \mathbf{u}_j$ for $j = 1, \ldots, r$. Then all the secret vectors $\mathbf{s}_j$ can be packed into only one masked opening $\mathbf{f} = x_0\mathbf{t} + \sum_j x_j\mathbf{s}_j$ where the secret vectors are separated by different challenges x_j. The verifier can then still compute $\mathbf{d} = \sum_j x_j\mathbf{u}_j - \mathbf{A}\mathbf{f} = -x_0\mathbf{A}\mathbf{t} + \sum_j x_j\mathbf{e}_j$, which is a masked opening of all the error vectors $\mathbf{e}_j$ in the same form as $\mathbf{f}$. The masked opening $\mathbf{f}$ is the second biggest part of our basic protocol after the Merkle tree paths and so amortizing its size over many equations gives a further saving of about a factor of 2 in the per-equation cost. Now, if one computes a quadratic expression of the form $\mathbf{f} \circ (\mathbf{f} - \sum_j x_j\mathbf{1})$ then the terms x_j^2 vanish if and only if the $\mathbf{s}_j$ have binary coefficients. We take the challenges x_j to be evaluations $x_j = \ell_j(x)$ of Lagrange interpolation polynomials at the same evaluation point x. This has the advantage that the number of non-vanishing garbage terms that appear in the quadratic expression (i.e. the terms to which we need to commit and transmit the commitments) for proving 0/1 scales only linearly in r, as will be explained in Section 4. If one has m^2 secret coefficients distributed over m vectors, each of length m, then our final amortized protocol has communication cost of order m, i.e. of order square root in the number of secret coefficients. This is because there is only one masked opening $\mathbf{f}$ of length m for all m^2 secret coefficients.

2 Preliminaries

Notation. Let q be a prime. We write $\mathbb{Z}_q$ for the ring of integers modulo q. Bold letters as in $\mathbf{v} \in \mathbb{Z}_q^l$ will denote vectors over $\mathbb{Z}_q$ and matrices will be written as regular capital letters M.

Reed-Solomon Codes. Let l, k' be positive integers. Let $a_1, \ldots, a_l$ be distinct elements of $\mathbb{Z}_q$. The subspace $\mathcal{C} \subset \mathbb{Z}_q^l$ of $\mathbb{Z}_q^l$ of degree k' consisting of all l-tuples $\mathsf{Enc}(f) = (f(a_1), \ldots, f(a_l))$ where f is a polynomial of degree less than k' with coefficients in $\mathbb{Z}_q$ is a so-called *Reed-Solomon* code. We write $d(\mathbf{V}, \mathbf{W})$ for the Hamming distance between two elements of $\mathbb{Z}_q^l$. Since a polynomial of degree less than k' can have at most $k' - 1$ roots, it is clear that the minimum distance between codewords in $\mathcal{C}$ is $d = l - k' + 1$. This means that if $\mathbf{V}$ is a vector in $\mathbb{Z}_q^l$ that we know has distance at most $(d-1)/2$ from $\mathcal{C}$, then the vector can be uniquely decoded to the closest codeword in $\mathcal{C}$.

The usefulness of Reed-Solomon codes for zero-knowledge proofs stems from the following facts.

Firstly, the encoding function is homomorphic. More precisely, polynomial addition and multiplication translate to coefficient-wise addition and multiplication on the codewords.

Secondly, assume that the prover has committed to several codewords. Then, let the verifier only get to see a small number, say τ, of openings of random positions of his choice from all of the committed codewords. If he now checks that a random linear combination of these positions coincide with the positions of a fixed known codeword, then he will be convinced that the prover has honestly committed to codewords with sufficiently few errors that they decode to polynomials whose linear combination is the same as the decoding of the known codeword.

Moreover, if the k' coefficients of the input polynomials consist of m message coefficients and τ randomness coefficients, then the random τ openings do not reveal any information about the message coefficients.

We will write $\mathsf{Enc}\left(\mathbf{m},\mathbf{r}\right)$ to denote the Reed-Solomon codeword corresponding to the input polynomial $f=\sum_{i=0}^{m-1}\mathbf{m}_i X^i + X^m\sum_{i=0}^{\tau-1}\mathbf{r}_i X^i$ with message coefficients $\mathbf{m}$ and randomness coefficients $\mathbf{r}$. We will also extend this notation in the straight-forward way to split the input polynomial into even more coefficient vectors.

So, in summary, Reed-Solomon codes offer a way to commit to message vectors with the ability to prove linear relations between these vectors in zero-knowledge. We perform hash-based commitments to codewords with the technique from [BCG+17] that we recall in the next paragraphs.

Commitments. Let $M=\left(\mathbf{m}_1\ \mathbf{m}_2\ \ldots\ \mathbf{m}_l\right)$ be a matrix made up of column vectors $\mathbf{m}_i\in\mathbb{Z}_q^r$. Let Commit be any binding (but not necessarily hiding) commitment scheme, with message space $\mathbb{Z}_q^r$. Define $\mathsf{CommitCols}$ to be the function which takes M as input and returns the list of commitments $\mathsf{Commit}(\mathbf{m}_i)$.

Merkle Trees. We will commit to many Reed-Solomon codewords $\mathbf{H}_i\in\mathcal{C}$, $i=1,\ldots,r$, in the following way. Firstly, take the codewords to be the rows of the matrix $M=(\mathbf{V}_i)\in\mathbb{Z}_q^{r\times l}$. Secondly, apply $\mathsf{CommitCols}$ to commit to the columns of M. Finally, produce a Merkle tree with all these column commitments as leafs. This means that the commitments $\mathsf{CommitCols}(M)$ are taken to be the leafs of a binary tree of height $\log l$ where each inner node is the hash of its two children. This results is a single hash $\mathcal{M}=\mathsf{Merkle}(\mathsf{CommitCols}(M))$ in the root of the tree. This gives a commitment to all codewords $\mathbf{V}_i$ and allows to simultaneously open all codewords at an arbitrary position $j\in[l]$ by revealing the position j of every codeword and the nodes in the Merkle tree that are needed to compute the path to the root node $\mathcal{M}$. For $I\subset[l]^\tau$ we will later write $\mathsf{MerklePaths}|_I$ for the set of all nodes in the Merkle tree needed to compute the paths for all codeword position in I. This construction is binding because if it would be possible to produce to differing openings at the same position then there must be a hash collision somewhere in the path to the root of the Merkle tree.

3 Basic Protocol

In this section, we present our basic protocol tailored towards single SIS instances, where the solution has entries lying in $\{-1,0,1\}$. The protocol can incorporate larger sets in the obvious way, as explained in the caption of Figure 1. At a high level, it implements the strategy used in [BLS19] and explained in the introduction. But it uses Reed-Solomon codes to instantiate the commitment scheme and their associated zero-knowledge proofs, rather than lattice-based commitments.

Proof systems using code-based commitment schemes often require a proximity test, to prove that the (possibly malicious) values hashed by the prover are close to codewords, and therefore represent valid encodings of messages. Following [RVW13], this is often done by checking that an auxiliary random linear combination of the committed and possibly noisy codewords is itself close to the code. Recently, [BCI+20] investigated the use of a more structured linear combination, with coefficients x^i for some random x, in proximity testing. We will use the same strategy for proximity testing with powers of x as part of our scheme.[4]

We cannot use a structured linear combination in the amortised case since with this strategy, the probability that the verifier can catch a cheating prover decreases as the number of committed secrets increases. Our amortised result thus uses a random linear combination of all of the hashed vectors to prove that each hashed vector is close to a codeword.

The complete protocol is given in Figure 1.

