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 $\mathbf{I} \otimes \mathbf{A}'$, 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 a 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 that the latter are applied directly to 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

[⋆] This is the full version of the paper presented at ESORICS 2021.

techniques of this paper will prove knowledge of $\mathbf{s}$ and $\mathbf{e}$ with small coefficients[3] 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 such relaxed proofs only prove knowledge of larger $\bar{\mathbf{s}}$ and $\bar{\mathbf{e}}$ than the ones used by an honest prover, we must increase the sizes of other parameters (like q and n, m) in order to obtain the same security level, 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

[3] If the coefficients of $\mathbf{s}$ are unrestricted, as in some applications of LWE, then the proof will be slightly more efficient because we do not have to prove the shortness of $\mathbf{s}$. If $\mathbf{s}$ has a size-restricted distribution, but different from $\mathbf{e}$, then we can apply the transformation of [ACPS09] to convert the instance to one where the distribution of the secret is the same as of the error.

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 uses some of the algebraic techniques from previous lattice-based works, which design zero-knowledge arguments for (1) directly based on homomorphic, lattice-based commitments. The main difference is that we replace the the lattice-based commitments with the 'encode-then-hash' commitment scheme implicit in [BCG$^+$17]. This scheme is asymptotically more efficient than when using lattice-based commitments (which are linear in the size of the message), but has a large additive overhead. This overhead grows logarithmically in the number of instances, and linearly in the size of the domain of the errors in (1). We show that when the errors come from a small domain, then the overhead is amortized away as the number of instances grows.

The main difference between our work and IOP constructions such as [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 code-words over $\mathbb{Z}_2$. The work of [Beu20] used a cut-and-choose approach to leverage the larger field size in (1) 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 $q \approx 2^{32}$, the proofs (with 2^{-128} soundness error) are around 400 KB long. Further building on these results, the proof of [ENS20] is a little under 50KB, and a recent improvement in [LNS21] reduces it by another 30%.

For direct comparisons with the results in this paper, we also computed the parameter sizes using the techniques in [ENS20, LNS21] for $n = m = 2048, q \approx 2^{60}$, and 2^{-57} soundness. The proof size for these parameters is around 45KB.[4]

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, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ 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])[5].

In Fig. 1, 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 Fig. 1,[6] we see that the amortized proof is about 3 KB per instance, which is about an order of magnitude improvement in size.

[4] We point out that the parameters n and m from this paper do not correspond to those with the same name from [ENS20, Appendix B.1]. The length of the secret $\mathbf{s}$ in [ENS20] already includes the error vector. So even though the length of the secret is 2048 there, it is actually broken down into a vector of dimension 1024 which gets multiplied by $\mathbf{A}$, and another 1024 dimensional vector, which is the error. Thus the secret length we are comparing to in this paper is twice as large as in the original [ENS20]. The comparisons to [ENS20] for other dimensions / modulus would be fairly similar.

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

[6] The secret size in the first line is chosen from 4 elements, rather than 3 as in [ENS20]. Reducing the set to 3 will not give us any noticeable reduction.

Size of Secret Set	Proof Size (KB)	Prover Time (sec)	Verifier Time (sec)
4	217	0.78	0.06
8	232	1.12	0.06
16	262	1.77	0.06
32	322	3.13	0.07
64	442	6.03	0.08
128	682	12.97	0.10
256	1162	31.27	0.15

Fig. 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 2 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 values of τ and l (see Section 3 for their definitions) are 512 and 2^{19}, respectively). 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 2, 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, and about 30% smaller using improvements from [LNS21], when the size of the secret set is 3.

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.

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 $c \in \mathbb{Z}_q$, $\mathbf{s} + \mathbf{t}c = \mathtt{Open}(S + Tc)$. For now, let us treat $\mathtt{Com}$ as a black box. Let $\mathbf{1}$ be a vector all of whose coefficients are 1, and for two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^m$, let $\mathbf{v} \circ \mathbf{w}$ denote the component-wise product of $\mathbf{v}$ and $\mathbf{w}$. We will now give a simplified version of the protocol in Figure 2 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

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

which satisfies

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

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 $W_0 = \mathtt{Com}((\mathbf{A}\mathbf{t}) \circ (\mathbf{1} - 2))$.

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 hash-based commitments where the input to the i^{th} commitment are all the elements in the i^{th} position of all codewords. These commitments are then hashed into a Merkle tree and the root is transmitted as the final commitment. Proving 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 the codeword in those τ positions. If τ is smaller than the number of randomness positions, then revealing τ positions of the codeword information-theoretically hides the message and so the commitment scheme is 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, $l \approx 2^{19}$, 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 $\zeta_1, \ldots, \zeta_l$ be distinct elements of $\mathbb{Z}_q^\times$. The subspace $\mathcal{C} \subset \mathbb{Z}_q^l$ of $\mathbb{Z}_q^l$ of degree k' consisting of all l-tuples $\mathsf{Enc}(f) = (f(\zeta_1), \ldots, f(\zeta_l))$ 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 in 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, suppose that the verifier is allowed 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 τ openings do not reveal any information about the message coefficients. We explain this in more detail in Appendix A.

We will write $\mathsf{Enc}(\mathbf{m}, \mathbf{r})$ 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 Commit be a commitment scheme with message space $\mathbb{Z}_q^r$ and randomness space R. Let $M = \big(\mathbf{m}_1\ \mathbf{m}_2\ \ldots\ \mathbf{m}_l\big)$ be a matrix made up of column vectors $\mathbf{m}_i \in \mathbb{Z}_q^r$. Let $\rho = \big(\rho_1\ \rho_2\ \ldots\ \rho_l\big)$ be a list of random strings $\rho_i \in R$.

Define $\mathsf{CommitCols}$ to be the function which takes M and ρ as input and returns the list of commitments $\mathsf{Commit}(\mathbf{m}_i; \rho_i)$.

In our protocols, we will instantiate Commit with the folklore hash-based commitment scheme where $\mathsf{Commit}(\mathbf{m}_i; \rho_i) = h(\rho_i || \mathbf{m}_i)$, for a hash-function $h\colon \{0,1\}^* \to \{0,1\}^{256}$, and randomness ρ_i uniformly sampled from $\{0,1\}^{128}$. When h is modelled as a random-oracle, Commit is a computationally-hiding and computationally-binding commitment scheme (see e.g. [MF21, Proposition 8.12]).

