Zero-Knowledge Proofs from Learning Parity with Noise: Optimization, Verification, and Application

Thomas Haines
Australian National University
Australia
thomas.haines@anu.edu.au

Rafieh Mosaheb
SnT/University of Luxembourg
Luxembourg
rafieh.mosaheb@uni.lu

Johannes Müller
LORIA/INRIA/CNRS
France
johannes.mueller@inria.fr

Reetika
Indian Institute of Space Science and Technology
India
reetika.reetika.2002@gmail.com

***Abstract*—Zero-Knowledge Proofs (ZKPs) are cryptographic building blocks of many privacy-preserving security protocols. An important research focus in this area is the development of post-quantum ZKPs. These are ZKPs whose security is reduced to computational hardness assumptions that are assumed to be intractable even by scalable quantum computers.**

In this paper, we study the post-quantum ZKPs of Jain, Krenn, Pietrzak, and Tentes (Asiacrypt 2012). These are the only ZKPs for proving arbitrary binary statements whose security reduces to the Learning Parity with Noise (LPN) problem–a very conservative post-quantum hardness assumption.

We make the following contributions to further develop the potential and understanding of these ZKPs. First, we optimize the efficiency of the verifier by several orders of magnitude, making this part as computationally light as that of the prover. Second, we show that the only open source implementation of these ZKPs does not implement them correctly, allowing a malicious prover to convince the verifier of false statements. Third, we formally verify for the first time the security of these (optimized) ZKPs in EasyCrypt. Fourth, we show how these ZKPs can be used to construct the first code-based ZKP of shuffle and verifiable e-voting protocol.

I. Introduction

Zero-Knowledge Proofs (ZKPs) are cryptographic building blocks of many privacy-preserving security protocols. Using a ZKP, a party can convince another party that a certain statement is correct without revealing any (possibly sensitive) information beyond the correctness of that statement. For example, ZKPs can be employed to authenticate users without learning their private credentials, or to verify that encrypted ballots in an e-voting protocol were tallied correctly without learning the links between individual voters and their choices.

Over the last three decades, research in ZKP has led to a wide variety of solutions. These systems differ from each other in various aspects, in particular with respect to the type of statements (specific vs. arbitrary) and the cryptographic objects (commitments, signatures, public-key ciphertexts, etc.) they support, their efficiency, and the security assumptions on which they are based (discrete logarithm, lattice-based, etc.).

In this paper, we will focus on ZKPs whose security is based on the difficulty of solving a well-studied code-based problem, namely the hardness of *decoding random linear codes*. This assumption is characterized by the fact that, despite decades of research, there has been no significant speedup in its cryptanalysis, neither in the classical nor in the quantum model. Therefore, decoding random linear codes is considered an extremely conservative hardness assumption in cryptography and especially in post-quantum cryptography.

One of the most prominent ZKPs in this field is the protocol by Jain, Krenn, Pietrzak, and Tentes [35] (Asiacrypt 2012). Their ZKP can be used to prove arbitrary binary statements among committed bit vectors. The security of their ZKP and commitment scheme reduces to the *Learning Parity with Noise (LPN)* problem, which is equivalent to the hardness of decoding random linear codes. Specifically, the LPN problem asks to distinguish "noisy" systems of binary vectors of the form $\mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$, where $\mathbf{e}$ is the noise, from uniformly random binary vectors. This means that LPN is a specific instance, for $p = 2$, of the more general *Learning With Errors (LWE)* problem, which is defined for arbitrary vector spaces $\mathbb{Z}_p$.

Since the Jain et al. system operates directly at the binary level, it is a reasonable candidate for applications with hardware constraints that require fast low-level implementations. For example, Prada-Delgado, Baturone, Dittmann, Jelitto and Kind [38] demonstrated how to use this ZKP to implement a *Physical Unclonable Function (PUF)* on a microcontroller.

However, despite these contributions, the potential and understanding of the original ZKPs of Jain et al. [35] and the work building on it is limited in the following aspects:

1) *Slow verifier*: While the prover of these ZKPs is computationally light, the computational complexity of the verifier of the original ZKPs is several orders of magnitude higher and therefore hardly practical. This is reflected, for example, in the benchmarks presented by Bellini, Gaborit, Hasikos, and Mateu [11] (Table 5 in their paper). Their performance results show that in the underlying Σ-protocol of the basic ZKP (i.e., proving the well-formedness of a single commitment), the prover needs

less than 2 ms to produce the proof, while the verifier on average needs more than 116 ms to verify it.
2) *Security understanding*: The understanding of whether and how these ZKPs provide the desired security properties needs to be improved. First, while Jain et al. have provided handwritten security proofs for their commitment scheme and (the basic version of) their ZKP, the security of their schemes has not yet been formally verified. Second, the complexity of these ZKPs makes it difficult to understand them and, as we will show, to implement them correctly in a way that guarantees security in practice.
3) *Limited application scenarios*: The applications of these ZKPs in the literature is currently focused on the Internet of Things (IoT). However, it has not been researched whether these cryptographic protocols can also be used for other privacy-preserving applications.

A. Our contributions

We make the following contributions to further develop the potential and understanding of this code-based ZKP:

1) *Optimization*: We show how the original ZKPs of Jain et al. can be modified in a simple but effective way so that the computational complexity of the verifier becomes as light as that of the prover. Our modification fully preserves the security of the original ZKPs: the main idea behind this optimization is that the prover sends some additional information to the verifier that the verifier would otherwise have to compute itself in a relatively inefficient way. Since we are not aware of any related work using the same trick, we suspect that it may be of independent interest. Our benchmarks show that our version of these ZKPs can be realized several orders of magnitude more efficiently than before.
2) *Security review*: We reviewed the only open source implementation [11] of the original ZKP and found that it does not implement it correctly. We show that a malicious prover in their implementation can exploit this inconsistency to convince an honest verifier of the correctness of a false statement.
3) *Formal verification*: We formally prove for the first time the security of the commitment scheme and of the full (optimized) ZKPs in EasyCrypt.
4) *New application*: We show how to create an efficient ZKP of shuffle whose security reduces to the LPN problem. These cryptographic protocols are common building blocks for verifiable mixing networks [29] that can anonymize sensitive data in a verifiable way. As a concrete application, we present the first code-based verifiable e-voting protocol.

B. Related work

1) Code-based ZKPs: The original idea of code-based ZKPs dates back to the seminal work of Stern [42]. Since then, many code-based ZKPs have been proposed in the literature. There are essentially two groups of code-based ZKPs, depending on whether they prove specific or arbitrary statements. The ZKPs in the first group (e.g., [13], [33], [14], [27], [36]) prove knowledge of a committed or encrypted value, which can be used to create efficient code-based signatures. In the second group, however, there are much fewer protocols in the literature, namely the one by Jain et al. [35] (which we study in this paper) and [11], [24]. From a security perspective, the main difference between [35] and [11], [24] is the metric that they apply to measure the distance of a code: the first work uses the Hamming metric and the two other use the rank metric. While assuming the rank metric can lead to more efficient constructions, this general approach is controversial because several rank-based cryptographic primitives have already been broken (see, e.g., [9] and the references therein). This means that, to the best of our knowledge, the ZKP of [35] is the only one in the literature to prove arbitrary statements among committed values and whose security reduces to the (currently) undisputed assumption that it is computationally intractable to decode random linear codes in the Hamming metric.

The ZKPs of Jain et al. [35], which we verified, are examples of "Stern-style" sigma protocols. Stern's work [42] does not use the language of sigma protocols since it predates that notion, but it has the requisite 3-round structure. The kind of soundness achieved by a Stern sigma protocol is not the usual definition of special soundness but rather what has subsequently been called l-*special soundness* [16], since the extractor is allowed l accepting transcripts with distinct messages; specifically for "Stern-style" sigma protocols l is 3. The basic problem being addressed which prevents the use of more standard sigma protocols is that the witness has low weight and so masking it in a way which preserves both soundness and zero-knowledge is non-trivial. The solution is to commit to the mask and masked secret as normally with a cut-and-choose ZKP, along with permuted versions of these two values. Depending on the challenge either check the consistency of mask and permuted mask, masked secret and permuted masked secret, or the weight of the permuted secret.

2) Formal verification of code-based cryptography: To the best of our knowledge, there has been no specific work on the formal verification of code-based cryptography outside of information-theoretic analysis. In this sense, our work breaks new ground; however, as we have already observed, the LPN problem is a specific instance of the LWE problem. LWE, and lattice-based cryptography more generally, has been a significant focus of formal verification including works on Saber [34], Dilithium [7], and Kyber [3]. In fact, we use the EasyCrypt libraries designed for lattice-based cryptography with additional constraints to restrict us to the LPN setting.

Sigma protocols have seen a significant amount of formal verification including [10], [2], [25], [28]; to our knowledge of these only [28] supports l-special soundness. Moreover, to our knowledge all the existing work targets Schnorr style sigma protocols [40] (proving knowledge of some preimage) or simple cut and choose with a challenge space of 2. We preferred to make use the existing support for LWE in EasyCrypt rather than the existing support for l-special sound sigma protocols in Coq. Extending the work of [10], [2] to cover Stern style

sigma protocols would be interesting but we expect it would be complicated, if not impossible, given that the Stern style protocols do not appear to have the homomorphic property which is relied upon.

3) Post-quantum ZKP of shuffle: Several post-quantum ZKPs of shuffle have been proposed in the literature [4], [5], [32], [31], [21], [20]. Their security reduces to the *Ring Learning With Errors (RLWE)* assumption, which is often assumed to create more efficient lattice-based ZKPs. However, this advantage comes at the cost of a potentially lower security guarantee because RLWE requires specially structured lattices in contrast to the pure LWE assumption. In contrast, the security of our proof of shuffle reduces to the hardness of decoding random linear codes, which, as mentioned above, is a more conservative hardness assumption.

4) Post-quantum verifiable e-voting: There are basically two ways to achieve verifiable e-voting, which differ in the way ballots are verifiably counted: homomorphic tallying and verifiable mixing.

For *homomorphic tallying*, the encryption (or commitment) scheme to encrypt the voters' ballots is additively homomorphic. This property is used to first aggregate the voters' ballots homomorphically and finally decrypt (or open) the aggregated ciphertext in a verifiable way. The main advantage of this approach is that the aggregation is verifiable by anyone without any further proof. Since the voters need to prove the well-formedness of their encrypted ballots, which can become expensive in terms of memory and time, homomorphic tallying is particularly useful for elections with simple ballot types (e.g., yes/no elections). The only post-quantum solutions for homomorphic tallying in the literature are the lattice-based e-voting protocols Evolve [23] and Epoque [18].

For *verifiable mixing*, the encrypted ballots are processed through a mixing network. This is a sequence of so-called *mixing servers* that successively re-randomize and shuffle the voters' encrypted ballots. Since voters do not have to prove the well-formedness of their ballots, this approach can handle arbitrarily complex ballot types. There are several techniques for making a mixing network verifiable [29]. The ones that have been used in the literature for post-quantum e-voting are the lattice-based ZKPs of shuffle mentioned above and the so-called *trip-wire technique* [17]. Unlike the existing post-quantum ZKPs of shuffle, which are all based on the RLWE assumption, the trip-wire technique can also be realized efficiently with cryptographic primitives that are based on the (more conservative) pure LWE problem, and it avoids a lattice-based ZKP of correct decryption [26], [41]. On the other hand, the trip-wire technique requires an active verifier, while the ZKPs of shuffle can also be verified passively.