[4] From a technical perspective, [BCI+20] proves that except with small probability, these structured linear combinations have the same distance from the code as the maximum distance of all the codewords in the linear combination. This involves a deep and technical proof. Here, we can tolerate some decrease in the distance, and prove a weaker result (3.4) as part of our soundness proof. We find this interesting as the proof is significantly simpler.

$$
\begin{array}{lcl}
\underline{\text{Prover } \mathcal{P}} & & \underline{\text{Verifier } \mathcal{V}} \\
\text{Inputs:} & & \\
A \in \mathbb{Z}_q^{n\times m}, \mathbf{u} \in \mathbb{Z}_q^n, \mathbf{s} \in \{-1,0,1\}^m & & A, \mathbf{u} \\
\mathbf{u} = A\mathbf{s} + \mathbf{e}, \mathbf{e} \in \{-1,0,1\}^n & & \\
\hline
\mathbf{t} \leftarrow \mathbb{Z}_q^m & & \\
\mathbf{f}(X) = \mathbf{t}X + \mathbf{s} & & \\
\frac{1}{X}\mathbf{f}(X) \circ [\mathbf{f}(X) - \mathbf{1}^m] \circ [\mathbf{f}(X) + \mathbf{1}^m] & & \\
\quad = \mathbf{v}_2 X^2 + \mathbf{v}_1 X + \mathbf{v}_0 & & \\
\mathbf{d}(X) = \mathbf{u} - A\mathbf{f}(X) & & \\
\frac{1}{X}\mathbf{d}(X) \circ [\mathbf{d}(X) - \mathbf{1}^n] \circ [\mathbf{d}(X) + \mathbf{1}^n] & & \\
\quad = \mathbf{w}_2 X^2 + \mathbf{w}_1 X + \mathbf{w}_0 & & \\
\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2 \leftarrow \mathbb{Z}_q^\tau & & \\
\mathbf{H}_2 = \mathsf{Enc}\left(\mathbf{0}^m, \mathbf{v}_2, \mathbf{w}_2, \mathbf{r}_2\right) & & \\
\mathbf{H}_1 = \mathsf{Enc}\left(\mathbf{t}, \mathbf{v}_1, \mathbf{w}_1, \mathbf{r}_1\right) & & \\
\mathbf{H}_0 = \mathsf{Enc}\left(\mathbf{s}, \mathbf{v}_0, \mathbf{w}_0, \mathbf{r}_0\right) & & \\
E = \begin{pmatrix} \mathbf{H}_2 \\ \mathbf{H}_1 \\ \mathbf{H}_0 \end{pmatrix} & & \\
\mathcal{M} = \mathsf{Merkle}(\mathsf{CommitCols}(E)) & & \\
 & \xrightarrow{\mathcal{M}} & \\
 & \xleftarrow{x} & x \xleftarrow{\$} \mathbb{Z}_q^\times \\
\bar{\mathbf{f}} = \mathbf{f}(x) & & \\
\bar{\mathbf{r}} = \mathbf{r}_2 x^2 + \mathbf{r}_1 x + \mathbf{r}_0 & \xrightarrow{\bar{\mathbf{f}}, \bar{\mathbf{r}}} & \\
 & \xleftarrow{I} & I \xleftarrow{\$} [l]^{(\tau)}, |I| = \tau \\
 & \xrightarrow{E|_I, \mathsf{MerklePaths}_I} & \\
 & & \text{Check } E|_I \text{ against } \mathcal{M} \text{ and } \mathsf{MerklePaths}_I \\
 & & \text{Compute } \bar{\mathbf{d}} = \mathbf{u} - A\bar{\mathbf{f}} \\
 & & \mathsf{Enc}\left(\bar{\mathbf{f}}, \frac{1}{x}\bar{\mathbf{f}} \circ \left[\bar{\mathbf{f}} - \mathbf{1}^m\right] \circ \left[\bar{\mathbf{f}} + \mathbf{1}^m\right], \frac{1}{x}\bar{\mathbf{d}} \circ \left[\bar{\mathbf{d}} - \mathbf{1}^n\right] \circ \left[\bar{\mathbf{d}} + \mathbf{1}^n\right], \bar{\mathbf{r}}\right)\Bigg|_I \\
 & & \quad \stackrel{?}{=} \mathbf{H}_2|_I x^2 + \mathbf{H}_1|_I x + \mathbf{H}_0|_I
\end{array}
$$

Fig. 1. Simple hash-based proof of knowledge of a ternary solution to a linear equation over $\mathbb{Z}_q$. To use a secret set S of size σ different from $\{-1,0,1\}$, one would change $\frac{1}{X}\mathbf{f}(X) \circ [\mathbf{f}(X) - \mathbf{1}^m] \circ [\mathbf{f}(X) + \mathbf{1}^m]$ to $\frac{1}{X}\bigcirc_{i\in S}[f(X) - i^m] = \sum w_j X^j$. The analogous this is done for the line $\frac{1}{X}\mathbf{d}(X) \circ [\mathbf{d}(X) - \mathbf{1}^n] \circ [\mathbf{d}(X) + \mathbf{1}^n]$. One would also accordingly increase the number of terms $\mathbf{r}_i$ and the number of rows $\mathbf{H}_j = \mathsf{Enc}\left(\mathbf{0}^m, \mathbf{v}_j, \mathbf{w}_j, \mathbf{r}_j\right)$

Theorem 3.1 (Completeness). *The protocol in Figure 1 is perfectly complete.*

Proof. As usual this easily follows from careful inspection of the protocol. □

Theorem 3.2 (Special Honest Verifier Zero Knowledge). *There exists an efficient simulator $\mathcal{S}$ which, given values for the random challenges x and I from the protocol in Figure 1, outputs a protocol transcript distributed identically to a real transcript from the interaction between an honest prover and an honest verifier.*

Proof. Firstly, the simulator $\mathcal{S}$ picks $\bar{\mathbf{f}}$ and $\bar{\mathbf{r}}$ uniformly at random from $\mathbb{Z}_q^m$ and $\mathbb{Z}_q^\tau$ respectively. At that point, $\bar{\mathbf{d}}$ is fully determined. Next, the simulator chooses $\mathbf{H}_2|_I, \mathbf{H}_1|_I$ uniformly at random from $\mathbb{Z}_q^\tau$. As a result, $\mathbf{H}_0|_I$ is determined completely by the last verification equation. Thus, $\mathcal{S}$ has simulated all of $E|_I$. Finally, the simulator $\mathcal{S}$ sets $E|_{I^c}$ to be all zeroes, and computes $\mathcal{M}$ as well as $\mathsf{MerklePaths}_I$. □

Theorem 3.3 (Knowledge Soundness). *Let $\mathcal{C} \subset \mathbb{Z}_q^l$ be a Reed-Solomon code of dimension $k' = 2m+n+\tau$ and length l with encoding function $\mathsf{Enc}()$. Let $k' \le k \le l$. Suppose that there is an efficient deterministic prover $\mathcal{P}^*$ that convinces the honest verifier $\mathcal{V}$ on input $A, \mathbf{u}$ to accept with probability*

$$\varepsilon > \max\left\{2\left(\frac{k}{l-\tau}\right)^\tau, \frac{2}{q-1} + \left(1 - \frac{k-k'}{9l}\right)^\tau, 2\left(1 - \frac{2(k-k')}{3l}\right)^\tau, \frac{12}{q-1}\right\}.$$

Then, there exists an efficient probabilistic extraction algorithm $\mathcal{E}$ which, given rewindable black-box access to $\mathcal{P}^*$, produces a witness $\mathbf{s} \in \{-1, \dots, 1\}^m$ such that $\mathbf{u} - A\mathbf{s} \in \{-1, 0, 1\}^n$.

