Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability

Jonathan Bootle[1](✉), Andrea Cerulli[1], Essam Ghadafi[2], Jens Groth[1], Mohammad Hajiabadi[3], and Sune K. Jakobsen[1]

[1] University College London, London, UK
{jonathan.bootle.14,andrea.cerulli.13, j.groth,s.jakobsen}@ucl.ac.uk
[2] University of the West of England, Bristol, UK
essam.ghadafi@uwe.ac.uk
[3] University of California, Berkeley, CA, USA
mdhajiabadi@berkeley.edu

Abstract. We give computationally efficient zero-knowledge proofs of knowledge for arithmetic circuit satisfiability over a large field. For a circuit with N addition and multiplication gates, the prover only uses $\mathcal{O}(N)$ multiplications and the verifier only uses $\mathcal{O}(N)$ additions in the field. If the commitments we use are statistically binding, our zero-knowledge proofs have unconditional soundness, while if the commitments are statistically hiding we get computational soundness. Our zero-knowledge proofs also have sub-linear communication if the commitment scheme is compact.

Our construction proceeds in three steps. First, we give a zero-knowledge proof for arithmetic circuit satisfiability in an ideal linear commitment model where the prover may commit to secret vectors of field elements, and the verifier can receive certified linear combinations of those vectors. Second, we show that the ideal linear commitment proof can be instantiated using error-correcting codes and non-interactive commitments. Finally, by choosing efficient instantiations of the primitives we obtain linear-time zero-knowledge proofs.

Keywords: Zero-knowledge · Arithmetic circuit · Ideal linear commitments

1 Introduction

A zero-knowledge proof [GMR85] is a protocol between two parties: a prover and a verifier. The prover wants to convince the verifier that an instance u belongs to a specific language $\mathcal{L}_{\mathcal{R}}$ in NP. She has a witness w such that (u, w) belongs

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007–2013)/ERC Grant Agreement no. 307937.

E. Ghadafi and M. Hajiabadi—Part of the work was done while at University College London.

T. Takagi and T. Peyrin (Eds.): ASIACRYPT 2017, Part III, LNCS 10626, pp. 336–365, 2017.
https://doi.org/10.1007/978-3-319-70700-6_12

to the NP relation $\mathcal{R}$ defining the language, but wants to convince the verifier that the statement $u \in \mathcal{L}_{\mathcal{R}}$ is true without revealing the witness or any other confidential information.

Zero-knowledge proofs are widely used in cryptography since it is often useful to verify that a party is following a protocol without requiring her to divulge secret keys or other private information. Applications range from digital signatures and public-key encryption to secure multi-party computation and verifiable cloud computing.

Efficiency is crucial for large and complex statements such as those that may arise in the latter applications. Important efficiency parameters include the time complexity of the prover, the time complexity of the verifier, the amount of communication measured in bits, and the number of rounds the prover and verifier need to interact. Three decades of research on zero-knowledge proofs have gone into optimizing these efficiency parameters and many insights have been learned.

For zero-knowledge proofs with unconditional soundness where it impossible for any cheating prover to convince the verifier of a false statement, it is possible to reduce communication to the witness size [IKOS09, KR08, GGI+14]. For zero-knowledge arguments where it is just computationally intractable for the prover to cheat the verifier we can do even better and get sub-linear communication complexity [Kil92].

There are many constant-round zero-knowledge proofs and arguments, for instance Bellare et al. [BJY97] construct four round arguments based on one-way functions. In the common reference string model, it is even possible to give non-interactive proofs where the prover computes a convincing zero-knowledge proof directly without receiving any messages from the verifier [BFM88].

The verifier computation is in general at least proportional to the instance size because the verifier must read the entire instance in order to verify it. However, the verifier computation can be sub-linear in the time it takes for the relation to verify a witness for the statement being true [GKR08], which is useful in both zero-knowledge proofs and verifiable computation.

Having reduced the cost of many other efficiency parameters, today the major bottleneck is the prover's computation. Classical number-theoretic constructions for circuit satisfiability such as [CD98] require a linear number of exponentiations, i.e., the cost is $\mathcal{O}(\lambda N)$ group multiplications where N is the number of gates and λ is a security parameter. Relying on different techniques and underlying cryptography [DIK10] has reduced the computational overhead further to being $\mathcal{O}(\log(\lambda))$. This leaves a tantalizing open question of whether we can come all the way down to constant overhead $\mathcal{O}(1)$, i.e., make the prover's cost within a constant factor of the time it takes to verify $(u, w) \in R$ directly.

1.1 Our Contributions

We construct zero-knowledge proofs of knowledge for the satisfiability of arithmetic circuits. An instance is an arithmetic circuits with N fan-in 2 addition and multiplication gates over a finite field $\mathbb{F}$ and a specification of the values of some of the wires. A witness consists of the remaining wires such that the values are consistent with the gates and the wire values specified in the instance.

Our zero-knowledge proofs are highly efficient asymptotically:

– Prover time is $\mathcal{O}(N)$ field additions and multiplications.
– Verifier time is $\mathcal{O}(N)$ field *additions*.

This is optimal up to a constant factor for both the prover and verifier. The prover only incurs a constant overhead compared to the time needed to evaluate the circuit from scratch given an instance and a witness, and for instances of size equivalent to $\Omega(N)$ field elements the verifier only incurs a constant overhead compared to the time it takes to read the instance. The constants are large, so we do not recommend implementing the zero-knowledge proof as it is, but from a theoretical perspective we consider it a big step forward to get constant overhead for both prover and verifier.

Our zero-knowledge proofs have perfect completeness, i.e., when the prover knows a satisfactory witness she is always able to convince the verifier. Our constructions are proofs of knowledge, that is, not only does the prover demonstrate the statement is true but also that she knows a witness. The proofs have special honest-verifier zero-knowledge, which means that given a set of verifier challenges it is possible to simulate a proof answering the challenges without knowing a witness. The flavour of knowledge soundness and special honest-verifier zero-knowledge depends on the underlying commitment scheme we use. When instantiated with statistically binding commitment schemes, we obtain proofs (statistically knowledge sound) with computational zero-knowledge. When we use statistically hiding commitments we obtain arguments of knowledge with statistical special honest verifier zero-knowledge. The communication complexity of our proofs with unconditional soundness is only $\mathcal{O}(N)$ field elements, while our arguments with computational soundness have sub-linear communication of $\text{poly}(\lambda)\sqrt{N}$ field elements when the commitments are compact. Round complexity is also low, when we optimize for computational efficiency for prover and verifier we only use $\mathcal{O}(\log\log N)$ rounds.

1.2 Construction and Techniques

Our construction is modular and consists of three steps. First, we construct a proof in a communication model we call the Ideal Linear Commitment (ILC) channel. In the ILC model, the prover can commit vectors of secret field elements to the channel. The verifier may later query openings to linear combinations of the committed vectors, which the channel will answer directly. We show that idealizing the techniques by Groth et al. [Gro09, BCC+16] gives us efficient proofs in the ideal linear commitment model. By optimizing primarily for prover computation and secondarily for round efficiency, we get a round complexity of $\mathcal{O}(\log\log N)$ rounds, which is better than the $\mathcal{O}(\log N)$ rounds of Bootle et al. [BCC+16] that optimized for communication complexity.

Next, we compile proofs in the ILC model into proof and argument systems using non-interactive commitment schemes; however, unlike previous works we do not commit directly to the vectors. Instead, we encode the vectors as randomized codewords using a linear error-correcting code. We now consider the codewords as rows of a matrix and commit to the columns of that matrix. When the verifier asks for a linear combination of the vectors, the prover simply tells the verifier what the linear combination is. However, the verifier does not have to have blind confidence in the prover because she can ask for openings of some of the committed columns and use them to spot check that the resulting codeword is correct.

Finally, we instantiate the scheme with concrete error-correcting codes and non-interactive commitment schemes. We use the error-correcting codes of Druk and Ishai [DI14], which allow the encoding of k field elements using $\mathcal{O}(k)$ additions in the field. Statistically hiding commitment schemes can be constructed from collision-resistant hash functions, and using the recent hash functions of Applebaum et al. [AHI+17] we can hash t field elements at a cost equivalent to $\mathcal{O}(t)$ field additions. Statistically binding commitment schemes on the other hand can be built from pseudorandom number generators. Using the linear-time computable pseudorandom number generators of Ishai et al. [IKOS08] we get linear-time computable statistically binding commitments. Plugging either of the commitment schemes into our construction yields zero-knowledge proofs with linear-time computation for both prover and verifier.