II. Preliminaries

We introduce the notation we use throughout this paper and the definitions of commitment schemes, zero-knowledge proofs, Σ protocols, and the (exact) Learning Parity with Noise (LPN) assumption. We have adopted the notation and definitions from the original paper for consistency.

A. Notation

We denote vectors by bold lowercase letters $\mathbf{a}$, matrices by bold uppercase letters $\mathbf{A}$, probabilistic polynomial time (ppt) algorithms by sans-serif letters A, and sets by calligraphic letters $\mathcal{A}$.

We use $a \xleftarrow{R} \mathcal{A}$ to denote that a is drawn uniformly at random from the set $\mathcal{A}$, $a \xleftarrow{R} \chi$ to denote that a is drawn according to a probability distribution χ, and $a \xleftarrow{R} \mathsf{A}$ to denote that a is a result of a randomized algorithm A.

We define the sets $\mathcal{I} = \{0,1\}$ and $\mathcal{I}^k = \{0,1\}^k$. The Hamming weight of a vector $\mathbf{a} \in \mathcal{I}^k$ is $\|\mathbf{a}\|_1 = \sum_{i=1}^{k} \mathbf{a}[i]$. We define the set of all k-bit vectors of weight exactly w as $\mathcal{I}_w^k = \{\mathbf{a} \in \mathcal{I}^k \colon \|a\|_1 = w\}$. We denote the all-zeroes vector by $\mathbf{0}^k$ and the all-ones vector by $\mathbf{1}^k$. We denote the concatenation of $\mathbf{a}$ and $\mathbf{b}$ by $\mathbf{a}\|\mathbf{b}$.

We denote the symmetric group on k elements by $\mathcal{S}_k$, and the permutation of a vector $\mathbf{a} \in \mathcal{I}^k$ with permutation $\pi \in \mathcal{S}_k$ by $\pi(\mathbf{a})[\cdot] = \mathbf{a}[\pi(\cdot)]$.

B. Commitment scheme

A commitment scheme allows a party to commit to a message without revealing the message itself (see App. A).

The commitment scheme we study satisfies the following security properties: (1) *correctness* guarantees that if the public key pk and the commitment are generated correctly, then the verification algorithm accepts (with overwhelming probability), (2) *perfectly binding* ensures that a commitment can only be opened to one message, and (3) *computationally hiding* guarantees that a commitment does not reveal the committed message. See App. A for the formal definitions of these properties.

C. Learning Parity with Noise

The Learning Parity with Noise (LPN) problem asks to distinguish "noisy" systems of binary linear equations of the form $\mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$, where $\mathbf{e}$ is the noise, from uniformly random binary vectors.

Definition 1 (Learning Parity with Noise (LPN)): For $k, \ell \in \mathbb{N}$, let χ be an error distribution over $\mathcal{I}^k$. The *decisional* (χ, ℓ, k)*-LPN problem* is (s, ϵ)-hard if for every distinguisher D of size s:

$$\left| \Pr_{\mathbf{A},\mathbf{x},\mathbf{e}} [\mathsf{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{e}) = 1] - \Pr_{\mathbf{A},\mathbf{r}} [\mathsf{D}(\mathbf{A}, \mathbf{r}) = 1] \right| \leq \epsilon,$$

where $\mathbf{A} \xleftarrow{R} \mathcal{I}^{k \times \ell}$, $\mathbf{e} \xleftarrow{R} \chi$, $\mathbf{r} \xleftarrow{R} \mathcal{I}^k$, and $\mathbf{x} \xleftarrow{R} \mathcal{I}^\ell$.

For $\tau \in (0, \frac{1}{2})$, we denote by LPN_τ the version of LPN, where χ is the Bernoulli distribution and every bit $\mathbf{e}[i]$ is chosen independently and identically distributed with $Pr[\mathbf{e}[i] = 1] = \tau$.

Jain et al. [35] introduced the following new version of the LPN problem on which the security of their commitment scheme and ZKP will be based. In the *exact* LPN problem with parameter τ, the Hamming weight of the error vector is exactly $\lfloor k\tau \rceil$, not only of expected weight $k\tau$ as in LPN_τ.

Definition 2 (Exact Learning Parity with Noise (xLPN)): Let $k, \ell \in \mathbb{N}$ and let $0 < \tau < 1/2$ be the noise parameter.

The *decisional exact* (τ, ℓ, k)*-LPN problem* is (s, ϵ)-hard if for every distinguisher D of size s:

$$\left| \Pr_{\mathbf{A},\mathbf{x},\mathbf{e}} [\mathsf{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{e}) = 1] - \Pr_{\mathbf{A},\mathbf{r}} [\mathsf{D}(\mathbf{A}, \mathbf{r}) = 1] \right| \leq \epsilon,$$

where $\mathbf{A} \xleftarrow{R} \mathcal{I}^{k \times \ell}$, $\mathbf{e} \xleftarrow{R} \mathcal{I}^k_{[k\tau]}$, $\mathbf{r} \xleftarrow{R} \mathcal{I}^k$, and $\mathbf{x} \xleftarrow{R} \mathcal{I}^\ell$ is fixed and secret.

Jain et al. proved that the hardness of the decisional xLPN$_\tau$ problem is polynomially related to the hardness of the search LPN$_\tau$ problem, which itself is based on the hardness of decoding random linear codes [15]. We will choose the parameters of the commitment scheme and ZKP such that these reductions hold true.

D. Zero-knowledge proofs of knowledge and Σ-protocols

A Zero-Knowledge Proof (ZKP) is an interactive protocol between two parties, the prover P and the verifier V. The prover P wants to convince the verifier V that a certain statement x is true without revealing any further information.

More formally, the common input to the prover and the verifier is a binary relation $\mathcal{R}$ and a statement x. The secret input to the prover is a witness w such that $(x, w) \in \mathcal{R}$. Now, the prover wants to convince the verifier that there exists w such that $(x, w) \in \mathcal{R}$ without revealing any information about w. For example, $\mathcal{R}$ can model that the messages of two commitment vectors are equal modulo permutation, when x contains the two commitment vectors and w contains the messages and the permutation.

The ZKPs that we study in this work are created with so-called Σ*-protocols* (see App. A). They can be turned into non-interactive ZKPs with the Fiat-Shamir heuristic or into interactive ZKPs with standard techniques, such as [22].

III. Original Scheme

A. Commitment scheme

The commitment to a message $\mathbf{m} \in \mathcal{I}^v$ is $\mathsf{Com}(\mathbf{m}) = \mathbf{A} \cdot (\mathbf{r} \| \mathbf{m}) \oplus \mathbf{e}$, where $\mathbf{A} \in \mathcal{I}^{k \times (\ell+v)}$ is public random binary matrix, $\mathbf{r} \in \mathcal{I}^\ell$ is a uniformly random vector, and $\mathbf{e} \in \mathcal{I}^k$ is a uniformly random vector with fixed low weight.

Definition 3 (xLPN commitment scheme [35]): Let $\ell \in \mathbb{N}$ be the main security parameter, $0 < \tau < 0.25$ be the noise parameter, $v \in \mathbb{N}$ be the message length, $k \in \Theta(\ell + v)$, and $w = [\tau k]$ be the weight. We assume these parameters as implicit input to all algorithms.

- KeyGen:
  1) Sample $\mathbf{A}' \xleftarrow{R} \mathcal{I}^{k \times \ell}$ and $\mathbf{A}'' \xleftarrow{R} \mathcal{I}^{k \times v}$.
  2) Return public key $\mathbf{A} = \mathbf{A}' \| \mathbf{A}'' \in \mathcal{I}^{k \times (\ell+v)}$.
- $\mathsf{Com}(\mathbf{A}, \mathbf{m})$:
  1) Sample $\mathbf{r} \xleftarrow{R} \mathcal{I}^\ell$ and $\mathbf{e} \xleftarrow{R} \mathcal{I}^k_w$.
  2) Set $\mathbf{c} = \mathbf{A} \cdot (\mathbf{r} \| \mathbf{m}) \oplus \mathbf{e}$.
  3) Return commitment/opening pair $(\mathbf{c}, (\mathbf{r}, \mathbf{e}))$.
- $\mathsf{Ver}(\mathbf{A}, \mathbf{c}, \mathbf{m}', \mathbf{r}')$:
  1) Compute $\mathbf{e}' = \mathbf{A} \cdot (\mathbf{r}' \| \mathbf{m}') \oplus \mathbf{c}$.
  2) Return 1 if $\|\mathbf{e}'\|_1 = w$ and 0 otherwise.

Jain et al. provided a handwritten proof that this commitment scheme is perfectly binding and computationally hiding if the parameters are chosen such that the respective xLPN$_\tau$ problem is hard and such that (with overwhelming probability) the public key $\mathbf{A}$ has weight larger than $2w$.

B. Σ-protocol

Jain et al. proposed three LPN-based Σ-protocols for specific relations that can be combined to construct an LPN-based Σ-protocol for any relation. The first Σ-protocol proves the well-formedness of a single commitment, the second one linear and the third one multiplicative relations between commitments. Essentially, the Σ-protocols for linear and multiplicative relations are built with several instances of the Σ-protocol of well-formedness and some mechanisms for assembling these instances.

Since the Σ-protocol of well-formedness is the main component of the general-purpose Σ-protocol, we restrict our following summary to this Σ-protocol. (We have still formally verified all Σ-protocols.) The full protocol is presented in Fig. 1. In this protocol, the joint input to the prover and the verifier is a pair $(\mathbf{A}, \mathbf{c})$, and the prover wants to convince the verifier that it knows $(\mathbf{e}, \mathbf{s})$ such that $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$ and $\|\mathbf{e}\|_1 = w$ hold true.

1) Main idea: The main idea behind this protocol is as follows. In the first step, the prover creates a mask $\mathbf{t} = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f}$. Now, at a higher level, depending on the challenge of the verifier, the prover either reveals the mask or the masked commitment. To ensure that the masked commitment does not leak any information about the original commitment $\mathbf{c}$, in particular about the noise vector $\mathbf{e}$, Jain et al. used the following trick. First, the vector $\mathbf{f}$ in the mask is chosen uniformly at random from $\mathcal{I}^k$ (and not from $\mathcal{I}^k_w$ as in an xLPN commitment). Second, the prover shuffles $\mathbf{f} \oplus \mathbf{e}$ with a random permutation π that serves as an additional mask.

2) Protocol flow: Based on this idea, the Σ-protocol works as follows. In the first step, the prover creates two masks: a random permutation π and a vector $\mathbf{t}_0 = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f}$, where $\mathbf{v}$ and $\mathbf{f}$ are chosen uniformly at random. Then, the prover creates the following three commitments (here, the commitment scheme can be any perfectly binding commitment scheme): c_0 commits to the pair of masks π and $\mathbf{t}_0$, c_1 to the permutation of $\mathbf{f}$ with π, and c_2 to the permutation of $\mathbf{f} \oplus \mathbf{e}$ with π. Afterwards, the prover sends the commitments $a = (c_0, c_1, c_2)$ to the verifier, who responds with a random challenge $e \in \{0, 1, 2\}$. If $e = 0$, the prover opens c_0 and c_1, if $e = 1$, the prover opens c_0 and c_2, and if $e = 2$, the prover opens c_1 and c_2. Then, the verifier checks the correctness of the respective openings and, depending on the challenge, also verifies the following. If $e = 0$, the verifier checks whether $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1)$ is in the image of $\mathbf{A}$ and whether π is a permutation. If $e = 1$, the verifier checks whether $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c}$ is in the image of $\mathbf{A}$. If $e = 2$, the verifier checks whether $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 = w$.

