Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts

Rafael del Pino[1⋆], Vadim Lyubashevsky[2], and Gregor Seiler[2,3]

[1] ENS Paris
Rafael.Del.Pino@ens.fr
[2] IBM Research – Zurich
vadim.lyubash@gmail.com, Switzerland
[3] ETH Zurich, Switzerland
gseiler@inf.ethz.ch

Abstract. In applications of fully-homomorphic encryption (FHE) that involve computation on encryptions produced by several users, it is important that each user proves that her input is indeed well-formed. This may simply mean that the inputs are valid FHE ciphertexts or, more generally, that the plaintexts m additionally satisfy $f(m) = 1$ for some public function f. The most efficient FHE schemes are based on the hardness of the Ring-LWE problem and so a natural solution would be to use lattice-based zero-knowledge proofs for proving properties about the ciphertext. Such methods, however, require larger-than-necessary parameters and result in rather long proofs, especially when proving general relationships.

In this paper, we show that one can get much shorter proofs (roughly 1.25KB) by first creating a Pedersen commitment from the vector corresponding to the randomness and plaintext of the FHE ciphertext. To prove validity of the ciphertext, one can then prove that this commitment is indeed to the message and randomness and these values are in the correct range. Our protocol utilizes a connection between polynomial operations in the lattice scheme and inner product proofs for Pedersen commitments of Bünz et al. (S&P 2018). Furthermore, our proof of equality between the ciphertext and the commitment is very amenable to amortization – proving the equivalence of k ciphertext / commitment pairs only requires an additive factor of $O(\log k)$ extra space than for one such proof. For proving additional properties of the plaintext(s), one can then directly use the logarithmic-space proofs of Bootle et al. (Eurocrypt 2016) and Bünz et al. (IEEE S&P 2018) for proving arbitrary relations of discrete log commitment.

Our technique is not restricted to FHE ciphertexts and can be applied to proving many other relations that arise in lattice-based cryptography. For example, we can create very efficient verifiable encryption / decryption schemes with short proofs in which confidentiality is based on the hardness of Ring-LWE while the soundness is based on the discrete logarithm problem. While such proofs are not fully post-quantum, they are adequate in scenarios where secrecy needs to be future-proofed, but one only needs to be convinced of the validity of the proof in the pre-quantum era. We furthermore show that our zero-knowledge protocol can be easily modified to have the property that breaking soundness implies solving discrete log in a short amount of time. Since building quantum computers capable of solving discrete logarithm in seconds requires overcoming many more fundamental challenges, such proofs may even remain valid in the post-quantum era.

1 Introduction

Fully-homomorphic encryption (FHE) allows for evaluations of arbitrary functions over encrypted data. The traditional application of this primitive is outsourcing – a user encrypts his data and

⋆ Work done while at IBM Research – Zurich, Switzerland

sends it to a server who performs the (intensive) computation and returns back the encrypted result. In this scenario, the user is the only one affected by the outcome of the computation, and so it is not necessary for him to prove that his ciphertexts he submitted to the server are properly formed.

There are other applications of FHE, however, that involve computations on ciphertexts submitted by several users [LTV12,MW16,PS16]. For example, multi-key FHE allows the server to compute over ciphertexts encrypted under different keys and produce a result that can then be jointly decrypted by the participating parties. One can also use FHE in a "distributed ledger" (e.g. [ABB+18]) setting where users can submit ciphertexts encrypted under some particular public key and a computation can be performed by anyone on behalf of the holder of the secret key to produce an encrypted output. This is useful in scenarios where certain entities (the holder of the secret key in our example) wish to perform only a limited amount of computation.

For the above scenarios where more than one user is involved, it is important that each party provides a zero-knowledge proof that his input is a valid FHE ciphertext – otherwise the final output may, unknowingly to anyone else, be constructed from invalid data. It may furthermore be necessary to prove that the encrypted message satisfies certain additional properties dictated by the protocol. For encryptions based on the discrete logarithm problem, such proofs can be very efficiently constructed for certain relations using techniques in [CS03] and for general circuits using the more recent *logarithmic space* proofs for discrete logarithms [BCC+16,BBB+17]. FHE schemes, on the other hand, are constructed from LWE (or LWE-like) encryption schemes (e.g. [BGV12]), which unfortunately do not enjoy such practical proofs. For example, the most efficient verifiable encryption scheme for Ring-LWE [LN17] ciphertexts only handles linear relations $\mathbf{B}\vec{\mathbf{m}} = \vec{\mathbf{t}}$ and gives proofs of knowledge of an $\vec{\mathbf{m}}'$ satisfying $\mathbf{B}\vec{\mathbf{m}}' = c \cdot \vec{\mathbf{t}}$, where c is some polynomial with small coefficients. This is satisfactory in some scenarios (see [LN17] for examples), but is not general enough for many other applications. Obtaining proofs without the polynomial c even for simple relations would make the proof sizes on the order of megabytes (cf. [LLNW18]).

In this work, we take a different approach for creating such proofs. An FHE (or more generally, a Ring-LWE) ciphertext can be written as

$$\mathbf{A}\vec{\mathbf{s}} = \vec{\mathbf{t}} \tag{1}$$

where $\mathbf{A}$ is the public key, $\vec{\mathbf{t}}$ is the ciphertext, and $\vec{\mathbf{s}}$ consists of the randomness and the message. All operations are performed over some polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(\mathbf{f})$ for some integer q and a monic, irreducible polynomial $\mathbf{f} \in \mathbb{Z}[X]$ of degree d.

The main result of the current work is an efficient protocol for proving knowledge of $\vec{\mathbf{s}}$ with small coefficients in the above equation. Our strategy is to first create a joint Pedersen commitment $t = \mathrm{Com}(\vec{\mathbf{s}})$ to all the coefficients in $\vec{\mathbf{s}}$, and prove in zero-knowledge that these coefficients, when interpreted as a polynomial vector $\vec{\mathbf{s}}$, satisfy (1). At the same time, the proof will also show that the coefficients of $\vec{\mathbf{s}}$ are in the required range for valid Ring-LWE ciphertexts. Moreover, if we have many Ring-LWE ciphertexts $\vec{\mathbf{t}}_1, \ldots, \vec{\mathbf{t}}_k$, then the size of our proof is only approximately an *additive factor* of $O(\log k)$ larger than the proof for one equation in (1).

Once we have a Pedersen commitment of the coefficients of $\vec{\mathbf{s}}$, we can additionally use the aforementioned very efficient zero-knowledge proofs for discrete logarithm commitments [BCC+16,BBB+17] to prove arbitrary properties of the plain-text contained in $\vec{\mathbf{s}}$. This gives us a verifiable encryption scheme (and also a verifiable decryption scheme) for Ring-LWE ciphertexts (see Section 1.5). As an example of the proof size, a proof of ciphertext validity of a Ring-LWE encryption scheme in (9) requires only 1.25KB.

1.1 Post-Quantum Security

One of the side advantages of FHE based on Ring-LWE is that the encryption scheme remains secure against quantum attacks (assuming that the Ring-LWE problem is post-quantum secure). Since Pedersen commitments are statistically-hiding and all the proofs are statistical zero-knowledge, the secrecy of the ciphertext and the Pedersen commitment is still based on just Ring-LWE. The soundness of the proofs, however, is based on the hardness of the discrete log problem and is therefore not post-quantum.

Having the soundness of the proof not be post-quantum is still, for many scenarios, acceptable even if we do foresee quantum computers appearing in the future. For example, all proofs created until quantum computers capable of breaking discrete log actually appear would still be valid. Furthermore, the protocol can be easily altered to force the prover to create his Pedersen commitment and the zero-knowledge proof with "fresh" randomly-chosen generators and complete his proof in a specified amount of time.[4] Breaking the soundness of this proof system would thus require solving the discrete log problem using a quantum computer within a prescribed (e.g. several seconds) time interval.

While building a quantum computer capable of breaking cryptographic problems presents a very substantial scientific and engineering challenge, building one that is capable of solving such problems in *seconds* is a potentially significantly harder problem. For a 2048-bit number, under some reasonable assumptions on the error rate and the speed of each gate computation on a superconducting platform, this would take around 27 hours and a billion physical qubits [FMMC12]. A trapped-ion based computer with very low error rate would need 110 days to perform the same operation [LWF+17]. One can sometimes decrease the running time by utilizing more qubits, but there are several other roadblocks that would keep the computation time from decreasing beyond certain barriers (c.f. [Gid18] for a discussion). While it is too early to guess when (or if) it will be possible to run Shor's algorithm in under a minute, it certainly appears to be a problem that will require overcoming many more fundamental challenges even after a "basic" fault-tolerant universal quantum computer is built.

1.2 Other Applications

Our general result gives a way to prove knowledge that the secret $\vec{\mathbf{s}}$ in the linear equation (1) is the same as in the commitment $\mathrm{Com}(\vec{\mathbf{s}})$, where $\mathrm{Com}(\cdot)$ is a Pedersen commitment to the individual coefficients of $\vec{\mathbf{s}}$. Because (1) is quite generic, it can be used to represent many relations throughout lattice cryptography. For example, ciphertexts, commitments, public keys in encryption / signature schemes, etc. are all of this form. One can therefore apply our protocol as a first step in a larger protocol that needs to prove something about the secret $\vec{\mathbf{s}}$. For example, verifiable encryption and decryption schemes (where the prover or decryptor needs to prove that the plaintext m satisfies $f(m) = 1$ for some public function f) has many applications (c.f. [CS03]) and such schemes that retain the post-quantum secrecy of the ciphertext can thus be built using our techniques. We sketch the construction in Section 1.5 and note that proving validity of FHE ciphertexts is just a special case of verifiable encryption.

[4] If the proof is to be made non-interactive, the randomness for creating the generators could come from some public randomness beacons (e.g. the NIST randomness beacon).

1.3 Previous Related Work

A connection between Ring-LWE and discrete log commitments has been previously explored by Benhamouda et al. [BCK+14]. The construction in the current paper is completely different and enjoys significant advantages (both theoretical and practical) over the aforementioned prior work. Firstly, the modulus q in (1) has to be the same as the group size underlying the discrete log commitment for the proof in [BCK+14] – and taking $q \approx 2^{256}$ would require making the Ring-LWE / FHE scheme significantly less efficient than it needs to be (typical sizes of q are $\approx 2^{30}$). Secondly, the protocol in [BCK+14] requires a separate Pedersen commitment for every coefficient of $\vec{\mathbf{s}}$ rather than one commitment for all the coefficients of $\vec{\mathbf{s}}$. Thirdly, the proof is a Σ-protocol with soundness error $1/d$ (where n is the degree of $\mathbf{f}$) and so needs to be repeated around a dozen times. While [BCK+14] did not provide concrete parameters, we would estimate that our proofs would be shorter by 2 - 3 orders of magnitude. And additionally, our current proof can be amortized for proving k equations as in (1) while only incurring an $O(\log k)$ additive overhead.

Our work can also be seen as complementary to that of Fiore, Gennaro, and Pastro [FGP14] where they give a succinct proof that the evaluation in the FHE scheme was performed correctly for certain types of functions.

1.4 High Level Overview of the Protocol

Our general proof is for k copies of (1) – in other words a proof of a matrix $\mathbf{S} \in \mathcal{R}_q^{m \times k}$ with bounded coefficients such that

$$\mathbf{AS} = \mathbf{T} \bmod (\mathbf{f}, q). \tag{2}$$

We will explicitly write out which modular reductions occur as it will change throughout the protocol.

In this overview, we will sketch the proof of a simpler version of (2), which is just a Ring-LWE / Ring-SIS equation

$$\sum_{i=1}^{m} \mathbf{a}_i \mathbf{s}_i = \mathbf{t} \bmod (\mathbf{f}, q) \tag{3}$$

where $\mathbf{a}_i, \mathbf{t}, \mathbf{s}_i \in \mathcal{R}_q$ and the coefficients of $\mathbf{s}_i$ have absolute value less than B. Afterwards, we will explain how this can be extended to the full proof of (2). Let G be a group of size $p \leq 2^{256}$ in which the discrete problem is hard.