1.3 Related Work

There is a rich body of research on zero-knowledge proofs. Early practical zero-knowledge proofs such as Schnorr [Sch91] and Guillou-Quisquater [GQ88] used number-theoretic assumptions. There have been several works extending these results to prove more general statements [CDS94,CD98,Gro09,BCC+16] with the latter giving discrete-logarithm based arguments for arithmetic circuit satisfiability with logarithmic communication complexity and a linear number of exponentiations for the prover, i.e., a computation cost of $\mathcal{O}(\lambda N)$ group multiplications for λ-bit exponents and a circuit with N multiplication gates.

Ishai et al. [IKOS09] showed how to use secure multi-party computation (MPC) protocols to construct zero-knowledge proofs. The intuition behind this generic construction is that the prover first executes in *her head* an MPC protocol for computing a circuit verifying some relation R and then commits to the views of all the virtual parties. The verifier asks the prover to open a subset of those views and then verifies their correctness and consistency with each other. Soundness and zero-knowledge follow from robustness and privacy of the MPC protocol. Applying this framework to efficient MPCs gives asymptotically efficient zero-knowledge proofs. For example, the perfectly secure MPC of [DI06] is used in [IKOS09] to obtain zero-knowledge proofs for the satisfiability of Boolean circuits with communication linear in the circuit size, $\mathcal{O}(N)$, and a computational cost of $\Omega(\lambda N)$, for circuits of size N and security parameter λ. Damgård et al. [DIK10] used the MPC framework to construct zero-knowledge proofs for the satisfiability of arithmetic circuits. Their construction has more balanced efficiency and achieves $\mathcal{O}(\text{polylog}(\lambda)N)$ complexity for both computation and communication.

Jawurek et al. [JKO13] gave a very different approach to building zero-knowledge proofs based on garbled circuits. Their approach proved [FNO15, CGM16] to be very efficient in practice for constructing proofs for languages represented as Boolean circuits. These techniques are appealing for proving small statements as they require only a constant number of symmetric-key operations per gate, while the main bottleneck is in their communication complexity. Asymptotically, this approach yields computational and communication complexity of $\mathcal{O}(\lambda N)$ bit operations and bits, respectively, when λ is the cost of a single symmetric-key operation. Recently, these techniques found applications in zero-knowledge proofs for checking the execution of RAM programs [HMR15, MRS17]. For instances that can be represented as RAM programs terminating in T steps and using memory of size M, these zero-knowledge proofs yield communication and computation with $\text{polylog}(M)$ overhead compared to the running time T of the RAM program.

Cramer et al. [CDP12] introduce zero-knowledge proofs for verifying multiplicative relations of committed values using techniques related to ours. When applied to zero-knowledge proofs for the satisfiability of Boolean circuits, the asymptotic communication and computation complexities of [CDP12] are close to [IKOS09], although the constants are smaller. Unlike [CDP12], we do not require any homomorphic property from the commitment scheme, and instead of relying on linear secret sharing schemes with product reconstruction, we use linear error-correcting codes.

In past years, a lot of attention has been dedicated to the study of succinct non-interactive arguments of knowledge (SNARKs) [Gro10, BCCT12, GGPR13, BCCT13, PHGR13, BCG+13, Gro16]. These are very compact arguments offering very efficient verification time. In the most efficient cases, the arguments consist of only a constant number of group elements and verification consists of a constant number of pairings and a number of group exponentiations that is linear in the instance size but independent of the witness size. The main bottleneck of these arguments is the computational complexity of the prover which requires $\mathcal{O}(N)$ group exponentiations.

Recently, Ben-Sasson et al. [BSCS16] proposed the notion of interactive oracle proofs (IOPs), which are interactive protocols where the prover may send a probabilisticaly checkable proof (PCP) in each round. Ben-Sasson et al. [BSCG+16] construct a 3-round public-coin IOP (with soundness error 1/2) for Boolean circuit satisfiability with linear proof length and quasi-linear running times for both the prover and the verifier. Moreover, the constructed IOP has constant query complexity (the number of opening queries requested by the verifier), while prior PCP constructions require sub-linear query complexity. Another follow-up work by Ben-Sasson et al. [BSCGV16] gives 2-round zero-knowledge IOPs (duplex PCPs) for any language in $\text{NTIME}(T(n))$ with quasi-linear prover computation in $n + T(n)$.

Efficiency Comparison. All the proofs we list above have super-linear cost for the prover. This means our zero-knowledge proofs are the most efficient zero-knowledge proofs for arithmetic circuits for the prover. We also know that our verification time is optimal for an instance of size $\Omega(N)$ field elements since the verification time is comparable to the time it takes just to read the instance.

Another well-studied class of languages is Boolean circuit satisfiability but here our techniques do not fare as well since there would be an overhead in representing bits as field elements. We therefore want to make clear that our claim of high efficiency and a significant performance improvement over the state of the art relates only to arithmetic circuits. Nevertheless, we find the linear cost for arithmetic circuits a significant result in itself. This is the first time for any general class of NP-complete language that true linear cost is achieved for the prover when compared to the time it takes to evaluate the statement directly given the prover's witness.

2 Preliminaries

2.1 Notation and Computational Model

We write $y \leftarrow A(x)$ for an algorithm outputting y on input x. When the algorithm is randomized, and we wish to explicitly refer to a particular choice of random coins r chosen by the algorithm, we write $y \leftarrow A(x; r)$. We write PPT/DPT for algorithms running in probabilistic polynomial time and deterministic polynomial time in the size of their inputs. Typically, the size of inputs and output will be polynomial in a *security parameter* λ, with the intention that larger λ means better security. For functions $f, g : \mathbb{N} \rightarrow [0, 1]$, we write $f(\lambda) \approx g(\lambda)$ if $|f(\lambda) - g(\lambda)| = \frac{1}{\lambda^{\omega(1)}}$. We say a function f is *overwhelming* if $f(\lambda) \approx 1$ and f is *negligible* if $f(\lambda) \approx 0$.

Throughout the paper, we will be working over a finite field $\mathbb{F}$. To get negligible risk of an adversary breaking our zero-knowledge proofs, we need the field to be large enough such that $\log |\mathbb{F}| = \omega(\lambda)$. When considering efficiency of our zero-knowledge proofs, we will assume the prover and verifier are RAM machines where operations on W-bit words have unit cost. We assume a field element is represented by $\mathcal{O}(\frac{\log |\mathbb{F}|}{W})$ words and that additions in $\mathbb{F}$ carry a cost of $\mathcal{O}\left(\frac{\log |\mathbb{F}|}{W}\right)$ machine operations. We expect multiplications to be efficiently computable as well but at a higher cost of $\omega\left(\frac{\log |\mathbb{F}|}{W}\right)$ machine operations.

For a positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$. We use bold letters such as $\boldsymbol{v}$ for row vectors. For $\boldsymbol{v} \in \mathbb{F}^n$ and a set $J = \{j_1, \ldots, j_k\} \subset [n]$ with $j_1 < \cdots < j_k$ we define the vector $\boldsymbol{v}|_J$ to be $(\boldsymbol{v}_{j_1}, \ldots, \boldsymbol{v}_{j_k})$. Similarly, for a matrix $V \in \mathbb{F}^{m \times n}$ we let $V|_J \in \mathbb{F}^{m \times k}$ be the submatrix of V restricted to the columns indicated in J.

2.2 Proofs of Knowledge

A *proof system* is defined by a triple of stateful PPT algorithms $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, which we call the setup *generator*, the *prover* and *verifier*, respectively. The setup generator $\mathcal{K}$ creates public parameters pp that will be used by the prover and the verifier. We think of pp as being honestly generated, however, in the proofs we construct it consists of parts that are either publicly verifiable or could be generated by the verifier, so we use the public parameter model purely for simplicity and efficiency of our proofs, not for security.

The prover and verifier communicate with each other through a *communication channel* $\overset{\text{chan}}{\longleftrightarrow}$. When $\mathcal{P}$ and $\mathcal{V}$ interact on inputs s and t through a communication channel $\overset{\text{chan}}{\longleftrightarrow}$ we let $\mathsf{view}_{\mathcal{V}} \leftarrow \langle \mathcal{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathcal{V}(t)\rangle$ be the view of the verifier in the execution, i.e., all inputs he gets including random coins and let $\mathsf{trans}_{\mathcal{P}} \leftarrow \langle \mathcal{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathcal{V}(t)\rangle$ denote the transcript of the communication between prover and channel. This overloads the notation $\leftarrow \langle \mathcal{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathcal{V}(t)\rangle$ but it will always be clear from the variable name if we get the verifier's view or the prover's transcript. At the end of the interaction the verifier accepts or rejects. We write $\langle \mathcal{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathcal{V}(t)\rangle = b$ depending on whether the verifier rejects ($b = 0$) or accepts ($b = 1$).