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 $M = (\mathbf{H}_i) \in \mathbb{Z}_q^{r \times l}$. Secondly, apply $\mathsf{CommitCols}$ to commit to the columns of M. Finally, produce a Merkle tree with all these column commitments as leaves. This means that the commitments $\mathsf{CommitCols}(M; \rho)$ are taken to be the leaves of a binary tree of height $\log l$ where each inner node is the hash of its two children. This results in a single hash $\mathcal{M} = \mathsf{Merkle}(\mathsf{CommitCols}(M; \rho))$ at the root of the tree. This gives a commitment to all codewords $\mathbf{H}_i$ and allows simultaneous opening of all codewords at arbitrary positions $j \in [l]$ 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 were possible to produce two different openings at the same position then there would be a hash collision somewhere on the path to the root of the Merkle tree. The construction is hiding because the unopened leaves are all hiding commitments, and so the nodes in the Merkle paths which are derived from them do not leak any information about the unopened codeword entries.

3 Basic Protocol

In this section, we present our basic protocol tailored towards single LWE 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 2. 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. Previously, [BKS18] investigated the use of the structured linear combination $(1, x)$ 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.[7]

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 2.

Theorem 3.1 (Completeness). *The protocol in Figure 2 is perfectly complete.*

Proof. This follows from a 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 2, outputs a protocol transcript whose distribution is indistinguishable from that of a real transcript from the interaction between an honest prover and an honest verifier.*

Proof. First, we will analyse the distribution of the transcripts generated by an honest prover. Then, we will give an efficient simulator.

The honest prover. The randomness ρ consists of the uniformly random bits sampled for use in $\mathsf{CommitCols}$, and so $\rho|_I$ is uniformly random. The commitment scheme Commit is computationally hiding, and so the output of $\mathsf{CommitCols}$ is computationally indistinguishable from a list of commitments to all-zero messages.

Since $\mathbf{s}$ and $\mathbf{r}_0$ are both sampled uniformly at random, $\bar{\mathbf{f}}$ and $\bar{\mathbf{r}}$ are also uniformly distributed.

Finally, since each of the Reed-Solomon encodings $\mathbf{H}_0$, $\mathbf{H}_1$ and $\mathbf{H}_2$ was computed using randomness vectors $\mathbf{r}_0$, $\mathbf{r}_1$, and $\mathbf{r}_2$, which were sampled uniformly at random from $\mathbb{Z}_q^\tau$, any τ entries of each codeword are uniformly distributed. Therefore, since $|I| = \tau$, each row of the matrix $E|_I$ must be uniformly distributed, conditioned on satisfying the verification equations.

The simulator. First, 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 $E|_I$. The simulator samples uniformly random bits ρ, and includes $\rho|_I$ as part of the simulated transcript. Finally, the simulator $\mathcal{S}$ sets $E|_{I^c}$ to be all zeroes, and computes $\mathcal{M}$ as well as $\mathsf{MerklePaths}_I$. □

[7] From a technical perspective, [BKS18, Theorem 4.1] 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. Here, we can tolerate some decrease in the distance in our soundness proof, and prove a weaker result (Lemma 3.5)via a simpler method.

Prover $\mathcal{P}$'s input: $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{u} \in \mathbb{Z}_q^n, \mathbf{s} \in \{-1,0,1\}^m$ and $\mathbf{e} \in \{-1,0,1\}^n$ such that $\mathbf{u} = \mathbf{A}\mathbf{s} + \mathbf{e}$.

Verifier $\mathcal{V}$'s input: $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{u} \in \mathbb{Z}_q^n$.

The protocol proceeds as follows.

$\mathcal{P}$:

- The prover samples $\mathbf{t} \leftarrow \mathbb{Z}_q^m$ and computes the polynomials $\mathbf{f}(X) = \mathbf{t}X + \mathbf{s}$ and $\mathbf{d}(X) = \mathbf{u} - \mathbf{A}\mathbf{f}(X)$.
- The prover computes the polynomials

$$\frac{1}{X}\mathbf{f}(X) \circ [\mathbf{f}(X) - \mathbf{1}^m] \circ [\mathbf{f}(X) + \mathbf{1}^m] = \mathbf{v}_2X^2 + \mathbf{v}_1X + \mathbf{v}_0 \ , \text{ and}$$
$$\frac{1}{X}\mathbf{d}(X) \circ [\mathbf{d}(X) - \mathbf{1}^n] \circ [\mathbf{d}(X) + \mathbf{1}^n] = \mathbf{w}_2X^2 + \mathbf{w}_1X + \mathbf{w}_0 \ .$$

- The prover samples $\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2 \leftarrow \mathbb{Z}_q^\tau$ and computes the encodings

$$\mathbf{H}_2 = \mathsf{Enc}\left(\mathbf{0}^m, \mathbf{v}_2, \mathbf{w}_2, \mathbf{r}_2\right) \ , \mathbf{H}_1 = \mathsf{Enc}\left(\mathbf{t}, \mathbf{v}_1, \mathbf{w}_1, \mathbf{r}_1\right) \text{ and } \mathbf{H}_0 = \mathsf{Enc}\left(\mathbf{s}, \mathbf{v}_0, \mathbf{w}_0, \mathbf{r}_0\right) \ .$$

- The prover sets $E = \begin{pmatrix}\mathbf{H}_2\\ \mathbf{H}_1\\ \mathbf{H}_0\end{pmatrix}$, samples randomness ρ for $\mathsf{CommitCols}$, and computes $\mathcal{M} = \mathsf{Merkle}(\mathsf{CommitCols}(E;\rho))$.

$\mathcal{P} \to \mathcal{V}$: The prover sends $\mathcal{M}$ to the verifier.

$\mathcal{P} \leftarrow \mathcal{V}$: The verifier samples $x \leftarrow \mathbb{Z}_q^\times$ and sends x to the prover.

$\mathcal{P} \to \mathcal{V}$: The prover computes $\bar{\mathbf{f}} = \mathbf{f}(x)$ and $\bar{\mathbf{r}} = \mathbf{r}_2x^2 + \mathbf{r}_1x + \mathbf{r}_0$ and sends them to the verifier.

$\mathcal{P} \leftarrow \mathcal{V}$: The verifier samples $I \leftarrow [l]^\tau$ such that $|I| = \tau$ and sends I to the prover.

$\mathcal{P} \to \mathcal{V}$: The prover computes $E|_I$, $\rho|_I$ and $\mathsf{MerklePaths}_I$ and sends them to the verifier.