Proof. We define an extractor $\mathcal{E}$ which runs as follows. The extractor runs the prover $\mathcal{P}^*$ until an accepting transcript is produced. Since each transcript is accepting with probability ϵ, this is expected to take $1/\epsilon$ attempts.

Let I_1 be the set of indices that the prover opened in the first accepting transcript. The extractor $\mathcal{E}$ now replays the prover and verifier using the same x but different challenge sets I, as follows.

- Suppose that r accepting transcripts have been collected so far.
- Let $J = \bigcup_{i=1}^r I_r$ be the set of indices for which the prover has provided openings so far.
- The extractor repeatedly replays the prover and verifier using the same x until an accepting transcript with $I_{r+1} \not\subset J$ is obtained.
- The extractor stops when J contains at least k indices.

Now, we analyse the number of attempts that it takes the extractor to obtain all of these transcripts. Suppose that J does not yet contain k indices. We compute an upper bound on the probability that $I \subset J$ for a random I. The probability that $I \subset J$ is $\binom{|J|}{\tau}/\binom{l}{\tau}$, which is bounded above by $\binom{k}{\tau}/\binom{l}{\tau}$ since $|J| < k$. This in turn is bounded above by $\left(\frac{k}{l-\tau}\right)^\tau$.

By a standard 'heavy rows' argument, there is probability at least $1/2$ that x is such that, on replaying the proof with the same x and new random I, the prover's success probability is at least $\epsilon/2$. If so, then the probability that every subsequent transcript is accepting but uses a challenge set $I \not\subset J$ is at least $\epsilon/2 - (k/(l-\tau))^\tau$. Therefore, in this case, the expected number of further attempts that the extractor has to make to obtain openings at k indices is bounded above by $(k-\tau)/(\epsilon/2 - (k/(l-\tau))^\tau)$.

Let T be the total expected running time for this process of gathering transcripts. By the Markov inequality, the probability that the extractor's running time for this process exceeds $4T$ is at most $1/4$. This means that with probability at least $3/4$, the extractor terminates within time $4T$. With probability at least $1/2$, then the extractor succeeds in gathering enough accepting transcripts. Therefore, there is some overlap between these two events, and the probability that the extractor terminates within time $4T$ *and* manages to gather k transcripts is at least $1/4$.

Now we argue that if these extracted openings cannot be decoded to give a valid witness, then the prover's success probability must be lower than the bound given in the statement of the theorem.

Firstly, note that the indices from the challenge sets $I_1, I_2, \ldots, I_r$ which make up J may overlap. If the prover's commitment openings at the same index are not consistent, then since the check on the values of $E|_I$ and the Merkle tree root $\mathcal{M}$ and Merkle paths was satisfied in each case, the extractor has found a hash collision. Further, if the prover could produce another accepting proof, using Merkle tree openings on any element of J that are different from the extracted ones, with good probability, then the extractor could run the prover once more and obtain a hash collision, with the same probability.

Otherwise, let

$$E^* = \begin{pmatrix} \mathbf{H}_2^* \\ \mathbf{H}_1^* \\ \mathbf{H}_0^* \end{pmatrix} \in \mathbb{Z}_q^{3\times k}$$

be the matrix of openings that the verifier sees. We can assume J contains only k indices, by ignoring any extra ones. Let $\mathcal{C}'$ be the code $\mathcal{C}$ restricted to the indices of J. Then $\mathcal{C}'$ is a code of length k with minimum distance $d' = k - k' + 1$.

One way that the prover could attempt to cheat is by committing to vectors that are not codewords. We will prove a stronger statement than necessary to prevent this type of cheating, and show that if there exists some linear combination $\mathbf{c}^* E^*$ such that $d'(\mathcal{C}', \mathbf{c}^* E^*) \geq d'/3$, then the prover's success probability is bounded above.

Lemma 3.4. *If there exists some $\mathbf{c}^* \in \mathbb{Z}_q^3$ such that $d'(\mathcal{C}', \mathbf{c}^* E^*) \geq d'/3$, then*

$$\Pr_{x \xleftarrow{\$} \mathbb{Z}_q^\times} \left[d'(\mathcal{C}', (x^2, x, 1)E^*) < \frac{d'}{9} \right] < \frac{3}{q-1}.$$

Proof. If the probability in the statement of the lemma is at least $3/(q-1)$, then there exist three distinct $x_1, x_2, x_3 \in \mathbb{Z}_q^\times$ such that

$$d'(\mathcal{C}', (x_i^2, x_i, 1)E^*) < \frac{d'}{9}.$$

Let $B \in (\mathbb{Z}_q^\times)^{3\times 3}$ be the square matrix with entries $B_{i,j} = x_i^{3-j}$. Set $BE^* = M$, so that the ith row of M is equal to $\mathbf{m}_i = (x_i^2, x_i, 1)E^*$. Since B is invertible, we also get $E^* = B^{-1}M$ and so $c^*E^* = c^*B^{-1}M$. Consequently, c^*E^* can be written as a $\mathbb{Z}_q$-linear combination of the vectors $\mathbf{m}_i = (x_i^2, x_i, 1)E^*$, say $c^*E^* = \sum_{i=1}^3 \mu_i \mathbf{m}_i$ for some $\mu_1, \mu_2, \mu_3 \in \mathbb{Z}_q$. Then, by the properties of linear codes and the triangle inequality we get

$$\frac{d'}{3} \leq d'(\mathcal{C}', \mathbf{c}^* E^*) = d'\left(\mathcal{C}', \sum_{i=1}^3 \mu_i \mathbf{m}_i\right) \leq \sum_{i=1}^3 d'(\mathcal{C}', \mu_i \mathbf{m}_i) = \sum_{i=1}^3 d'(\mathcal{C}', \mathbf{m}_i) < \frac{d'}{3}$$

which is a contradiction. Hence, if x is chosen uniformly at random from $\mathbb{Z}_q^\times$, there is at most a $2/(q-1)$ fraction of x for which $d'(\mathcal{C}', (x^2, x, 1)E^*) < d'/9$. □

Corollary 3.5. *If there exists some $\mathbf{c}^* \in \mathbb{Z}_q^3$ such that $d'(\mathcal{C}', \mathbf{c}^* E^*) \geq d'/3$, then the prover's success probability is bounded above by*

$$\frac{2}{q-1} + \left(1 - \frac{d'}{9l}\right)^\tau < \frac{2}{q-1} + \left(1 - \frac{k-k'}{9l}\right)^\tau.$$

Proof. If such a $\mathbf{c}^*$ exists, then for random x, the previous lemma shows that $d'(\mathcal{C}', (x^2, x, 1)E^*) \geq d'/9$ except with probability at most $2/(q-1)$.

If $d'(\mathcal{C}', (x^2, x, 1)E^*) \geq d'/9$, then since

$$\mathsf{Enc}\left(\bar{\mathbf{f}}, x^{-1}\bar{\mathbf{f}} \circ \left[\bar{\mathbf{f}} - \mathbf{1}^m\right] \circ \left[\bar{\mathbf{f}} + \mathbf{1}^m\right], x^{-1}\bar{\mathbf{d}} \circ \left[\bar{\mathbf{d}} - \mathbf{1}^n\right] \circ \left[\bar{\mathbf{d}} + \mathbf{1}^n\right], \bar{\mathbf{r}}\right)$$

is a codeword, we know that it differs from $(x^2, x, 1)E^* = \mathbf{H}_2^* x^2 + \mathbf{H}_1^* x + \mathbf{H}_0^*$ in at least $d'/9$ positions.