In the *standard channel* $\longleftrightarrow$, all messages are forwarded between prover and verifier. We also consider an *ideal linear commitment* channel, $\overset{\mathsf{ILC}}{\longleftrightarrow}$, or simply ILC, described in Fig. 1. When using the ILC channel, the prover can submit a `commit` command to commit to vectors of field elements of some fixed length k, specified in pp_{ILC}. The vectors remain secretly stored in the channel, and will not be forwarded to the verifier. Instead, the verifier only learns how many vectors the prover has committed to. The verifier can submit a `send` command to the ILC to send field elements to the prover. In addition, the verifier can also submit `open` queries to the ILC for obtaining the opening of any linear combinations of

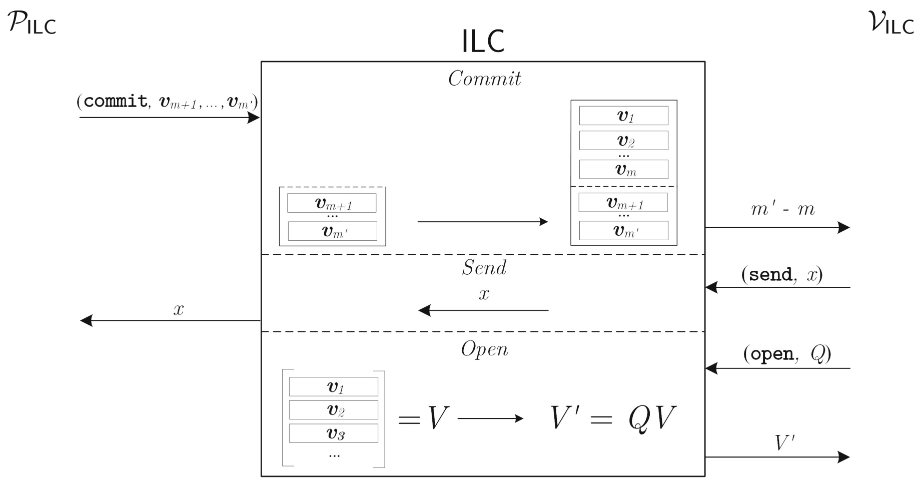

Fig. 1. Description of the ILC channel.

the vectors sent by the prover. We stress that the verifier can request several linear combinations within a single `open` query, as depicted in Fig. 1.

In a proof system over the ILC channel, sequences of `commit`, `send` and `open` queries could alternate in an arbitrary order. We call a proof system over the ILC channel *non-adaptive* if the verifier only makes one *open* query to the ILC channel before terminating his interaction with the channel, otherwise we call it *adaptive*. Although adaptive proof systems are allowed by the model, in this paper we will only consider non-adaptive ILC proof systems to simplify the exposition.

We remark that ILC proof systems are different from linear interactive proofs considered in [BCI+13]. In linear interactive proofs both the prover and verifier send vectors of field elements, but the prover can only send linear (or affine) transformations of the verifier's previously sent vectors. However, for our constructions it is important that the prover can compute on field elements received by the verifier and for instance evaluate polynomials.

We say a proof system is *public coin* if the verifier's messages to the communication channel are chosen uniformly at random and independently of the actions of the prover, i.e., the verifier's messages to the prover correspond to the verifier's randomness ρ.

We will consider relations $\mathcal{R}$ consisting of tuples (pp, u, w), and define $\mathcal{L}_{\mathcal{R}} = \{(pp, u) | \exists w : (pp, u, w) \in \mathcal{R}\}$. We refer to u as the *instance* and w as the *witness* that $(pp, u) \in \mathcal{L}_{\mathcal{R}}$. The *public parameter* pp will specify the security parameter λ, perhaps implicitly through its length, and may also contain other parameters used for specifying the specific relation, e.g. a description of a field. Typically, pp will also contain parameters that do not influence membership of $\mathcal{R}$ but may aid the prover and verifier, for instance, a description of an encoding function that they will use.

We will construct SHVZK proofs of knowledge for the relation $\mathcal{R}_{\mathsf{AC}}$, where the instances are arithmetic circuits over a field $\mathbb{F}$ specified by pp. An instance consists of many fan-in 2 addition and multiplication gates over $\mathbb{F}$, a description of how wires in the circuit connect to the gates, and values assigned to some of the input wires. Witnesses w are the remaining inputs such that the output of the circuit is 0. For an exact definition of how we represent an arithmetic circuit, see Sect. 3. We would like to stress the fact that the wiring of the circuit is part of the instance and we allow a fully adaptive choice of the arithmetic circuit. This stands in contrast to pairing-based SNARKs that usually only consider circuits with fixed wires, i.e., the arithmetic circuit is partially non-adaptive, and getting full adaptivity through a universal circuit incurs an extra efficiency overhead.

The protocol $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is called a *proof of knowledge* over communication channel $\overset{\text{chan}}{\longleftrightarrow}$ for relation $\mathcal{R}$ if it has perfect completeness and computational knowledge soundness as defined below.

Definition 1 (Perfect Completeness). *The proof is* perfectly complete *if for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathcal{K}(1^\lambda); (u, w) \leftarrow \mathcal{A}(pp) : \\ (pp, u, w) \notin \mathcal{R} \ \vee \ \langle \mathcal{P}(pp, u, w) \overset{chan}{\longleftrightarrow} \mathcal{V}(pp, u)\rangle = 1 \end{array}\right] = 1.$$

Definition 2 (Knowledge soundness). *A public-coin proof system has* computational (strong black-box) knowledge soundness *if for all DPT $\mathcal{P}^*$ there exists an expected PPT extractor $\mathcal{E}$ such that for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathcal{K}(1^\lambda); (u,s) \leftarrow \mathcal{A}(pp); w \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(s) \overset{chan}{\longleftrightarrow} \mathcal{V}(pp,u)\rangle}(pp,u) : \\ b = 1 \ \wedge \ (pp,u,w) \notin \mathcal{R} \end{array}\right] \approx 0.$$

Here the oracle $\langle \mathcal{P}^*(s) \overset{chan}{\longleftrightarrow} \mathcal{V}(pp,u)\rangle$ runs a full protocol execution and if the proof is successful it returns a transcript of the prover's communication with the channel. The extractor $\mathcal{E}$ can ask the oracle to rewind the proof to any point in a previous transcript and execute the proof again from this point on with fresh public-coin challenges from the verifier. We define $b \in \{0,1\}$ to be the verifier's output in the first oracle execution, i.e., whether it accepts or not, and we think of s as the state of the prover. The definition can then be paraphrased as saying that if the prover in state s makes a convincing proof, then we can extract a witness.

If the definition holds also for unbounded $\mathcal{P}^*$ and $\mathcal{A}$ we say the proof has statistical knowledge soundness.

If the definition of knowledge soundness holds for a non-rewinding extractor, i.e., a single transcript of the prover's communication with the communication channel suffices, we say the proof system has knowledge soundness with straight-line extraction.

We will construct public-coin proofs that have special honest-verifier zero-knowledge. This means that if the verifier's challenges are known, or even adversarially chosen, then it is possible to simulate the verifier's view without the witness. In other words, the simulator works for verifiers who may use adversarial coins in choosing their challenges but they follow the specification of the protocol as an honest verifier would.