3) Security: Jain et al. presented a handwritten proof why this protocol is in fact a Σ-protocol according to Definition 5.

We formally verified this theorem in EasyCrypt (see Section VI). We now explain on a more intuitive level why the security holds true.

To see why this protocol is special honest-verifier zero-knowledge, observe that for challenge $e = 0$, the output of the verifier does not contain any information about $\mathbf{c}$; for challenge $e = 1$, the secret vector $\mathbf{v}$ perfectly blinds $\mathbf{f}$, which itself perfectly blinds $\mathbf{e}$; for challenge $e = 2$, the secret permutation π perfectly blinds $\mathbf{f}$ and $\mathbf{f} \oplus \mathbf{e}$.

This protocol provides special soundness because when we assume that the checks for all possible challenges e succeed, then the validity of all individual checks implies the existence of $(\mathbf{s}', \mathbf{e}')$ with $\mathbf{A} \cdot \mathbf{s}' \oplus \mathbf{e}'$ and $\|\mathbf{e}'\|_1 = w$.

IV. Optimization

We show how the computational efficiency of the verifier algorithm in the original Σ-protocols (Section III) can be improved by several orders of magnitudes in a simple way without affecting security. Since it is easy to see that our improvement applies to all Σ-protocols of this family (linear and multiplicative relations), we here focus on the basic protocol for proving knowledge of a valid opening.

A. Problem

The verifier in the original protocol checks whether $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \stackrel{?}{\in} \text{img}(\mathbf{A})$ for $e = 0$, and $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \stackrel{?}{\in} \text{img}(\mathbf{A})$ for $e = 1$.

While numerical linear algebra provides different algorithms to verify whether a given vector is in the image of a linear function (here, the linear function defined by $\mathbf{A}$), the computationally best algorithms (to give an exact, not only an approximate answer) we know have efficiency $\mathcal{O}(n^3)$, where n is the size of the vector. This issue is for example reflected in the benchmarks of [11] who implemented the original Σ-protocol [35]. They report that the time for checking one of these two equations is more than 170 ms, while each other operation (i.e., sampling random elements and permutations, computing and verifying commitments, checking Hamming weights) requires about 0.5 ms or less.

Since the Σ-protocol needs to be repeated several times to reduce the knowledge error, this drawback creates a significant practical overhead that makes the original Σ-protocol infeasible for many realistic applications.

B. Our solution

Our main observation is that in the original protocol, the verifier gets enough information to efficiently compute all solutions of the equations $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \stackrel{?}{\in} \text{img}(\mathbf{A})$ for $e = 0$, and $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \stackrel{?}{\in} \text{img}(\mathbf{A})$ for $e = 1$.

We therefore propose to augment the output of the prover by its solutions to these equations, namely $\mathbf{v}$ in the first case and $\mathbf{v}' = \mathbf{v} \oplus \mathbf{s}$ in the second one. While this modification does not affect the zero-knowledge property of the scheme, it allows the verifier to check these two equations more efficiently by $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \stackrel{?}{=} \mathbf{A} \cdot \mathbf{v}$ for $e = 0$, and $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \stackrel{?}{=} \mathbf{A} \cdot \mathbf{v}'$ for $e = 1$.

We have modified the corresponding checks in Bellini et al.'s implementation[1] accordingly and tested their efficiency on a standard laptop. The parameters $\ell = 128$, $k = 1320$, $v = 2640$, $w = 284$ were chosen such that the underlying decoding problem requires at least 2^{128} operations with a quantum or classical computer; see Section 5.1 in [11] for details.

We ran both the original and our optimized subroutine on standard machine. We found that the optimization makes the subroutine 7137 times faster on average. Applying this factor to the runtime of more than 170 ms reported by Bellini et al., we interpolate that the runtime of the optimized subroutine in their setup requires less than 0.025 ms. Interestingly, our (simple) improvement makes Jain et al.'s ZKP similarly fast to Bellini et al.'s rank-based ZKP, even though it is based on a more conservative hardness assumption.

We note that the size of the transcript of the Σ-protocol is bounded by $7\frac{2}{3} \cdot k = 1.265$ Kilobytes, when we choose the parameters as above and instantiate the commitment scheme Com' with the Jain et al.'s commitment scheme (Section III).

C. Security

We provide two proofs that the optimized version is a Σ-protocol for the (original) relation $\mathcal{R} = \{((\mathbf{A}, \mathbf{c}), (\mathbf{s}, \mathbf{e})) \colon \mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e} \wedge \|e\|_1 = w\}$. Our first proof is a pen-and-paper proof that we present in this section. Our second proof is formally verified (see Sec. VI).

a) Correctness.: We show that if the prover and verifier are honest, then the transcript is always accepted by the verifier:

- If the challenge is 0, we have $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) = (\mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f}) \oplus \mathbf{f} = \mathbf{A} \cdot \mathbf{v}$.
- If the challenge is 1, we have $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = \mathbf{A}(\mathbf{v} \oplus \mathbf{s})$.
- If the challenge is 2, we have $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 = \|\mathbf{f} \oplus (\mathbf{f} \oplus \mathbf{e})\|_1 = \|\mathbf{e}\|_1 = w$.

b) Special soundness.: We show that from three accepting transcripts for challenges 0, 1, and 2, we can efficiently extract a witness $(\mathbf{s}, \mathbf{e})$ for the statement $(\mathbf{A}, \mathbf{c})$. If the transcripts are accepted, it follows from the binding property of the commitment scheme $(\mathsf{Com}', \mathsf{Ver}')$ that $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) = \mathbf{A} \cdot \mathbf{v}$ and $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = \mathbf{A}(\mathbf{v} \oplus \mathbf{s})$. By adding these equations, we obtain the first part of the witness, $\mathbf{s}$, and by adding $\mathbf{A} \cdot \mathbf{s}$ to $\mathbf{c}$, we obtain the second part $\mathbf{e}$.

c) Special honest-verifier zero-knowledge.: For each challenge $e \in \{0, 1, 2\}$, we show how to modify the real protocol with the knowledge of e but without knowledge of the witness, so that the resulting family of transcripts is indistinguishable from the family of transcripts of the real protocol. Our simulator is identical to the real protocol except for the following changes:

- $e = 0$: We set $(c_2, d_2) \xleftarrow{R} \mathsf{Com}'(0)$.
- $e = 1$: We sample $\mathbf{t}_2 \xleftarrow{R} \mathcal{I}^k$, $\mathbf{v}' \xleftarrow{R} \mathcal{I}^{\ell+v}$, and $\pi \xleftarrow{R} \mathcal{S}_k$. We set $\mathbf{t}_0 \leftarrow \mathbf{A} \cdot \mathbf{v}' \oplus \mathbf{c} \oplus \pi^{-1}(\mathbf{t}_2)$ and $(c_1, d_1) \xleftarrow{R} \mathsf{Com}'(0)$.

[1] See https://github.com/Crypto-TII/2020-CANS-rank_commitments (accessed 26.04.2024).

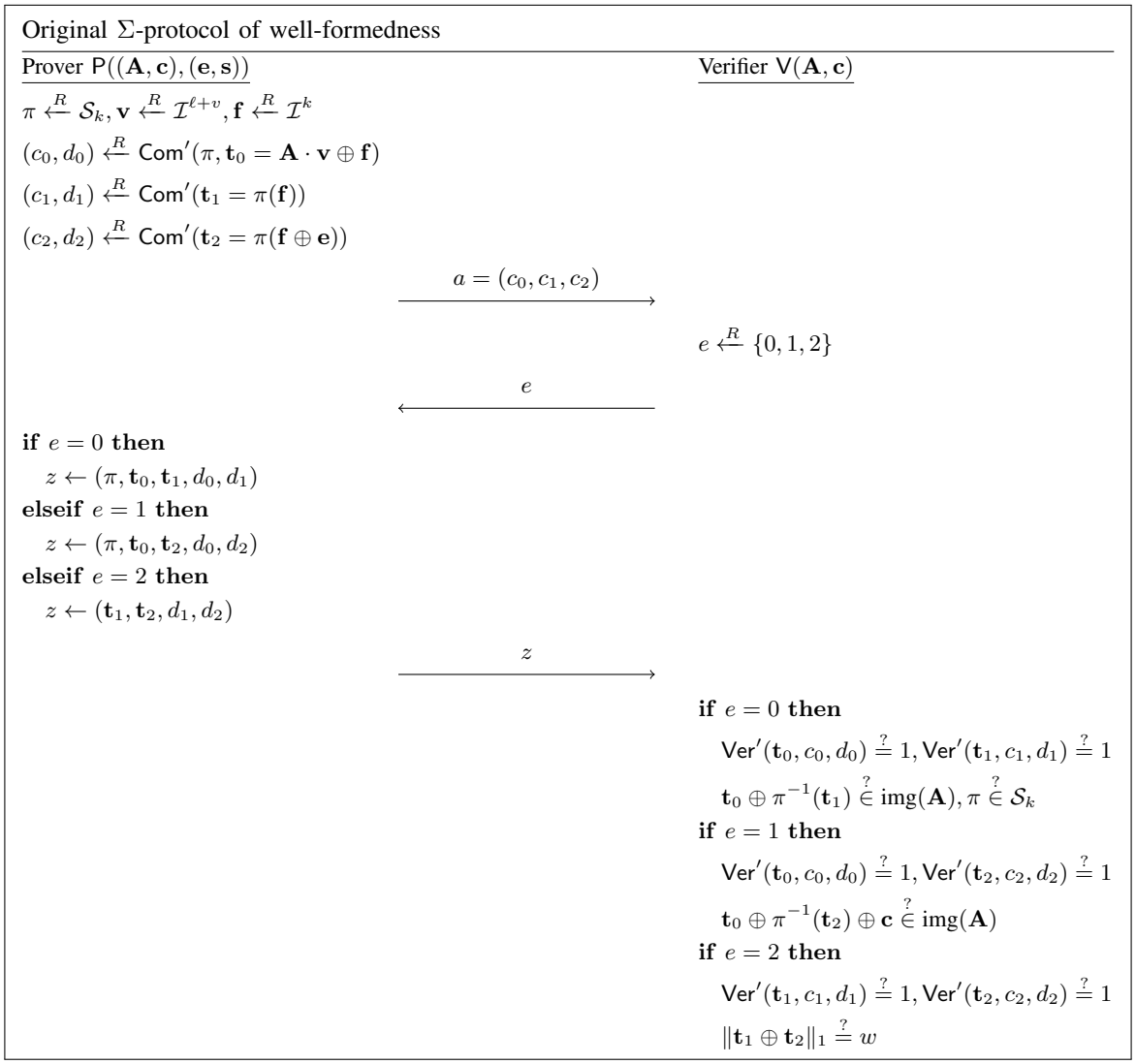

Fig. 1: Original Σ-protocol for $\mathcal{R} = \{(\mathbf{A}, \mathbf{c}), (\mathbf{e}, \mathbf{s}) \colon \mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e} \wedge \|\mathbf{e}\|_1 = w\}$ (well-formedness of a commitment).

- $e = 2$: We sample $\mathbf{t}_1 \xleftarrow{R} \mathcal{I}^k$ and $\mathbf{t}_2 \xleftarrow{R} \mathcal{I}_w^k$. We set $(c_0, d_0) \xleftarrow{R} \mathsf{Com}'(0)$.