$\mathcal{V}$:

- The verifier checks $E|_I$ and $\rho|_I$ against $\mathcal{M}$ and $\mathsf{MerklePaths}_I$.
- The verifier computes $\bar{\mathbf{d}} = \mathbf{u} - \mathbf{A}\bar{\mathbf{f}}$ and checks whether

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

Fig. 2. 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 b 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 $\frac{1}{X}\bigcirc_{i\in S}[\mathbf{f}(X)-i^m] = \sum \mathbf{v}_jX^j$. The analogous of 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 $\mathbf{r}_j$ 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 3.3 (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 $k' \leq k \leq l$. Suppose that there is an efficient deterministic prover $\mathcal{P}^*$ that convinces the honest verifier $\mathcal{V}$ on input $\mathbf{A}, \mathbf{u}$ to accept with probability*

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

Then, there exists a probabilistic extraction algorithm $\mathcal{E}$ which, given rewindable black-box access to $\mathcal{P}^*$, produces a witness $\mathbf{s} \in \{-1,0,1\}^m$ such that $\mathbf{u} - \mathbf{A}\mathbf{s} \in \{-1,0,1\}^n$ or finds a hash collision in expected time at most $64T$ where

$$T := \frac{3}{\varepsilon} + \frac{k - \tau}{\varepsilon/2 - (k/(l-\tau))^{\tau}}$$

and running $\mathcal{P}^*$ once is assumed to take unit time.

Remark 3.4 (on the parameter k). The parameter k in Theorem 3.3 is not used by the prover and verifier algorithms, but only by the extraction algorithm $\mathcal{E}$. In more detail, the extraction algorithm $\mathcal{E}$ attempts to collect openings for k out of the l entries of each committed codeword. The extraction algorithm attempts to decode the partial codewords obtained in order to find a witness.

Analysing the probability that $\mathcal{E}$ can decode these codewords leads to the dependence of ε on k. This indirectly affects the proof size, as choosing k to reduce the knowledge soundness error ε means that the proof requires fewer parallel repetitions to achieve negligible soundness. This also applies to our amortised protocol.

Asymptotically, when $l = ck'$ for some constant $c > 1$, one could set $k = c'k'$ for constant $c' \in (1, c)$. For concrete parameters, we choose the value of k in our basic and amortised protocols using computer experiments.

Proof. We first recap the standard "heavy-rows" strategy [Dam10]. Namely, let $H \in \{0,1\}^{(q-1)\times\binom{l}{\tau}}$ be a binary matrix where rows and columns are indexed by $\mathbb{Z}_q^\times$ and $[l]^\tau$ respectively. We define $H[x][I]$ to be equal to 1 if and only if the (deterministic) prover $\mathcal{P}^*$ outputs a valid proof given challenges x and I. Clearly, this matrix has ε fraction of 1s. We say that a row is *heavy* if it contains $\varepsilon/2$ fraction of 1s. Then, the heavy-rows lemma states that more than a half of 1s in H are located in the heavy rows.

Now, we turn to defining an extractor $\mathcal{E}$ which runs as follows. The extractor runs the prover $\mathcal{P}^*$ until an accepting transcript is produced. However, if it takes longer than $8/\varepsilon$ time to obtain such a transcript, it aborts.

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 E^* and ρ^* so far.

- The extractor repeatedly replays the prover and verifier using the same x until an accepting transcript with challenge $I_{r+1} \not\subset J$ and openings $E|_{I_{r+1}}, \rho|_{I_{r+1}}$ is obtained. Next, for each index $i \in I_{r+1} \cap J$, $\mathcal{E}$ checks whether the openings $E|_{I_{r+1}}$ and E^* differ on index i and analogously for $\rho|_{I_{r+1}}, \rho^*$. If so, $\mathcal{E}$ checks the corresponding Merkle paths (with the same root $\mathcal{M}$), finds a hash collision and terminates. Otherwise, it continues.
- The extractor stops when J contains at least k indices.

If the whole procedure, which starts when the very first transcript is obtained, takes more than $8(k-\tau)/(\varepsilon/2-(k/(l-\tau))^\tau)$ of time, it aborts.

For the final step, $\mathcal{E}$ re-runs the prover $\mathcal{P}^*$ $16/\varepsilon$ times with fresh challenges x, I and obtains the corresponding openings $E|_I, \rho|_I$. If for some trial, $\mathcal{P}^*$ outputs a valid proof and there exists an index $i \in I \cap J$ such that $E|_I$ and E^* differ in position i then $\mathcal{E}$ outputs a hash collision. It also does an identical check for $\rho|_I$ and ρ^*. Otherwise, $\mathcal{E}$ terminates after $16/\varepsilon$ trials.

Note that the total run-time of $\mathcal{E}$ is at most:

$$8\left(\frac{3}{\varepsilon} + \frac{k-\tau}{\varepsilon/2-(k/(l-\tau))^\tau}\right) = 8T.$$

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 $\left(\frac{k}{l-\tau}\right)^\tau$.

Assuming that the first accepting transcript with a challenge x was obtained, the heavy-rows lemma says that 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 $\varepsilon/2$. If so, then the probability that every subsequent transcript is accepting but uses a challenge set $I \not\subset J$ is at least $\varepsilon/2-(k/(l-\tau))^\tau$. 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 $(k-\tau)/(\varepsilon/2-(k/(l-\tau))^\tau)$.

Finally, we compute the upper-bound on the probability that $\mathcal{E}$ aborts. Let abort_1 be the event that $\mathcal{E}$ aborts in the first step i.e. it does not obtain the first valid transcript in time $8/\varepsilon$. Since each transcript is accepting with probability ε, this is expected to take $1/\varepsilon$ attempts. Now, by the Markov inequality, we have that $\Pr[\mathsf{abort}_1] \leq 1/8$.

Now suppose that $\neg\mathsf{abort}_1$ holds and an accepting transcript with challenges x and I_1 was obtained. We define abort_2 to be the event that $\mathcal{E}$ aborts in the second phase. We are interested in bounding $\Pr[\mathsf{abort}_2|\neg\mathsf{abort}_1]$. Let heavy be the event that the x-row of H is heavy. By the heavy-rows lemma, $\Pr[\neg\mathsf{heavy}|\neg\mathsf{abort}_1] \leq 1/2$. On the other hand, by the Markov inequality and the reasoning above, we obtain:

$$\Pr[\mathsf{abort}_2|\mathsf{heavy} \wedge \neg\mathsf{abort}_1] \leq 1/8.$$

Hence,

$$\Pr[\mathsf{abort}_2|\neg\mathsf{abort}_1] \leq \Pr[\mathsf{abort}_2|\mathsf{heavy} \wedge \neg\mathsf{abort}_1] + \Pr[\neg\mathsf{heavy}|\neg\mathsf{abort}_1] \leq 5/8.$$

To sum up, the probability that $\mathcal{E}$ does not abort, is at least $1/4$.

Eventually, we analyse the final step of the algorithm $\mathcal{E}$. Let $\mathsf{success}$ be the event that a prover outputs a valid proof in the last step. Moreover, define $\mathsf{consistent}$ as the event when after running $\mathcal{P}^*$ for fresh random challenges x, I, the openings $E|_I, \rho|_I$ are consistent with the ones gathered by $\mathcal{E}$ earlier, i.e. for all $i \in I \cap J$, $E|_I$ and E^* match in the i-th position and similarly for $\rho|_I$ and ρ^*.