Definition 3 (Special Honest-Verifier Zero-Knowledge). *The proof of knowledge is* computationally special honest-verifier zero-knowledge (SHVZK) *if there exists a PPT simulator $\mathcal{S}$ such that for all stateful interactive PPT adversaries $\mathcal{A}$ that output (u,w) such that $(pp,u,w) \in R$ and randomness ρ for the verifier*

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathcal{K}(1^\lambda); (u,w,\rho) \leftarrow \mathcal{A}(pp); \\ \mathsf{view}_{\mathcal{V}} \leftarrow \langle \mathcal{P}(pp,u,w) \overset{chan}{\longleftrightarrow} \mathcal{V}(pp,u;\rho)\rangle : \mathcal{A}(\mathsf{view}_{\mathcal{V}}) = 1 \end{array}\right]$$
$$\approx \Pr\left[pp \leftarrow \mathcal{K}(1^\lambda); (u,w,\rho) \leftarrow \mathcal{A}(pp); \mathsf{view}_{\mathcal{V}} \leftarrow \mathcal{S}(pp,u,\rho) : \mathcal{A}(\mathsf{view}_{\mathcal{V}}) = 1\right].$$

We say the proof is statistically SHVZK *if the definition holds also against unbounded adversaries, and we say the proof is* perfect SHVZK *if the probabilities are exactly equal.*

From Honest-Verifier to General Zero-Knowledge. Honest-verifier zero-knowledge only guarantees the simulator works for verifiers following the proof system specifications. It might be desirable to consider general zero-knowledge where the simulator works for arbitrary malicious verifiers that may deviate from the specification of the proof. However, honest-verifier zero-knowledge is a first natural stepping stone to get efficient zero-knowledge proofs. We recall that our proofs are public coin, which means that the verifier's messages are chosen uniformly at random and independently from the messages received from the verifier. Below we recall few options to obtain general zero-knowledge proofs from a public-coin SHVZK proof. All these transformations are very efficient in terms of computation and communication such that the efficiency properties of our special honest-verifier zero-knowledge protocols are preserved.

In the Fiat-Shamir transform [FS86] the verifier's challenges are computed using a cryptographic hash function applied to the transcript up to the challenge. The Fiat-Shamir transform is more generally used to turn a public-coin proof into a non-interactive one. Since interaction with the verifier is no longer needed, general zero-knowledge is immediately achieved. If the hash function can be computed in linear time in the input size, then the Fiat-Shamir transform only incurs an additive linear overhead in computation for the prover and verifier. The drawback of the Fiat-Shamir transform is that security is usually proved in the random oracle model [BR93] where the hash function is modelled as an ideal random function.

Assuming a common reference string and relying on trapdoor commitments, Damgård [Dam00] gave a transformation yielding concurrently secure protocols for Σ-Protocols. The transformation can be optimized [Gro04] using the idea that for each public-coin challenge x, the prover first commits to a value x', then the verifier sends a value x'', after which the prover opens the commitment and uses the challenge $x = x' + x''$. The coin-flipping can be interleaved with the rest of the proof, which means the transformation preserves the number of rounds and only incurs a very small efficiency cost to do the coin-flipping for the challenges.

If one does not wish to rely on a common reference string for security, one can use a private-coin transformation where the verifier does not reveal the random coins used to generate the challenges sent to the prover (hence the final protocol is no longer public coin). One example is the Micciancio and Petrank [MP03] transformation (yielding concurrently secure protocols) while incurring a small overhead of $\omega(\log \lambda)$ with respect to the number of rounds as well as the computational and communication cost in each round. The transformation preserves the soundness and completeness errors of the original protocol; however, it does not preserve statistical zero-knowledge as the obtained protocol only has computational zero-knowledge.

There are other public-coin transformations to general zero-knowledge e.g. Goldreich et al. [GSV98]. The transformation relies on a random-selection protocol between the prover and verifier to specify a set of messages and restricting the verifier to choose challenges from this set. This means to get negligible soundness error these transformations require $\omega(1)$ sequential repetitions so the round complexity goes up.

2.3 Linear-Time Linear Error-Correcting Codes

A *code* over an alphabet Σ is a subset $\mathcal{C} \subseteq \Sigma^n$. A code $\mathcal{C}$ is associated with an encoding function $E_\mathcal{C} : \Sigma^k \to \Sigma^n$ mapping messages of length k into *codewords* of length n. We assume there is a setup algorithm $\mathrm{Gen}_{\mathsf{E}_\mathcal{C}}$ which takes as input a finite field $\mathbb{F}$ and the parameter $k \in \mathbb{N}$, and outputs an encoding function $E_\mathcal{C}$.

In what follows, we restrict our attention to $\mathbb{F}$-*linear codes* for which the alphabet is a finite field $\mathbb{F}$, the code $\mathcal{C}$ is a k-dimensional linear subspace of $\mathbb{F}^n$, and $E_\mathcal{C}$ is an $\mathbb{F}$-linear map. The *rate* of the code is defined to be $\frac{k}{n}$. The *Hamming distance* between two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^n$ is denoted by $\mathsf{hd}(\boldsymbol{x}, \boldsymbol{y})$ and corresponds to the number of coordinates in which $\boldsymbol{x}, \boldsymbol{y}$ differ. The *(minimum) distance* of a code is defined to be the minimum Hamming distance $\mathsf{hd}_{\min}$ between distinct codewords in $\mathcal{C}$. We denote by $[n, k, \mathsf{hd}_{\min}]_\mathbb{F}$ a linear code over $\mathbb{F}$ with length n, dimension k and minimum distance $\mathsf{hd}_{\min}$. The *Hamming weight* of a vector $\boldsymbol{x}$ is $\mathsf{wt}(\boldsymbol{x}) = |\{i \in [n] : \boldsymbol{x}_i \neq 0\}|$.

In the next sections, we will use families of linear codes achieving asymptotically good parameters. More precisely, we require codes with linear length, $n = \Theta(k)$, and linear distance, $\mathsf{hd}_{\min} = \Theta(k)$, in the *dimension* k of the code. We recall that random linear codes achieve with high probability the best trade-off between distance and rate. However, in this work we are particularly concerned with computational efficiency of the encoding procedure and random codes are not known to be very efficient.

To obtain zero-knowledge proofs and arguments with linear cost for prover and verifier, we need to use codes that can be encoded in linear time. Starting from the seminal work of Spielman [Spi95], there has been a rich stream of research [GI01, GI02, GI03, GI05, DI14, CDD+16] regarding linear codes with linear-time encoding. Our construction can be instantiated, for example, with one of the families of codes presented by Druk and Ishai [DI14]. These are defined over a generic finite field $\mathbb{F}$ and meets all the above requirements.

Theorem 1 ([DI14])**.** *There exist constants $c_1 > 1$ and $c_2 > 0$ such that for every finite field $\mathbb{F}$ there exists a family of $[\lceil c_1 k \rceil, k, \lfloor c_2 k \rfloor]_\mathbb{F}$ linear codes, which can be encoded by a uniform family of linear-size arithmetic circuit of addition gates.*

2.4 Commitment Schemes

A non-interactive commitment scheme allows a sender to commit to a secret message and later reveal the message in a verifiable way. Here we are interested in commitment schemes that take as input an arbitrary length message so the message space is $\{0, 1\}^*$. A commitment scheme is defined by a pair of PPT algorithms $(\mathsf{Setup}, \mathsf{Commit})$.

$\mathsf{Setup}(1^\lambda) \to ck$**:** Given a security parameter, this returns a commitment key ck.
$\mathsf{Commit}_{ck}(m) \to c$**:** Given a message m, this picks a randomness $r \leftarrow \{0, 1\}^{\mathrm{poly}(\lambda)}$ and computes a commitment $c = \mathsf{Commit}_{ck}(m; r)$.

A commitment scheme must be *binding* and *hiding*. The binding property means that it is infeasible to open a commitment to two different messages, whereas the hiding property means that the commitment does not reveal anything about the committed message.

Definition 4 (Binding). *A commitment scheme is* computationally binding *if for all PPT adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} ck \leftarrow \mathsf{Setup}(1^\lambda);\ (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck) : \\ m_0 \neq m_1 \ \wedge\ \mathsf{Commit}_{ck}(m_0; r_0) = \mathsf{Commit}_{ck}(m_1; r_1) \end{array}\right] \approx 0.$$

If this holds also for unbounded adversaries, we say the commitment scheme is statistically binding.

Definition 5 (Hiding). *A commitment scheme is* computationally hiding *if for all PPT stateful adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} ck \leftarrow \mathsf{Setup}(1^\lambda);\ (m_0, m_1) \leftarrow \mathcal{A}(ck);\ b \leftarrow \{0,1\}; \\ c \leftarrow \mathsf{Commit}_{ck}(m_b) :\ \mathcal{A}(c) = b \end{array}\right] \approx \frac{1}{2},$$

where $\mathcal{A}$ *outputs messages of equal length* $|m_0| = |m_1|$. *If the definition holds also for unbounded adversaries, we say the commitment scheme is* statistically hiding.