Now, the verifier will choose a random set I of τ indices, independent of the codeword, on which to check the verification equation given by

$$\begin{aligned}&\mathsf{Enc}\left(\bar{\mathbf{f}}, x^{-1}\bar{\mathbf{f}} \circ \left[\bar{\mathbf{f}} - \mathbf{1}^m\right] \circ \left[\bar{\mathbf{f}} + \mathbf{1}^m\right], x^{-1}\bar{\mathbf{d}} \circ \left[\bar{\mathbf{d}} - \mathbf{1}^n\right] \circ \left[\bar{\mathbf{d}} + \mathbf{1}^n\right], \bar{\mathbf{r}}\right)|_I \\ &\stackrel{?}{=} \mathbf{H}_2|_I x^2 + \mathbf{H}_1|_I x + \mathbf{H}_0|_I .\end{aligned}$$

What is the probability that the verification equation, which is restricted to codeword positions in I, is satisfied, given that there are at least $d'/9$ positions for which that equation does not hold? The probability that the first index in I is outside the set of bad indices is $1 - d'/(9l)$. Given that the first index is outside the set of bad positions, the probability that the second index in I is also outside the set of bad positions is $1 - d'/(9(l-1))$ which is bounded above by $1 - d'/(9l)$. Continuing in a similar fashion, we see that the probability of acceptance in this case is at most $\left(1 - \frac{d'}{9l}\right)^\tau$. The result follows by a union bound. □

We have bounded the prover's success probability in the case that there exists some linear combination of the rows of E^* which has distance at least $d'/3$ from the code. From now on, we focus on the case where such a linear combination does not exist, and bound the prover's success probability in this case.

This means that from now on, we can assume that for all $\mathbf{c} \in \mathbb{Z}_q^3$, $d'(\mathcal{C}', \mathbf{c}^* E^*) < d'/3$. Then, for each $j = 0, 1, 2$, there exist unique $\mathbf{s}_j^*, \mathbf{v}_j^*, \mathbf{w}_j^*, \mathbf{r}_j^*$ such that $\mathbf{H}_j^*$ is within distance $d'/3$ from $\mathsf{Enc}_J\left(\mathbf{s}_j^*, \mathbf{v}_j^*, \mathbf{w}_j^*, \mathbf{r}_j^*\right)$. This defines matrices V^* and R^* of messages and randomness. Reed-Solomon codes are efficiently decodable, so we can compute V^* and R^* efficiently from E^*.

We actually need to be convinced of more than what is implied by Lemma 3.4, and show that when we take a linear combination of the prover's committed vectors, which are close to codewords, then that linear combination decodes to the value that we would expect. That is, we want to be sure that a linear combination of vectors close to codewords, when decoded, gives the same linear combination of the decodings of those vectors. This is the statement of the following lemma, which is taken from the second claim in Appendix B of [BCG+17].

Lemma 3.6. *If $d'(\mathcal{C}', \mathbf{c}E^*) < d'/3$ for all $\mathbf{c} \in \mathbb{Z}_q^3$, then for any $\mathbf{y} \in \mathbb{Z}_q^3$, we have that*

$$d'\left(\mathsf{Enc}_J\left(\mathbf{y}V^*, \mathbf{y}R^*\right), \mathbf{y}E^*\right) < d'/3.$$

Corollary 3.7. *Suppose that $d'(\mathcal{C}', \mathbf{c}E^*) < d'/3$ for all $\mathbf{c} \in \mathbb{Z}_q^3$. If there exist less than $(\varepsilon/2)(q-1)$ accepting transcripts with pairwise different first challenges $x \in \mathbb{Z}_q^\times$ such that for the response $\bar{\mathbf{f}}$ the following conditions are true*

$$\bar{\mathbf{f}} = \mathbf{s}_2^* x^2 + \mathbf{s}_1^* x + \mathbf{s}_0^*, \tag{8}$$

$$x^{-1}\bar{\mathbf{f}} \circ \left(\bar{\mathbf{f}} - \mathbf{1}^m\right) \circ \left(\bar{\mathbf{f}} + \mathbf{1}^m\right) = \mathbf{v}_2^* x^2 + \mathbf{v}_1^* x + \mathbf{v}_0^*, \tag{9}$$

$$x^{-1}\bar{\mathbf{d}} \circ \left(\bar{\mathbf{d}} - \mathbf{1}^n\right) \circ \left(\bar{\mathbf{d}} + \mathbf{1}^n\right) = \mathbf{w}_2^* x^2 + \mathbf{w}_2^* x + \mathbf{w}_0^*, \tag{10}$$

then the prover's success probability ε is bounded above by

$$2\left(1 - \frac{2d'}{3l}\right)^\tau < 2\left(1 - \frac{2(k-k')}{3l}\right)^\tau .$$

Proof. Recall that a challenge $x \in \mathbb{Z}_q^\times$ defines a "heavy row" if interacting with $\mathcal{P}^*$ and sending x as the first challenge and a uniformly random second challenge set I results in an accepting transcript with probability at least $\varepsilon/2$. From the heavy rows lemma we know that a completely randomly sampled accepting transcript lies in a heavy row with probability at least $1/2$. Moreover, by a simple extension of the lemma, there are at least $(\varepsilon/2)(q-1)$ heavy rows.

Now assume that there is a heavy row with challenge $x \in \mathbb{Z}_q^\times$ and corresponding prover's response $\bar{\mathbf{f}}$ such that one of the conditions in the statement is not true. From Lemma 3.6 we know that

$$d'\left(\mathsf{Enc}_J\left((x^2, x, 1)V^*, (x^2, x, 1)R^*\right), (x^2, x, 1)E^*\right) < \frac{d'}{3}.$$

Note that the conditions mean that the vectors $(x^2, x, 1)(V^*, R^*)$ and $\left(\bar{\mathbf{f}}, x^{-1}\bar{\mathbf{f}} \circ \left(\bar{\mathbf{f}} - \mathbf{1}^m\right) \circ \left(\bar{\mathbf{f}} + \mathbf{1}^m\right), x^{-1}\bar{\mathbf{d}} \circ \left(\bar{\mathbf{d}} - \mathbf{1}^n\right) \circ \left(\bar{\mathbf{d}} + \mathbf{1}^n\right), \bar{\mathbf{r}}\right)$ are equal. Since one of the conditions is not true, the distance between the corresponding codewords must be at least d'. It then follows from the reverse triangle inequality that

$$\begin{aligned}&d'\left(\mathsf{Enc}_J\left(\bar{\mathbf{f}}, x^{-1}\bar{\mathbf{f}} \circ \left(\bar{\mathbf{f}} - \mathbf{1}^m\right) \circ \left(\bar{\mathbf{f}} + \mathbf{1}^m\right), x^{-1}\bar{\mathbf{d}} \circ \left(\bar{\mathbf{d}} - \mathbf{1}^n\right) \circ \left(\bar{\mathbf{d}} + \mathbf{1}^n\right), \bar{\mathbf{r}}\right), (x^2, x, 1)E^*\right) \\ &> \frac{2d'}{3}.\end{aligned}$$