Suppose that $\Pr[\mathsf{success} \wedge \neg\mathsf{consistent}] \geq \varepsilon/2$. If the prover's commitment openings at the same index are not consistent, then since the check on the values of $E|_I, \rho|_I$ and the Merkle tree root $\mathcal{M}$ and Merkle paths was satisfied in each case, the extractor has found a hash collision. Then, by the Markov inequality, the probability that $\mathcal{E}$ does not find a hash collision after $16/\varepsilon$ trials is at most $1/8$. In conclusion, the total probability of $\mathcal{E}$ not aborting and finding a hash collision in the last step is at least $1/8$. Therefore, we run $\mathcal{E}$ 8 times in order to get the required transcripts and possibly find a hash collision.

For the remaining part of the proof, we assume that $\Pr[\mathsf{success}\wedge\neg\mathsf{consistent}] < \varepsilon/2$ which implies that

$$\Pr[\mathsf{success} \wedge \mathsf{consistent}] > \varepsilon/2.$$

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.

Recall that

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

is 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 $d' = k - k' + 1$. Additionally, for any $\mathbf{x} \in \mathbb{Z}_q^k$ we define $d'(\mathcal{C}', \mathbf{x}) := \min_{\mathbf{c}' \in \mathcal{C}'} d(\mathbf{c}', \mathbf{x})$.

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

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

Proof. Suppose that 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 $\mathbf{c}^*E^* = \mathbf{c}^*B^{-1}M$. Consequently, $\mathbf{c}^*E^*$ can be written as a $\mathbb{Z}_q$-linear combination of the vectors $\mathbf{m}_i = (x_i^2, x_i, 1)E^*$, say $\mathbf{c}^*E^* = \sum_{i=1}^3 \mu_i \mathbf{m}_i$ 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

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

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 $d'(\mathcal{C}', (x^2, x, 1)E^*) < d'/9$. $\square$

We apply Lemma 3.5 to upper-bound the success probability ε of the prover $\mathcal{P}^*$ in the case when there is some $\mathbf{c}^* \in \mathbb{Z}_q^3$ so that $d'(\mathcal{C}', \mathbf{c}^*E^*) \geq d'/3$.

Corollary 3.6. *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*

$$\frac{4}{q-1} + 2\left(1 - \frac{d'}{9l}\right)^\tau < \frac{4}{q-1} + 2\left(1 - \frac{k-k'}{9l}\right)^\tau.$$

Proof. Note that

$$\begin{aligned}
\varepsilon/2 &< \Pr\left[\mathsf{success} \wedge \mathsf{consistent}\right] \\
&< \Pr\left[\mathsf{success} \wedge \mathsf{consistent} | d'(\mathcal{C}', (x^2, x, 1)E^*) \geq d'/9\right] + \Pr\left[d'(\mathcal{C}', (x^2, x, 1)E^*) < d'/9\right] \\
&< \Pr\left[\mathsf{success} \wedge \mathsf{consistent} | d'(\mathcal{C}', (x^2, x, 1)E^*) \geq d'/9\right] + \frac{2}{q-1}
\end{aligned}$$

where x is the $\mathbb{Z}_q^\times$ challenge used in the final step of $\mathcal{E}$ and the last inequality follows from Lemma 3.5.

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 $\left(1 - \frac{d'}{9l}\right)^{\tau}$. Hence, the result follows. $\square$

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 3.5, 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 3.7. *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 3.8. *Suppose that $d'(\mathcal{C}', \mathbf{c}E^*) < d'/3$ for all $\mathbf{c} \in \mathbb{Z}_q^3$. If there exist are no more than $(\varepsilon/4)(q-1)$ accepting transcripts with pairwise different first challenges $i \in \mathbb{Z}_q^\times$ such that for the response $\bar{\mathbf{f}}$ the following conditions are true*

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

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

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

$$\bar{\mathbf{r}} = \mathbf{r}_2^* i^2 + \mathbf{r}_1^* i + \mathbf{r}_0^* \tag{11}$$

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

$$4\left(1 - \frac{2d'}{3l}\right)^{\tau} < 4\left(1 - \frac{2(k-k')}{3l}\right)^{\tau}.$$

Proof. Let $p_i = \Pr[\mathsf{success} \wedge \mathsf{consistent} | x = i]$ where $i \in \mathbb{Z}_q^\times$ and x is the $\mathbb{Z}_q^\times$ challenge variable generated in the last step of $\mathcal{E}$. Let M be the number of all $i \in \mathbb{Z}_q^\times$ such that $p_i > \varepsilon/4$. We claim that $M > (\varepsilon/4)(q-1)$. Indeed, note that

$$\varepsilon/2 < \Pr[\mathsf{success} \wedge \mathsf{consistent}] = \frac{\sum_{i=1}^{q-1} p_i}{q-1} \leq (q-1-M)\frac{\varepsilon}{4(q-1)} + \frac{M}{q-1} < \varepsilon/4 + \frac{M}{q-1}.$$

Now assume that there is indeed a challenge $i \in \mathbb{Z}_q^\times$ such that $p_i > \varepsilon/4$ and the corresponding prover's response $\bar{\mathbf{f}}$ such that one of the conditions in the statement is not true. From Lemma 3.7 we know that

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

Note that the conditions mean that the vectors $(i^2, i, 1)(V^*, R^*)$ and $(\bar{\mathbf{f}}, i^{-1}\bar{\mathbf{f}} \circ \left(\bar{\mathbf{f}} - \mathbf{1}^m\right) \circ \left(\bar{\mathbf{f}} + \mathbf{1}^m\right), i^{-1}\bar{\mathbf{d}} \circ \left(\bar{\mathbf{d}} - \mathbf{1}^n\right) \circ \left(\bar{\mathbf{d}} + \mathbf{1}^n\right), \bar{\mathbf{r}})$ 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

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

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 3.6, 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

$$\frac{\varepsilon}{4} < p_i \leq \left(1 - \frac{2d'}{3l}\right)^\tau.$$

□

Finally, if we substitute Equation (8) into Equation (9) and multiply by i we find a polynomial over $\mathbb{Z}_q^m$ in indeterminate X of degree at most 6 that we know has more than $(\varepsilon/4)(q-1)$ roots i. If the polynomial is not the zero polynomial then it must be that $\varepsilon < 24/(q-1)$. 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

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

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

3.1 Proof Size and Concrete Parameter Choices

For one iteration of the proof in Figure 2 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 $\mathbf{H}_j$ at τ positions, with the randomness that was used to commit to them. Together these are $m + 4\tau$ elements of $\mathbb{Z}_q$, and τ bitstrings of commitment randomness, each of size 128 bits. 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 leaves into h trees with l/h leaves 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 sent. In total we see that one iteration of the protocol has a bandwidth requirement of

$$\left((m + 4\tau)\lceil\log(q)\rceil + 128\tau + 256\left(h + \tau\left\lceil\log\left(\frac{l}{h}\right)\right\rceil\right)\right)\bigg/ 8192$$

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

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

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 2 can be operated in a simple amortized mode which has a total size of

$$\left(r\,(m + 4\tau)\lceil\log(q)\rceil + 128\tau + 256\left(h + \tau\left\lceil\log\left(\frac{l}{h}\right)\right\rceil\right)\right)\bigg/ 8192$$

kilobytes for r equations.