We will be interested in using highly efficient commitment schemes. We say a commitment scheme is *linear-time* if the time to compute $\mathsf{Commit}_{ck}(m)$ is $\mathrm{poly}(\lambda) + \mathcal{O}(|m|)$ bit operations, which we assume corresponds to $\mathrm{poly}(\lambda) + \mathcal{O}(\frac{|m|}{W})$ machine operations on our W-bit RAM machine. We will also be interested in having small size commitments. We say a commitment scheme is compact if there is a polynomial $\ell(\lambda)$ such that commitments have size at most $\ell(\lambda)$ regardless of how long the message is. We say a commitment scheme is *public coin* if there is a polynomial $\ell(\lambda)$ such that $\mathsf{Setup}(1^\lambda)$ picks the commitment key uniformly at random as $ck \leftarrow \{0,1\}^{\ell(\lambda)}$. We will now discuss some candidate linear-time commitment schemes. Applebaum et al. [AHI+17] gave constructions of low-complexity families of collision-resistant hash functions, where it is possible to evaluate the hash function in linear time in the message size. Their construction is based on the binary shortest vector problem assumption, which is related to finding non-trivial low-weight vectors in the null space of a matrix over $\mathbb{F}_2$. To get down to linear-time complexity, they conjecture the binary shortest vector problem is hard when the matrix is sparse, e.g., an LDPC parity check matrix [Gal62]. Halevi and Micali [HM96] show that a collision-resistant hash function gives rise to a compact statistically hiding commitment scheme. Their transformation is very efficient, so starting with a linear-time hash function, one obtains a linear-time statistically hiding compact commitment scheme. Moreover, if we instantiate the hash function with the one by Applebaum et al. [AHI+17], which is public coin, we obtain a linear-time public-coin statistically hiding commitment scheme. Ishai et al. [IKOS08] propose linear-time computable pseudorandom generators. If we have statistically binding commitment scheme this means we can commit to an arbitrary length message m by picking a seed s for the pseudorandom generator, stretch it to $t = \mathrm{PRG}(s)$ of length $|m|$ and let $(\mathsf{Commit}_{ck}(s), t \oplus m)$ be the commitment to m. Assuming the commitment scheme is statistically binding, this gives us a linear-time statistically binding commitment scheme for arbitrary length messages. It can also easily be seen that commitments have the same length as the messages plus an additive polynomial overhead that depends only on the security parameter. The construction also preserves the public-coin property of the seed commitment scheme.

3 Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability in the Ideal Linear Commitment Model

In this section, we construct a SHVZK proof of knowledge for arithmetic circuit satisfiability relations $\mathcal{R}_{\mathsf{AC}}$ in the ILC model. Our proof can be seen as an abstraction of the zero-knowledge argument of Groth [Gro09] in an idealized vector commitment setting. In the ILC model, the prover can `commit` to vectors in $\mathbb{F}^k$ by sending them to the channel. The ILC channel stores the received vectors and communicates to verifier the number of vectors it received. The verifier can `send` messages to the prover via the ILC channel, which in the case of Groth's and our proof system consist of field elements in $\mathbb{F}$. Finally, the verifier can query the channel to `open` arbitrary linear combinations of the committed vectors sent by the prover. The field $\mathbb{F}$ and the vector length k is specified by the public parameter pp_{ILC}. It will later emerge that to get the best communication and computation complexity for arithmetic circuits with N gates, k should be approximately $\sqrt{N}$.

Consider a circuit with a total of N fan-in 2 gates, which can be either addition gates or multiplication gates over a field $\mathbb{F}$. Each gate has two inputs (left and right) and one output wire, and each output wire can potentially be attached as input to several other gates. In total, we have $3N$ inputs and outputs to gates. Informally, the description of an arithmetic circuit consists of a set of gates, the connection of wires between gates and known values assigned to some of the inputs and outputs. A circuit is said to be satisfiable if there exists an assignment complying with all the gates, the wiring, and the known values specified in the instance.

At a high level, the idea of the proof is for the prover to commit to the $3N$ inputs and outputs of all the gates in the circuit, and then prove that these assignments are consistent with the circuit description. This amounts to performing the following tasks:

- Prove for each value specified in the instance that this is indeed the value the prover has committed to.
- Prove for each addition gate that the committed output is the sum of the committed inputs.

– Prove for each multiplication gate that the committed output is the product of the committed inputs.
– Prove for each wire that all committed values corresponding to this wire are the same.

To facilitate these proofs, we arrange the committed values into row vectors $\boldsymbol{v}_i \in \mathbb{F}^k$ similarly to [Gro09]. Without loss of generality we assume both the number of addition gates and the number of multiplication gates are divisible by k, which can always be satisfied by adding few dummy gates to the circuit. We can then number addition gates from $(1, 1)$ to (m_A, k) and multiplication gates $(m_A + 1, 1)$ to $(m_A + m_M, k)$. We insert assignments to left inputs, right inputs and outputs of addition gates into entries of three matrices $A, B, C \in \mathbb{F}^{m_A \times k}$, respectively. We sort entries to the matrices so that wires attached to the same gate correspond to the same entry of the three matrices, as shown in Fig. 2. A valid assignment to the wires then satisfies $A + B = C$. We proceed in a similar way for the $m_M \cdot k$ multiplication gates to obtain three matrices $D, E, F \in \mathbb{F}^{m_M \times k}$ such that $D \circ E = F$, where $\circ$ denotes the Hadamard (i.e. entry-wise) product of matrices. All the committed wires then constitute a matrix

$$V = \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix} \in \mathbb{F}^{(3m_A + 3m_M) \times k}$$

Without loss of generality, we also assume the gates to be sorted so that the wire values specified in the instance correspond to *full* rows in V. Again, this is without loss of generality because we can always add a few dummy gates to the circuit and to the instance to complete a row.

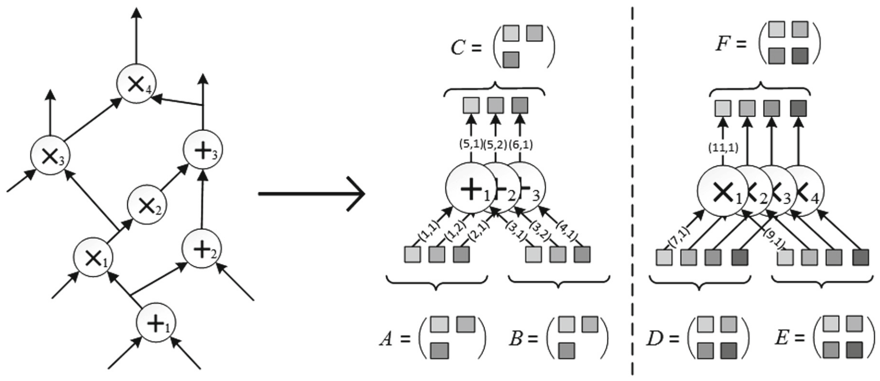

Fig. 2. Representation of an arithmetic circuit and arrangements of the wires into 6 matrices.

With these transformations in mind, let us write the arithmetic circuit relation as follows

$$\mathcal{R}_{\mathsf{AC}} = \left\{ \begin{array}{ll} \multicolumn{2}{l}{(pp, u, w) = \Big((\mathbb{F}, k, *),\ (m_A, m_M, \pi, \{\boldsymbol{v}_i\}_{i \in S}),\ (\{\boldsymbol{v}_i\}_{i \in \bar{S}}) \Big) :} \\ \quad m = 3m_A + 3m_M & \wedge\ \pi \in \Sigma_{[m] \times [k]} \\ \wedge\ S \subseteq [m] & \wedge\ \bar{S} = [m] \setminus S \\ \wedge\ A = (\boldsymbol{v}_i)_{i=1}^{m_A} & \wedge\ D = (\boldsymbol{v}_i)_{i=3m_A+1}^{3m_A+m_M} \\ \wedge\ B = (\boldsymbol{v}_i)_{i=m_A+1}^{2m_A} & \wedge\ E = (\boldsymbol{v}_i)_{i=3m_A+m_M+1}^{3m_A+2m_M} \\ \wedge\ C = (\boldsymbol{v}_i)_{i=2m_A+1}^{3m_A} & \wedge\ F = (\boldsymbol{v}_i)_{i=3m_A+2m_M+1}^{3m_A+3m_M} \\ \wedge\ A + B = C & \wedge\ D \circ E = F \\ \wedge\ V = (\boldsymbol{v}_i)_{i=1}^{m} & \wedge\ V_{i,j} = V_{\pi(i,j)}\ \forall\ (i,j) \in [m] \times [k] \end{array} \right\}$$

The role of the permutation π is to specify the wiring of the arithmetic circuit. For each wire, we can write a cycle $((i_1, j_1), \ldots, (i_t, j_t))$ that lists the location of the committed values corresponding to this wire. Then we let $\pi \in \Sigma_{[m] \times [k]}$ be the product of all these cycles, which unambiguously defines the wiring of the circuit. To give an example using the circuit in Fig. 2, the output wire of the first addition gate also appears as input of the first multiplication gate and the second addition gate. Therefore, if they appear as entries $(5,1), (9,1), (1,2)$ in the matrix V defined by the rows $\boldsymbol{v}_i$, then we would have the cycle $((5,1), (9,1), (1,2))$ indicating entries that must to be identical. The output of the second addition gate feeds into the third addition gate, so this might give us a cycle $((5,2), (4,1))$ of entries that should have the same value. The permutation π is the product of all these cycles that define which entries should have the same value.