The prover first rewrites (3) so that it is entirely over the ring $\mathbb{Z}[X]$ – i.e. there are no reductions modulo q and $\mathbf{f}$:

$$\sum_{i=1}^{m} \mathbf{a}_i \mathbf{s}_i = \mathbf{t} - \mathbf{r}_1 \cdot q - \mathbf{r}_2 \cdot \mathbf{f}. \tag{4}$$

The polynomials $\mathbf{r}_1$ and $\mathbf{r}_2$ are not unique, but we would like them to simultaneously have small coefficients and be of small degree. We show that $\mathbf{r}_1$ can be of degree $2(d-1)$ and have coefficients of absolute value at most $\frac{d}{2}(Bm + \|\mathbf{f}\|_\infty)$, while $\mathbf{r}_2$ can have degree $d-2$ with coefficients having absolute value at most $\frac{1}{2}(q-1)$.

The prover creates a Pedersen commitment $t = \mathrm{Com}(\mathbf{s}_1, \ldots, \mathbf{s}_m, \mathbf{r}_1, \mathbf{r}_2) \in G$ where each integer coefficient of $\mathbf{s}_i$ and $\mathbf{r}_i$ is in the exponent of a different generator g_j.[5] The prover sends t to the verifier.

[5] If we would like to achieve post-quantum security based on the assumption that discrete log cannot be solved in a prescribed amount of time, then the g_i should not be known to the prover before the start of the proof. This

The verifier chooses a random challenge element $\alpha \in \mathbb{Z}_p$ and sends it to the prover. The prover now needs to give several proofs. In the real protocol, all these will be combined into one proof, but for ease of exposition, we will explain them separately here. The first proof is a range proof $\pi_{s,r}$ from [BBB+17] showing that all the committed values in t are in the correct ranges. The second proof is a proof that (4) evaluated at α holds true over the field $\mathbb{Z}_p$. By the Schwartz-Zippel lemma, this implies that with probability $> 1 - 2d/|G|$, this equation also holds true over the polynomial ring $\mathbb{Z}_p[X]$. Since we have already proven that the coefficients of $\mathbf{s}_i$ and $\mathbf{r}_i$ are relatively small and we assumed that q is also small (compared to p), we know that if (4) holds true in $\mathbb{Z}_p[X]$, then it also holds over $\mathbb{Z}[X]$ because no reduction modulo p takes place. This will complete the proof. We now just have to prove that (4) evaluated at α holds true mod p.

Define the matrices

$$U = \begin{bmatrix}\mathbf{a}_1(\alpha) \cdots \mathbf{a}_m(\alpha) & q & \mathbf{f}(\alpha)\end{bmatrix} \bmod p,\ S = \begin{bmatrix} - & \mathbf{s}_1 & - \\ & \cdots & \\ - & \mathbf{s}_m & - \\ - & \mathbf{r}_1 & - \\ - & \mathbf{r}_2 & - \end{bmatrix},\ V = \begin{bmatrix} 1 \\ \alpha \\ \cdots \\ \alpha^{d-1} \end{bmatrix} \bmod p, \tag{5}$$

where the rows of S consist of the integer coefficients of $\mathbf{s}_i$ and $\mathbf{r}_i$ with the constant coefficient being in the leftmost column row and the coefficients of X^{d-1} being in the rightmost (e.g. if $\mathbf{s}_i = \sum_{j=0}^{d-1} \sigma_j X^j$, then the i^{th} row of S is $\begin{bmatrix}\sigma_0 & \sigma_1 & \cdots & \sigma_{d-1}\end{bmatrix}$. With this notation, observe that the matrix product

$$SV = \begin{bmatrix}\mathbf{s}_1(\alpha) & \cdots & \mathbf{s}_m(\alpha) & \mathbf{r}_1(\alpha) & \mathbf{r}_2(\alpha)\end{bmatrix}^T \bmod p,$$

and so

$$USV = \sum_{i=1}^{m} \mathbf{a}_i(\alpha)\mathbf{s}_i(\alpha) + \mathbf{r}_1(\alpha)q + \mathbf{r}_2(\alpha)\mathbf{f}(\alpha) \bmod p.$$

Thus if we prove that

$$USV = \mathbf{t}(\alpha) \bmod p, \tag{6}$$

then we will end up proving that (4) evaluated at α is true modulo p. Since U, V and $\mathbf{t}(\alpha)$ are public and we have a commitment to the coefficients of S, we can apply an extension of the inner-products proofs from [BCC+16,BBB+17] to prove our linear relation.[6] To complete the protocol, the prover simply sends $\pi, \pi_{s,r}$ to the verifier and he accepts if all the proofs are correct.

Combining the Two Proofs. In the real protocol which we describe in Section 5, we combine the two proofs $\pi_{s,r}$ and π into one. The reason is that the range proof $\pi_{s,r}$ in [BBB+17] works by writing each coefficient in binary, storing a matrix of these coefficients, and then giving a proof that each coefficient of the decomposition is 0 or 1 (the number of these coefficients then implies the range). Due to the fact that the ranges of the $\mathbf{s}_i$ and $\mathbf{r}_i$ are different, storing these in the same matrix would require us to increase the size of the matrix to accommodate the largest coefficients,

can be arranged by either having the verifier sending them (or more precisely, send a short seed that expands into the prescribed number of generators) at the start of the protocol or using a randomness beacon in non-interactive proofs.

[6] The "inner-product" proofs in [BCC+16,BBB+17] show that the vectors committed to in a Pedersen commitment satisfy a linear relation. This can also be extended to matrices.

which would be wasteful. Thus instead of proving the matrix equation (6), we write these out as a series of appropriate equations (each of varying lengths) where the coefficients of S are in binary and prove those instead. This allows us to do a range proof and the proof of (6) in one step.

We provide explicit details of the above algorithm in Section 5. We additionally obtain a tighter security proof of the inner-product proof of [BCC$^+$16,BBB$^+$17] by using a different extraction strategy, described in Section 3. In addition, our zero-knowledge range proof is somewhat simpler than the one in [BBB$^+$17] because our range proof is constructed on top of a zero-knowledge inner product proof instead of the original Bulletproof inner product proof which is not zero-knowledge. This allows for not blinding the vectors in the range proof simplifying extraction and saving two rounds of the protocol. The additional complexity in the inner product proof is basically just a Schnorr proof (see Section 4). These small improvements may be of independent interest.

Some observations about the proof strategy. The reason that we converted (3) into (4) and then used the Schwartz-Zippel lemma for proving (4) is for reducing the *time* complexity of the proof. An alternate, simpler, procedure for proving (3) would have been the following: first write (3) as

$$\sum_{i=1}^{m} \mathbf{a}_i \mathbf{s}_i = \mathbf{t} + \mathbf{r}_1 q \bmod \mathbf{f}, \tag{7}$$

and create the commitments t_s and t_{r_1} as before. Now, observe that polynomial multiplication $\mathbf{a}_i \mathbf{s}_i$ can be written as a matrix / vector product $\mathbf{A}\mathbf{s}$, where column j (labeled from 0 to $d-1$) of $\mathbf{A}$ consists of the coefficients of the $d-1$ degree polynomial $\mathbf{a}_i X^j \bmod \mathbf{f}$ and $\mathbf{s}$ is a vector of coefficients of $\mathbf{s}_i$. Thus $\sum_{i=1}^{m} \mathbf{a}_i \mathbf{s}_i$ can be written as a matrix / vector product itself. Then one could directly apply the modified inner-product proof to prove (7) modulo p, which would again imply that this equation holds true over $\mathbb{Z}$ (since the coefficients are all much smaller than p), and so this implies (3).

The main problem with the above approach is that the matrices $\mathbf{A}$ are $d \times d$ matrices, and so the proof of matrix/vector product would require $O(d^2)$ exponentiations (or multiplications in elliptic curve groups) in G. For typical values of $d > 1000$, this operation is quite expensive and could take several minutes even on a reasonably powerful machine. Our proof, on the other hand, takes advantage of the fact that the operations can be interpreted over the ring $\mathbb{Z}_p[X]$ for a very large p and one can then prove polynomial equality via the Schwartz-Zippel lemma. Since polynomial evaluation is an inner-product of d-dimensional vectors, constructing a matrix product proof only requires $O(d)$ exponentiations per evaluation. Note that this is also the reason that our proofs would be much less computationally efficient for proving relations over $\mathbb{Z}$ (i.e. LWE / SIS relations).

Another issue to draw attention to is that the polynomial equations we want to prove are modulo q, whereas the proofs are done modulo a larger p. As mentioned before, the reason for this is that in typical cryptographic applications of the Ring-LWE / Ring-SIS problems (such as FHE), the modulus q is not very large (smaller than 2^{40}). On the other hand, the discrete log commitments must be performed over a much larger-size group. If, however, an application called for the modulus q to be a large prime, then our proof could use $q = p$, and we would never need to switch to working over $\mathbb{Z}[X]$ – we could always work over $\mathbb{Z}_q[X]$ and have no need for the polynomial $\mathbf{r}_1$.

Simultaneously proving k polynomial equations. The proof for proving knowledge of $\mathbf{S}$ satisfying (2) is a straightforward extension of the above-described algorithm with the strategy for the proof being the same. First, we will prove that in the analogue of (4),

$$\mathbf{AS} = \mathbf{T} - q\mathbf{R}_1 - \mathbf{f}\mathbf{R}_2, \tag{8}$$

all the coefficients of $\mathbf{S}, \mathbf{R}_1, \mathbf{R}_2$ are small and then prove that the above equation holds, with high probability, over the ring $\mathbb{Z}_p[X]$ for a very large p. This will imply that (8) also holds over $\mathbb{Z}[X]$, and thus (2) is true. We now describe the protocol in slightly more detail.

The first step of the protocol remains virtually identical with the prover committing to $\mathbf{S}$ and $\mathbf{R}_1, \mathbf{R}_2$. After receiving the challenge α, the prover again wishes to show that the coefficients of $\mathbf{S}, \mathbf{R}_1, \mathbf{R}_2$ are in the appropriate ranges and prove the equality of (8) where each polynomial is evaluated at α.

If we define $I_n \in \mathbb{Z}^{n \times n}$ to be the identity matrix, then one can rewrite what we would like to prove as

$$\begin{bmatrix}\mathbf{A}(\alpha) & qI_n & \mathbf{f}(\alpha)I_n\end{bmatrix} \cdot \begin{bmatrix}\mathbf{S}(\alpha) \\ \mathbf{R}_1(\alpha) \\ \mathbf{R}_2(\alpha)\end{bmatrix} = \mathbf{T}(\alpha) \bmod p.$$

If, for a polynomial $m \times k$ matrix $\mathbf{S}$, we create the $m \times (kd)$ integer matrix $\vec{S}$ by writing each polynomial in $\mathbf{S}$ as a row consisting of its d coefficients (the way way that $\mathbf{s}_i$ were expanded in the matrix S in (5)), then we can rewrite the above equation as

$$\begin{bmatrix}\mathbf{A}(\alpha) & qI_n & \mathbf{f}(\alpha)I_n\end{bmatrix} \cdot \begin{bmatrix}\vec{S} \\ \vec{R}_1 \\ \vec{R}_2\end{bmatrix} \cdot \left(I_k \otimes \begin{bmatrix}1 \\ \alpha \\ \cdots \\ \alpha^{d-1}\end{bmatrix} \right) = \mathbf{T}(\alpha) \bmod p.$$

Since all the matrices in the above equation except $\begin{bmatrix}\vec{S} \\ \vec{R}_1 \\ \vec{R}_2\end{bmatrix}$ are public, we can again apply the modified inner-product proof from [BCC$^+$16,BBB$^+$17] to prove the equality modulo p. And again, as before, our real protocol would combine the range proof and modified inner-product proof into one proof.

1.5 Application to Verifiable Encryption and Decryption for Ring-LWE Ciphertexts

Notice that the first step of our proof involved creating a Pedersen commitment t to the coefficients of $\mathbf{S}$. The rest of the proof then went on to show that the commitment is really to an $\mathbf{S}$ satisfying (2). Since at the end of the protocol, we end up with a Pedersen commitment to $\mathbf{S}$, we can use another SNARK (e.g. one from [BBB$^+$17]) that proves arbitrary relations of its committed values. Thus just proving knowledge of $\mathbf{S}$ naturally gives rise to verifiable encryption and decryption schemes for Ring-LWE encryption, as we sketch below.

In a verifiable encryption scheme, the encryptor produces an encryption of a message m and a ZKPoK that the ciphertext is a valid encryption to m and that $f(m) = 1$ for a public function f. Consider the following "usual" encryption scheme based on Ring-LWE [LPR13]:

The secret key are polynomials $\mathbf{s}, \mathbf{e}$ with small, bounded coefficients and the public key consists of a random polynomial $\mathbf{a} \in \mathcal{R}_q$ and $\mathbf{t} = \mathbf{as} + \mathbf{e} \in \mathcal{R}_q$.