From the hiding property of the commitment scheme $(\mathsf{Com}', \mathsf{Ver}')$, it follows that the real and the modified protocol are indistinguishable.

V. Security review of implementation

We discovered that the only open-source implementation of the ZKP [35], namely the one by Bellini et al. [11] (see Fig. 3), is not correct and therefore does not provide soundness. In fact, for the following reason, a malicious prover in their implementation can cheat without being caught. We note that the same soundness issue also applies to the implementation of the rank-metric ZKP that [11] proposed themselves.

The issue is that the first commitment c_0 of the prover does not contain the permutation π, i.e., the prover in their implementation computes c_0 as $\mathsf{Com}(\mathbf{t}_0 = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f})$ instead of $\mathsf{Com}(\pi, \mathbf{t}_0 = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f})$, as defined in [35]. In this way, a malicious prover can convince an honest verifier that a vector $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$ is supposedly well-formed even when $\|\mathbf{e}\|_1 \neq w$.

For example, consider the case that $k = 3$, $w = 1$, $\mathbf{A} = [(1,0,0), (0,1,0), (0,1,0)]$, and $\mathbf{c} = \mathbf{A} \cdot \mathbf{s}$ for some arbitrary $\mathbf{s}$. Although $(\mathbf{A}, \mathbf{c})$ is not a valid statement (because $\mathbf{e} = \mathbf{0}^3$), a malicious prover can convince an honest verifier otherwise, as described next.

First, the dishonest prover chooses $\mathbf{t}_0 = \mathbf{0}^3$, $\mathbf{t}_1 = (1,0,1)^t$, and $\mathbf{t}_2 = (0,0,1)^t$. If $e = 0$, the prover returns $\mathbf{t}_0, \mathbf{t}_1$, and $\pi_0 := (12)(3)$ to the verifier, and if $e = 1$, it returns $\mathbf{t}_0, \mathbf{t}_2$, and $\pi_1 := (13)(2)$. If $e = 2$, the prover just sends $\mathbf{t}_1$ and $\mathbf{t}_2$. Although $\|\mathbf{e}\|_1 = 0 \neq w$, the verifier in their implementation accepts the transcript because all equations are correct (note that the vectors in $\mathrm{img}(\mathbf{A})$ are of the form $(a, b, b)^t$ for $a, b \in \mathcal{I}$):

- $e = 0 : \mathbf{t}_0 \oplus \pi_0^{-1}(\mathbf{t}_1) = (0,1,1)^t \in \mathrm{img}(\mathbf{A})$.
- $e = 1 : \mathbf{t}_0 \oplus \pi_1^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = (1,0,0)^t \oplus \mathbf{A} \cdot \mathbf{s} \in \mathrm{img}(\mathbf{A})$.
- $e = 2 : \|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 = \|(1,0,0)^t\|_1 = 1 = w$.

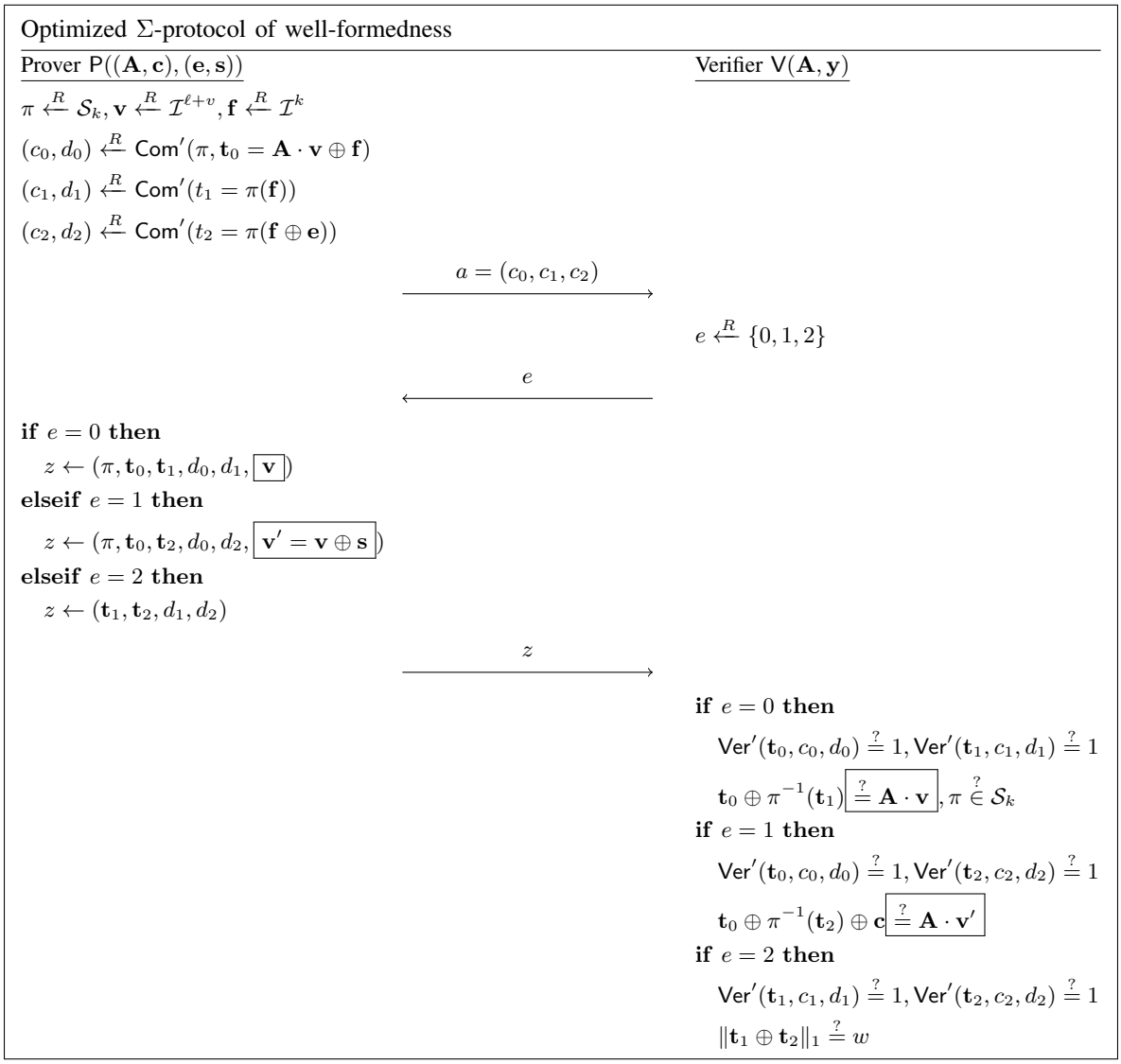

Fig. 2: Optimized Σ-protocol for $\mathcal{R} = \{(\mathbf{A}, \mathbf{c}), (\mathbf{e}, \mathbf{s}) \colon \mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e} \wedge \|\mathbf{e}\|_1 = w\}$ (well-formedness of a commitment)

VI. Verification

We have verified all the main results of [35] in the interactive theorem prover EasyCrypt; note that EasyCrypt only considers classical adversaries so while we reduce to a Post-Quantum assumption we do not prove the result for quantum adversaries. We expect that these results could be lifted to Quantum Adversary's using EasyPQC [8] but we have not done so. Our proof files and verification instructions can be found at https://github.com/gerlion/zkp-xpln.

Our EasyCrypt proofs use the standard EasyCrypt definition of a Σ-protocol with some modifications to handle the need for three accepting transcripts to be extracted, and the restriction of the challenge space to be the integers modulo 3. We use a standard linear algebra library (DynMatrix) restricted to the case where the underlying field has two elements. We introduce a variant of the standard commitment protocol which gives the adversary access to certain auxiliary information in the hiding experiment, this information does not depend on the message to be committed to. We include a discussion of the axioms we used in Appendix C. Below we include the various final lemmas we proved in EasyCrypt; we do not have space to include all the definitions, which can be found in the supplementary material, but this should nevertheless give a feel for our results.

a) Theorem 3.1: The correctness, binding, and hiding of the commitment scheme.
JKPT12 commitment scheme has prefect correctness.

```
lemma JKPT12_correctness:
    hoare[Correctness(JKPT12).main: true ==> res].
```

For any adversary the chance it can win the binding game is bounded by the chance that the sampled generator matrix of the linear code does not have distance larger than $2w$. This value is negligible for given proper choices of k, l, and v.

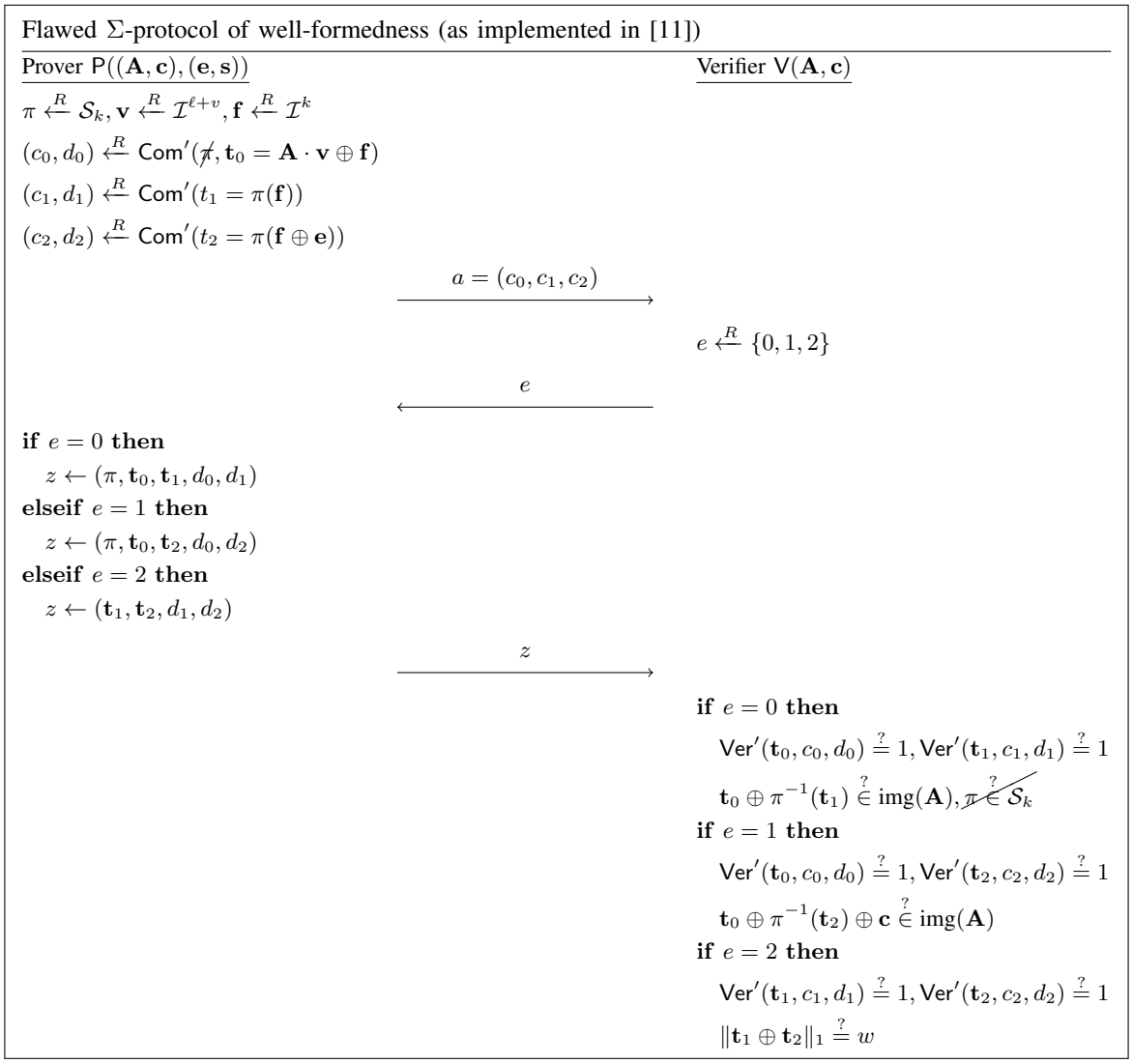

Fig. 3: Implemented Σ-protocol for $\mathcal{R} = \{(\mathbf{A}, \mathbf{c}), (\mathbf{e}, \mathbf{s}) \colon \mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e} \wedge \|\mathbf{e}\|_1 = w\}$ (well-formedness of a commitment) by [11].

```
lemma JKPT12_computational_binding (B<:Binder) &m:
  islossless B.bind =>
  Pr[BindingExperiment(JKPT12, B).main() @ &m : res]
    <= mDist k l v.
```