In the proof for arithmetic circuit satisfiability, the prover starts by committing to all values $\{\boldsymbol{v}_i\}_{i=1}^{m}$. She will then call suitable sub-proofs to handle the four constraints these committed values should specify. We describe all the sub-proofs after the main proof given in Fig. 3 and refer to the full version of the paper for the detailed constructions.

$\mathcal{P}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, w)$

- Parse $u = (m_A, m_M, \pi, \{\boldsymbol{v}_i\}_{i \in S})$
- Parse $w = (\{\boldsymbol{v}_i\}_{i \in \bar{S}})$
- Send $(\texttt{commit}, \{\boldsymbol{v}_i\}_{i=1}^{m})$ to the ILC channel
- The vectors define $V \in \mathbb{F}^{m \times k}$ and sub-matrices A, B, C, D, E, F as described earlier
- Let $U = (\boldsymbol{v}_i)_{i \in S}$
- Run $\mathcal{P}_{\mathrm{eq}}(pp_{\mathsf{ILC}}, (\{\boldsymbol{v}_i\}_{i \in S}, [U]))$
- Run $\mathcal{P}_{\mathrm{sum}}(pp_{\mathsf{ILC}}, ([A], [B], [C]))$
- Run $\mathcal{P}_{\mathrm{prod}}(pp_{\mathsf{ILC}}, ([D], [E], [F]))$
- Run $\mathcal{P}_{\mathrm{perm}}(pp_{\mathsf{ILC}}, (\pi, [V], [V]))$

$\mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u)$

- Parse $u = (m_A, m_M, \pi, \{\boldsymbol{u}_i\}_{i \in S})$
- Run $\mathcal{V}_{\mathrm{eq}}(pp_{\mathsf{ILC}}, (\{\boldsymbol{u}_i\}_{i \in S}, [U]))$
- Run $\mathcal{V}_{\mathrm{sum}}(pp_{\mathsf{ILC}}, ([A], [B], [C]))$
- Run $\mathcal{V}_{\mathrm{prod}}(pp_{\mathsf{ILC}}, ([D], [E], [F]))$
- Run $\mathcal{V}_{\mathrm{perm}}(pp_{\mathsf{ILC}}, (\pi, [V], [V]))$
- Return 1 if all the sub-proofs are accepted and 0 otherwise

Fig. 3. Arithmetic circuit satisfiability proof in the ILC model.

Here we use the convention that when vectors or matrices are written in square brackets, i.e., when we write $[A]$ in the instance, it means that these are values that have already been committed to the ILC channel. The prover knows these values, but the verifier may not know them. The first sub-proof $\left\langle \mathcal{P}_{\mathrm{eq}}\left(pp_{\mathsf{ILC}}, (\{\boldsymbol{v}_i\}_{i\in S}, [U])\right) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathrm{eq}}\left(pp_{\mathsf{ILC}}, (\{\boldsymbol{u}_i\}_{i\in S}, [U])\right)\right\rangle$ allows the verifier to check that values included in the instance are contained in the corresponding commitments the prover previously sent to the ILC channel. The second sub-proof $\left\langle \mathcal{P}_{\mathrm{sum}}\left(pp_{\mathsf{ILC}}, ([A],[B],[C])\right) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathrm{sum}}\left(pp_{\mathsf{ILC}}, ([A],[B],[C])\right)\right\rangle$ is used to prove the committed matrices A, B and C satisfy $A + B = C$. The sub-proof $\left\langle \mathcal{P}_{\mathrm{prod}}\left(pp_{\mathsf{ILC}}, ([D],[E],[F])\right) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathrm{prod}}\left(pp_{\mathsf{ILC}}, ([D],[E],[F])\right)\right\rangle$ is used to prove that the committed matrices D, E and F satisfy $D \circ E = F$. The last sub-proof $\left\langle \mathcal{P}_{\mathrm{perm}}\left(pp_{\mathsf{ILC}}, (\pi,[A],[B])\right) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathrm{perm}}\left(pp_{\mathsf{ILC}}, (\pi,[A],[B])\right)\right\rangle$ is used to prove that A has the same entries as B except they have been permuted according to the permutation π. Note that when we call the permutation sub-proof with $B = A$, then the statement is that A remains unchanged when we permute the entries according to π. This in turn means that all committed values that lie on the same cycle in the permutation must be identical, i.e., the matrix A respects the wiring of the circuit.

Theorem 2. *$(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ is a proof system for $\mathcal{R}_{\mathsf{AC}}$ in the ILC model with perfect completeness, statistical knowledge soundness with straight-line extraction, and perfect special honest-verifier zero-knowledge.*

Proof. Perfect completeness follows from the perfect completeness of the sub-proofs.

Perfect SHVZK follows from the perfect SHVZK of the sub-proofs. A simulated transcript is obtained by combining the outputs of the simulators of all the sub-proofs.

Also statistical knowledge soundness follows from the knowledge soundness of the sub-proofs. The statistical knowledge soundness of the equality sub-proof guarantees that commitments to values included in the instance indeed contain the publicly known values. The correctness of the addition gates and multiplication gates follows from the statistical knowledge soundness of the respective sub-proofs. Finally, as we have argued above, the permutation sub-proof guarantees the committed values respect the wiring of the circuit.

Since all sub-proofs have knowledge soundness with straight line extraction, so does the main proof. □

The efficiency of our arithmetic circuit satisfiability proof in the ILC model is given in Fig. 4. A detailed breakdown of the costs of each sub-protocol can be found in the full version of the paper. The asymptotic results displayed below are obtained when the parameter k specified by pp_{ILC} is approximately $\sqrt{N}$. The query complexity qc is the number of linear combinations the verifier queries from the ILC channel in the opening query. The verifier communication C_{ILC} is the number of messages sent from the verifier to the prover via the ILC channel

Prover computation	$T_{\mathcal{P}_{\mathsf{ILC}}} = \mathcal{O}(N)$ multiplications in $\mathbb{F}$
Verifier computation	$T_{\mathcal{V}_{\mathsf{ILC}}} = \mathcal{O}(N)$ additions in $\mathbb{F}$
Query complexity	$\text{qc} = 20$
Verifier communication	$C_{\mathsf{ILC}} = \mathcal{O}(\log\log(N))$ field elements
Round complexity	$\mu = \mathcal{O}(\log\log(N))$
Total number of committed vectors	$t = \mathcal{O}\left(\sqrt{N}\right)$ vectors in $\mathbb{F}^k$

Fig. 4. Efficiency of our arithmetic circuit satisfiability proof in the ILC model $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ for $(pp, u, w) \in \mathcal{R}_{\mathsf{AC}}$.

and in our proof system it is proportional to the number of rounds. Let μ be the number of rounds in the ILC proof and $t_1, \ldots, t_\mu$ be the numbers of vectors that the prover sends to the ILC channel in each round, and let $t = \sum_{i=1}^{\mu} t_i$.

4 Compiling Ideal Linear Commitment Proofs into Standard Proofs

In this section, we show how to compile a proof of knowledge with straight-line extraction for relation $\mathcal{R}$ over the communication channel ILC into a proof of knowledge without straight-line extraction for the same relation over the standard channel. Recall that the ILC channel allows the prover to submit vectors of length k to the channel and the verifier can then query linear combinations of those vectors.