The encryption of a message $\mathbf{m} \in \mathcal{R}_q$, where all coefficients of $\mathbf{m}$ are in the range $[0, p)$, is created as in the below equation, where $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$ are polynomials with bounded coefficients.

$$\begin{bmatrix} p\mathbf{a} & p & 0 & 0 \\ p\mathbf{t} & 0 & p & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \tag{9}$$

For a verifiable encryption scheme, we can use our proof system with $\mathbf{A} \in \mathcal{R}_q^{2\times 4}$ and $\mathbf{S} \in \mathcal{R}_q^{4\times 1}$ to create a Pedersen commitment(s) to $\mathbf{S}$ and prove that all the coefficients of $\mathbf{r}, \mathbf{e}_i, \mathbf{m}$ lie within their prescribed bounds and that (9) is satisfied by the commitment(s) representing $\mathbf{S}$. The preceding proves knowledge of the plaintext $\mathbf{m}$ for the ciphertext $\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$.

To decrypt a ciphertext $\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$, the decryptor first computes

$$\mathbf{v} - \mathbf{us} = p(\mathbf{er} + \mathbf{e}_2 - \mathbf{se}_1) + \mathbf{m}. \tag{10}$$

Since all the coefficients of the above equation are small, no reduction modulo q takes place and this equation holds true over $\mathbb{Z}[X]$. Computing $\mathbf{v} - \mathbf{us} \bmod p$ therefore recovers $\mathbf{m}$.

To construct a verifiable decryption scheme, let $\mathbf{g} = \mathbf{er} + \mathbf{e}_2 - \mathbf{se}_1$ from the above equation. Let β be a bound on $\mathbf{g}$ such that no reduction modulo q takes place in (10) and so decryption still works (i.e. β should be less than approximately q/p). Then the decryptor should be able to prove knowledge of $\mathbf{s}, \mathbf{e}, \mathbf{g}, \mathbf{m}$ in the following equation with coefficients of $\mathbf{s}, \mathbf{e}$ having the appropriate bounds and $\mathbf{m}$ having all coefficients in $[0, p)$.

$$\begin{bmatrix} \mathbf{a} & 1 & 0 & 0 \\ \mathbf{u} & 0 & p & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s} \\ \mathbf{e} \\ \mathbf{g} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathbf{v} \end{bmatrix}. \tag{11}$$

Proving the above shows that $\mathbf{m}$ is a valid decryption. To show that there is only one possible decryption (i.e. only one possible solution to the above equation), suppose there exist two solutions:

$$\begin{bmatrix} \mathbf{a} & 1 & 0 & 0 \\ \mathbf{u} & 0 & p & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s} \\ \mathbf{e} \\ \mathbf{g} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathbf{v} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{a} & 1 & 0 & 0 \\ \mathbf{u} & 0 & p & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s}' \\ \mathbf{e}' \\ \mathbf{g}' \\ \mathbf{m}' \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathbf{v} \end{bmatrix}. \tag{12}$$

If $\mathbf{s} \neq \mathbf{s}'$, then the first row of (12) implies a non-zero solution to

$$\mathbf{a}(\mathbf{s} - \mathbf{s}') + (\mathbf{e} - \mathbf{e}') = 0.$$

Writing $\mathbf{a}$ as above can either be shown to be impossible either via an information-theoretic argument or via the computational assumption that the Ring-SIS problem [PR06,LM06] is hard.[7]

[7] In general, the polynomial $\mathbf{a}$ is created as $H(seed)$, where H is a cryptographic hash function and the seed is public. It is therefore a valid assumption that $\mathbf{a}$ is random in $\mathcal{R}_q$.

If $\mathbf{s} = \mathbf{s}'$, then the second row of (12) implies that $p(\mathbf{g}-\mathbf{g}')+(\mathbf{m}-\mathbf{m}') = \mathbf{0}$. Since the coefficients are small enough that no reduction modulo q takes place, the preceding implies that $\mathbf{m}-\mathbf{m}'$ is a multiple of p, which implies that $\mathbf{m} = \mathbf{m}'$ (since the coefficients of $\mathbf{m}-\mathbf{m}'$ are in the range $(-p, p)$.)

1.6 Open Problems

We have shown how linear relations over polynomial rings can have very compact proofs by converting the problem into a form that is compatible with the compact SNARKs in [BCC$^+$16,BBB$^+$17]. While the proofs are small, creating such proofs may require on the order of hundreds of thousands of exponentiations. It would therefore be interesting to see whether one can transform the problem into a form compatible with SNARKS that are less compact but may require fewer operations, such as for example those in [WTS$^+$18]. Since the latter proofs are particularly tailored to parallelizable functions, they may also result in rather efficient proofs for LWE / SIS ciphertexts, and not require one to work over polynomial rings. We leave this direction as an open problem.

2 Notation

We use bold letters $\mathbf{f}$ for polynomials, arrows for column vectors as in $\vec{v}$, and capital letters A for matrices. Vectors and matrices of polynomials are denoted by bold letters $\vec{\mathbf{v}}$ with arrows and bold capital letters $\mathbf{M}$, respectively. We write $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{f})$ for the ring of integer polynomials modulo a monic irreducible polynomial $\mathbf{f} \in \mathbb{Z}[X]$, $\mathcal{R}_q$ for the quotient ring $\mathcal{R}/q\mathcal{R}$ for some prime q and similarly $\mathbb{Z}_p$ for $\mathbb{Z}/p\mathbb{Z}$.

Let $\vec{v}_1 \in \mathbb{Z}_p^n$ and $\vec{v}_2 \in \mathbb{Z}_p^n$ be two vectors over $\mathbb{Z}_p$. Then we write $\langle \vec{v}_1, \vec{v}_2 \rangle \in \mathbb{Z}_p$, $\vec{v}_1 \circ \vec{v}_2 \in \mathbb{Z}_p^n$ and $\vec{v}_1 \otimes \vec{v}_2 \in \mathbb{Z}_p^{n^2}$ for their inner product, componentwise product and tensor product, respectively.

Norms. The absolute value $|a|$ of an element $a \in \mathbb{Z}_q$ is defined to be the absolute value of the centralized representative in $\{-(q-1)/2, \ldots, (q-1)/2\}$. The infinity norm $\|\mathbf{s}\|_\infty$ of a polynomial $\mathbf{s} \in \mathcal{R}_q$ is the maximum absolute value of all of its coefficients. Likewise, the infinity norm $\|\vec{\mathbf{s}}\|_\infty$ of a vector of polynomials is the maximum over the infinity norms of its coefficient polynomials.

Multi exponentiations. For a group G of order p, written multiplicatively, and vectors $\vec{g} = (g_1, \ldots, g_n)^T \in G^n$ and $\vec{a} = (a_1, \ldots, a_n)^T \in \mathbb{Z}_p^n$ we use the notation

$$\vec{g}^{\vec{a}} = g_1^{a_1} \ldots g_n^{a_n} \in G.$$

Throughout the paper the group G will be understood to be cyclic of prime order p with hard computational discrete-log problem. A Pedersen multi-commitment over generators $\vec{g} \in G^n, u \in G$ to a vector $\vec{v} \in \mathbb{Z}_p^n$ with randomness $\rho \xleftarrow{\$} \mathbb{Z}_p$ is given by the multi-exponentiation $t = \vec{g}^{\vec{v}} u^\rho$. This is clearly perfectly hiding and computationally binding under the assumption that it is hard to compute a non-trivial discrete-log relation between the generators $\vec{g}, u$. The latter problem is easily seen to be equivalent to the discrete-log problem.

Serializing matrices to vectors. We will need to serialize matrices $A \in \mathbb{Z}_p^{n\times m}$ to vectors. For this reason we define functions

$$\text{Serialize}\colon \mathbb{Z}_p^{n\times m} \to \mathbb{Z}_p^{nm},\ A \mapsto \vec{a}$$

where $\vec{a}$ contains the coefficients of A in row major order. So if $A = (a_{ij})$, $0 \leq i \leq n-1$, $0 \leq j \leq m-1$, then $\vec{a} = (a_i)$ with $a_{mi+j} = a_{ij}$. In many programming languages, most notably C, this is how matrices are stored in memory so that Serialize is a non-operation in these languages. We extend Serialize to polynomial matrices over $\mathbb{Z}[X]$ by first expanding each polynomial to its row coefficient vector and then proceeding as before.

Expanding integers to their binary representation. We will also need to map integers to their binary representation, including negative integers. For this we define the function

$$\text{Binary}_b\colon \{-2^{b-1}, \ldots, 2^{b-1}-1\} \to \{0,1\}^b,\ z \mapsto \vec{z}$$

that maps a signed b-bit integer to its binary representation using two's complement. More precisely, $\vec{z} = (z_0, \ldots, z_{b-1})^T$ is defined by

$$z = z_0 + z_1 2 + \cdots + z_{b-2}2^{b-2} - z_{b-1}2^{b-1}.$$

Again this representation for signed integers is used by all modern CPU's and Binary is a non-operation. We extend Binary to vectors where Binary is applied to each coefficient individually.

3 Forking Lemma

For proving the security of proof systems based on the Bulletproof technique from [BBB+17] one needs a special forking lemma which shows that it is possible to obtain many accepting transcripts from a prover for challenges that are organized in a large tree. The forking lemma used in the Bulletproof paper goes back to [BCC+16, Lemma 1]. It is only stated in terms asymptotic in the security parameter. Moreover, the tree finding algorithm for computing the tree that is given and analyzed in the proof of the forking lemma does not try to avoid collisions between the challenges. But it is necessary that there are no collision so that the transcripts can be used for extraction. Therefore, in order to compute the success probability of the tree finding algorithm, the collision probability has to be taken into account in addition to the failure probability of the prover. For a 256 bit curve, the collision probability gets quite large for moderately sized trees and as a result of this the reasoning of the forking lemma only applies to provers whose failure probability $1-\varepsilon$ is small. Concretely, to obtain a tree of accepting transcripts of height μ where every inner node has n children one needs $\varepsilon > n^\mu/2^{85}$. For example in the case of the Bulletproof inner product proof, where $n = 4$ and $\mu = \log l$ with l the length of the vectors, $\varepsilon > l^2/2^{85}$ and the forking lemma only proves the inner product proof to be sound with soundness error 2^{-35} if $l = 2^{25}$, a length easily reached in our application. One would need to repeat the proof four times in order to get below 2^{-128}.

We give a different forking lemma with a different extraction algorithm together with a concrete analysis in this section. Our forking lemma achieves negligible soundness error. It is still non-tight though, which is unavoidable as one needs to obtain $n^\mu = l^{\log n}$ transcripts. We stress that we do not think that this non-tightness in the security proof allows for any actual attacks for 256 bit curves. Let us start by recalling the definition of a tree of accepting transcripts.

Definition 3.1. *Let $\mathcal{P}^*$ be a deterministic prover for a $(2\mu+1)$-move interactive proof protocol where the honest verifier $\mathcal{V}$ sends μ challenges in steps $2, 4, \ldots, 2\mu$. An $(n_1, \ldots, n_\mu)$-*tree of accepting transcripts *associated with $\mathcal{P}^*$ is a tree of height μ of the following form. Every node in level i, $0 \leq i \leq \mu - 1$, has precisely n_{i+1} children, all nodes except the root are labeled by a challenge and each leaf additionally contains the transcript obtained by interacting with $\mathcal{P}^*$ and sending the challenges in the path from the root to this leaf. Moreover, the challenges in all nodes with the same parent are distinct and $\mathcal{V}$ accepts all transcripts in the leaves.*

Lemma 3.2. *Let $\mathcal{P}^*$ be a deterministic prover for a $(2\mu+1)$-move interactive proof protocol where the honest verifier $\mathcal{V}$ sends $\mu = \log(l)$ uniformly random challenges from a set $\mathcal{C}$ of size p in steps $2, 4, \ldots, 2\mu$. Then there exists an algorithm* TREE-FINDER *that, when given rewindable black-box access to $\mathcal{P}^*$, computes an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts with probability at least $1/4$ in expected time at most*

$$O\left(\frac{l^{\log n + \log \alpha} \log l}{\varepsilon}\right) \quad (l \to \infty)$$

for every $\alpha > \left(\frac{1}{1-n/p}\right)^2$ and with $n = \max_{1 \leq i \leq \mu-1} n_i$ under the assumption that $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability $\varepsilon \geq \frac{\alpha^\mu}{\alpha - 1} \frac{n_\mu}{p} = \frac{l^{\log \alpha}}{\alpha - 1} \frac{n_\mu}{p}$. Running $\mathcal{P}^*$ once is assumed to take unit time.