But the verifier checks the equality of these codewords on the τ coefficients in the second challenge I. By similar reasoning as in the proof of Corollary 3.5, the probability that I does not contain one of the $2d'/3$ bad coefficients is at most $(1 - 2d'/(3l))^\tau$. So we must have

$$\frac{\varepsilon}{2} \le \left(1 - \frac{2d'}{3l}\right)^\tau.$$

□

Finally, if we substitute Equation (8) into Equation (9) and multiply by x we find a polynomial over $\mathbb{Z}_q^m$ in indeterminate X of degree at most 6 that we know has at least $(\varepsilon/2)(q-1)$ roots x. If the polynomial is not the zero polynomial then it must be that $\varepsilon \le 12/(q-1)$. Otherwise, the constant coefficient shows

$$\mathbf{s}_0^* \circ (\mathbf{s}_0^* - \mathbf{1}^m) \circ (\mathbf{s}_0^* + \mathbf{1}^m) = 0$$

which implies $\mathbf{s}_0^* \in \{-1, 0, 1\}^m$. If we further substitute

$$\bar{\mathbf{d}} = \mathbf{u} - A\bar{\mathbf{f}} = -A\mathbf{s}_2^* x^2 - A\mathbf{s}_1^* x + (\mathbf{u} - A\mathbf{s}_0^*)$$

into Equation (10) then we similarly find $\mathbf{u} - A\mathbf{s}_0^* \in \{-1, 0, 1\}^n$. □

3.1 Proof Size and Concrete Parameter Choices

For one iteration of the proof in Figure 1 the prover has to send the linear masked secrets $\bar{\mathbf{f}} \in \mathbb{Z}_q^m$ and $\bar{\mathbf{r}} \in \mathbb{Z}_q^\tau$, and open the 3 codewords $\mathbf{H}_i$ at τ positions. Together these are $m + 4\tau$ elements of $\mathbb{Z}_q$. Moreover, the prover has to send the initial commitment $\mathcal{M}$ and τ Merkle paths. As specified in the protocol this needs $1 + \tau\lceil\log l\rceil$ 32-byte hashes per iteration. It is obvious that the latter can be optimized by splitting the tree with l leafs into h trees with l/h leafs each. This results in shorter Merkle paths at the costs of h root hashes in the commitment $\mathcal{M}$. Concretely, with this optimization in one iteration of the protocol $h + \tau\lceil\log(l/h)\rceil$ hashes need to be send. In total we see that one iteration of the protocol has a bandwidth requirement of

$$\left((m + 4\tau)\lceil\log(q)\rceil + 256\left(h + \tau\left\lceil\log\left(\frac{l}{h}\right)\right\rceil\right)\right)\Big/ 8192$$

kilobytes.

4 Amortized Protocol For a Fixed Public Randomness

A closer look at the communication cost of the protocol from the previous section shows that the initial commitment and the Merkle paths are actually responsible for the majority of the proof size.

Now, the size of the Merkle trees does not depend on the number of Reed-Solomon codewords that the prover commits to, since each leaf can be the hash of codeword positions from arbitrarily many codewords. It is therefore clear that when one has to prove many (different) equations

$$\mathbf{u}_j = A_j\mathbf{s}_j + \mathbf{e}_j$$

for $j = 1, \dots, r$ at once, then one can commit to all of the codewords in the corresponding proofs using only one set of Merkle trees. Moreover, when the individual proofs are executed in a parallel fashion where the same challenges are used in all parallel proofs, then one also only needs one set of τ Merkle paths for all the proofs. So we see that the protocol in Figure 1 can be operated in a simple amortized mode which has a total size of

$$\left(r\left(m+4\tau\right)\lceil\log(q)\rceil + 256\left(h + \tau\left\lceil\log\left(\frac{l}{h}\right)\right\rceil\right)\right)\Big/8192$$

kilobytes for r equations.

In this section we improve on this result by giving a sublinear amortized proof protocol that allows to prove $r = m$ equations $\mathbf{u}_j = A\mathbf{s}_j + \mathbf{e}_j$ with the same matrix A and a total number of m^2 secret coefficients in size that scales only with the square root m of the number of secret coefficients. As an independent generalization, the protocol of this section allows to directly prove the secret coefficient to lie in an arbitrary interval $\{0, \dots, b-1\}$. We chose not do this in the basic protocol from Section 3 to keep that protocol as simple as possible.

The key technique for the sublinearity is to replace the r masked secrets $\bar{\mathbf{f}}_j = x\mathbf{t}_j + \mathbf{s}_j$ of total length rm by a single length-m packed vector that masks all secrets $\mathbf{s}_j$ at once. After the Merkle tree roots for the commitment and the Merkle tree paths, the masked secret $\bar{\mathbf{f}}$ is the next largest element of the basic proof protocol. So by being able to amortize its size over many equations we can also significantly improve the concrete per-equation cost. One way of doing this is to separate the secret vectors with independent challenges $x_j \in \mathbb{Z}_q$ and use

$$\bar{\mathbf{f}} = \sum_{j=0}^{r} x_j \mathbf{s}_j$$

where $\mathbf{s}_0$ now is the single masking vector. So then $\bar{\mathbf{f}}$ is the evaluation of a multivariate polynomial in r indeterminates. This is compatible with the lattice equations because we assume that they use the same matrix A. The verifier can compute

$$\sum_{j=0}^{r} x_j \mathbf{u}_j - A\bar{\mathbf{f}} = -x_0 A\mathbf{s}_0 - \sum_{j=1}^{r} x_j \mathbf{e}_j$$

which is a masking of the error vectors with masking vector $\mathbf{e}_0 = -A\mathbf{s}_0$. The problem with this approach is that when we work with equations of degree b in these polynomials to prove that all $\mathbf{s}_j$ and $\mathbf{e}_j$ have coefficients in $\{0, \dots, b-1\}$, we arrive at multivariate polynomials of total degree b that contain in the order of r^b monomials. So this means we need to commit to that many garbage terms and prove openings of these commitments which has communication cost in the order of r^b. On the other hand, the multivariate Schwartz-Zippel lemma still only gives that a multivariate polynomial of total degree b vanishes at a random point with probability b/q so we do not profit from the large challenge space of size q^r in the soundness error.

Therefore a better approach is to not use independent challenges x_j but instead choose them to be evaluations of Lagrange interpolation polynomials at the same evaluation point as in [GGPR13].

So let $a_1, \dots, a_r$ be distinct interpolation points in $\mathbb{Z}_q$. Then, for $j \in \{1, \dots, r\}$, let

$$\ell_j(X) = \prod_{i \neq j} \frac{X - a_i}{a_j - a_i}$$

be the jth Lagrange interpolation polynomial and let $\ell_0(X) = \prod_{j=1}^{k}(X - a_j)$.

By polynomial interpolation every polynomial $f \in \mathbb{Z}_q[X]/(\ell_0(X))$ can be written uniquely as

$$f(X) = \sum_{j=1}^{r} a_j \ell_j(X)$$

with $a_j \in \mathbb{Z}_q$. Since $\ell_j(X)^2 \equiv \ell_j(X) \pmod{\ell_0(X)}$ and $\ell_i(X)\ell_j(X) \equiv 0 \pmod{\ell_0(X)}$ for all $i, j \in \{1, \dots, r\}$, $i \neq j$, multiplication is coefficient-wise in this representation. That is, for instance,

$$f(X)^2 = \sum_{j=1}^{r} a_j^2 \ell_j(X).$$

Now, higher-degree polynomials $f \in \mathbb{Z}_q[X]/(\ell_0(X)^b)$ can be written uniquely as

$$f(X) = \sum_{i=0}^{b-1} \sum_{j=1}^{r} \ell_j(X)\ell_0(X)^i.$$

For our application this means that when we separate the secret vectors using Lagrange polynomials, i.e.

$$\bar{\mathbf{f}} = \sum_{j=0}^{r} \ell_j(x)\mathbf{s}_j,$$

then equations of degree b in $\bar{\mathbf{f}}$ will only contain br terms $\mathbf{v}_{i,j}\ell_j(x)\ell_0(x)^i$ for $j \in \{1, \dots, r\}$, $i \in \{0, \dots, b-1\}$. So this is again linear in r and suffices for our sublinearity result.