The adversary chance of breaking hiding reduces to the exact LPN problem.

```
lemma JKPT12_comp_hiding (U<:Unhider) &m:
 islossless U.choose =>
 islossless U.guess =>
 Pr[HidingExp(JKPT12,U).main() @ &m : res] -1%r/2%r =
 Pr[GameL(A(U)).main() @ &m :res]-
 Pr[GameR(A(U)).main() @ &m : res].
```

b) Theorem 4.1: The completeness, soundness, and (honest-verifier) zero-knowledge of the valid opening ZKP. The valid open ZKP has perfect completeness.

```
lemma ValidOpenProt_Completeness s w' &m: R s w' =>
  Pr[Completeness(VOProt).main(s, w') @ &m : res]
    = 1%r.
```

The valid open ZKP has perfect binding.

```
lemma ValidOpenProt_Sound (F <: SigmaFaker) &m:
    hoare[SpecialSoundnessExp(VOProt,
      VOAlg, F).main : true ==> res].
```

The adversary's advantage against the hiding property of the valid open ZKP is bounded by its ability to break the hiding of the used commitment scheme.

```
lemma ValidOpen_ZK s w e &m :
  R s w =>
  e \in range 0 3 =>
  islossless D.distinguish =>
  Pr[SHVZK(VOProt, VOAlg, D).simulate(s,e) @ &m : res]
- Pr[SHVZK(VOProt, VOAlg, D).real(s,w,e) @ &m : res]
<= Pr[HidingExpL(U(D)).main((s,w,e)) @ &m : res] -
   Pr[HidingExpR(U(D)).main((s,w,e)) @ &m : res].
```

c) Theorem 4.2:: The completeness, soundness, and (honest-verifier) zero-knowledge of the linear relation ZKP. The linear relation ZKP has perfect completeness.

```
lemma LinerRelProt_Completeness s w' &m : s w' =>
  Pr[Completeness(LRProt).main(s, w') @ &m  : res] = 1%r.
```

The linear relation ZKP has perfect binding.

```
lemma LinearRelProt_Sound (F <: SigmaFaker)  &m:
    hoare[SpecialSoundnessExp(LRProt, LRAlg, F).main
      : true ==> res].
```

The adversary's advantage against the hiding property of the linear relation ZKP is bounded by its ability to break the hiding of the used commitment scheme.

```
lemma LinearRelPro_ZK s w e &m :
  R s w =>
  e \in range 0 3 =>
  islossless D.distinguish =>
  Pr[SHVZK(LRProt, LRAlg, D).simulate(s,e) @ &m : res]
-  Pr[SHVZK(LRProt, LRAlg, D).real(s,w,e) @ &m : res]
<= Pr[HidingExpL(U(D)).main((s,w,e)) @ &m : res] -
   Pr[HidingExpR(U(D)).main((s,w,e)) @ &m : res].
```

d) Theorem 4.3: The completeness, soundness, and (honest-verifier) zero-knowledge of the multiplicative relation ZKP.
The multiplicative relation ZKP has perfect completeness.

```
lemma MultiRelProt_Completeness s w' &m:
  R s w' =>
  Pr[Completeness(MRProt).main(s, w') @ &m  : res] = 1%r.
```

The multiplicative relation ZKP has perfect binding.

```
lemma MulitRelProt_Sound (F <: SigmaFaker)  &m:
  hoare[SpecialSoundnessExp(MRProt, MRAlg, F).main
    : true ==> res].
```

The adversary's advantage against the hiding property of the multiplicative relation ZKP is bounded by its ability to break the hiding of the used commitment scheme.

```
lemma MulitRelProt_ZK s w e &m :
  R s w =>
  e \in range 0 3 =>
  islossless D.distinguish =>
  Pr[SHVZK(MRProt, MRAlg, D).simulate(s,e) @ &m : res]
-  Pr[SHVZK(MRProt, MRAlg, D).real(s,w,e) @ &m : res]
<= Pr[HidingExpL(U(D)).main((s,w,e)) @ &m : res] -
   Pr[HidingExpR(U(D)).main((s,w,e)) @ &m : res].
```

The proofs of the ZKPs are tedious but relatively straightforward to construct and verify. The only real problem is that the sheer complexity of the linear and multiplicative ZKPs caused problems for EasyCrypt. It seems that the more recent versions of EasyCrypt are able to handle the complexity and we have been able to remove a number of the tricks we initally employed.

We discovered the flaw in the existing implementation (Section V) by comparing it to the EasyCrypt encoding. Since the EasyCrypt encoding is much closer to code and therefore much easier to compare, it makes, for example, explicit many of the input validations that are implicit in the original paper.

VII. Proof of Shuffle

We present the first proof of shuffle whose security reduces to the hardness of decoding random linear codes.

The concept of our proof of shuffle follows the cut-and-choose idea by Sako and Kilian [39], which they instantiated with ElGamal's public-key encryption scheme and corresponding zero-knowledge proof based on the hardness of the discrete logarithm. An implementation of their proof of shuffle is used, for example, in the Helios e-voting system [1]. In this section, we show how to instantiate this blueprint with the optimized LPN-based Σ-protocols from Sec. IV.

a) Notation.: In the following, we generalize our notation of commitments to vectors by applying the commitment scheme to individual entries of the vector.

A. Main idea

Assume that we have a vector of commitments $\vec{c}$ and a vector of plaintext messages $\vec{m}$ such that $\vec{c} \in \mathsf{Com}(\sigma(\vec{m}))$ for some secret permutation σ. In the first step of the underlying interactive protocol, the prover creates an intermediate commitment vector $\vec{\gamma} \xleftarrow{R} \mathsf{Com}(\pi(\vec{m}))$ to the same vector of messages $\vec{m}$ but with an independent random permutation π. Depending on the challenge from the verifier, the prover either opens $\vec{\gamma}$ to show that it is in fact a commitment to $\vec{m}$, or the prover reveals $\sigma' \leftarrow \sigma \circ \pi^{-1}$ and proves that the commitment vectors $\vec{c}$ and $\sigma'(\vec{\gamma})$ contain the same (secret) messages. To prove the latter property, the prover and the verifier run as a subroutine the Σ-protocol for proving equality of committed messages, which we can derive from the (optimized) Σ-protocol for linear relations, as described next.

As usual, this protocol can be turned into a NIZKP for the same relation using the Fiat-Shamir transformation and by repeating it several times to reduce the soundness error.

B. Subroutine

A main component of our proof of shuffle is the Σ-protocol of plaintext equality. We now describe how to realize this subroutine for the LPN-based commitments with the (optimized) Σ-protocol for linear relations.

In fact, the Σ-protocol for linear relations can be used to prove that three commitments $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$ satisfy $\mathbf{c}_i = \mathsf{Com}(\mathbf{m}_i)$ (for all $i \in \{1,2,3\}$) and $\mathbf{X}_1 \cdot \mathbf{m}_1 \oplus \mathbf{X}_2 \cdot \mathbf{m}_2 = \mathbf{m}_3$ for some (mutually known) matrices $\mathbf{X}_1, \mathbf{X}_2 \in \mathcal{I}^{v \times v}$, which define the linear relation. Now, the required Σ-protocol for plaintext equality is an instance of this general linear relation for $\mathbf{X}_1 = \mathbf{1}^{v \times v}$ and $\mathbf{X}_2 = \mathbf{0}^{v \times v}$. We fully describe this Σ-protocol in Fig. 4. Essentially, it consists of running the Σ-protocol of well-formedness (Fig. 2) twice, plus an additional (computationally negligible) extra check.

To generalize this Σ-protocol to commitment vectors, we can simply run it in parallel for all vector entries and merge these parallel runs into a joint one.

Optimized Σ-protocol of plaintext equality

Prover $\mathsf{P}((\mathbf{A}, \mathbf{c}_1, \mathbf{c}_2), (\mathbf{e}_1, \mathbf{e}_2, \mathbf{s}_1, \mathbf{s}_2))$		Verifier $\mathsf{V}(\mathbf{A}, \mathbf{c}_1, \mathbf{c}_2)$
$\mathbf{x} \xleftarrow{R} \mathcal{I}^v$		
foreach i **in**$\{1,2\}$ **do**		
$\pi_i \xleftarrow{R} \mathcal{S}_k, \mathbf{u}_i \xleftarrow{R} \mathcal{I}^\ell, \mathbf{f}_i \xleftarrow{R} \mathcal{I}^k$		
$\mathbf{v}_i \leftarrow \mathbf{u}_i \Vert \mathbf{x}$		
$(c_{i,0}, d_{i,0}) \xleftarrow{R} \mathsf{Com}'(\pi_i, \mathbf{t}_{i,0} = \mathbf{A} \cdot \mathbf{v}_i \oplus \mathbf{f}_i)$		
$(c_{i,1}, d_{i,1}) \xleftarrow{R} \mathsf{Com}'(t_{i,1} = \pi_i(\mathbf{f}_i))$		
$(c_{i,2}, d_{i,2}) \xleftarrow{R} \mathsf{Com}'(t_{i,2} = \pi_i(\mathbf{f}_i \oplus \mathbf{e}_i))$		
$a \leftarrow (c_{i,0}, c_{i,1}, c_{i,2})_{i \in \{1,2\}}$		
	$\xrightarrow{\quad a \quad}$	
		$e \xleftarrow{R} \{0,1,2\}$
	$\xleftarrow{\quad e \quad}$	
if $e = 0$ **then**		
$z \leftarrow (\pi_i, \mathbf{t}_{i,0}, \mathbf{t}_{i,1}, d_{i,0}, d_{i,1}, \mathbf{v}_i)_{i \in \{1,2\}}$		
elseif $e = 1$ **then**		
$\mathbf{v}'_i \leftarrow (\mathbf{u}'_i \Vert \mathbf{x}'_i) = \mathbf{v}_i \oplus \mathbf{s}_i$		
$z \leftarrow (\pi_i, \mathbf{t}_{i,0}, \mathbf{t}_{i,2}, d_{i,0}, d_{i,2}, \mathbf{v}'_i)_{i \in \{1,2\}}$		
elseif $e = 2$ **then**		
$z \leftarrow (\mathbf{t}_{i,1}, \mathbf{t}_{i,2}, d_{i,1}, d_{i,2})_{i \in \{1,2\}}$		
	$\xrightarrow{\quad z \quad}$	
		if $e = 0$ **then**
		foreach i **in**$\{1,2\}$ **do**
		$\mathsf{Ver}'(\mathbf{t}_{i,0}, c_{i,0}, d_{i,0}) \stackrel{?}{=} 1, \mathsf{Ver}'(\mathbf{t}_{i,1}, c_{i,1}, d_{i,1}) \stackrel{?}{=} 1$
		$\mathbf{t}_{i,0} \oplus \pi_i^{-1}(\mathbf{t}_{i,1}) \stackrel{?}{=} \mathbf{A} \cdot \mathbf{v}_i, \pi_i \stackrel{?}{\in} \mathcal{S}_k$
		$\mathbf{x}_1 \stackrel{?}{=} \mathbf{x}_2$ (where $(\mathbf{u}_i \Vert \mathbf{x}_i) = \mathbf{v}_i$)
		if $e = 1$ **then**
		foreach i **in**$\{1,2\}$ **do**
		$\mathsf{Ver}'(\mathbf{t}_{i,0}, c_{i,0}, d_{i,0}) \stackrel{?}{=} 1, \mathsf{Ver}'(\mathbf{t}_{i,2}, c_{i,2}, d_{i,2}) \stackrel{?}{=} 1$
		$\mathbf{t}_{i,0} \oplus \pi_i^{-1}(\mathbf{t}_{i,2}) \oplus \mathbf{c}_i \stackrel{?}{=} \mathbf{A} \cdot \mathbf{v}'_i$
		$\mathbf{x}'_1 \stackrel{?}{=} \mathbf{x}'_2$
		if $e = 2$ **then**
		foreach i **in**$\{1,2\}$ **do**
		$\mathsf{Ver}'(\mathbf{t}_{i,1}, c_{i,1}, d_{i,1}) \stackrel{?}{=} 1, \mathsf{Ver}'(\mathbf{t}_{i,2}, c_{i,2}, d_{i,2}) \stackrel{?}{=} 1$
		$\Vert \mathbf{t}_{i,1} \oplus \mathbf{t}_{i,2} \Vert_1 \stackrel{?}{=} w$