In this section we improve on this result by giving a sublinear amortized proof protocol that allows to prove $r = O(m)$ equations $\mathbf{u}_j = \mathbf{A}\mathbf{s}_j + \mathbf{e}_j$ with the same matrix $\mathbf{A}$ and a total number of $O(m^2)$ 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 that the coefficients of the secret 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 $\mathbf{A}$. The verifier can compute

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

which is a masking of the error vectors with masking vector $\mathbf{e}_0 = -\mathbf{A}\mathbf{s}_0$. 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 on 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 on 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 $\ell_0(X) = \prod_{j=1}^{r}(X - a_j)$.

By polynomial interpolation every polynomial $f \in \mathbb{Z}_q[X]/(\ell_0(X))$ can be written uniquely as a linear combination of $\ell_j(X)$'s, i.e.

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

with $\lambda_j \in \mathbb{Z}_q$. 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,

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

Now, higher-degree polynomials $f \in \mathbb{Z}_q[X]/(\ell_0(X)^b)$ can be written uniquely as a linear combination of the polynomials $\{\ell_j(X)\ell_0(X)^i\}_{i=0,j=1}^{b-1,r}$. 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 sent. 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 many 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 3. 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 padding these vectors with zeroes. We analyze the protocol in the following theorems.

Theorem 4.1 (Completeness). *The protocol in Figure 3 is perfectly complete.*

Proof. This 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, β, γ, δ, and I from the protocol in Figure 3, outputs a protocol transcript whose distribution is indistinguishable from that of a real transcript from the interaction between an honest prover and an honest verifier.*

Proof. First, we will analyse the distribution of the transcripts generated by an honest prover. Then, we will give an efficient simulator.

The honest prover. The randomness ρ consists of the uniformly random bits sampled for use in CommitCols, and so $\rho|_I$ is uniformly random. The commitment scheme Commit is computationally hiding, and so the output of CommitCols is

Prover $\mathcal{P}$'s input: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{u}_1, \dots, \mathbf{u}_r \in \mathbb{Z}_q^n, \mathbf{s}_1, \dots, \mathbf{s}_r \in \{0, \dots, b-1\}^m$ and $\mathbf{e}_1, \dots, \mathbf{e}_r \in \{0, \dots, b-1\}^n$ such that $\mathbf{u}_j = \mathbf{A}\mathbf{s}_j + \mathbf{e}_j$ for all $j \in [r]$.

Verifier $\mathcal{V}$'s input: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{u}_1, \dots, \mathbf{u}_r \in \mathbb{Z}_q^n$.

The protocol proceeds as follows.

$\mathcal{P}$:

- The prover samples $\mathbf{s}_0 \leftarrow \mathbb{Z}_q^m$ and computes the polynomials

$$\mathbf{f}(X) = \sum_{j=0}^{r} \mathbf{s}_j \ell_j(X) \qquad \text{and} \qquad \mathbf{d}(X) = \sum_{j=1}^{r} \mathbf{u}_j \ell_j(X) - \mathbf{A}\mathbf{f}(X) \ .$$

- The prover computes the polynomials

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

- The prover samples $\mathbf{y} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{r}^y \leftarrow \mathbb{Z}_q^\tau$, and computes the encoding $\mathbf{Y} = \mathsf{Enc}(\mathbf{y}, \mathbf{r}^y)$.
- For $j = 0, 1, \dots, r$, the prover samples $\mathbf{r}_j^s \leftarrow \mathbb{Z}_q^\tau$ and computes the encoding $\mathbf{S}_j = \mathsf{Enc}\left(\mathbf{s}_j, \mathbf{r}_j^s\right)$.
- For $i = 0, \dots, b-1$ and $j = 1, \dots, r$, the prover samples $\mathbf{r}_{i,j}^v,\ \mathbf{r}_{i,j}^w \varepsilon \mathbb{Z}_q^\tau$ and computes the encodings

$$\mathbf{V}_{i,j} = \mathsf{Enc}\left(\mathbf{v}_{i,j}, \mathbf{r}_{i,j}^v\right) \qquad \text{and} \qquad \mathbf{W}_{i,j} = \mathsf{Enc}\left(\mathbf{w}_{i,j}, \mathbf{r}_{i,j}^w\right) \ .$$

- The prover sets $S = \begin{pmatrix} \mathbf{S}_0 \\ \vdots \\ \mathbf{S}_r \end{pmatrix}$, $V = \begin{pmatrix} \mathbf{V}_{0,1} \\ \vdots \\ \mathbf{V}_{b-1,r} \end{pmatrix}$ and $W = \begin{pmatrix} \mathbf{W}_{0,1} \\ \vdots \\ \mathbf{W}_{b-1,r} \end{pmatrix}$, samples randomness ρ for $\mathsf{CommitCols}$, and computes

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

$\mathcal{P} \rightarrow \mathcal{V}$: The prover sends $\mathcal{M}$ to the verifier.

$\mathcal{P} \leftarrow \mathcal{V}$: The verifier samples $x \leftarrow \mathbb{Z}_q \setminus \{a_1, \dots, a_r\}$, $\beta \leftarrow \mathbb{Z}_q^{r+1}$, $\gamma \leftarrow \mathbb{Z}_q^{rb}$ and $\delta \leftarrow \mathbb{Z}_q^{rb}$ and sends them to the prover.

$\mathcal{P} \rightarrow \mathcal{V}$: The prover computes vectors

$$\bar{\mathbf{f}} = \mathbf{f}(x) \ , \qquad \bar{\mathbf{r}}^f = \sum_{j=0}^{r} \mathbf{r}_j^s \ell_j(x) \ , \qquad \bar{\mathbf{r}}^v = \sum_{i=0}^{b-1} \sum_{j=1}^{r} \mathbf{r}_{i,j}^v \ell_j(x) \ell_0(x)^{i+1} \ ,$$
$$\bar{\mathbf{r}}^w = \sum_{i=0}^{b-1} \sum_{j=1}^{r} \mathbf{r}_{i,j}^w \ell_j(x) \ell_0(x)^{i+1} \ , \qquad \bar{\mathbf{z}} = \mathbf{y} + \sum_{j=0}^{r} \beta_j \mathbf{s}_j + \sum_{i=0}^{b-1} \sum_{j=1}^{r} (\gamma_{i,j} \mathbf{v}_{i,j} + \delta_{i,j} \mathbf{w}_{i,j}) \ , \text{ and}$$
$$\bar{\mathbf{r}}^z = \mathbf{r}^y + \sum_{j=0}^{r} \beta_j \mathbf{r}_j^f + \sum_{i=0}^{b-1} \sum_{j=1}^{r} (\gamma_{i,j} \mathbf{r}_{i,j}^v + \delta_{i,j} \mathbf{r}_{i,j}^w) \ ,$$

and sends them to the verifier.