After we have replaced the individual masked secrets $\bar{\mathbf{f}}_j$ by just one, note that we need $r+1$ commitments to the secret vectors in order to prove that $\bar{\mathbf{f}}$ is correctly formed whereas the range proofs for the secret coefficients need br commitments to the garbage terms in the equations of degree b in $\bar{\mathbf{f}}$ or $\bar{\mathbf{d}}$. So we don't want to put the secret vectors and the garbage terms in the same Reed-Solomon codewords as we have done in the basic protocol in Section 3. This would only prove that $\bar{\mathbf{f}}$ is the evaluation of a polynomial of degree $br - 1$ instead of a polynomial of degree r, which is bad for large b as it results in a higher soundness error. Splitting the codewords into several smaller ones has the downside that more openings need to be send. Therefore for small b it would actually be better not to do this.

The last difference to the basic protocol is that for technical reasons already explained in Section 3, the prover sends an auxiliary masking of a random linear combination with independent coefficients of all of the messages in the codewords. This is necessary since the prover commits to much more codewords and he must prove that they are really close to codewords. Using a variant of the technique of the basic protocol and leverage the linear combination of the codewords with $\ell_j(x)$ as coefficients would result in a much larger soundness error.

The complete sublinear protocol is given in Figure 2. We only use one Reed-Solomon code of dimension $k' = m + \tau$ and length l and also use it for the shorter message vectors of length $n \leq m$ by zero padding these vectors. We analyze the protocol in the coming Theorems.

Theorem 4.1 (Completeness). *The protocol in Figure 2 is perfectly complete.*

Proof. Follows by inspection of the protocol. □

Theorem 4.2 (Special Honest Verifier Zero Knowledge). *There exists an efficient simulator $\mathcal{S}$ which given values for the random challenges x, α, β, γ, δ, and I from the protocol in Figure 2, outputs a protocol transcript distributed identically to a real transcript from the interaction between an honest prover and an honest verifier.*

Proof. Firstly, the simulator $\mathcal{S}$ picks $\bar{\mathbf{f}}$ and $\bar{\mathbf{r}}$ uniformly at random from $\mathbb{Z}_q^m$ and $\mathbb{Z}_q^\tau$ respectively. At that point, $\bar{\mathbf{d}}$ is fully determined. Next, the simulator chooses $\mathbf{F}_1|_I, \dots, \mathbf{F}_r|_I$, $\mathbf{V}_{i,j}|_I$, and $\mathbf{W}_{i,j}|_I$ for $(i, j) \neq (0, 1)$ uniformly at random from $\mathbb{Z}_q^\tau$. As a result, $\mathbf{F}_0|_I$, $\mathbf{V}_{0,1}|_I$, $\mathbf{W}_{0,1}|_I$, and $\mathbf{Y}|_I$ are determined completely by the verification equations. Thus, $\mathcal{S}$ has simulated all of $E|_I$. Finally, the simulator $\mathcal{S}$ sets $E|_{I^c}$ to be all zeroes, and computes $\mathcal{M}$ as well as $\mathsf{MerklePaths}_I$. □

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

Inputs:

$A \in \mathbb{Z}_q^{n\times m}, \mathbf{u}_j \in \mathbb{Z}_q^n, \mathbf{s}_j \in \{0,\dots,b-1\}^m, j \in [r]$ | $A, \mathbf{u}_j$

$\mathbf{u}_j = A\mathbf{s}_j + \mathbf{e}_j, \mathbf{e}_j \in \{0,\dots,b-1\}^n$

$\mathbf{s}_0 \leftarrow \mathbb{Z}_q^m$

$$\mathbf{f}(X) = \sum_{j=0}^{r} \mathbf{s}_j \ell_j(X)$$

$$\frac{1}{\ell_0(X)} \bigcirc_{i=0}^{b-1} [\mathbf{f}(X) - i\mathbf{1}] = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \mathbf{v}_{i,j}\ell_j(X)\ell_0(X)^i$$

$$\mathbf{d}(X) = \sum_{j=1}^{r} \mathbf{u}_j \ell_j(X) - A\mathbf{f}(X)$$

$$\frac{1}{\ell_0(X)} \bigcirc_{i=0}^{b-1} [\mathbf{d}(X) - i\mathbf{1}] = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \mathbf{w}_{i,j}\ell_j(X)\ell_0(X)^i$$

$\mathbf{y} \leftarrow \mathbb{Z}_q^m,\ \mathbf{r}^y \leftarrow \mathbb{Z}_q^\tau$

$\mathbf{Y} = \mathsf{Enc}(\mathbf{y}, \mathbf{r}^y)$

For $0 \le j \le r$:

  $\mathbf{r}_j^s \leftarrow \mathbb{Z}_q^\tau$

  $\mathbf{S}_j = \mathsf{Enc}(\mathbf{s}_j, \mathbf{r}_j^s)$

For $0 \le i \le b-1$ and $j \in [r]$:

  $\mathbf{r}_{i,j}^v,\ \mathbf{r}_{i,j}^w \leftarrow \mathbb{Z}_q^\tau$

  $\mathbf{V}_{i,j} = \mathsf{Enc}(\mathbf{v}_{i,j}, \mathbf{r}_{i,j}^v)$

  $\mathbf{W}_{i,j} = \mathsf{Enc}(\mathbf{w}_{i,j}, \mathbf{r}_{i,j}^w)$

$$S = \begin{pmatrix}\mathbf{S}_0\\ \vdots \\ \mathbf{S}_r\end{pmatrix},\ V = \begin{pmatrix}\mathbf{V}_{0,1}\\ \vdots \\ \mathbf{V}_{b-1,r}\end{pmatrix},\ W = \begin{pmatrix}\mathbf{W}_{0,1}\\ \vdots \\ \mathbf{W}_{b-1,r}\end{pmatrix}$$

$$\mathcal{M} = \mathsf{Merkle}\left(\mathsf{CommitCols}\left(\begin{pmatrix}\mathbf{Y}\\ S\\ V\\ W\end{pmatrix}\right)\right)$$

$$\xrightarrow{\mathcal{M}}$$

$$\xleftarrow{x,\alpha,\beta,\gamma,\delta} \qquad x,\alpha \overset{\$}{\leftarrow} \mathbb{Z}_q, \beta \overset{\$}{\leftarrow} \mathbb{Z}_q^{r+1}, \gamma,\delta \overset{\$}{\leftarrow} \mathbb{Z}_q^{rb}$$

$\bar{\mathbf{f}} = \mathbf{f}(x)$

$$\bar{\mathbf{r}}^f = \sum_{j=0}^{r} \mathbf{r}_j^s \ell_j(x)$$

$$\bar{\mathbf{r}}^v = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \mathbf{r}_{i,j}^v \ell_j(x)\ell_0(x)^{i+1}$$

$$\bar{\mathbf{r}}^w = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \mathbf{r}_{i,j}^w \ell_j(x)\ell_0(x)^{i+1}$$

$$\bar{\mathbf{z}} = \alpha\mathbf{y} + \sum_{j=0}^{r}\beta_j\mathbf{s}_j + \sum_{i=0}^{b-1}\sum_{j=1}^{r}(\gamma_{i,j}\mathbf{v}_{i,j} + \delta_{i,j}\mathbf{w}_{i,j})$$

$$\bar{\mathbf{r}}^z = \alpha\mathbf{r}^y + \sum_{j=0}^{r}\beta_j\mathbf{r}_j^f + \sum_{i=0}^{b-1}\sum_{j=1}^{r}(\gamma_{i,j}\mathbf{r}_{i,j}^v + \delta_{i,j}\mathbf{r}_{i,j}^w) \qquad \xrightarrow{\bar{\mathbf{f}}, \bar{\mathbf{r}}^f, \bar{\mathbf{r}}^v, \bar{\mathbf{r}}^w, \bar{\mathbf{z}}, \bar{\mathbf{r}}^z}$$

$$\xleftarrow{I} \qquad I \overset{\$}{\leftarrow} [l]^{(\tau)}, |I| = \tau$$

$$\xrightarrow[\mathsf{MerklePaths}_I]{\mathbf{Y}|_I, S|_I, V|_I, W|_I,}$$

Check $\mathbf{Y}|_I, S|_I, V|_I, W|_I$ against $\mathcal{M}$ and $\mathsf{MerklePaths}_I$