Proof. We construct TREE-FINDER = TREE-FINDER(1) as a recursive algorithm with TREE-FINDER(i), $i = 1, \ldots, \mu$, interacting with $\mathcal{P}^*$ from the $2i$-th move onward. A naive first approach would be as follows. For $i < \mu$, TREE-FINDER(i) would run $\mathcal{P}^*$ until and including move $2i+1$ sending a uniformly random challenge $c_i \in \mathcal{C}$ in step $2i$. Then the algorithm would call TREE-FINDER($i+1$). Afterwards it would rewind $\mathcal{P}^*$ back to just after step $2(i-1)+1$ and repeat the process for a total of n_i *different* challenges. So in the second iteration TREE-FINDER(i) would sample a uniform challenge from $\mathcal{C} \setminus \{c_i\}$. The tree-finding algorithm TREE-FINDER(μ) in the last level would send a last challenge c_μ and check whether the interaction with $\mathcal{P}^*$ led to a valid proof, i.e. $\mathcal{V}$ would accept the proof. Then it would repeat for as many last challenges c_μ as needed to get n_μ valid proofs for n_μ different c_μ. The problem with this approach is that in any level for many challenges c_i there might only be very few continuations $c_{i+1}, \ldots, c_\mu$ that lead to valid proofs (or none at all). Hence the tree-finding algorithm might run into dead ends where TREE-FINDER(μ) runs for a very long time or does not terminate at all.

For fixed challenges $c_1, \ldots, c_{i-1}$, let ε_i be the acceptance probability over all uniform continuations $c_i, \ldots, c_\mu$. In particular $\varepsilon_1 = \varepsilon$. Then for some c_i let $\varepsilon_{i+1} = \varepsilon_{i+1}(c_i)$ be the acceptance probability under the additional condition that the i-th challenge is c_i. Now from a standard heavy rows / averaging argument we know $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$, $\alpha > 1$, for at least a fraction of $1 - 1/\alpha$ of the c_i. Therefore our solution to the problem is as follows. After choosing c_i, TREE-FINDER(i) estimates ε_{i+1} by running $\mathcal{P}^*$ until the end for many continuations $c_{i+1}, \ldots, c_\mu$ and counting the number of valid proofs. Then the tree finding algorithm only continues with c_i if the acceptance probability does not decrease too much by fixing c_i. The complete algorithm is as follows where $1 < \lambda < \sqrt{\alpha}$ and T_i are specified later.

1: **function** TREE-FINDER(i)
2: Initialize *tree* as a tree containing only an empty root
3: $\mathcal{C}' = \emptyset$
4: **while** $|\mathcal{C}'| < n_i$ **do**
5: **if** $i = \mu$ **then**
6: Run $\mathcal{P}^*$ until the end using a fresh challenge $c_\mu \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}'$ and let tr be the transcript of the full interactive proof
7: **if** proof is valid **then**
8: Append new leaf (c_μ, tr) to the root of *tree*
9: $\mathcal{C}' = \mathcal{C}' \cup \{c_\mu\}$
10: **end if**
11: **else**
12: **repeat**
13: Run $\mathcal{P}^*$ up to and including step $2i+1$ using a fresh challenge $c_i \xleftarrow{\$} \mathcal{C} \setminus \mathcal{C}'$
14: $count = 0$
15: **for** $j = 1, \ldots, T_i$ **do**
16: Run $\mathcal{P}^*$ until the end with fresh challenges $c_{i+1}, \ldots, c_\mu \xleftarrow{\$} \mathcal{C}$
17: **if** proof is valid **then**
18: $count = count + 1$
19: **end if**
20: Rewind $\mathcal{P}^*$ back to just after step $2i+1$
21: **end for**
22: **if** $count < \lambda T_i \frac{\varepsilon}{\alpha^i}$ **then**
23: Rewind $\mathcal{P}^*$ back to just after step $2(i-1)+1$
24: **end if**
25: **until** $count >= \lambda T_i \frac{\varepsilon}{\alpha^i}$
26: $tree' \leftarrow$TREE-FINDER$(i+1)$
27: Label root of $tree'$ by c_i and append $tree'$ to the root of *tree*
28: $\mathcal{C}' = \mathcal{C}' \cup \{c_i\}$
29: **end if**
30: **end while**
31: **return** *tree*
32: **end function**

We analyze the algorithm under the assumption $\varepsilon_i \geq \varepsilon/\alpha^{i-1}$. The challenge c_i is chosen and the acceptance probability $\varepsilon_{i+1} = \varepsilon_{i+1}(c_i)$ estimated during the loop in lines $12-25$. We define the

following probabilities in one iteration of the loop.

$$\begin{aligned}
p_0 &= \Pr\left[\mathit{count} < \lambda T_i \frac{\varepsilon}{\alpha^i}\right], \\
p_1 &= \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \text{ and } \varepsilon_{i+1}(c_i) \geq \frac{\varepsilon}{\alpha^i}\right], \\
p_2 &= \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \text{ and } \varepsilon_{i+1}(c_i) < \frac{\varepsilon}{\alpha^i}\right].
\end{aligned}$$

So p_0, p_1 and p_2 are the probabilities of continuing the loop, choosing a "good" challenge c_i, and choosing a "bad" challenge, respectively. Note that $p_0 + p_1 + p_2 = 1$. By the heavy rows argument, with probability at least $1 - 1/\sqrt{\alpha} - n/p$, $\varepsilon_{i+1}(c_i) \geq \varepsilon/(\sqrt{\alpha} \cdot \alpha^{i-1})$. Therefore and by the Chernoff bound,

$$\begin{aligned}
p_1 &= \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \text{ and } \varepsilon_{i+1} \geq \frac{\varepsilon}{\alpha^i}\right] \\
&\geq \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \text{ and } \varepsilon_{i+1} \geq \frac{\varepsilon}{\sqrt{\alpha} \cdot \alpha^{i-1}}\right] \\
&= \Pr\left[\varepsilon_{i+1} \geq \frac{\varepsilon}{\sqrt{\alpha} \cdot \alpha^{i-1}}\right] \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \,\middle|\, \varepsilon_{i+1} \geq \frac{\varepsilon}{\sqrt{\alpha} \cdot \alpha^{i-1}}\right] \\
&\geq \left(1 - \frac{1}{\sqrt{\alpha}} - \frac{n}{p}\right) \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \,\middle|\, \varepsilon_{i+1} \geq \frac{\varepsilon}{\sqrt{\alpha} \cdot \alpha^{i-1}}\right] \\
&\geq \left(1 - \frac{1}{\sqrt{\alpha}} - \frac{n}{p}\right) \Pr\left[\mathit{count} \geq \frac{\lambda}{\sqrt{\alpha}} T_i \varepsilon_{i+1} \,\middle|\, \varepsilon_{i+1} \geq \frac{\varepsilon}{\sqrt{\alpha} \cdot \alpha^{i-1}}\right] \\
&\geq \left(1 - \frac{1}{\sqrt{\alpha}} - \frac{n}{p}\right) \left(1 - \Pr\left[\mathit{count} \leq \frac{\lambda}{\sqrt{\alpha}} T_i \varepsilon_{i+1} \,\middle|\, \varepsilon_{i+1} \geq \frac{\varepsilon}{\sqrt{\alpha} \cdot \alpha^{i-1}}\right]\right) \\
&\geq \left(1 - \frac{1}{\sqrt{\alpha}} - \frac{n}{p}\right) \left(1 - \exp\left(-\frac{(1 - \lambda/\sqrt{\alpha})^2}{2} T_i \varepsilon_{i+1}\right)\right) \\
&\geq \left(1 - \frac{1}{\sqrt{\alpha}} - \frac{n}{p}\right) \left(1 - \exp\left(-\frac{(\sqrt{\alpha} - \lambda)^2}{2\sqrt{\alpha}} T_i \frac{\varepsilon}{\alpha^i}\right)\right) = p_1'
\end{aligned}$$

On the other hand we find for p_2,

$$\begin{aligned}
p_2 &= \Pr\left[\varepsilon_{i+1} < \frac{\varepsilon}{\alpha^i}\right] \Pr\left[\mathit{count} \geq \lambda T_i \frac{\varepsilon}{\alpha^i} \,\middle|\, \varepsilon_{i+1} < \frac{\varepsilon}{\alpha^i}\right] \\
&\leq \Pr\left[\mathit{count} \geq (1 + \delta) T_i \varepsilon_{i+1} \,\middle|\, \varepsilon_{i+1} < \frac{\varepsilon}{\alpha^i}\right] \\
&\leq \exp\left(-\frac{1}{3} \min(\delta, \delta^2) \varepsilon_{i+1} T_i\right)
\end{aligned}$$

where we have set $\delta > 0$ such that $(1 + \delta)\varepsilon_{i+1} = \lambda\varepsilon/\alpha^i$, i.e. $\delta = \frac{\lambda\varepsilon}{\alpha^i \varepsilon_{i+1}} - 1$. We want to bound $\min(\delta, \delta^2)\varepsilon_{i+1}$ from below. Notice that

$$\delta^2 \varepsilon_{i+1} = \frac{\lambda^2 \varepsilon^2}{\alpha^{2i} \varepsilon_{i+1}} - \frac{2\lambda\varepsilon}{\alpha^i} + \varepsilon_{i+1}$$

is strictly decreasing on the interval $\varepsilon_{i+1} \in [0, \varepsilon/\alpha^i[$. Hence,

$$\delta^2 \varepsilon_{i+1} > \frac{\lambda^2 \varepsilon}{\alpha^i} - \frac{2\lambda\varepsilon}{\alpha^i} + \frac{\varepsilon}{\alpha^i} = \frac{\varepsilon}{\alpha^i}(\lambda - 1)^2.$$

Moreover, $\delta\varepsilon_{i+1} > (\lambda-1)\frac{\varepsilon}{\alpha^i}$ and therefore

$$p_2 < \exp\left(-\frac{(\lambda-1)^2}{3}T_i\frac{\varepsilon}{\alpha^i}\right) = p_2'.$$

We set λ such that the arguments of the exponential function in p_1' and p_2' are equal; that is,

$$\frac{(\sqrt{\alpha}-\lambda)^2}{2\sqrt{\alpha}} = \frac{(\lambda-1)^2}{3}.$$

Then $p_1' = (1-1/\sqrt{\alpha}-n/p)(1-p_2')$. With these probabilities we now calculate the probability that the loop ends with a bad c_i. It is given by

$$p_{\text{bad}} = \sum_{j=0}^{\infty} p_0^k p_2 = \frac{p_2}{1-p_0} = \frac{p_2}{p_1+p_2} = \frac{1}{1+p_1/p_2} < \frac{1}{1+p_1'/p_2'} = \frac{p_2'}{p_1'+p_2'}.$$

The probability that the first-level TREE-FINDER(1) chooses n_1 good challenges c_1 is $(1-p_{\text{bad}})^{n_1}$. Under this condition our assumption $\varepsilon_2 \geq \varepsilon/\alpha$ is true for the second-level tree finders and they all choose only good challenges with probability $(1-p_{\text{bad}})^{n_1 n_2}$. Write $N = \sum_{i=1}^{\mu-1}(n_1 \dots n_i) \leq \sum_{i=1}^{\mu-1} n^i = \frac{n^\mu - n}{n-1} < n^\mu = (2^{\log n})^\mu = (2^\mu)^{\log n} = l^{\log n}$ for $n = \max_{1\leq i\leq \mu-1} n_i$. We see that with probability $(1-p_{\text{bad}})^N$ only good challenges are chosen in the whole execution of the tree-finding algorithm and the assumption is true for all invocations of TREE-FINDER(i). Now, by the Bernoulli inequality,

$$\begin{aligned}(1-p_{\text{bad}})^N &\geq 1 - Np_{\text{bad}} > 1 - \frac{Np_2'}{p_2'+p_1'} \\ &= 1 - \frac{Np_2'}{p_2' + (1-1/\sqrt{\alpha}-n/p)(1-p_2')} \\ &> 1 - \frac{Np_2'}{1-1/\sqrt{\alpha}-n/p},\end{aligned}$$

which is bigger than $1/2$ if $p_2' \leq (1-1/\sqrt{\alpha}-n/p)/(2N)$, which in turn is implied by

$$T_i = \frac{3}{(\lambda-1)^2}\frac{\alpha^i}{\varepsilon}\ln\left(\frac{2N}{1-1/\sqrt{\alpha}-n/p}\right) = O\left(\frac{l^{\log\alpha+\log n}}{\varepsilon}\right) \quad (l\to\infty).$$