The idea behind the compilation of an ILC proof is that instead of committing to vectors $\boldsymbol{v}_\tau$ using the channel ILC, the prover encodes each vector $\boldsymbol{v}_\tau$ as $\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_\tau)$ using a linear error-correcting code $\mathsf{E}_{\mathcal{C}}$. In any given round, we can think of the codewords as rows $\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_\tau)$ in a matrix $\mathsf{E}_{\mathcal{C}}(V)$. However, instead of committing to the rows of the matrix, the prover commits to the columns of the matrix. When the verifier wants to open a linear combination of the original vectors, he sends the coefficients $\boldsymbol{q} = (q_1, \ldots, q_t)$ of the linear combination to the prover, and the prover responds with the linear combination $\boldsymbol{v}_{(\boldsymbol{q})} \leftarrow \boldsymbol{q}V$. Notice that we will use the notation $\boldsymbol{v}_{(\boldsymbol{q})}$, and later on $\boldsymbol{v}_{(\boldsymbol{\gamma})}$, to denote vectors that depend on $\boldsymbol{q}$ and $\boldsymbol{\gamma}$: the $\boldsymbol{q}$ and $\boldsymbol{\gamma}$ are not indices. Now, to spot check that the prover is not giving a wrong $\boldsymbol{v}_{(\boldsymbol{q})}$, the verifier may request the jth element of each committed codeword $\boldsymbol{e}_\tau$. This corresponds to revealing the jth column of error-corrected matrix $\mathsf{E}_{\mathcal{C}}(V)$. Since the code $\mathsf{E}_{\mathcal{C}}$ is linear, the revealed elements should satisfy $\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_{(\boldsymbol{q})})_j = \sum_{\tau=1}^{t} q_\tau \mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_\tau)_j = \boldsymbol{q}(\mathsf{E}_{\mathcal{C}}(V)|_j)$. The verifier will spot check on multiple columns, so that if the code has sufficiently large minimum distance and the prover gives a wrong $\boldsymbol{v}_{(\boldsymbol{q})}$, then with overwhelming probability, the verifier will open at least one column j where the above equality does not hold.

Revealing entries in a codeword may leak information about the encoded vector. To get SHVZK, instead of using $\mathsf{E}_{\mathcal{C}}$, we use a randomized encoding $\tilde{\mathsf{E}}_{\mathcal{C}}$ defined by $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}; \boldsymbol{r}) = (\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}) + \boldsymbol{r}, \boldsymbol{r})$. This doubles the code-length to $2n$ but ensures that when you reveal entry j, but not entry $j+n$, then the verifier only learns a random field element. The spot checking technique using $\tilde{\mathsf{E}}_{\mathcal{C}}$ is illustrated in Fig. 5. In the following, we use the notation $\boldsymbol{e}_\tau = (\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_\tau) + \boldsymbol{r}_\tau, \boldsymbol{r}_\tau)$ and $E = (\mathsf{E}_{\mathcal{C}}(V) + R, R)$. We also add a check, where the verifier sends an extra

$$\begin{array}{ccc}
\begin{pmatrix} \boldsymbol{v}_0 \\ \vdots \\ \boldsymbol{v}_t \end{pmatrix} & \xrightarrow{\tilde{\mathsf{E}}_{\mathcal{C}}} & \left(\begin{array}{c|c} \mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_0) + \boldsymbol{r}_0 & \boldsymbol{r}_0 \\ \vdots & \vdots \\ \mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_t) + \boldsymbol{r}_t & \boldsymbol{r}_t \end{array}\right) \\
\boldsymbol{q}\downarrow & & \boldsymbol{q}\downarrow_{j_1} \cdots \boldsymbol{q}\downarrow_{j_\lambda} \quad \boldsymbol{q}\downarrow_{j_{\lambda+1}} \cdots \boldsymbol{q}\downarrow_{j_{2\lambda}} \\
\left(\boldsymbol{v}_{(\boldsymbol{q})} \right) & \xrightarrow{\tilde{\mathsf{E}}_{\mathcal{C}}} & \left(\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_{(\boldsymbol{q})}) + \boldsymbol{r}_{(\boldsymbol{q})} \;\middle|\; \boldsymbol{r}_{(\boldsymbol{q})} \right)
\end{array}$$

Fig. 5. Vectors $\boldsymbol{v}_\tau$ organized in matrix V are encoded row-wise as matrix $E = \tilde{\mathsf{E}}_{\mathcal{C}}(V; R)$. The vertical line in the right matrix and vector denotes concatenation of matrices respectively vectors. The prover commits to each column of E. When the prover given $\boldsymbol{q}$ wants to reveal the linear combination $\boldsymbol{v}_{(\boldsymbol{q})} = \boldsymbol{q}V$ she also reveals $\boldsymbol{r}_{(\boldsymbol{q})} = \boldsymbol{q}R$. The verifier now asks for openings of 2λ columns $J = \{j_1, \ldots, j_{2\lambda}\}$ in E and verifies for these columns that $\boldsymbol{q}E|_J = \tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\boldsymbol{q})}; \boldsymbol{r}_{(\boldsymbol{q})})|_J$. To avoid revealing any information about $\mathsf{E}_{\mathcal{C}}(V)$, we must ensure that $\forall j \in [n] : j \in J \Rightarrow j + n \notin J$. If the spot checks pass, the verifier believes that $\boldsymbol{v}_{(\boldsymbol{q})} = \boldsymbol{q}V$.

random linear combination $\boldsymbol{\gamma} \in \mathbb{F}^t$ to ensure that if a malicious prover commits to values of $\boldsymbol{e}_\tau$ that are far from being codewords, the verifier will most likely reject. The reason the challenges $\boldsymbol{q}$ from the ILC proof are not enough to ensure this is that they are not chosen uniformly at random. One could, for instance, imagine that there was a vector $\boldsymbol{v}_\tau$ that was never queried in a non-trivial way, and hence the prover could choose it to be far from a codeword. To make sure this extra challenge $\boldsymbol{\gamma}$ does not reveal information to the verifier, the prover picks a random blinding vector $\boldsymbol{v}_0$, which is added as the first row of V and will be added to the linear combination of the challenge $\boldsymbol{\gamma}$.

4.1 Construction

Let $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ be a *non-adaptive* μ-round SHVZK proof of knowledge with straight-line extraction over ILC for a relation $\mathcal{R}$. Here, non-adaptive means that the verifier waits until the last round before querying linear combinations of vectors and they are queried all at once instead of the queries depending on each other.[1] Let $\mathrm{Gen}_{\mathsf{E}_{\mathcal{C}}}$ be a generator that given field $\mathbb{F}$ and length parameter

[1] The construction can be easily modified to an adaptive ILC proof. For each round of queries in the ILC proof, there will one extra round in the compiled proof.

k outputs a constant rate linear code $\mathsf{E}_\mathcal{C}$ that is linear-time computable given its description and has linear minimum distance. Define the $\tilde{\mathsf{E}}_\mathcal{C}$ with code length $2n$ as above: $\tilde{\mathsf{E}}_\mathcal{C}(\boldsymbol{v};\boldsymbol{r}) = (\mathsf{E}_\mathcal{C}(\boldsymbol{v}) + \boldsymbol{r}, \boldsymbol{r})$. Finally, let $(\mathsf{Setup}, \mathsf{Commit})$ be a non-interactive commitment scheme.