Compute $\bar{\mathbf{d}} = \sum_{j=1}^{r} \mathbf{u}_j\ell_j(x) - A\mathbf{f}$

$$\mathsf{Enc}\left(\bar{\mathbf{f}}, \bar{\mathbf{r}}^f\right)\Big|_I \overset{?}{=} \sum_{j=0}^{r} \ell_j(x)\mathbf{S}_j|_I$$

$$\mathsf{Enc}\left(\bigcirc_{i=0}^{b-1}\left[\bar{\mathbf{f}} - i\mathbf{1}^m\right], \bar{\mathbf{r}}^v\right)\Big|_I \overset{?}{=} \sum_{i=0}^{b-1}\sum_{j=1}^{r} \ell_j(x)\ell_0(x)^{i+1}\mathbf{V}_{i,j}|_I$$

$$\mathsf{Enc}\left(\bigcirc_{i=0}^{b-1}\left[\bar{\mathbf{d}} - i\mathbf{1}^n\right], \bar{\mathbf{r}}^w\right)\Big|_I \overset{?}{=} \sum_{i=0}^{b-1}\sum_{j=1}^{r} \ell_j(x)\ell_0(x)^{i+1}\mathbf{W}_{i,j}|_I$$

$$\mathsf{Enc}(\bar{\mathbf{z}}, \bar{\mathbf{r}}^z)|_I \overset{?}{=} \alpha\mathbf{Y}|_I + \sum_{j=0}^{r}\beta_j\mathbf{S}_j|_I + \sum_{i=0}^{b-1}\sum_{j=1}^{r}(\gamma_{i,j}\mathbf{V}_{i,j}|_I + \delta_{i,j}\mathbf{W}_{i,j}|_I)$$

Fig. 2. Sublinear hash-based proof of knowledge of short solutions to many linear equations over $\mathbb{Z}_q$.

Theorem 4.3 (Knowledge Soundness). *Let $\mathcal{C} \subset \mathbb{Z}_q^l$ be a Reed-Solomon code of dimension $k' = m + \tau$ and length l with encoding function $\mathsf{Enc}()$. Let $k' \leq k \leq l$. Suppose that there is an efficient deterministic prover $\mathcal{P}^*$ that convinces the honest verifier $\mathcal{V}$ on input $A, \mathbf{u}_j$ to accept with probability*

$$\varepsilon > \max\left\{2\left(\frac{k}{l-\tau}\right)^\tau, \frac{1}{q} + \left(1 - \frac{k-k'}{6l}\right)^\tau, 2\left(1 - \frac{2(k-k')}{3l}\right)^\tau, \frac{2(b+1)r}{q}\right\}.$$

Then, there exists an efficient probabilistic extraction algorithm $\mathcal{E}$ which, given rewindable black-box access to $\mathcal{P}^*$, produces vectors $\mathbf{s}_j \in \{0,\dots,b-1\}^m$ such that $\mathbf{u}_j - A\mathbf{s}_j \in \{0,\dots,b-1\}^n$ for all $j = 1,\dots,r$.

Proof. The extractor wants to obtain openings at k positions of all of the codewords $\mathbf{Y}$, $\mathbf{S}_j$, $\mathbf{V}_{i,j}$, $\mathbf{W}_{i,j}$, which might have errors. Exactly as in the proof of Theorem 3.3, this is possible in expected time at most

$$\frac{1}{\varepsilon} + 4\frac{k-\tau}{\varepsilon/2 - (k/(l-\tau))^\tau}.$$

Let the code $\mathcal{C}'$ be the restriction of $\mathcal{C}$ to the k positions at which the extractor succeeded to obtain codeword openings. Then the openings can be regarded as noisy codewords of the punctured code $\mathcal{C}'$ and we write them as

$$\mathbf{Y}^*, \mathbf{S}_0^*, \dots, \mathbf{S}_r^*, \mathbf{V}_{0,1}^*, \dots, \mathbf{V}_{b-1,r}^*, \mathbf{W}_{0,1}^*, \dots, \mathbf{W}_{b-1,r}^*.$$

Note that the punctured code has minimum distance $d' = k - k' + 1$.

For what follows it is important that for each new run of $\mathcal{P}^*$ that produces an accepting transcript, codeword openings at positions among those already obtained by the extractor must coincide with the previous openings. Otherwise there would be a hash collision in the Merkle paths.

Next we argue that it follows from the bound on the success probability ε of the prover $\mathcal{P}^*$ that there is no linear combination of the noisy codewords which has distance to $\mathcal{C}'$ greater or equal than $d'/3$. The strategy is essentially again that there can not be a larger distance since the verifier checks random linear combinations to be equal to a proper codeword at τ random positions, where the codeword is independent of the positions. So the verifier would catch the prover with probability bigger than $1 - \varepsilon$ if there were more errors. We can not use (a variant) of Lemma 3.4 because the number of errors that this lemma says exist, with high probability, decreases with the length of the linear combinations. For this reason, unlike in Protocol 1, the verifier checks a uniformly random linear combination of all of the codewords the last verification equation. Then we can use the first claim in Appendix B of [BCG+17]. It states that if there was some linear combination of all of the codewords that had distance greater or equal than $d'/3$ from $\mathcal{C}'$, then a uniformly random linear combination would have distance at least $d'/6$, with probability bigger than $1 - 1/q$. Therefore, as in the proof of Theorem 3.3, the verifier would accept with probability at most

$$\frac{1}{q} + \left(1 - \frac{d'}{6l}\right)^\tau < \frac{1}{q} + \left(1 - \frac{k-k'}{6l}\right)^\tau < \varepsilon.$$

This is in contradiction to the assumption that the verifier accepts with probability ε.

Now, since in particular every noisy codeword individually has distance less than $d'/3$ to the code $\mathcal{C}'$, the extractor can efficiently decode all of them. Let

$$\mathbf{y}^*, \mathbf{s}_0^*, \dots, \mathbf{s}_r^*, \mathbf{v}_{0,1}^*, \dots, \mathbf{v}_{b-1,r}^*, \mathbf{w}_{0,1}^*, \dots, \mathbf{w}_{b-1,r}^*$$

be the decoded messages (without the randomness vectors).