The expected number of iterations of the loop in lines $12-25$ under the condition that a good c_i is chosen is

$$\begin{aligned}\sum_{j=1}^{\infty} j\frac{p_0^{j-1}p_1}{p_1/(1-p_0)} &= (1-p_0)\sum_{j=1}^{\infty} jp_0^{j-1} \\ &= \frac{1}{1-p_0} = \frac{1}{p_1+p_2} \\ &< \frac{1}{p_1'} = \frac{1}{(1-1/\sqrt{\alpha}-n/p)(1-p_2')} \\ &= O(1)\end{aligned}$$

and each iteration takes time $T_i + 1$. So with probability at least $1/2$ the conditioned expected runtime of the whole tree finding algorithm is at most

$$\begin{aligned} t &= \sum_{i=1}^{\mu-1} n^i \frac{1}{p_1'}(T_i + 1) + \frac{n^{\mu-1} n_\mu}{\varepsilon/\alpha^{\mu-1} - n_\mu/p} \\ &< \frac{1}{p_1'} \sum_{i=1}^{\mu-1} n^i T_i + \frac{1}{p_1'} l^{\log n} + \frac{n_\mu}{n} \frac{l^{\log n + \log \alpha}}{\varepsilon} \\ &= \frac{1}{p_1'} \frac{3}{(\lambda-1)^2} \frac{1}{\alpha n - 1} \ln\left(\frac{2N}{1 - 1/\sqrt{\alpha} - n/p}\right) \frac{l^{\log n + \log \alpha}}{\varepsilon} \\ &\quad + \frac{n_\mu}{n} \frac{l^{\log n + \log \alpha}}{\varepsilon} + \frac{1}{p_1'} l^{\log n} \\ &= O\left(\frac{l^{\log n + \log \alpha} \log l}{\varepsilon}\right). \end{aligned}$$

Here we have used $\varepsilon \geq \alpha^\mu n_\mu / ((\alpha - 1)p)$ which implies $\varepsilon/\alpha^{\mu-1} - n_\mu/p \geq \varepsilon/\alpha^\mu = \varepsilon/l^{\log \alpha}$. When we are not so lucky and some bad challenges are chosen the algorithm might run for a long time but we just limit the runtime to $2t$. Then the probability for obtaining a full n-tree of accepting transcripts is at least $\frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{4}$ since the probability that an algorithm with expected runtime t runs longer than $2t$ is at most $1/2$. Notice that in expected time $8t$ we can obtain an n-tree of accepting transcripts. □

Example. The implied constant in the big-O statement for the runtime of the extractor is readily computed from the formulas in the proof of Lemma 3.2. For example in the case where $p \approx 2^{256}$, $n = 4$, $l = 2^{25}$ and $\alpha = 1.3$, one finds that $\lambda \approx 1.075$ and the implied constant is about 1564.

4 Zero-Knowledge Inner Product Proof

In an inner product proof there is a commitment $t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} u^\rho$ to two vectors whose inner product $x = \langle \vec{v}_1, \vec{v}_2 \rangle$ is publicly known. The goal is to prove knowledge of an opening to t that really fulfills this inner product relation. In this section we give a variant of the Bulletproof inner product proof which differs in that it is zero-knowledge. In the original protocol, after folding the vectors down to just 1-dimensional elements, the prover reveals the opening to the commitment. The main difference of the modified protocol from this section is that instead of revealing the opening it uses a Schnorr-type proof to prove knowledge of an opening in zero-knowledge, in a way that also proves the necessary product relation. With a zero-knowledge inner product proof at hand we can significantly simplify our main protocol compared to the similar Bulletproof range proof from [BBB+17]. For example, our proof is only three round compared to the five rounds of the range proof. The advantage stems from the fact that the secret vectors do not have to be blinded which is the reason for much of the complication in the Bulletproof range proof. We write $\triangleright\!\Pi_{\langle \cdot,\cdot \rangle}(\cdot;\cdot)$ for our inner product proof protocol, which is detailed in Figure 1.

The length l of the secret vectors $\vec{v}_1$, $\vec{v}_2$ is assumed to be a power of two. In the main protocol from Section 5 we need an inner product proof for vectors of arbitrary length but it is trivial to achieve this by just padding the vectors with zeros. If $t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} u^\rho$ is a commitment to two vectors

of length l which is not a power of two, we can just interpret this as a commitment to vectors of length $2^{\lceil \log l \rceil}$ over more generators $\vec{g}'$, $\vec{h}'$. Notice that the inner product of the padded vectors stays the same.

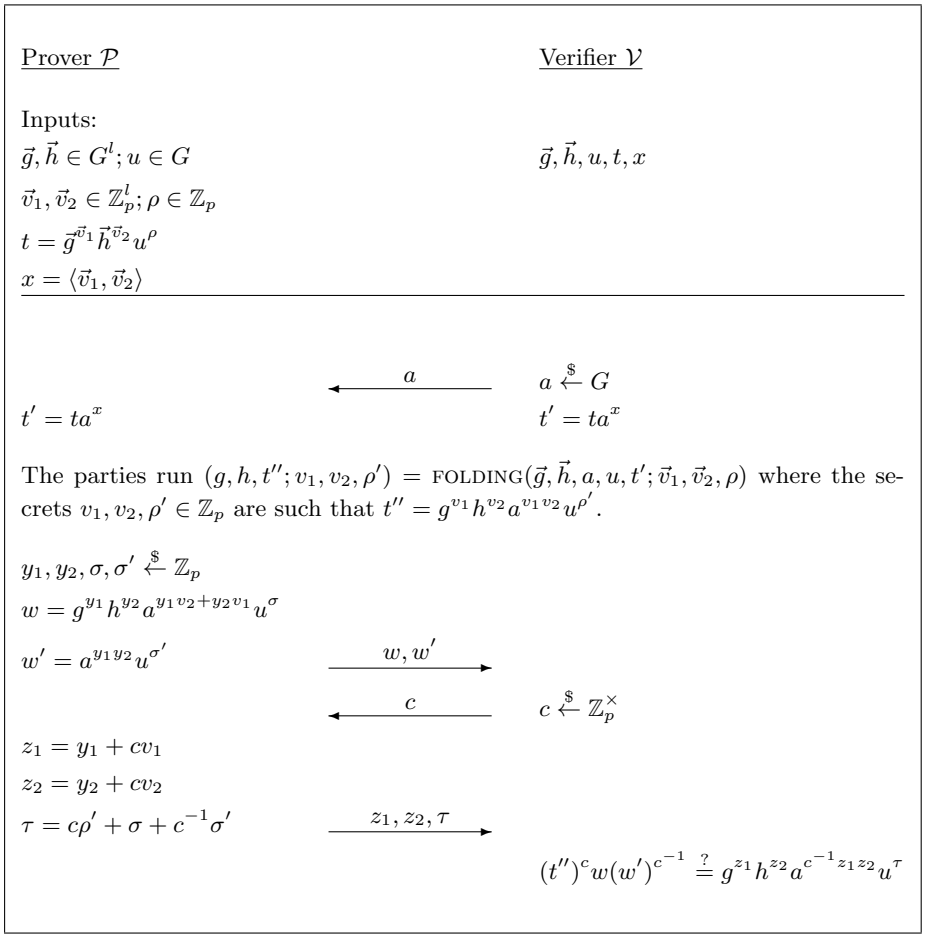

Fig. 1. Zero-knowledge inner product Bulletproof $\rhd\Pi_{\langle \cdot, \cdot \rangle}(\cdot;\cdot)$. It proves knowledge of an opening to a Pedersen commitment $t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} u^\rho$ such that the vectors $\vec{v}_1$ and $\vec{v}_2$ fulfill an inner product relation $\langle \vec{v}_1, \vec{v}_2 \rangle = x$.

Theorem 4.1. *The protocol given in Figures 1 and 2 is complete, perfectly honest verifier zero-knowledge and generalized special sound under the discrete-log assumption. So there is an extractor $\mathcal{E}$ that, when given rewindable black-box access to a deterministic prover $\mathcal{P}^*$, either outputs an opening $\vec{v}_1^*, \vec{v}_2^* \in \mathbb{Z}_p^l$, $\rho^* \in \mathbb{Z}_p$ of t, i.e. $t = \vec{g}^{\vec{v}_1^*} \vec{h}^{\vec{v}_2^*} u^{\rho^*}$, such that $x = \langle \vec{v}_1^*, \vec{v}_2^* \rangle$, or a non-trivial discrete-log relation between $\vec{g}, \vec{h}, u$ and two auxiliary generators $e, f \in G$. The extractor $\mathcal{E}$ runs in expected time at most $O(l^{2+\log \alpha} \log l / \varepsilon)$ for some $\alpha > 1$, for example $\alpha = 1.3$, when $\mathcal{P}^*$ has acceptance probability $\varepsilon \geq 10 \frac{\alpha}{\alpha - 1} l^{\log \alpha} / p$. Running $\mathcal{P}^*$ once is assumed to take unit time.*

Proof. The subprotocol without the first move is a $2\mu + 1$ move protocol for $\mu = \log(l) + 1$, which fulfills the prerequisites of the forking lemma given in Lemma 3.2. After sending a uniformly random generator $a = e^b$ of the group G for a uniform $b \in \mathbb{Z}_p$, the extractor $\mathcal{E}$ can thus use

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
Inputs:		
$\vec{g}, \vec{h} \in G^l; a, u \in G$		$\vec{g}, \vec{h}, a, u, t$
$\vec{v}_1, \vec{v}_2 \in \mathbb{Z}_p^l; \rho \in \mathbb{Z}_p$		
$t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} a^{\langle \vec{v}_1, \vec{v}_2 \rangle} u^{\rho}$		
Outputs:		
$g, h \in G$		g, h, t'
$v_1, v_2, \rho' \in \mathbb{Z}_p$		
$t' = g^{v_1} h^{v_2} a^{v_1 v_2} u^{\rho'}$		

If $l > 1$, define $l' = \frac{l}{2}$ and write $\vec{g} = \left(\frac{\vec{g}_t}{\vec{g}_b}\right)$, $\vec{h} = \left(\frac{\vec{h}_t}{\vec{h}_b}\right)$, $\vec{v}_i = \left(\frac{\vec{v}_{i,t}}{\vec{v}_{i,b}}\right)$, where $\vec{g}_j, \vec{h}_j, \vec{v}_{i,j} \in G^{l'}$ for $i = 1, 2$, $j = t, b$. Then,

$\sigma_{-1}, \sigma_1 \xleftarrow{\$} \mathbb{Z}_p$		
$t_{-1} = \vec{g}_t^{\vec{v}_{1,b}} \vec{h}_b^{\vec{v}_{2,t}} a^{\langle \vec{v}_{1,b}, \vec{v}_{2,t} \rangle} u^{\sigma_{-1}}$		
$t_1 = \vec{g}_b^{\vec{v}_{1,t}} \vec{h}_t^{\vec{v}_{2,b}} a^{\langle \vec{v}_{1,t}, \vec{v}_{2,b} \rangle} u^{\sigma_1}$	$\xrightarrow{t_{-1}, t_1}$	
	$\xleftarrow{c}$	$c \xleftarrow{\$} \mathbb{Z}_p^\times$
$\vec{v}_1' = \vec{v}_{1,t} + c^{-1} \vec{v}_{1,b}$		
$\vec{v}_2' = \vec{v}_{2,t} + c \vec{v}_{2,b}$		
$\rho'' = c^{-1} \sigma_{-1} + \rho + c \sigma_1$		

and both parties compute $\vec{g}' = \vec{g}_t \circ \vec{g}_b^c$, $\vec{h}' = \vec{h}_t \circ \vec{h}_b^{c^{-1}}$ and $t'' = t_{-1}^{c^{-1}} t t_1^c$. They recursively run $(g, h, t'; v_1, v_2, \rho') = \text{FOLDING}(\vec{g}', \vec{h}', a, u, t''; \vec{v}_1', \vec{v}_2', \rho'')$ where $\mathcal{P}$ knows $\vec{v}_1'$, $\vec{v}_2'$, ρ'' such that $t'' = (\vec{g}')^{\vec{v}_1'} (\vec{h}')^{\vec{v}_2'} a^{\langle \vec{v}_1', \vec{v}_2' \rangle} u^{\rho''}$.