$\mathcal{P} \leftarrow \mathcal{V}$: The verifier samples $I \leftarrow [l]^\tau$ such that $|I| = \tau$ and sends I to the prover.

$\mathcal{P} \rightarrow \mathcal{V}$: The prover computes $\mathbf{Y}|_I$, $S|_I$, $V|_I$, $W|_I$, $\rho|_I$ and $\mathsf{MerklePaths}_I$, and sends them to the verifier.

$\mathcal{V}$:

- The verifier checks $\mathbf{Y}|_I$, $S|_I$, $V|_I$, $W|_I$ and $\rho|_I$ against $\mathcal{M}$ and $\mathsf{MerklePaths}_I$.
- The verifier computes $\bar{\mathbf{d}} = \sum_{j=1}^{r} \mathbf{u}_j \ell_j(x) - \mathbf{A}\mathbf{f}$ and checks whether

$$\mathsf{Enc}\left(\bar{\mathbf{f}}, \bar{\mathbf{r}}^f\right)\Big|_I \overset{?}{=} \sum_{j=0}^{r} \ell_j(x) \mathbf{S}_j|_I \ , \qquad \mathsf{Enc}\left(\bigcirc_{i=0}^{b-1} \left[\bar{\mathbf{f}} - i\mathbf{1}^m\right], \bar{\mathbf{r}}^v\right)\Big|_I \overset{?}{=} \sum_{i=0}^{b-1} \sum_{j=1}^{r} \ell_j(x) \ell_0(x)^{i+1} \mathbf{V}_{i,j}|_I \ ,$$
$$\mathsf{Enc}\left(\bigcirc_{i=0}^{b-1} \left[\bar{\mathbf{d}} - i\mathbf{1}^n\right], \bar{\mathbf{r}}^w\right)\Big|_I \overset{?}{=} \sum_{i=0}^{b-1} \sum_{j=1}^{r} \ell_j(x) \ell_0(x)^{i+1} \mathbf{W}_{i,j}|_I \ , \text{ and} \qquad \mathsf{Enc}(\bar{\mathbf{z}}, \bar{\mathbf{r}}^z)|_I \overset{?}{=} \mathbf{Y}|_I + \sum_{j=0}^{r} \beta_j \mathbf{S}_j|_I + \sum_{i=0}^{b-1} \sum_{j=1}^{r} (\gamma_{i,j} \mathbf{V}_{i,j}|_I + \delta_{i,j} \mathbf{W}_{i,j}|_I) \ .$$

Fig. 3. Sublinear hash-based proof of knowledge of short solutions to many linear equations over $\mathbb{Z}_q$.

computationally indistinguishable from a list of commitments to all-zero messages.

Consider the linear combination $\bar{\mathbf{f}}$. Since the vector $\mathbf{s}_0$ is sampled uniformly at random, and multiplied by $\ell_0(x)$, which is non-zero because $x \in \mathbb{Z}_q \backslash \{a_1, \ldots, a_r\}$, $\bar{\mathbf{f}}$ must be uniformly distributed. Similar reasoning applies to $\bar{\mathbf{r}}^f$, $\bar{\mathbf{r}}^v$ and $\bar{\mathbf{r}}^w$ which use uniformly-random masking vectors $\mathbf{r}_0^s$, $\mathbf{r}_{0,1}^v$ and $\mathbf{r}_{0,1}^v$.

Since $\mathbf{y}$ and $\mathbf{r}^y$ are both sampled uniformly at random, $\bar{\mathbf{z}}$ and $\bar{\mathbf{r}}^z$ are also uniformly distributed.

Finally, since each of the Reed-Solomon encodings $\mathbf{Y}$, $\{\vec{S}_j\}_{j=0}^r$, $\{\vec{V}_{i,j}\}_{i=0,j=1}^{b-1,r}$ and $\{\vec{V}_{i,j}\}_{i=0,j=1}^{b-1,r}$ was computed using randomness vectors which were sampled uniformly at random from $\mathbb{Z}_q^\tau$, any τ entries of each codeword are uniformly distributed. Therefore, since $|I| = \tau$, the values of $\mathbf{Y}|_I$, $S|_I$, $V|_I$ and $W|_I$ must be uniformly distributed, conditioned on satisfying the verification equations.

The simulator. Now we give the simulator. First, 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 $\mathbf{F}_1|_I, \ldots, \mathbf{F}_r|_I$, $\mathbf{V}_{i,j}|_I$, 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 $\mathsf{MerklePaths}_I$. □

Theorem 4.3 (Knowledge Soundness). *Let $\mathcal{C} \subset \mathbb{Z}_q^l$ be a Reed-Solomon code of dimension $k' = \max(n,m) + \tau$ 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 $\mathbf{A}, \mathbf{u}_j$ to accept with probability*

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

Then, there exists an efficient probabilistic extraction algorithm $\mathcal{E}$ which, given rewindable black-box access to $\mathcal{P}^*$, either produces vectors $\mathbf{s}_j \in \{0, \ldots, b-1\}^m$ such that $\mathbf{u}_j - \mathbf{A}\mathbf{s}_j \in \{0, \ldots, b-1\}^n$ for all $j = 1, \ldots, r$ or finds a hash collision in expected time at most $64T$ where

$$T := \frac{3}{\varepsilon} + \frac{k-\tau}{\varepsilon/2 - (k/(l-\tau))^\tau}$$

and running $\mathcal{P}^*$ once is assumed to take unit time.

The parameter k is used as part of the strategy for the knowledge extractor $\mathcal{E}$, but not in the protocol itself, as discussed in Remark 3.4.

Proof. The extractor $\mathcal{E}$ 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 $64T$. As above, suppose that $n \leq m$. Let $\mathsf{success}$ and $\mathsf{consistent}$ be the events defined in the proof of

Theorem 3.3. Similarly as before, if $\Pr[\mathsf{success} \wedge \neg\mathsf{consistent}] \geq \varepsilon/2$ then $\mathcal{E}$ finds a hash collision. Otherwise, we assume that $\Pr[\mathsf{success} \wedge \mathsf{consistent}] > \varepsilon/2$.