By Lemma 3.6, a linear combination of the noisy codewords is not just close to *some* codeword but actually has distance less than $d'/3$ to the encoding of the corresponding linear combination of the above decoded messages. For example we have

$$d'\left(\mathsf{Enc}_J\left(\sum_{j=0}^r \ell_j(x)\mathbf{s}_j^*, \mathbf{r}\right), \sum_{j=0}^r \ell_j(x)\mathbf{S}_j^*\right) < \frac{d'}{3}$$

for some randomness vector $\mathbf{r} \in \mathbb{Z}_q^\tau$. Furthermore, if the vector $\bar{\mathbf{f}}$ in some accepting transcript is not equal to $\sum_{j=0}^r \ell_j(x)\mathbf{s}_j^*$, then encodings of the two vectors have distance at least d'. It then follows from the reverse triangle inequality that

$$d'\left(\mathsf{Enc}_J\left(\bar{\mathbf{f}}, \bar{\mathbf{r}}^f\right), \sum_{j=0}^r \ell_j(x)\mathbf{S}_j^*\right) > \frac{2d'}{3}.$$

But the verifier checks the equality of these two (noisy) codewords at τ random positions. Hence, he does not requests one of $2d'/3$ differing positions with probability only at most

$$\left(1 - \frac{2d'}{3l}\right)^\tau < \left(1 - \frac{2(k-k')}{3l}\right)^\tau < \frac{\varepsilon}{2}.$$

For a random accepting transcript the remaining success probability when fixing the first challenges and hence $\bar{\mathbf{f}}$ is at least $\varepsilon/2$ with probability at least $1/2$. That is, $\bar{\mathbf{f}}$ lies on a "heavy row" with probability at least $1/2$ and in this case $\bar{\mathbf{f}}$ must be such that

$$\bar{\mathbf{f}} = \sum_{j=0}^r \ell_j(x)\mathbf{s}_j^*.$$

With the same reasoning we get that at the same time

$$\bigcirc_{i=0}^{b-1}\left(\bar{\mathbf{f}} - i\mathbf{1}\right) = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \ell_j(x)\ell_0(x)^{i+1}\mathbf{v}_{i,j}^*, \tag{11}$$

$$\bigcirc_{i=0}^{b-1}\left(\bar{\mathbf{d}} - i\mathbf{1}\right) = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \ell_j(x)\ell_0(x)^{i+1}\mathbf{w}_{i,j}^*. \tag{12}$$

Finally, substituting the expression for $\bar{\mathbf{f}}$ into the second-to-last equation (11) shows that the polynomial

$$\bigcirc_{i=0}^{b-1}\left(\sum_{j=0}^r \ell_j(X)\mathbf{s}_j^* - i\mathbf{1}\right) - \sum_{i=0}^{b-1}\sum_{j=1}^{k} \ell_j(X)\ell_0(X)^{i+1}\mathbf{v}_{i,j}^*$$

of degree less than $(b+1)r$ has a root at x. As a further consequence of the heavy rows argument above we know that there are at least $\varepsilon/2 \cdot q$ heavy rows with distinct challenges x. So, since $\varepsilon > 2(b+1)r/q$, the polynomial has more roots than its maximum degree and hence must be the zero polynomial.

Reducing modulo $\ell_0(X)$ then shows that in particular

$$\sum_{j=1}^r \ell_j(X)\bigcirc_{i=0}^{b-1}\left(\mathbf{s}_j^* - i\mathbf{1}\right) = 0.$$

Since the Lagrange polynomials are linear independent we also get $\bigcirc_{i=0}^{b-1}(\mathbf{s}_j^* - i\mathbf{1}) = 0$. This implies $\mathbf{s}_j^*$, $j = 1, \ldots, r$, has coefficients in $\{0, \ldots, b-1\}$.

Moreover, when we define $\mathbf{e}_j^* = \mathbf{u}_j - A\mathbf{s}_j^*$ for $j = 1, \ldots, r$ and $\mathbf{e}_0^* = -A\mathbf{s}_0^*$, then $\bar{\mathbf{d}} = \sum_{j=1}^r \ell_j(x)\mathbf{u}_j - A\bar{\mathbf{f}} = \sum_{j=0}^r \ell_0(x)\mathbf{e}_j^*$. This can be substituted into Equation (12). Then the same reasoning as above yields $\mathbf{e}_j^* \in \{0, \ldots, b-1\}^n$, $j = 1, \ldots, r$. So we have found the small solutions we were looking for. $\square$

4.1 Proof Size

In one execution of the protocol in Figure 2 the prover sends only 6 masked secret vectors $\bar{\mathbf{f}} \in \mathbb{Z}_q^m$, $\bar{\mathbf{z}}$, and $\bar{\mathbf{r}}^\iota \in \mathbb{Z}_q^\tau$ for $\iota = f, v, w, z$ for all r equations with a total of $2m + 4\tau$ $\mathbb{Z}_q$-coefficients. Then he also sends τ codeword positions from all of the $(2b+1)r + 1$ codewords in the protocol. Note that from the soundness error in Theorem 4.3 it is clear that τ does not need to increase with the number of equations. So this part only scales linearly in r. Finally, the number of hashes sent is exactly the same as in the single-equation protocol from Section 3. That is, the prover sends $h + \tau\lceil log(l/h)\rceil$ 32-byte hashes. In summary, we see that the protocol has a bandwidth requirement of

$$\left((2m + ((2b+1)r + 5)\tau)\lceil \log q\rceil + 256\left(h + \tau\left\lceil \log\left(\frac{l}{h}\right)\right\rceil\right)\right)\Big/ 8192$$

kilobytes. When we have $r = m$ equations with m secret coefficients each, so m^2 secret coefficient in total, we see that the communication cost is of order m, which is the square root of the number of secret coefficients.

Example. Since this proof system is essentially many instances of the example from [BLS19,ENS20] (using the same public randomness $\mathbf{A}$), we compare the amortized output size to those papers. In particular, we consider the case where one wants to prove $r = m = 1024$ equations simultaneously. The last term in the expression for the soundness error in Theorem 4.3 limits it to at least 2^{-19}. So we search for parameters k, l and τ under the constraint that the soundness error stays below 2^{-18}. Then 7 iterations of the protocol give a negligible soundness error of less than 2^{-126}. Our search for such parameters minimizing the proof size resulted in

$$k = 37376, \quad l = 43521, \quad \text{and} \quad \tau = 88.$$

With these parameters and the same $h = 2\tau$ as before, the total proof size for all 7 iterations is 17509 kilobytes. So this translates to an amortized cost of 17.1 kilobytes per equation. We mention that, as we have explained above, for the small secret interval with $b = 3$ it would in fact be better to only have one set of codewords for all verification equations. Concretely, this would give a protocol with per-equation size of only about 8 kilobytes, which is a noticeable improvement over the 45KB proof size in [ENS20].

References

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

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

BBC$^+$18. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *CRYPTO*, pages 669–699, 2018.

BCC$^+$16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, pages 327–357, 2016.

BCG$^+$17. Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, pages 336–365, 2017.

BCI$^+$20. Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. *Electron. Colloquium Comput. Complex.*, 27:83, 2020.

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

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

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

BCS19. Carsten Baum, Daniele Cozzo, and Nigel P. Smart. Using topgear in overdrive: A more efficient zkpok for SPDZ. *IACR Cryptology ePrint Archive*, 2019:35, 2019.

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

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

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

DKL$^+$18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

dPLNS17. Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In *ACM Conference on Computer and Communications Security*, pages 1565–1581. ACM, 2017.

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

EKS$^+$20. Muhammed F. Esgin, Veronika Kuchta, Amin Sakzad, Ron Steinfeld, Zhenfei Zhang, Shifeng Sun, and Shumo Chu. Practical post-quantum few-time verifiable random function with applications to algorand. *IACR Cryptol. ePrint Arch.*, 2020:1222, 2020.

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. *IACR Cryptol. ePrint Arch.*, 2020:518, 2020. To appear in Asiacrypt 2020.

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

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

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

KKPP20. Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest. Scalable ciphertext compression techniques for post-quantum kems and their applications. *IACR Cryptol. ePrint Arch.*, 2020:1107, 2020. To appear in Asiacrypt 2020.

KR08. Yael Tauman Kalai and Ran Raz. Interactive PCP. In *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547. Springer, 2008.

KTX08. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389, 2008.

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

Mic00. Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

RRR16. Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62. ACM, 2016.

RVW13. Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013.

Sho. Victor Shoup. https://www.shoup.net/ntl/.

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

YAZ$^+$19. Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 147–175. Springer, 2019.