Else $g = \vec{g}, h = \vec{h} \in G$, and $\mathcal{P}$ knows $v_1 = \vec{v}_1, v_2 = \vec{v}_2, \rho' = \rho \in \mathbb{Z}_p$, such that $t' = t = g^{v_1} h^{v_2} a^{v_1 v_2} u^{\rho'}$.

Fig. 2. Bulletproof folding protocol $\text{FOLDING}(\vec{g}, \vec{h}, a, u, t; \vec{v}_1, \vec{v}_2, \rho)$. This reduces a Pedersen multi-commitment of the form $t = \vec{g}^{\vec{v}_1} \vec{h}^{\vec{v}_2} a^{\langle \vec{v}_1, \vec{v}_2 \rangle} u^{\rho}$ to a new commitment $t' = g^{v_1} h^{v_2} a^{v_1 v_2} u^{\rho'}$ with the same (inner) product structure but in dimension 1. Furthermore, given an opening for t' having the correct inner product structure, one can extract an opening for t that also has the inner product structure by using the extractor from the forking lemma (Lemma 3.2).

TREE-FINDER to obtain a $(4,\ldots,4,5)$-tree of accepting transcripts of this subprotocol. More precisely, with probability at least $1/2$ over the choice of a, the verifier $\mathcal{V}$ will accept with probability at least $\varepsilon/2 \geq \frac{\alpha^\mu}{\alpha-1} n_\mu/p$. Therefore TREE-FINDER will be successful with probability at least $1/8$. If it is not successful, $\mathcal{E}$ restarts.

Consider the 5 accepting transcripts from neighboring leaves with the same parent node. Only the last challenges differ in the transcripts and we have the 5 verification equations

$$(t'')^{c_i} w (w')^{c_i^{-1}} = g^{z_{1,i}} h^{z_{2,i}} a^{c_i^{-1} z_{1,i} z_{2,i}} u^{\tau_i} \tag{13}$$

for $i = 1,\ldots,5$ with distinct $c_i \in \mathbb{Z}_p$. Let $(\lambda_1,\lambda_2,\lambda_3)^T \in \mathbb{Z}_p^3$ be the solution of the linear system

$$\begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^{-1} & c_2^{-1} & c_3^{-1} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

It exists because it is well-known that the determinant of this Vandermonde matrix is equal to $-(c_1c_2c_3)^{-1}(c_1-c_2)(c_1-c_3)(c_2-c_3) \neq 0$. Now raise the first 3 equations in 13 for $i = 1,2,3$ to the powers of λ_i and multiply them. This gives

$$t'' = g^{v_1^*} h^{v_2^*} a^{x^*} u^{\tau^*}$$

where for example $v_1^* = \sum_{i=1}^3 \lambda_i z_{1,i}$. In the same manner we can extract openings for w and w',

$$\begin{aligned} w &= g^{y_1^*} h^{y_2^*} a^{x_w^*} u^{\sigma^*}, \\ w' &= g^{(y_1')^*} h^{(y_2')^*} a^{x_{w'}^*} u^{(\sigma')^*}. \end{aligned}$$

With these openings to t'', w and w' we can reconstruct the equations in (13) and get

$$\begin{aligned} &(t'')^{c_i} w (w')^{c_i^{-1}} \\ &= g^{c_i v_1^* + y_1^* + c_i^{-1}(y_1')^*} h^{c_i v_2^* + y_2^* + c_i^{-1}(y_2')^*} a^{c_i x^* + x_w^* + c_i^{-1} x_{w'}^*} u^{c_i \rho^* + \sigma^* + c_i^{-1}(\sigma')^*} \\ &= g^{z_{1,i}} h^{z_{2,i}} a^{c_i^{-1} z_{1,i} z_{2,i}} u^{\tau_i} \end{aligned}$$

By comparing exponents we either find a non-trivial discrete-log relation between g,h,a,u, which gives a relation between $\vec{g},\vec{h},u,e$ since $\mathcal{E}$ knows expressions of g,h,a,u as powers of $\vec{g},\vec{h},u,e$. Or we have

$$\begin{aligned} &c_i x^* + x_w^* + c_i^{-1} x_{w'}^* = c_i^{-1} z_{1,i} z_{2,i} \\ &= c_i^{-1} \left(c_i v_1^* + y_1^* + c_i^{-1}(y_1')^*\right)\left(c_i v_2^* + y_2^* + c_i^{-1}(y_2')^*\right). \end{aligned}$$

Multiplying this equation by c_i^3 yields a polynomial of degree 4 which has five roots c_i. Hence it must be the zero polynomial and from the leading coefficient we get $x^* = v_1^* v_2^*$ and thus

$$t'' = g^{v_1^*} h^{v_2^*} a^{v_1^* v_2^*} u^{\tau^*}.$$

The extractor performs this process for all parents in the second-to-last level $\mu - 1 = \log(l)$ of the tree of accepting transcripts. Then, with the same techniques and as is detailed in [BBB+17],

the extractor can invert all the $\log(l)$ folding steps and either compute a non-trivial discrete-log relation or an opening $\vec{v}_1, \vec{v}_2, x^*, \rho^*$ of $t' = ta^x$,

$$ta^x = \vec{g}^{\vec{v}_1^*} \vec{h}^{\vec{v}_2^*} a^{x^*} u^{\rho^*},$$

such that $x^* = \langle \vec{v}_1, \vec{v}_2 \rangle$. If $x^* = x$ then $\mathcal{E}$ has an opening of t as stated in the theorem. If not, $\mathcal{E}$ starts over from scratch but samples a challenge generator $a' = f^{b'} \in \mathbb{Z}_p$ for the first move. By this $\mathcal{E}$ obtains an opening

$$t(a')^x = \vec{g}^{\vec{v}_1^{**}} \vec{h}^{\vec{v}_2^{**}} (a')^{x^{**}} u^{\rho^{**}},$$

and can compute

$$\vec{g}^{\vec{v}_1^* - \vec{v}_1^{**}} \vec{h}^{\vec{v}_2^* - \vec{v}_2^{**}} e^{b(x^* - x)} f^{b'(x^{**} - x)} u^{\rho^* - \rho^{**}} = 1,$$

which is a non-trivial discrete-log relation. Not taking into account the simple arithmetic over $\mathbb{Z}_p$, the expected running time of $\mathcal{E}$ is at most 16 times the expected running time of TREE-FINDER.

We turn to the zero-knowledge property. The first message by the verifier containing the generator a and all the messages in the folding protocol are independently uniformly random. This is because all the cross-terms t_{-1}, t_1 are independently blinded with independently random factors $u^{\sigma_{-1}}$ and u^{σ_1}. So the simulator can just choose $a \xleftarrow{\$} G$ and all messages in the folding protocol uniformly randomly. From these messages the honest verifier computes the generators g, h and the commitment t''. Now it remains to simulate the Schnorr-type protocol at the end for proving knowledge of an opening of t'' that obeys the product relation. This is made possible by how we set up the verification equation. The simulator first samples $c \xleftarrow{\$} \mathbb{Z}_p$, and then $z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$, which are independent from the previously chosen messages because of y_1 and y_2, respectively. Then he chooses $w' \xleftarrow{\$} G$ which is independent because of the blinding factor $u^{\sigma'}$. Last the simulator samples $\tau \xleftarrow{\$} \mathbb{Z}_p$ which is still uniformly random because of σ. Now $w \in G$ is not independent anymore but instead fully determined by the previous choices and the simulator can compute it correctly as

$$w = (t'')^{-c} (w')^{-c^{-1}} g^{z_1} h^{z_2} a^{c^{-1} z_1 z_2} u^{\tau},$$

which clearly makes the verification equation true.

5 The Main Protocol

In this section we present in detail our protocol to prove knowledge of a matrix $\mathbf{S} \in \mathcal{R}_q^{m \times k}$ consisting of short polynomials of infinity norm less than B such that

$$\mathbf{AS} = \mathbf{T} \text{ over } \mathcal{R}_q \tag{14}$$

where $\mathbf{A} \in \mathcal{R}_q^{n \times m}$ and $\mathbf{T} \in \mathcal{R}_q^{n \times k}$ are public.

First, when $\mathbf{A}$, $\mathbf{S}$, $\mathbf{T}$ are lifted to matrices over $\mathbb{Z}[X]$, the equation is true modulo q and $\mathbf{f}$. So there are matrices $\mathbf{R}_1, \mathbf{R}_2$ over $\mathbb{Z}[X]$ such that

$$\mathbf{AS} + q\mathbf{R}_1 + \mathbf{f}\mathbf{R}_2 = \mathbf{T} \text{ over } \mathbb{Z}[X]. \tag{15}$$

More precisely, notice that $\mathbf{T} - \mathbf{AS} \in (\mathbb{Z}[X])^{n \times k}$ consists of polynomials of degree at most $2(d-1)$ and infinity norm less than $mdBq/2$ when we use central representatives for coefficients in $\mathbb{Z}_q$. Moreover, $\mathbf{T} - \mathbf{AS}$ is a multiple of $\mathbf{f}$ modulo q. So we can exactly divide $\mathbf{T} - \mathbf{AS}$ by $\mathbf{f}$ over $\mathbb{Z}_q[X]$ to

obtain $\mathbf{R}_2$ with polynomials of degree at most $d-2$ and coefficients in $\{-(q-1)/2, \ldots, (q-1)/2\}$. Then, dividing $\mathbf{T} - \mathbf{AS} - \mathbf{fR}_2$ by q yields $\mathbf{R}_1$ with polynomials of degree at most $2(d-1)$ and infinity norm less than $(mdB + d\|\mathbf{f}\|_\infty)/2$. Next, for a prime p we have

$$\mathbf{AS} + q\mathbf{R}_1 + \mathbf{fR}_2 = \mathbf{T} \text{ over } \mathbb{Z}_p[X], \tag{16}$$

and then for an $\alpha \in \mathbb{Z}_p$ the equation

$$\mathbf{A}(\alpha)\mathbf{S}(\alpha) + q\mathbf{R}_1(\alpha) + \mathbf{f}(\alpha)\mathbf{R}_2(\alpha) = \mathbf{T}(\alpha) \text{ over } \mathbb{Z}_p[X]. \tag{17}$$

Conversely, by the Schwartz-Zippel lemma, if Equation (17) is true for a uniformly random α, then Equation (16) holds with probability at least $1 - 2(d-1)/p$. In this case, if $p \geq 2(mdB + d\|\mathbf{f}\|_\infty)q$, Equation (15) is true since no reduction modulo p takes place, and Equation (14) follows. So in order to prove knowledge of a matrix $\mathbf{S} \in \mathcal{S}_B^{m\times k}$ as in Equation (14), it suffices to prove knowledge of matrices $\mathbf{S}$, $\mathbf{R}_1$ and $\mathbf{R}_2$ of integer polynomials whose coefficients have absolute value less than B, $B_1 = (mdB + d\|\mathbf{f}\|_\infty)/2$ and $B_2 = q/2$, respectively, such that Equation (17) is true for a uniformly random α.

We describe our strategy for conducting such a proof. If we expand all polynomials in the secret matrices $\mathbf{S}$, $\mathbf{R}_1$, $\mathbf{R}_2$ to their coefficient row vectors of dimensions d, $2d-1$ and $d-1$, respectively, and hence consider the matrices as integer matrices S, R_1, R_2, then, with $\vec{\alpha}_d = (1, \alpha, \ldots, \alpha^{d-1})^T$, we can equivalently write

$$\mathbf{A}(\alpha)S(I_k \otimes \vec{\alpha}_d) + qR_1(I_k \otimes \vec{\alpha}_{2d-1}) + \mathbf{f}(\alpha)R_2(I_k \otimes \vec{\alpha}_{d-1}) = \mathbf{T}(\alpha). \tag{18}$$

Now a natural strategy would be to produce a Pedersen multi-commitment over a group of order p to the secret matrices S, R_1, R_2. Then one could prove that the matrices fulfill Equation (18) by reducing them to integers using in the order of $\log(mkd)$ bulletproof folding steps. In addition one would also need to give a range proof that the coefficients of the matrices are sufficiently small. For increased efficiency we combine these proofs in one single proof.