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

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

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

Next we argue 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 cannot 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. However, we cannot use (a variant) of Lemma 3.5 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 2, the verifier checks a uniformly random linear combination of all of the codewords in the last verification equation. Thus 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 to $d'/3$ from $\mathcal{C}'$, then a uniformly random linear combination would have distance at least $d'/6$, with probability bigger than $1 - 1/(q-r)$. Therefore, as in the proof of Corollary 3.6, we have:

$$\varepsilon/2 < \Pr[\mathsf{success} \wedge \mathsf{consistent}] \leq \frac{1}{q-r} + \left(1 - \frac{d'}{6l}\right)^{\tau} < \frac{1}{q-r} + \left(1 - \frac{k-k'}{6l}\right)^{\tau}.$$

This leads to a contradiction by our assumption on 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

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

be the decoded messages (without the randomness vectors).

By Lemma 3.7, 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 $\mathbf{r} \in \mathbb{Z}_q^{\tau}$.

Now, we apply the exact strategy as in Corollary 3.8 to show that there are at least $\varepsilon(q-r)/4$ accepting transcripts with pairwise different first challenges

x such that:

$$\bar{\mathbf{f}} = \sum_{j=0}^{r} \ell_j(x)\mathbf{s}_j^*, \tag{12}$$

$$\bigcirc_{i=0}^{b-1} \left(\bar{\mathbf{f}} - i\mathbf{1}\right) = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \ell_j(x)\ell_0(x)^{i+1}\mathbf{v}_{i,j}^*, \tag{13}$$

$$\bigcirc_{i=0}^{b-1} \left(\bar{\mathbf{d}} - i\mathbf{1}\right) = \sum_{i=0}^{b-1}\sum_{j=1}^{r} \ell_j(x)\ell_0(x)^{i+1}\mathbf{w}_{i,j}^*. \tag{14}$$

Finally, substituting the expression for $\bar{\mathbf{f}}$ into the second-to-last equation (13) shows that the polynomial

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

of degree at most $(b+1)r$ has a root at x. So, since $\varepsilon > 4(b+1)r/(q-r)$, 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 $\mathbf{s}_j^*$, $j = 1,\dots,r$, has coefficients in $\{0,\dots,b-1\}$.

Moreover, when we define $\mathbf{e}_j^* = \mathbf{u}_j - \mathbf{A}\mathbf{s}_j^*$ for $j = 1,\dots,r$ and $\mathbf{e}_0^* = -\mathbf{A}\mathbf{s}_0^*$, then $\bar{\mathbf{d}} = \sum_{j=1}^{r} \ell_j(x)\mathbf{u}_j - \mathbf{A}\bar{\mathbf{f}} = \sum_{j=0}^{r} \ell_0(x)\mathbf{e}_j^*$. This can be substituted into Equation (14). Then the same reasoning as above yields $\mathbf{e}_j^* \in \{0,\dots,b-1\}^n$, $j = 1,\dots,r$. So we have found the small solutions we were looking for. □

4.1 Proof Size

In one execution of the protocol in Figure 3 the prover sends 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, and τ bitstrings of commitment randomness, each of size 128 bits. 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

$$\left((2m + ((2b+1)r+5)\tau)\lceil\log q\rceil + 128\tau + 256\left(h + \tau\left\lceil\log\left(\frac{l}{h}\right)\right\rceil\right)\right)\Big/ 8192$$

kilobytes. Asymptotically, when we have $r = O(m)$ equations with m secret coefficients each, so $O(m^2)$ secret coefficients in total, we see that the communication cost is of order $O(m)$, which is the square root of the number of secret coefficients. To be a little more precise, the parameter m is multiplied by $2 \log q$, while the parameter r is multiplied by $(2b+1)\tau \log q$. Thus the optimal setting of parameters occurs roughly when $m \approx b\tau r$. If the scenario in which we need to apply our proof system does not have this optimal relation between m and r because r is too large, we can transform our instance into one that does. For example, we can transform several relations of the form $\mathbf{u}_1 = \mathbf{A}\mathbf{s}_1 + \mathbf{e}_1, \ldots, \mathbf{u}_j = \mathbf{A}\mathbf{s}_j + \mathbf{e}_j$ into one relation

$$\begin{bmatrix} \mathbf{A}\ \mathbf{0} \cdots\ \mathbf{0}\ \mathbf{0} \\ \cdots \\ \cdots \\ \cdots \\ \mathbf{0}\ \mathbf{0} \cdots\ \mathbf{0}\ \mathbf{A} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s}_1 \\ \cdots \\ \cdots \\ \cdots \\ \mathbf{s}_j \end{bmatrix} + \begin{bmatrix} \mathbf{e}_1 \\ \cdots \\ \cdots \\ \cdots \\ \mathbf{e}_j \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \cdots \\ \cdots \\ \cdots \\ \mathbf{u}_j \end{bmatrix} \tag{15}$$

If we combine j relations into one in such manner, then the magnitude of m and n increase by a factor of j, while the number of equations r gets reduced by this same factor. By picking the j strategically, one can therefore end up with $m \approx b\tau r$ for the new values of m and r.

Example. Since the proof system in this section is multiple instances of the example from [BLS19, ENS20] (using the same public randomness matrix $\mathbf{A}$), we compare the amortized output size to those papers. As an example, we consider the case where $(q, R, N, M) = (\approx 2^{32}, 1026, 1024, 1024)$ and one wants to prove

$$\mathbf{u}_j = \mathbf{A}\mathbf{s}_j + \mathbf{e}_j \text{ and } \mathbf{s}_j, \mathbf{e}_j \in \{-1, 0, 1\}^M \text{ for } j \in [r] \text{ and } \mathbf{A} \in \mathbb{Z}_q^{N \times M}.$$

We take the value R larger than optimal to illustrate how one can use the idea from above (i.e. (15)) to optimally rearrange the equations. Firstly, note that for every divisor d of R,[8] we can define vectors

$$\mathbf{s}_i^* = \begin{pmatrix} \mathbf{s}_{(i-1)d+1} \\ \vdots \\ \mathbf{s}_{id} \end{pmatrix}, \mathbf{e}_i^* = \begin{pmatrix} \mathbf{e}_{(i-1)d+1} \\ \vdots \\ \mathbf{e}_{id} \end{pmatrix}, \mathbf{u}_i^* = \begin{pmatrix} \mathbf{u}_{(i-1)d+1} \\ \vdots \\ \mathbf{u}_{id} \end{pmatrix}$$

for $i \in [r/d]$ and a matrix $\mathbf{A}^* = \mathbf{I}_d \otimes \mathbf{A}$, where $\mathbf{I}_d$ is the $d \times d$ identity matrix. Clearly, our problem is reduced to proving that each $\mathbf{s}_i, \mathbf{e}_i \in \{-1, 0, 1\}^{Md}$ and

$$\mathbf{u}_i^* = \mathbf{A}^* \mathbf{s}_i^* + \mathbf{e}_i^* \text{ for } i \in [R/d].$$

Now, we can apply our protocol from Fig. 3 for $(b, n, m, r) = (3, N, Md, R/d)$ and find an appropriate divisor d to minimize the proof size per equation.