Fig. 4: Σ-protocol for $\mathcal{R} = \{((\mathbf{A}, \mathbf{c}_1, \mathbf{c}_2), (\mathbf{e}_1, \mathbf{e}_2, \mathbf{s}_1, \mathbf{s}_2)) \colon \mathbf{c}_1 = \mathbf{A} \cdot \mathbf{s}_1 \oplus \mathbf{e}_1 \wedge \mathbf{c}_2 = \mathbf{A} \cdot \mathbf{s}_2 \oplus \mathbf{e}_2 \wedge (\exists \mathbf{r}_1, \mathbf{r}_2 \in \mathcal{I}^\ell \ \exists \mathbf{m} \in \mathcal{I}^v \colon \mathbf{s}_1 = (\mathbf{r}_1 \Vert \mathbf{m}) \wedge s_2 = (\mathbf{r}_2 \Vert \mathbf{m}))\}$ (equality of plaintexts).

C. Protocol flow

The full proof of shuffle works as follows (see Fig. 5). The prover takes as input a statement/witness pair $((\vec{c}, \vec{m}), (\sigma, \vec{d}))$ that is in the shuffle relation

$$\mathcal{R} = \{((\vec{c}, \vec{m}), (\sigma, \vec{d})) \colon \mathsf{Ver}'(\sigma(\vec{m}), \vec{c}, \vec{d}) = 1\}$$

and the verifier takes as input the statement $(\vec{c}, \vec{m})$.

In the first step, the prover chooses a random permutation π, and then commits to $\pi(\vec{m})$ as $(\vec{\gamma}, \vec{\delta}) \xleftarrow{R} \mathsf{Com}(\pi(\vec{m}))$. The verifier gets $\vec{\gamma}$ and returns a random bit $e \xleftarrow{R} \{0,1\}$ as a challenge. If the challenge bit is 0, the prover returns $z \leftarrow (\pi, \vec{\delta})$ so that

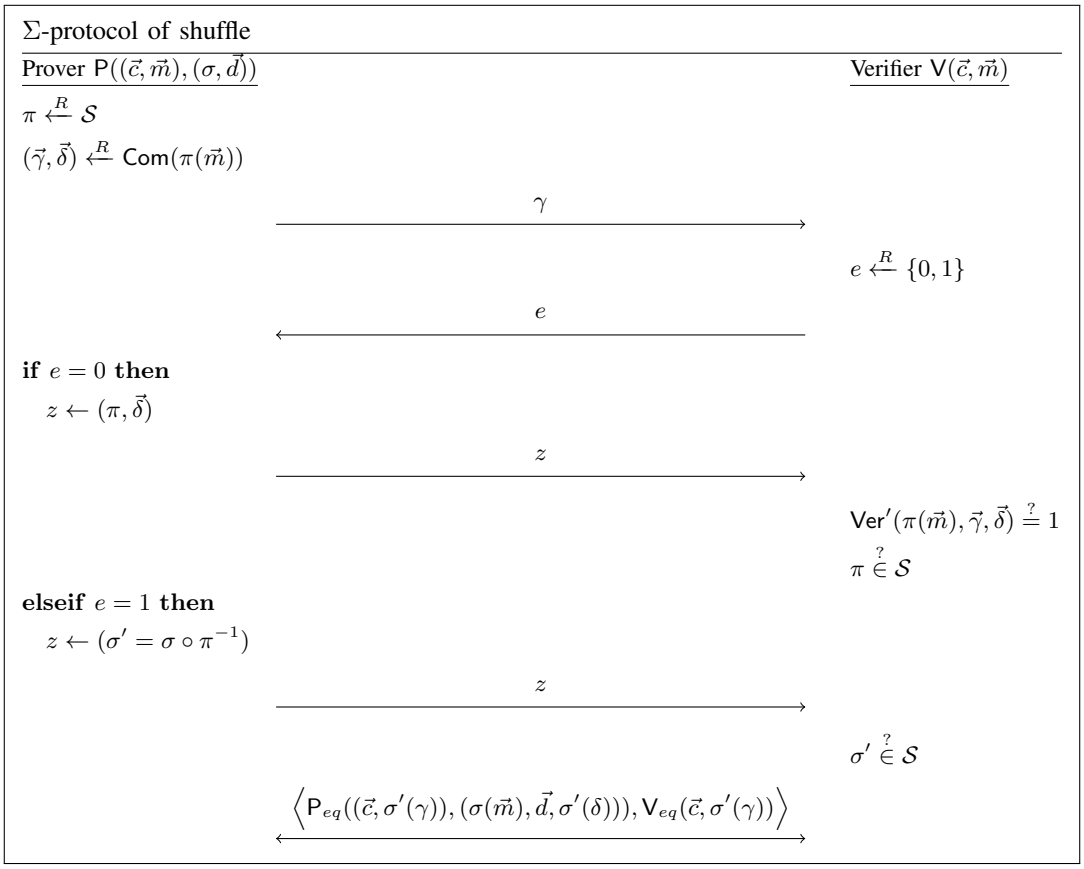

Fig. 5: Interactive proof for $\mathcal{R} = \{((\vec{c},\vec{m}),(\sigma,\vec{d}))\colon \mathsf{Ver}(\sigma(\vec{m}),\vec{c},\vec{d}) = 1\}$ (proof of shuffle). P_{eq} denotes the Σ-protocol of plaintext equality described in Fig. 4.

the verifier can check whether $\mathsf{Ver}'(\pi(\vec{m}),\vec{\gamma},\vec{\delta}) \stackrel{?}{=} 1$. Otherwise, if the challenge bit is 1, the prover returns $\sigma' = \sigma \circ \pi^{-1}$, and then the prover and the verifier run the Σ-protocol of plaintext equality P_{eq}, where the input to the prover is the statement/witness pair $((\vec{c},\sigma'(\vec{\gamma})),(\sigma(\vec{m}),\vec{d},\sigma'(\vec{\delta})))$ and the input to the verifier is the statement $(\vec{c},\sigma'(\vec{\gamma}))$. If the respective check is positive, the verifier returns 1, otherwise it returns 0.

D. Security

We refer to [30] for a formal security analysis, where an abstract version of our proof of shuffle was formally verified (with the cryptographic primitives as black boxes). Since we proved in the previous sections that the commitment scheme and the Σ-protocol of plaintext equality, which we use to instantiate the generic proof of shuffle, provide the required properties, it follows that our code-based instantiation of this protocol achieves the claimed features.

On an intuitive level, the underlying protocol offers special honest verifier zero-knowledge since in each run, either the individual links between the commitment vectors or the individual links between the intermediate commitments and the plaintext messages remain secret, but never both at the same time. The protocol offers special soundness since the prover's openings for both challenges of the same message imply the existence of the respective witness.

E. Efficiency

Since our interactive protocol inherits the soundness error $\frac{2}{3}$ of the underlying Σ-protocol for linear relations, we need to repeat it 28 or 55 times to get soundness error 2^{-16} or 2^{-32}, respectively.

The computational complexity of the proof of shuffle is dominated by computing and verifying the intermediate commitment vector (prover: 0.197 ms, verifier: 0.03 ms, for a single commitment, see Table 4 in [11]) and by running the underlying Σ-protocol of plaintext equality. We further observe that the Σ-protocol of plaintext equality is executed on average in half of the repetitions and that, as explained above, it essentially consists of running two times the Σ-protocol of well-formedness (prover: 1.897 ms, verifier: 0.891 ms, for single commitments, see Section IV). Putting these numbers together, we can estimate that the prover can "process" approximately 17 or 9 messages per second for soundness errors 2^{-16} or 2^{-32}, respectively. With our optimization, the speed of the verification algorithm is about two times better than the prover's, but without it, the verifier would need more than 2 *minutes* for the same numbers of messages instead of 1 *second*.

The size of the transcript of the proof of shuffle is essentially the average between the size of $\vec{\delta}$ and the size of the transcript of the Σ-protocol of plaintext equality. Therefore, the proof

size of the protocol is on average $\frac{1}{2}\cdot(2\cdot 1.265+0.165) = 1.348$ Kilobytes per message. Hence, for soundness errors 2^{-16} or 2^{-32}, we need around 1 Gigabit per 3312 or 1686 shuffled messages, respectively.

VIII. Verifiable E-Voting

We propose the first verifiable e-voting protocol whose security and vote privacy reduce to the hardness of decoding random linear codes.

A. Introduction

1) Overview: A voting protocol is (usually) run between an *election authority* EA, a set of *voters* $\mathsf{V}_1, \ldots, \mathsf{V}_n$, and a *trustee* T (sometimes called *tallier*). The election authority EA is responsible for setting up the election (date, set of candidates, voting method, etc.) and for registering voters. During the submission phase, the voters $\mathsf{V}_1, \ldots, \mathsf{V}_n$ cast their individual votes $v_1, \ldots, v_n$. In the tallying phase, the trustee T then takes these votes as input, applies the specified voting method ρ to these votes, and outputs the election result $\rho(v_1, \ldots, v_n)$; in our case, ρ will be the random shuffle function.

2) Security and privacy properties: It is obvious that, without any further measures, if T is dishonest, it can manipulate the election outcome undetectably. Therefore, *secure* e-voting protocols offer *public verifiability* [19] to ensure that everyone is able to verify that the final election result is correct, even if the trustee or other participants are corrupt. Since the mechanism for verifying the correctness of an election should not reveal how individual voters voted, such e-voting systems also strive for *vote privacy* [12], which guarantees that the links between individual voters and their votes in the public result remain secret.