The usual method for range proofs consists of expressing the coefficients by their binary representations so that the range follows from the number of bits used per coefficient. The proof that this representation really only contains bits in $\{0, 1\}$ is most easily done via an inner product proof as in [BBB+17]. Therefore we want to reduce Equation (18) to an inner product equation which then can be integrated into the range proof. To this end we first multiply from both sides by uniformly random vectors $\vec{\beta} \in \mathbb{Z}_p^k$ and $\vec{\gamma} \in \mathbb{Z}_p^n$, so that

$$\vec{\gamma}^T\mathbf{A}(\alpha)S(\vec{\beta} \otimes \vec{\alpha}_d) + q\vec{\gamma}^T R_1(\vec{\beta} \otimes \vec{\alpha}_{2d-1}) + \mathbf{f}(\alpha)\vec{\gamma}^T R_2(\vec{\beta} \otimes \vec{\alpha}_{d-1}) = \vec{\gamma}^T\mathbf{T}(\alpha)\vec{\beta}.$$

This equation implies Equation (18) with probability at least $1 - 2/p$. Next we serialize the secret matrices to column vectors $\vec{s} \in \mathbb{Z}^{mkd}$, $\vec{r}_1 \in \mathbb{Z}^{nk(2d-1)}$ and $\vec{r}_2 \in \mathbb{Z}^{nk(d-1)}$ in row-major order. With these the last equation is equivalent to the inner product equation

$$\begin{aligned}&\left\langle \mathbf{A}(\alpha)^T\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d, \vec{s} \right\rangle + \left\langle q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1}, \vec{r}_1 \right\rangle + \left\langle \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1}, \vec{r}_2 \right\rangle \\ &= \vec{\gamma}^T\mathbf{T}(\alpha)\vec{\beta}.\end{aligned}$$

Finally, we expand each secret vector one more time and replace the coefficients by their binary representation using two's complement for negative numbers. We get

$$\begin{aligned}
&\left\langle \mathbf{A}(\alpha)^T \vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b, \text{Binary}_b(\vec{s}) \right\rangle \\
+&\left\langle q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}, \text{Binary}_{b_1}(\vec{r}_1) \right\rangle \\
+&\left\langle \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2}, \text{Binary}_{b_2}(\vec{r}_2) \right\rangle \\
=&\vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta}, \qquad (19)
\end{aligned}$$

where $\vec{2}_b = (1, 2, \ldots, 2^{b-2}, -2^{b-1})^T$, $b = \lceil \log(B) \rceil + 1$, $b_1 = \lceil \log(B_1) \rceil + 1 = \lceil \log(mdB + d \left\| \mathbf{f} \right\|_\infty) \rceil$ and $b_2 = \lceil \log(B_2) \rceil + 1 = \lceil \log(q) \rceil$. For the sake of clarity in what follows we concatenate the public and secret vectors and define

$$\begin{aligned}
\vec{v} &= \mathbf{A}(\alpha)^T \vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b \parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1} \parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2}, \\
\vec{s}_1 &= \text{Binary}_b(\vec{s}) \parallel \text{Binary}_{b_1}(\vec{r}_1) \parallel \text{Binary}_{b_2}(\vec{r}_2)
\end{aligned}$$

so that we can write $\langle \vec{v}, \vec{s}_1 \rangle = \vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta}$.

It remains to prove that the secret vector $\vec{s}_1$ only contains coefficients in $\{0, 1\}$. As usual this is done by proving that there is a second vector $\vec{s}_2$, the vector with all bits flipped, such that $\vec{s}_1 \circ \vec{s}_2 = \vec{0}$ and $\vec{s}_1 + \vec{s}_2 = \vec{1}$. The first property holds with probability at least $1 - 1/p$ if $\langle \vec{\varphi}, \vec{s}_1 \circ \vec{s}_2 \rangle = \langle \vec{\varphi} \circ \vec{s}_2, \vec{s}_1 \rangle = 0$ for a uniformly random vector $\vec{\varphi}$. Similarly, the second property follows with overwhelming probability from $\langle \vec{\varphi}, \vec{s}_1 + \vec{s}_2 \rangle = \langle \vec{\varphi}, \vec{s}_1 \rangle + \langle \vec{\varphi} \circ \vec{s}_2, \vec{1} \rangle = \langle \vec{\varphi}, \vec{1} \rangle$. We incorporate both inner product equations into Equation (19) and arrive at

$$\left\langle \vec{v} + \vec{\varphi} \circ \vec{s}_2 + \psi\vec{\varphi}, \vec{s}_1 + \psi\vec{1} \right\rangle = \vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta} + \psi \left\langle \vec{v}, \vec{1} \right\rangle + (\psi + \psi^2) \left\langle \vec{\varphi}, \vec{1} \right\rangle$$

where $\psi \in \mathbb{Z}_p$ is another uniformly random field element with the purpose of separating the three inner product equations.

When given a Pedersen multi-commitment to the vectors $\vec{s}_2$ and $\vec{s}_1$ it is easy to compute a commitment to $\vec{v}_1 = \vec{v} + \vec{\varphi} \circ \vec{s}_2 + \psi\vec{\varphi}$ and $\vec{v}_2 = \vec{s}_1 + \psi\vec{1}$. It might be unclear at first how to multiply $\vec{s}_2$ componentwise with $\vec{\varphi}$ inside the multi-commitment, which means each coefficient has to be multiplied by a different value. There is a standard trick to do this. Suppose $\vec{g} \in G^l$ is the vector of generators underlying $\vec{s}_2$. Then we just reinterpret this part of the commitment as a commitment over generators $\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$. Since $\vec{g}^{\vec{s}_2} = (\vec{g}^{\vec{\varphi}^{-1}})^{\vec{\varphi} \circ \vec{s}_2} = (\vec{g}')^{\vec{s}_2}$, our original commitment containing $\vec{s}_2$ over $\vec{g}$ thus becomes a commitment containing $\vec{\varphi} \circ \vec{s}_2$ over $\vec{g}'$. Now given the commitment to $\vec{v}_1$ and $\vec{v}_2$ we prove that the inner product of these vectors of dimension $l = mkdb + nk(2d-1)b_1 + nk(d-1)b_2$ is equal to $x = \vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta} + \psi\langle \vec{v}, \vec{1} \rangle + (\psi + \psi^2)\langle \vec{\varphi}, \vec{1} \rangle$. It follows with overwhelming probability that $\vec{s}_1$ gives rise to a matrix $\mathbf{S} \in \mathcal{R}_q^{m \times k}$ of short polynomials such that $\mathbf{AS} = \mathbf{T}$ over $\mathcal{R}_q$. For the inner product proof we make use of Bulletproofs, which have communication cost logarithmic in l. But in contrast to the range proof in [BBB+17], we do not blind the vectors and instead use a variant of the Bulletproof inner product proof that is zero knowledge. Here one first reduces the vectors to dimension 1 and then uses a zero-knowledge Schnorr-type proof for the one-dimensional base case. See Figure 3 for the complete protocol and Theorem 5.1 for its security. We state the zero-knowledge inner product Bulletproof in Figure 1.

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
Inputs:		
$\mathbf{A} \in \mathcal{R}_q^{n\times m}, \mathbf{S} \in \mathcal{S}_B^{m\times k}$		$\mathbf{A}, \mathbf{T}, b, b_1, b_2, l, \vec{g}, \vec{h}, u$
$\mathbf{T} = \mathbf{AS} \in \mathcal{R}_q^{n\times k}$		
$b = \lceil \log(B) \rceil + 1$		
$b_1 = \lceil \log(mdB + d\left\|\mathbf{f}\right\|_\infty) \rceil$		
$b_2 = \lceil \log(q) \rceil$		
$l = mkdb + nk(2d-1)b_1$		
$\quad + nk(d-1)b_2$		
$\vec{g}, \vec{h} \in G^l, u \in G$		
$\mathbf{R}_2 = (\mathbf{T} - \mathbf{AS})/\mathbf{f}$ over $\mathbb{Z}_q[X]$		
$\mathbf{R}_1 = (\mathbf{T} - \mathbf{AS} - \mathbf{fR}_2)/q$ over $\mathbb{Z}[X]$		
$\vec{s} = \text{Serialize}(\mathbf{S}) \in \mathbb{Z}^{mkd}$		
$\vec{r}_1 = \text{Serialize}(\mathbf{R}_1) \in \mathbb{Z}^{nk(2d-1)}$		
$\vec{r}_2 = \text{Serialize}(\mathbf{R}_2) \in \mathbb{Z}^{nk(d-1)}$		
$\vec{s}_1 = \text{Binary}_b(\vec{s}) \parallel \text{Binary}_{b_1}(\vec{r}_1)$		
$\quad \parallel \text{Binary}_{b_2}(\vec{r}_2)$		
$\vec{s}_2 = \vec{s}_1 + \vec{1} \in \mathbb{Z}_2^l$ (XOR)		
$\rho \xleftarrow{\$} \mathbb{Z}_p$		
$w = \vec{g}^{\vec{s}_2}\vec{h}^{\vec{s}_1}u^\rho$	$\xrightarrow{w}$	
		$\alpha \xleftarrow{\$} \mathbb{Z}_p^\times, \vec{\beta} \xleftarrow{\$} (\mathbb{Z}_p^\times)^k, \vec{\gamma} \xleftarrow{\$} (\mathbb{Z}_p^\times)^n$
	$\xleftarrow{\alpha, \vec{\beta}, \vec{\gamma}, \vec{\varphi}, \psi}$	$\vec{\varphi} \xleftarrow{\$} (\mathbb{Z}_p^\times)^l, \psi \xleftarrow{\$} \mathbb{Z}_p^\times$
$\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$		$\vec{g}' = \vec{g}^{\vec{\varphi}^{-1}}$
$\vec{v} = \mathbf{A}(\alpha)^T\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b$		$\vec{v} = \mathbf{A}(\alpha)^T\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_d \otimes \vec{2}_b$
$\quad \parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}$		$\quad \parallel q\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{2d-1} \otimes \vec{2}_{b_1}$
$\quad \parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2} \in \mathbb{Z}_p^l$		$\quad \parallel \mathbf{f}(\alpha)\vec{\gamma} \otimes \vec{\beta} \otimes \vec{\alpha}_{d-1} \otimes \vec{2}_{b_2}$
$t = w(\vec{g}')^{\vec{v}+\psi\vec{\varphi}}\vec{h}^\psi$		$t = w(\vec{g}')^{\vec{v}+\psi\vec{\varphi}}\vec{h}^\psi$
$\vec{v}_1 = \vec{v} + \vec{\varphi} \circ \vec{s}_2 + \psi\vec{\varphi}$		
$\vec{v}_2 = \vec{s}_1 + \psi\vec{1}$		
$x = \langle \vec{v}_1, \vec{v}_2 \rangle$		$x = \vec{\gamma}^T\mathbf{T}(\alpha)\vec{\beta} + \psi \left\langle \vec{v}, \vec{1} \right\rangle$
		$\quad + (\psi + \psi^2)\langle \vec{\varphi}, \vec{1} \rangle \in \mathbb{Z}_p$

The parties run the zero-knowledge inner product proof $\rhd\Pi_{\langle\cdot,\cdot\rangle}(\vec{g}', \vec{h}, u, t, x; \vec{v}_1, \vec{v}_2, \rho)$ and the verifier $\mathcal{V}$ accepts if he accepts in $\rhd\Pi_{\langle\cdot,\cdot\rangle}(\cdot;\cdot)$.

Fig. 3. Discrete-log based zero-knowledge proof of knowledge of a short solution to a matrix equation over $\mathcal{R}_q$.