In this example, we pick $d = 18$, hence $r = 57$. The last term in the expression for the soundness error in Theorem 4.3 limits it to at least 2^{-24}. So we search for

[8] If d is not a divisor of R, we can pad the last equation with zeroes.

parameters k, l and τ under the constraint that the soundness error stays below 2^{-24}. Then 5 iterations of the protocol give a negligible soundness error of less than 2^{-120}. Our search for such parameters minimizing the proof size resulted in

$$k = 50304, \quad l = 55809, \quad \text{and} \quad \tau = 176.$$

With these parameters and $h = 2\tau$, the total proof size for all 5 iterations is 2384 kilobytes. So this translates to an amortized cost of 2.32 kilobytes per equation, which is a noticeable improvement over the 47KB proof size in [ENS20].

Acknowledgements

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

References

ACPS09. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO'09*, pages 595–618, 2009.

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 - CCS'17*, pages 2087–2104, 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 - IEEE S&P'18*, 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 *Advances in Cryptology - CRYPTO'18*, 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 *Advances in Cryptology - EUROCRYPT'16*, 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'17*, pages 336–365, 2017.

BCOS20. Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum snarks for RSIS and RLWE and their applications to privacy. In *International Conference on Post-Quantum Cryptography - PQCrypto'20*, pages 247–267, 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 *Advances in Cryptology - EUROCRYPT'19*, pages 103–128, 2019.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography Conference - TCC'16-B*, pages 31–60, 2016.

BCS19. Carsten Baum, Daniele Cozzo, and Nigel P. Smart. Using topgear in overdrive: A more efficient zkpok for SPDZ. In *Selected Areas in Cryptography - SAC'19*, pages 274–302, 2019.

BDL$^+$18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *International Conference on Security and Cryptography for Networks - SCN'18*, pages 368–385, 2018.

Beu20. Ward Beullens. Sigma protocols for MQ, PKP and SIS, and fishy signature schemes. In *Advances in Cryptology - EUROCRYPT'20*, pages 183–211, 2020.

BKS18. Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-case to average case reductions for the distance to a code. In *Computational Complexity Conference - CCC'18*, pages 1–23, 2018.

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *Advances in Cryptology - CRYPTO'19*, pages 176–202, 2019.

Dam10. Ivan Damgård. On Σ-Protocols. Lecture on Cryptologic Protocol Theory; Faculty of Science, University of Aarhus, 2010.

DKL$^+$18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. pages 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 - CCS'17*, pages 1565–1581, 2017.

dPLS19. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for FHE and ring-lwe ciphertexts. In *International Conference on Practice and Theory of Public Key Cryptography - PKC'19*, pages 344–373, 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 Cryptology ePrint Archive, Report 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. In *Advances in Cryptology - ASIACRYPT'20*, pages 259–288, 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 - CCS'19*, pages 567–584, 2019.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Advances in Cryptology - EUROCRYPT'13*, pages 626–645, 2013.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *ACM Symposium on the Theory of Computing - STOC'92*, pages 723–732, 1992.

KKPP20. Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest. Scalable ciphertext compression techniques for post-quantum kems and their applications. In *Advances in Cryptology - ASIACRYPT'20*, pages 289–320, 2020.

KR08. Yael Tauman Kalai and Ran Raz. Interactive PCP. In *International Colloquium on Automata, Languages and Programming - ICALP'08*, pages 536–547, 2008.

KTX08. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *Advances in Cryptology - ASIACRYPT'08*, pages 372–389, 2008.

LNS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. 12710:215–241, 2021.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *International Conference on Practice and Theory of Public Key Cryptography - PKC'13*, pages 107–124, 2013.

MF21. Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography.* Springer, 2021.

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

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO'08*, pages 554–571, 2008.

RRR16. Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *ACM Symposium on the Theory of Computing - STOC'16*, pages 49–62, 2016.

RVW13. Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *ACM Symposium on the Theory of Computing - STOC'13*, pages 793–802, 2013.

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

Ste93. Jacques Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology - CRYPTO'93*, 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 *Advances in Cryptology - CRYPTO'19*, pages 147–175, 2019.

A The Hiding Property of Reed-Solomon Codes

We show that when $\mathbf{r}$ is sampled uniformly at random from $\mathbb{Z}_q^\tau$, then any τ entries of the Reed-Solomon encoding $\mathsf{Enc}(\mathbf{m}, \mathbf{r})$ corresponding to the input polynomial $f = \sum_{i=0}^{m-1} \mathbf{m}_i X^i + X^m \sum_{i=0}^{\tau-1} \mathbf{r}_i X^i$ follow the uniform distribution over $\mathbb{Z}_q^\tau$.

Let $\zeta_1, \dots, \zeta_l$ be distinct elements of $\mathbb{Z}_q^\times$ used as the evaluation points for the Reed-Solomon code. We can write the encoding $\mathsf{Enc}(\mathbf{m}, \mathbf{r})$ as a matrix multiplication:

$$\mathsf{Enc}(\mathbf{m}, \mathbf{r}) = \begin{bmatrix} 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^{m-1} \\ 1 & \zeta_2 & \zeta_2^2 & \cdots & \zeta_2^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_l & \zeta_l^2 & \cdots & \zeta_l^{m-1} \end{bmatrix} \mathbf{m} + \begin{bmatrix} \zeta_1^m & \cdots & \zeta_1^{m+\tau-1} \\ \zeta_2^m & \cdots & \zeta_2^{m+\tau-1} \\ \vdots & \ddots & \vdots \\ \zeta_l^m & \cdots & \zeta_l^{m+\tau-1} \end{bmatrix} \mathbf{r} = A\mathbf{m} + B\mathbf{r}$$

Let I be a subset of $[l]$ with $|I| = \tau$, and let $B(I)$ be the submatrix of B formed by restricting to the rows in I. Observe $B(I)$ forms a Vandermonde matrix where the i-th row has been multiplied by ζ_i^m. Since the ζ_i are distinct and non-zero, $B(I)$ is invertible. Hence, if $\mathbf{r}$ is sampled uniformly at random from $\mathbb{Z}_q^\tau$, then $B(I)\mathbf{r}$ is uniformly distributed over $\mathbb{Z}_q^\tau$. This argument shows that any τ entries of $\mathsf{Enc}\left(\mathbf{m}, \mathbf{r}\right)$ are uniformly distributed.