To combine public verifiability and vote privacy, an additional party, called the *public bulletin board* PBB, is usually employed. The role of PBB is to broadcast all the data necessary to verify the correctness of the final result.

3) Verifiable mixing approach: The e-voting protocol that we design in this paper follows the verifiable mixing approach to combine public verifiability and vote privacy. We now describe the high-level protocol flow of this approach.

During the submission phase, the voters V_i commit to their votes v_i, send their opening values to the trustee over private channels, and post their commitments to PBB. In the tallying phase, the trustee randomly permutes the voters' choices and posts the shuffled votes to PBB. The trustee additionally computes and publishes a proof of shuffle so that everyone can verify that the final result is correct with respect to the voters' commitments, while keeping their opening values and the overall permutation secret.

B. Protocol

1) Cryptographic primitives: We use the following cryptographic primitives:

- The xLPN-based commitment scheme described Sec. III.
- The xLPN-based non-interactive zero-knowledge proof (NIZKP) of knowledge of well-formedness Π_{wf} derived from the optimized Σ-protocol presented in Sec. IV.
- The xLPN-based NIZKP of shuffle Π_{sh} derived from the protocol designed in Sec. VII.
- An (arbitrary) code-based IND-CCA secure public-key encryption scheme $(\mathsf{KeyGen}_{pk}, \mathsf{Enc}, \mathsf{Dec})$ (e.g., based on the KEMs BIKE or HQC).

2) Participants: The protocol is run between the election authority EA, the voters $\mathsf{V}_1, \ldots, \mathsf{V}_n$, the trustee T, and the public bulletin board PBB. We assume that all messages from the election authority, the voters, and the trustee on the public bulletin board are authenticated (e.g., by using code-based digital signatures).

3) Setup phase: The election authority EA determines the main security parameter $\ell \in \mathbb{N}$, the noise parameter $0 < \tau < 0.25$, the message length $v \in \mathbb{N}$, the parameter $k \in \Theta(\ell + v)$, and the weight $w = [\tau k]$; we assume that these parameters are implicit inputs to all algorithms of the commitment scheme. The election authority determines the public key $\mathbf{A} \leftarrow \mathsf{KeyGen}$ of the commitment scheme. Finally, EA posts these parameters on PBB. The trustee T runs the key generation algorithm KeyGen_{pk} of the public-key encryption scheme to generate its public/private (encryption/decryption) key pair (pk, sk) and publishes pk.

4) Submission phase: Voter V_i reads all parameters from PBB and then runs the following program:

1) Pick favorite choice $\mathbf{m} \in \mathcal{I}^v$.
2) Commit to $\mathbf{m}$ as $(\mathbf{c}, (\mathbf{r}, \mathbf{e})) \leftarrow \mathsf{Com}(\mathbf{A}, \mathbf{m})$.
3) Create a NIZKP π_{wf} of well-formedness of $\mathbf{c}$ as $\pi_{wf} \leftarrow \Pi_{wf}(\mathbf{A}, \mathbf{c}, \mathbf{m}, \mathbf{r}, \mathbf{e})$.
4) Encrypt $(\mathbf{m}, \mathbf{c}, \mathbf{r})$ under pk as $e \leftarrow \mathsf{Enc}(pk, (\mathbf{m}, \mathbf{c}, \mathbf{r}))$.
5) Post the ballot $b_i \leftarrow (i, \mathbf{c}, \pi_{wf}, e)$ to PBB.

5) Tallying phase: Trustee T reads all encrypted ballots $(i, \mathbf{c}, \pi_{wf}, e)$ from PBB. First, it discards all ballots that do not have the right format. Second, it discards all ballots with duplicate entries or with invalid proofs of well-formedness.[2] For all remaining ballots, T uses its secret key sk to decrypt the ciphertext e on that ballot and checks whether the resulting plaintext is a valid opening of the commitment of the same ballot; if not, it discards this ballot.

After this filtering step, the trustee T computes a uniformly random permutation σ and applies it to the votes to determine the final result $r \leftarrow (\mathbf{m}_{\sigma(i)})_{i\in[n]}$.[3] Then, T uses the opening values $(\mathbf{r}_i, \mathbf{e}_i)_{i\in[n]}$ of all commitments $(\mathbf{c}_i)_{i\in[n]}$ and the permutation σ to compute a proof of shuffle $\pi_{sh} \leftarrow \Pi_{sh}(((\mathbf{c}_i)_{i\in[n]}, (\mathbf{m}_i)_{i\in[n]}), (\sigma, (\mathbf{r}_i, \mathbf{e}_i)_{i\in[n]}))$. Finally, T posts (r, π_{sh}) to the public bulletin board PBB.

6) Verification phase: Everyone can verify whether the proof of shuffle π_{sh} is correct w.r.t. the committed votes $(\mathbf{c}_i)_{i\in[n]}$ and the final result $r \leftarrow (\mathbf{m}_{\sigma(i)})_{i\in[n]}$. Furthermore, every voter can check whether her submitted ballot b_i is in the input to the tallying phase.

[2] This is to prevent so-called *replay attacks* [37], where a targeted voter's ballot is duplicated in order to amplify its choice in the final election result and thus undermine its vote privacy.

[3] For simplicity of notation, we assume that all ballots were correct.

C. Security

We now state and explain why the protocol designed in Sec. VIII-B provides public verifiability and vote privacy. For a formal security analysis, we refer to Section 5.2 in [6], where an abstract version of our protocol (with the cryptographic primitives as black boxes) has been formally verified. Since we proved in the previous sections that the commitment scheme and the zero-knowledge proofs, which we use to instantiate the voting protocol, provide the required properties, it follows that our code-based voting protocol achieves the claimed features.

1) Verifiability: The protocol provides verifiability under the (general) trust assumption that the public bulletin board PBB is honest, while the voters and the tallier can be malicious, and under the xLPN hardness assumption. This essentially follows from the soundness of the well-formedness proofs of the voters, which guarantees that even a malicious voters cannot create a malformed ballot that is accepted for tallying; and from the soundness of the shuffling proof of the tallier, which ensures that the votes in the final result match the messages in the income commitments (modulo permutation).

2) Vote privacy: The protocol guarantees vote privacy under the (general) trust assumption that the public bulletin board PBB and the tallier are honest, while the voters can be malicious, under the xLPN hardness assumption and under the IND-CCA assumption of the public-key encryption scheme. This follows essentially from the hiding property of the commitment scheme and the secrecy of the public-key encryption scheme, which ensure that no one can learn the votes from the commitments and ciphertexts; from the zero-knowledge feature of the well-formedness proofs, which guarantees that these proofs do not leak any information about the random coins used to commit to the vote; from the special soundness (proof of *knowledge*) of the well-formedness proofs, which ensures that ballots are mutually independent; and from the zero-knowledge feature of the shuffle proof, which keeps all information about the global permutation secret.

D. Performance

1) Voter: Each voter creates a commitment to its vote and a NIZKP of knowledge. For the parameters we chose (see Section IV), the size of the commitment is 0.17 Kilobytes and it takes less than 0.2 ms to compute it, according to Table 4 in [11]. The size of the voter's NIZKP is 35.42 Kilobytes and it takes 28 ms to compute it, according to Table 5 in [11] (for soundness error 2^{-16}). With our optimization, verifying the correctness of a voter's NIZKP can also be done in less than 28 ms, which is practical even for large electorates. The size and the speed of the ciphertext that encrypts the voter's opening values depends, of course, on the specific instantiation of the PKE scheme; this part can also be realized efficiently, for example, with the BIKE or HQC KEMs. In summary, the workload for voters is sufficiently low, even if they are using computers with limited computing power or poor Internet connection.

2) Trustee: The main work for the trustee is to compute the NIZKP of shuffle. Our calculations in Section VII show that the trustee can process about 17 messages per second. We can improve the speed of the trustee by observing that most of the trustee's computations can be done independently of the specific messages being shuffled. First, in the commitment phase of the underlying Σ-protocol of plaintext equality, the permutation and random vectors to hide the secret messages, the first two commitments, and the randomness for the third commitment can be computed in advance. Second, in the commitment phase of the interactive proof of shuffle, the permutation and the randomness for the commitments can be computed in advance.

When a trustee computes this data in an offline phase (which is realistic for e-voting), the performance results in [11] show that speed of tallying process can improved by a factor of about 3. For example, the trustee could create a NIZKP for more than $1.000.000$ voters in less than 6 hours.

Without our improvement, using the original version, it would take more than 1300 hours to verify a NIZKP of this size. However, with our computationally improved verifier, a NIZKP of this size can be verified in less than about 12 hours, making this practically interesting application feasible.

IX. Conclusion

We have improved the understanding and potential of the only conservative code-based ZKP for arbitrary statements. We have shown how to make the verifier of this ZKP as efficient as the prover. We formally verified the security of the resulting optimized ZKP in EasyCrypt. We have shown that it is possible to create a code-based ZKP of shuffle and how to use it to design the first code-based verifiable e-voting protocol.

We see several open challenges for future research. For example, it is interesting to develop new code-based proofs of shuffle that are lighter in terms of both computational power and memory requirements, and to develop a code-based e-voting protocol in which trust in the tallier for vote privacy is distributed.

Acknowledgments

Thomas Haines is the recipient of an Australian Research Council Australian Discovery Early Career Award (project number DE220100595). Rafieh Mosaheb was supported by the Luxembourg National Research Fund (FNR) under the CORE project EquiVox (C19/IS/13643617/EquiVox/Ryan). Johannes Müller was supported by FNR under the CORE Junior project FP2 (C20/IS/14698166/FP2/Mueller). This work benefited from funding managed by the French National Research Agency under the France 2030 programme with the reference ANR-22-PECY-0006.

References

[1] Ben Adida. Helios: Web-based Open-Audit Voting. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348. USENIX Association, 2008.

[2] José Bacelar Almeida, Endre Bangerter, Manuel Barbosa, Stephan Krenn, Ahmad-Reza Sadeghi, and Thomas Schneider. A Certifying Compiler for Zero-Knowledge Proofs of Knowledge Based on Sigma-Protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, volume 6345 of *Lecture Notes in Computer Science*, pages 151–167. Springer, 2010.

[3] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, Antoine Séré, and Pierre-Yves Strub. Formally verifying Kyber Episode IV: Implementation correctness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):164–193, 2023.

[4] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 1467–1481. ACM, 2023.

[5] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Lattice-Based Proof of Shuffle and Applications to Electronic Voting. In Kenneth G. Paterson, editor, *Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12704 of *Lecture Notes in Computer Science*, pages 227–251. Springer, 2021.

[6] Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical Everlasting Privacy. In David A. Basin and John C. Mitchell, editors, *Principles of Security and Trust - Second International Conference, POST 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7796 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2013.

[7] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 358–389. Springer, 2023.

[8] Manuel Barbosa, Gilles Barthe, Xiong Fan, Benjamin Grégoire, Shih-Han Hung, Jonathan Katz, Pierre-Yves Strub, Xiaodi Wu, and Li Zhou. Easypqc: Verifying post-quantum cryptography. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2564–2586. ACM, 2021.

[9] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. Revisiting algebraic attacks on minrank and on the rank decoding problem. *Des. Codes Cryptogr.*, 91(11):3671–3707, 2023.