Theorem 5.1. *If $p \geq 2(mdB + d\left\|\mathbf{f}\right\|_\infty)q$, then the protocol in Figure 3 is complete, perfectly honest verifier zero-knowledge and generalized special sound under the discrete-log assumption in the sense that there is an extractor $\mathcal{E}$ with the following properties. When given rewindable black-box access to a deterministic prover $\mathcal{P}^*$ that convinces the honest verifier with probability $\varepsilon \geq 100l/p$, $\mathcal{E}$ either outputs a solution $\mathbf{S}^* \in \mathcal{R}_q^{m \times k}$ to $\mathbf{AS}^* = \mathbf{T}$, which consists of polynomials whose coefficients fit in $b = \lceil \log(B) \rceil + 1$ bits, or a non-trivial discrete-log relation between generators of the group G. The extractor $\mathcal{E}$ runs in expected time at most $O(l^{2.4} \log l/\varepsilon)$. Running $\mathcal{P}^*$ once is assumed to take unit time.*

Proof. Completeness is clear from the discussion at the beginning of Section 5 and the zero-knowledge property follows immediately from the fact that the inner product proof is honest verifier zero-knowledge; see Theorem 4.1. Let us now prove soundness. The extractor $\mathcal{E}$ runs $\mathcal{P}^*$, sends uniformly random challenges in the second move and then uses the extractor for the inner product proof assuming acceptance probability $\varepsilon/2$ to get an opening for t, c.f. Theorem 4.1. From an averaging argument we know that for at least half of the challenges in the second move the inner product proof π is valid with probability at least $\varepsilon/2$. Then, since $\varepsilon/2 > 10\alpha l^{\log \alpha}/((\alpha - 1)p)$ for $\alpha \geq 1.3$, the conditions of Theorem 4.1 are met. So after an expected number of 2 trials we can assume that $\mathcal{E}$ either has a non-trivial discrete-log relation or an opening $\vec{v}_1^*, \vec{v}_2^*, \rho^*$ of t, i.e.

$$t = (\vec{g}')^{\vec{v}_1^*} \vec{h}^{\vec{v}_2^*} u^{\rho^*},$$

such that $\langle \vec{v}_1^*, \vec{v}_2^* \rangle = x$. Since $t = w(\vec{g}')^{\vec{v}+\psi\vec{\varphi}}\vec{h}^{\psi}$, we get the opening $\vec{\varphi} \circ \vec{s}_2^* = \vec{v}_1^* - \vec{v} - \psi\vec{\varphi}$, $\vec{s}_1^* = \vec{v}_2^* - \psi\vec{1}$, ρ^* for w such that

$$\begin{aligned}
&\langle \vec{v}, \vec{s}_1^* \rangle + \left\langle \vec{v}, \psi\vec{1} \right\rangle + \langle \vec{\varphi} \circ \vec{s}_2^*, \vec{s}_1^* \rangle + \left\langle \vec{\varphi} \circ \vec{s}_2^*, \psi\vec{1} \right\rangle + \langle \psi\vec{\varphi}, \vec{s}_1^* \rangle + \left\langle \psi\vec{\varphi}, \psi\vec{1} \right\rangle \\
&= \langle \vec{v}, \vec{s}_1^* \rangle + \langle \vec{\varphi}, \vec{s}_1^* \circ \vec{s}_2^* \rangle + \psi \langle \vec{\varphi}, \vec{s}_1^* + \vec{s}_2^* \rangle + \psi^2 \left\langle \vec{\varphi}, \vec{1} \right\rangle + \psi \left\langle \vec{v}, \vec{1} \right\rangle \\
&= \vec{\gamma}^T \mathbf{T}(\alpha) \vec{\beta} + \psi \left\langle \vec{v}, \vec{1} \right\rangle + (\psi + \psi^2) \left\langle \vec{\varphi}, \vec{1} \right\rangle.
\end{aligned}$$

The last equation is equivalent to

$$\langle \vec{v}, \vec{s}_1^* \rangle + \langle \vec{\varphi}, \vec{s}_1^* \circ \vec{s}_2^* \rangle + \psi \left\langle \vec{\varphi}, \vec{s}_1^* + \vec{s}_2^* - \vec{1} \right\rangle = \vec{\gamma}^T \mathbf{T}(\alpha) \vec{\beta},$$

which can be interpreted as a multivariate polynomial P over $\mathbb{Z}_p$ in $n + k + l + 2$ variables that evaluates to zero at $(\alpha, \vec{\beta}, \vec{\gamma}, \vec{\varphi}, \psi)$. If the polynomial is the zero polynomial it follows that

$$\vec{s}_1^* \circ \vec{s}_2^* = 0 \text{ and } \vec{s}_1^* + \vec{s}_2^* = \vec{1}$$

so $\vec{s}_1^*$ is a binary vector with entries $s_{1,i}^* \in \{0, 1\}$. Write $\mathbf{S}^* \in (\mathbb{Z}[X])^{m \times k}$ for the polynomial matrix in which the coefficient of X^ν, $0 \leq \nu \leq d-1$, of the polynomial in the (i, j)-th entry, $0 \leq i \leq m-1$, $0 \leq j \leq k-1$, is given by

$$\begin{aligned}
&s_{1,bdki+bdj+b\nu}^* + s_{1,bdki+bdj+b\nu+1}^* 2 + \cdots + s_{1,bdki+bdj+b\nu+(b-2)}^* 2^{b-2} \\
&- s_{1,bdki+bdj+b\nu+(b-1)}^* 2^{b-1}.
\end{aligned}$$

Proceed similarly for $\mathbf{R}_1^*, \mathbf{R}_2^* \in (\mathbb{Z}[X])^{n\times k}$ starting from coefficient $s^*_{1,bdkm}$ and $s^*_{1,bdkm+b_1(2d-1)kn}$ of $\vec{s}_1^*$, respectively. In other words, $\mathbf{S}^*, \mathbf{R}_1^*$ and $\mathbf{R}_2^*$ are such that

$$\begin{aligned}&\text{Binary}_b(\text{Serialize}(\mathbf{S}^*)) \parallel \text{Binary}_{b_1}(\text{Serialize}(\mathbf{R}_1^*)) \parallel \text{Binary}_{b_2}(\text{Serialize}(\mathbf{R}_2^*))\\ &= \vec{s}_1^*.\end{aligned}$$

By construction the polynomials in $\mathbf{S}^*$, $\mathbf{R}_1^*$ and $\mathbf{R}_2^*$ have coefficients that fit in b, b_1 and b_2 bits, respectively. Then, since $\langle \vec{v}, \vec{s}_1^* \rangle = \vec{\gamma}^T \mathbf{T}(\alpha)\vec{\beta}$, it follows by inspection

$$\vec{\gamma}^T \Big(\mathbf{A}\mathbf{S}^* + q\mathbf{R}_1^* + \mathbf{f}\mathbf{R}_2^* - \mathbf{T} \Big)(\alpha)\vec{\beta} = 0 \text{ in } \mathbb{Z}_p.$$

The coefficient of X^ν of the polynomial in the (i,j)-th entry of the matrix in the middle corresponds to the coefficient of $\alpha^\nu \beta_j \gamma_i$ of our multivariate polynomial P that we assume to be zero. So,

$$\mathbf{A}\mathbf{S}^* + q\mathbf{R}_1^* + \mathbf{f}\mathbf{R}_2^* = \mathbf{T} \text{ over } \mathbb{Z}_p[X]$$

but from our assumption on p this equation is even true over $\mathbb{Z}[X]$ and we finally get $\mathbf{A}\mathbf{S}^* = \mathbf{T}$ over $\mathcal{R}_q$.

It remains to consider the case where $P \neq 0$. Note that in this case the polynomial is of total degree at most $2d$. Consequently, it can evaluate to zero at no more than $2dp^{n+k+l+1}$ points in $\mathbb{Z}_p^{n+k+l+2}$ (this is just a counting version of the Schwartz-Zippel lemma). Now the extractor $\mathcal{E}$ reruns $\mathcal{P}^*$ but sends a uniform challenge $(\alpha, \vec{\beta}, \vec{\gamma}, \vec{\varphi}, \psi) \in \mathbb{Z}_p^{n+k+l+2}$ from the set of non-roots of P. Then $\mathcal{E}$ again tries to extract from the inner product proof and continues in this fashion until he is successful for a second time. At least for a fraction of $\frac{1}{2} - \frac{2d}{p}$ of the non-roots, the inner product proof is accepted with probability at least $\varepsilon/2$. So after an expected number of roughly 2 trials $\mathcal{E}$ will get a non-trivial discrete-log relation or new multivariate polynomial P' that is zero outside of the small set of roots of our original polynomial P so that P' must be different to P. But then, since P and P' are in one-to-one correspondence to openings of the commitment t, we must have two different openings and can compute a non-trivial discrete-log relation. We see the total expected runtime of $\mathcal{E}$ is at most 4 times the expected runtime of the extractor of the inner product proof. □

5.1 Proof size

The communication size of our protocol from Figure 3 is very small. Instead of all the individual challenges in the second move the verifier can just send a short seed that is expanded to the challenges with the help of a XOF. Moreover, in the non-interactive version of the protocol via the Fiat-Shamir transform the challenges are expanded from public information and the first message. So such a non-interactive proof only consists of the first message and the inner product proof of size logarithmic in l. Simple counting shows that one full non-interactive proof consists of $2\lceil \log l \rceil + 3$ group elements and 3 elements of $\mathbb{Z}_p$. If a 256 bit elliptic curve is used for G, then this results in $64\lceil \log l \rceil + 192$ bytes per proof.

5.2 Number of exponentiations

Computing multi-exponentiations over G is by far the most time-consuming operation in our main protocol. We count the number of exponentiations to be performed by the prover and verifier in

order to estimate the time needed to execute the protocol. The prover computes l exponentiations for $\vec{g}'$, $l+1$ exponentiations for t and only 1 exponentiation for w ($\vec{s}_1$ and $\vec{s}_2$ are binary) plus the exponentiations in the inner product proof. The verifier computes $2l+1$ exponentiations and those from the inner product proof. In the inner product proof the prover has to compute $2 \cdot 2^{\lceil \log l \rceil - i} + 6$ exponentiations in the i-th folding level, $i = 0, \ldots, \lceil \log l \rceil - 1$. This amounts to $4 \cdot 2^{\lceil \log l \rceil} + 6\lceil \log l \rceil - 4 < 8l + 6 \log l + 2$ exponentiations for the full Bulletproof folding. In addition there are 6 exponentiations needed for the Schnorr-type proof. The verifier performs $4\lceil \log l \rceil < 4 \log l + 1$ exponentiations for the folding protocol and 6 exponentiations for the verification equation. This can be heavily optimized by delaying exponentiations; see [BBB+17, Section 6.2]. We conclude that the total exponentiation costs for the prover and verifier are less than $10l + 6 \log l + 10$ and $2l + 4 \log l + 10$ exponentiations.

5.3 Example

We return to the example of a verifiable encryption scheme from Section 1.5. In the case of verifiable encryption, one has to prove a matrix equation $\mathbf{A}\vec{\mathbf{s}} = \vec{\mathbf{t}}$ with parameters $n = 2$, $m = 4$, $k = 1$, $B = 4$. For the ring $\mathcal{R}_q$, a common example for encrypting messages that are binary polynomials (c.f. [ADPS16]) is setting $\mathbf{f} = X^{1024} + 1$ and q being a prime of about 13 bits, and $p = 2$. With these parameters we find the length l of the secret vectors $\vec{s}_1$ and $\vec{s}_2$ in the inner product proof to be equal to 100296. It then follows from above that the prover and verifier need to compute about 724986 and 200667 exponentiations to run our protocol for this application. With current CPUs one exponentiation on a 256 bit elliptic curve can be computed in about 35000 cycles (see `https://bench.cr.yp.to/results-dh.html`), which amounts to roughly 85000 exponentiations per second. So computing one of our proofs should be possible in less than 10 seconds. This can then be improved by using specialized algorithms for computing multi-exponentiations, in particular Pippenger's algorithm [Pip80]. The size of the proof is 1.25 kilobytes.

Acknowledgements

We thank the reviewers for their careful reading of the proofs and useful suggestions. This work was supported in part by the SNSF ERC Transfer Grant CRETP2-166734 FELICITY and H2020 FutureTPM.

References

ABB+18. Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: A distributed operating system for permissioned blockchains. *CoRR*, abs/1801.10228, 2018.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *USENIX*, pages 327–343, 2016.

BBB+17. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. *IACR Cryptology ePrint Archive*, 2017:1066, 2017.

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 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 327–357, 2016.

BCK+14. Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT*, pages 551–572, 2014.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

CS03. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, pages 126–144, 2003.

FGP14. Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 844–855, 2014.

FMMC12. Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012.

Gid18. Craig Gidney. Why will quantum computers be slow? `http://algassert.com/post/1800`. Last accessed January 18, 2019., 2018.

LLNW18. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *CRYPTO*, pages 700–732, 2018.

LM06. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155, 2006.

LN17. Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT*, pages 293–323, 2017.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. Preliminary version appeared in EUROCRYPT 2010.

LTV12. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

LWF+17. Bjoern Lekitsch, Sebastian Weidt, Austin G. Fowler, Klaus Mølmer, Simon J. Devitt, Christof Wunderlich, and Winfried K. Hensinger. Blueprint for a microwave trapped ion quantum computer. *Science Advances*, 3(2), 2017.

MW16. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 735–763, 2016.

Pip80. Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM J. Comput.*, 9(2):230–250, 1980.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166, 2006.

PS16. Chris Peikert and Sina Shiehian. Multi-key FHE from lwe, revisited. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 217–238, 2016.

WTS+18. Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 926–943, 2018.