[10] Gilles Barthe, Daniel Hedin, Santiago Zanella Béguelin, Benjamin Grégoire, and Sylvain Heraud. A Machine-Checked Formalization of Sigma-Protocols. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 246–260. IEEE Computer Society, 2010.

[11] Emanuele Bellini, Philippe Gaborit, Alexandros Hasikos, and Víctor Mateu. Enhancing Code Based Zero-Knowledge Proofs Using Rank Metric. In Stephan Krenn, Haya Schulmann, and Serge Vaudenay, editors, *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings*, volume 12579 of *Lecture Notes in Computer Science*, pages 570–592. Springer, 2020.

[12] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 499–516. IEEE Computer Society, 2015.

[13] Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Yann Connan, and Philippe Gaborit. A gapless code-based hash proof system based on RQC and its applications. *Des. Codes Cryptogr.*, 90(12):3011–3044, 2022.

[14] Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Nicolas Sendrier. Quasi-Cyclic Stern Proof of Knowledge. In *IEEE International Symposium on Information Theory, ISIT 2022, Espoo, Finland, June 26 - July 1, 2022*, pages 1459–1464. IEEE, 2022.

[15] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[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 Marc Fischlin and Jean-Sébastien Coron, editors, *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*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016.

[17] Xavier Boyen, Thomas Haines, and Johannes Müller. A Verifiable and Practical Lattice-Based Decryption Mix Net with External Auditing. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 336–356. Springer, 2020.

[18] Xavier Boyen, Thomas Haines, and Johannes Müller. Epoque: Practical End-to-End Verifiable Post-Quantum-Secure E-Voting. In *IEEE European Symposium on Security and Privacy, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, pages 272–291. IEEE, 2021.

[19] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. SoK: Verifiability Notions for E-Voting Protocols. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 779–798. IEEE Computer Society, 2016.

[20] Núria Costa, Ramiro Martínez, and Paz Morillo. Proof of a Shuffle for Lattice-Based Cryptography. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevicius, editors, *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, volume 10674 of *Lecture Notes in Computer Science*, pages 280–296. Springer, 2017.

[21] Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-Based Proof of a Shuffle. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala, editors, *Financial Cryptography and Data Security - FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11599 of *Lecture Notes in Computer Science*, pages 330–346. Springer, 2019.

[22] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest Verifier vs Dishonest Verifier in Public Coin Zero-Knowledge Proofs. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 325–338. Springer, 1995.

[23] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical Quantum-Safe Voting from Lattices. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1565–1581. ACM, 2017.

[24] Xuan-Thanh Do, Dang Truong Mac, and Quoc-Huy Vu. zk-SNARKs from Codes with Rank Metrics. In Elizabeth A. Quaglia, editor, *Cryptography and Coding - 19th IMA International Conference, IMACC 2023, London, UK, December 12-14, 2023, Proceedings*, volume 14421 of *Lecture Notes in Computer Science*, pages 99–119. Springer, 2023.

[25] Denis Firsov and Dominique Unruh. Zero-Knowledge in EasyCrypt. In *36th IEEE Computer Security Foundations Symposium, CSF 2023, Dubrovnik, Croatia, July 10-14, 2023*, pages 1–16. IEEE, 2023.

[26] Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter B. Rønne, and Tjerand Silde. Verifiable Decryption in the Head. In Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo, editors, *Information Security and Privacy - 27th Australasian Conference, ACISP 2022, Wollongong, NSW, Australia, November 28-30, 2022, Proceedings*, volume 13494 of *Lecture Notes in Computer Science*, pages 355–374. Springer, 2022.

[27] Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup. *Cryptogr.*, 6(1):5, 2022.

[28] Thomas Haines, Rajeev Goré, and Mukesh Tiwari. Machine-checking Multi-Round Proofs of Shuffle: Terelius-Wikstrom and Bayer-Groth. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 6471–6488. USENIX Association, 2023.

[29] Thomas Haines and Johannes Müller. SoK: Techniques for Verifiable Mix Nets. In *33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*, pages 49–64. IEEE, 2020.

[30] Thomas Haines and Johannes Müller. A Novel Proof of Shuffle: Exponentially Secure Cut-and-Choose. In Joonsang Baek and Sushmita Ruj, editors, *Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings*, volume 13083 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2021.

[31] Javier Herranz, Ramiro Martínez, and Manuel Sánchez. Shorter Lattice-Based Zero-Knowledge Proofs for the Correctness of a Shuffle. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops - CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers*, volume 12676 of *Lecture Notes in Computer Science*, pages 315–329. Springer, 2021.

[32] Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. Concrete NTRU Security and Advances in Practical Lattice-Based Electronic Voting. *IACR Cryptol. ePrint Arch.*, page 933, 2023.

[33] Rong Hu, Kirill Morozov, and Tsuyoshi Takagi. Proof of plaintext knowledge for code-based public-key encryption revisited. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 535–540. ACM, 2013.

[34] Andreas Hülsing, Matthias Meijers, and Pierre-Yves Strub. Formal Verification of Saber's Public-Key Encryption Scheme in EasyCrypt. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 622–653. Springer, 2022.

[35] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and Efficient Zero-Knowledge Proofs from Learning Parity with Noise. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 663–680. Springer, 2012.

[36] Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The Return of the SDitH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 564–596. Springer, 2023.

[37] David Mestel, Johannes Müller, and Pascal Reisert. How efficient are replay attacks against vote privacy? A formal quantitative analysis. *J. Comput. Secur.*, 31(5):421–467, 2023.

[38] Miguel Ángel Prada-Delgado, Iluminada Baturone, Gero Dittmann, Jens Jelitto, and Andreas Kind. PUF-derived IoT identities in a zero-knowledge protocol for blockchain. *Internet Things*, 9:100057, 2020.

[39] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer, 1995.

[40] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[41] Tjerand Silde. Short Paper: Verifiable Decryption for BGV. In Shin'ichiro Matsuo, Lewis Gudgeon, Ariah Klages-Mundt, Daniel Perez Hernandez, Sam Werner, Thomas Haines, Aleksander Essex, Andrea Bracciali, and Massimiliano Sala, editors, *Financial Cryptography and Data Security. FC 2022 International Workshops - CoDecFin, DeFi, Voting, WTSC, Grenada, May 6, 2022, Revised Selected Papers*, volume 13412 of *Lecture Notes in Computer Science*, pages 381–390. Springer, 2022.

[42] Jacques Stern. A New Identification Scheme Based on Syndrome Decoding. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.

APPENDIX

A. Commitment schemes

Definition 4 (Commitment scheme): A *commitment scheme* is a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Ver})$ that satisfies the following properties:

- The *key generation* algorithm KeyGen takes as input 1^ℓ and outputs a public *commitment key* pk.
- The *commitment* algorithm Com takes as input a message m from a message space $\mathcal{M}$ and a commitment key pk, and outputs a *commitment/opening pair* (c, d).
- The *verification* algorithm Ver takes as input a key pk, a message m, a commitment c and an opening d, and output 1 or 0.

The Σ-protocols we study in this work satisfy the following properties:

- *Correctness:*

$$Pr[\mathsf{Ver}(pk, m, c, d) = 1; pk \xleftarrow{R} \mathsf{KeyGen}(1^\ell), \\ m \in \mathcal{M}, (c, d) \xleftarrow{R} \mathsf{Com}(m, pk)] = 1$$

- *Perfectly binding:* For all $pk \in \mathsf{KeyGen}$, $m, m' \in \mathcal{M}$ and all c, d, d', if $\mathsf{Ver}(pk, m, c, d) = 1$ and $\mathsf{Ver}(pk, m', c, d') = 1$, then $m = m'$.
- *Computationally hiding:* With overwhelming probability over the choice of $pk \xleftarrow{R} \mathsf{KeyGen}(1^\ell)$, for every $m, m' \in \mathcal{M}$ and $(c, d) \xleftarrow{R} \mathsf{Com}(m, pk)$, $(c', d') \xleftarrow{R} \mathsf{Com}(m', pk)$, the distributions of c and c' are computationally indistinguishable.

B. Σ-protocols

Definition 5 (Σ-protocol): Let (P, V) be a pair of connected ppt Turing machines. Let $\mathcal{R}$ be a binary relation. Then (P, V) is *Σ-protocol for relation $\mathcal{R}$* with challenge set $\mathcal{E}$ if the following conditions are satisfied:

1) *Form*: The protocol is of the following form:
   a) P creates a commitment a and sends it to V.
   b) V draws a challenge $e \xleftarrow{R} \mathcal{E}$ and sends it to P.
   c) P sends a response z to V.
   d) V returns 0 or 1.

   A protocol transcript (a, e, z) is called *accepting* if V returns 1.
2) *Completeness*: For all $(x, w) \in \mathcal{R}$, the verifier in $\langle \mathsf{P}(x, w), \mathsf{V}(x) \rangle$ returns 1.
3) *Special soundness*: There exists a ppt algorithm E (the *knowledge extractor*) that takes accepting transcripts

$\{(a, e, z_e) : e \in \mathcal{E}\}$ with the same commitment a as input, and outputs w' such that $(x, w') \in \mathcal{R}$.

4) *Special honest-verifier zero-knowledge*: There exists a ppt algorithm S (the *simulator*) that takes x and $e \in \mathcal{E}$ as inputs, and that outputs triples (a, e, z) whose distribution is (computationally) indistinguishable from accepting protocol transcripts generated by real protocol runs.

C. EasyCrypt Axioms

When discussing axioms it is necessary to distinguish between axioms on classes of things, which become proof obligations when the abstract is replaced with the specific, and directly assumed axioms. In the list below, we mention those axioms in the second class. We do, however, rely on classes of things such as algebraic structures and existence of perfectly correct and binding commitment schemes.

Our axioms are included across the various files as they become necessary; we briefly summarize the axioms here:

a) Int axioms: Our results are defined over integer parameters k, l, v and w which we assume are all greater than zero, or in the case of w greater than or equal to.

b) Distribution axioms: We require two distributions, one over the field and one distribution over vectors of low weight. We use axioms to ensure these distributions are lossless and uniform, and for the distribution over vectors returns vectors of the expected length and weight.

c) Islossless axioms: We assume that all our adversaries always terminate when called.

d) Ring axioms: We assume the underlying ring has two elements.

e) Linear code generator axioms: We axiomatize the chance that the generator matrix has distance greater than $2w$.

f) Mapping from int lists to vectors: We introduce abstract operators to map from lists of integers to vectors and from vectors to lists of integers. We axiomitize the injectivity of lists of integers to vectors.

g) Multiplicative Relation ZKP axioms: To aid the run time of EasyCrypt in the checking the Multiplicative Relation ZKP we introduce a number of abstract operators alongside a single axiom which defines the operator. We could just as easily remove the axiom and define the operators as the axioms does but this would change the behavior of the simplify tactic and slow down the proof checking.

h) Multiplicative Relation Matrix axioms: In section 4.3 of [35] they discuss drawing a matrix of size $(v \times 4v)$ of full rank such that $R.\tilde{m}_i = m_i$. We introduce this as an abstract distribution with the properties axiomitized. A matrix is drawn from a ostensibly different distribution by the simulator which we axiomitize as `R_sample2`, our axiom `R_sample_equiv` says that the two distribution are identical since they are both uniformly random matrices of full rank with the specified restriction on the weight of its rows. Removing these axioms would strengthen our result but the linear algebra involved is largely orthogonal to the reasoning in the sigma protocols themselves.