We now define a proof of knowledge $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ in Fig. 6, where we use the following notation: given matrices $V_1, \dots, V_\mu$, $R_1, \dots, R_\mu$ and $E_1, \dots, E_\mu$ we define

```
$\mathcal{P}(pp, u, w)$
- Parse input:
    • Parse $pp = (pp_{\mathsf{ILC}}, \mathsf{E}_\mathcal{C}, ck)$
    • Parse $pp_{\mathsf{ILC}} = (\mathbb{F}, k)$
    • Get $n$ from $\mathsf{E}_\mathcal{C}$
- Round 1:
    • $\boldsymbol{v}_0 \leftarrow \mathbb{F}^k$
    • $\boldsymbol{e}_0 \leftarrow \tilde{\mathsf{E}}_\mathcal{C}(\boldsymbol{v}_0; \boldsymbol{r}_0)$
    • $(\mathtt{commit}, V_1) \leftarrow \mathcal{P}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, w)$
    • $E_1 \leftarrow \tilde{\mathsf{E}}_\mathcal{C}(V_1; R_1)$
    • Let $E_{01} = \begin{pmatrix} \boldsymbol{e}_0 \\ E_1 \end{pmatrix}$
    • $\boldsymbol{c}_1 = \mathsf{Commit}(E_{01}; \boldsymbol{s}_1)$
    • Send $(\boldsymbol{c}_1, t_1)$ to $\mathcal{V}$
- Rounds $2 \le i \le \mu$:
    • Get challenge $x_{i-1}$ from $\mathcal{V}$
    • $(\mathtt{commit}, V_i) \leftarrow \mathcal{P}_{\mathsf{ILC}}(x_{i-1})$
    • $E_i \leftarrow \tilde{\mathsf{E}}_\mathcal{C}(V_i; R_i)$
    • $\boldsymbol{c}_i = \mathsf{Commit}(E_i; \boldsymbol{s}_i)$
    • Send $(\boldsymbol{c}_i, t_i)$ to $\mathcal{V}$
- Round $\mu + 1$:
    • Get $(\boldsymbol{\gamma}, Q)$ from $\mathcal{V}$
    • $\boldsymbol{v}_{(\boldsymbol{\gamma})} \leftarrow \boldsymbol{v}_0 + \boldsymbol{\gamma} V$
    • $\boldsymbol{r}_{(\boldsymbol{\gamma})} \leftarrow \boldsymbol{r}_0 + \boldsymbol{\gamma} R$
    • $V_{(Q)} \leftarrow QV$
    • $R_{(Q)} \leftarrow QR$
    • Send $(\boldsymbol{v}_{(\boldsymbol{\gamma})}, \boldsymbol{r}_{(\boldsymbol{\gamma})}, V_{(Q)}, R_{(Q)})$ to $\mathcal{V}$
- Round $\mu + 2$:
    • Get $J \subset [2n]$ from $\mathcal{V}$
    • Send $(E_{01}|_J, \boldsymbol{s}_1|_J, \dots, E_\mu, \boldsymbol{s}_\mu|_J)$ to $\mathcal{V}$

$\mathcal{K}(1^\lambda)$
- $pp_{\mathsf{ILC}} \leftarrow \mathcal{K}_{\mathsf{ILC}}(1^\lambda)$
- Parse $pp_{\mathsf{ILC}} = (\mathbb{F}, k)$
- $\mathsf{E}_\mathcal{C} \leftarrow \mathrm{Gen}_{\mathsf{E}_\mathcal{C}}(\mathbb{F}, k)$
- $ck \leftarrow \mathsf{Setup}(1^\lambda)$
- Return $pp = (pp_{\mathsf{ILC}}, \mathsf{E}_\mathcal{C}, ck)$

$\mathcal{V}(pp, u)$
- Parse input
    • Parse $pp = (pp_{\mathsf{ILC}}, \mathsf{E}_\mathcal{C}, ck)$
    • Parse $pp_{\mathsf{ILC}} = (\mathbb{F}, k)$
    • Get $n$ from $\mathsf{E}_\mathcal{C}$
    • Give input $(pp_{\mathsf{ILC}}, u)$ to $\mathcal{V}_{\mathsf{ILC}}$
- Rounds $1 \le i < \mu$:
    • Receive $(\boldsymbol{c}_i, t_i)$
    • $(\mathtt{send}, x_i) \leftarrow \mathcal{V}_{\mathsf{ILC}}(t_i)$
    • Send $x_i$ to $\mathcal{P}$
- Round $\mu$:
    • Receive $(\boldsymbol{c}_\mu, t_\mu)$
    • $\boldsymbol{\gamma} \leftarrow \mathbb{F}^{\sum_{i=1}^{\mu} t_i}$
    • $(\mathtt{open}, Q) \leftarrow \mathcal{V}_{\mathsf{ILC}}(t_\mu)$
    • Send $(\boldsymbol{\gamma}, Q)$ to $\mathcal{P}$
- Round $\mu + 1$:
    • Receive $(\boldsymbol{v}_{(\boldsymbol{\gamma})}, \boldsymbol{r}_{(\boldsymbol{\gamma})}, V_{(Q)}, R_{(Q)})$
    • Choose random allowed $J \subset [2n]$
    • Send $J$ to $\mathcal{P}$
- Round $\mu + 2$:
    • Receive $(E_{01}|_J, \boldsymbol{s}_1|_J, \dots, E_\mu, \boldsymbol{s}_\mu|_J)$
    • Check $\boldsymbol{c}_1|_J = \mathsf{Commit}(E_{01}|_J; \boldsymbol{s}_1|_J)$,
      $\dots, \boldsymbol{c}_\mu|_J = \mathsf{Commit}(E_\mu|_J; \boldsymbol{s}_\mu|_J)$
    • Check $\tilde{\mathsf{E}}_\mathcal{C}(\boldsymbol{v}_{(\boldsymbol{\gamma})}, \boldsymbol{r}_{(\boldsymbol{\gamma})})|_J = \boldsymbol{e}_0|_J + \boldsymbol{\gamma} E|_J$
    • Check $\tilde{\mathsf{E}}_\mathcal{C}(V_{(Q)}, R_{(Q)})|_J = QE|_J$
    • If all checks pass, return decision of
      $\mathcal{V}_{\mathsf{ILC}}(V_{(Q)})$, else return 0
```

Fig. 6. Construction of $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ from $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$, commitment scheme $(\mathsf{Setup}, \mathsf{Commit})$ and error-correcting code $\mathcal{C}$.

$$V = \begin{pmatrix} V_1 \\ \vdots \\ V_\mu \end{pmatrix} \qquad R = \begin{pmatrix} R_1 \\ \vdots \\ R_\mu \end{pmatrix} \qquad E = \begin{pmatrix} E_1 \\ \vdots \\ E_\mu \end{pmatrix}.$$

The matrices $V_1, \ldots, V_\mu$ are formed by the row vectors $\mathcal{P}_{\mathsf{ILC}}$ commits to, and we let $t_1, \ldots, t_\mu$ be the numbers of vectors in each round, i.e., for all i we have $V_i \in \mathbb{F}^{t_i \times k}$.

We say that a set $J \subset [2n]$ is *allowed* if $|J \cap [n]| = \lambda$ and $|J \setminus [n]| = \lambda$ and there is no $j \in J$ such that $j + n \in J$. In the following we will always assume codewords have length $n \geq 2\lambda$. We use $\tilde{\mathsf{E}}_{\mathcal{C}}(V; R)$ to denote the function that applies $\tilde{\mathsf{E}}_{\mathcal{C}}$ *row-wise.* In the protocol for $\mathcal{V}$, we are using that $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}; \boldsymbol{r})|_J$ can be computed from just $\boldsymbol{v}$ and $\boldsymbol{r}|_{\{j \in [n] : j \in J \vee j+n \in J\}}$. We use $\mathsf{Commit}(E; \boldsymbol{s})$ to denote the function that applies Commit *column-wise* on E and returns a vector $\boldsymbol{c}$ of $2n$ commitments. We group all $\mathcal{V}_{\mathsf{ILC}}$'s queries in one matrix $Q \in \mathbb{F}^{\mathrm{qc} \times t}$, where t is the total number of vectors committed to by $\mathcal{P}$ and qc is the query complexity of $\mathcal{V}_{\mathsf{ILC}}$, i.e., the total number of linear combinations $\boldsymbol{q}$ that $\mathcal{V}_{\mathsf{ILC}}$ requests to be opened.

4.2 Security Analysis

Theorem 3 (Completeness). *If $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ is complete for relation $\mathcal{R}$ over* ILC*, then $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ in Fig. 6 is complete for relation $\mathcal{R}$.*

Proof. All the commitment openings are correct, so they will be accepted by the verifier. In the execution of $\langle \mathcal{P}(pp, u, w) \longleftrightarrow \mathcal{V}(pp, u) \rangle$, the fact that $\mathsf{E}_{\mathcal{C}}$ is linear implies $\tilde{\mathsf{E}}_{\mathcal{C}}$ is linear and hence all the linear checks will be true. If $(pp, u, w) \in \mathcal{R}$ then $(pp_{\mathsf{ILC}}, u, w) \in \mathcal{R}$ and being complete $\langle \mathcal{P}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, w) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, stm) \rangle = 1$ so $\mathcal{V}$'s internal copy of $\mathcal{V}_{\mathsf{ILC}}$ will accept. Thus, in this case, $\langle \mathcal{P}(pp, u, w) \longleftrightarrow \mathcal{V}(pp, u) \rangle = 1$, which proves completeness. $\square$

Theorem 4 (Knowledge Soundness). *If $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ is statistically knowledge sound with a straight-line extractor for relation $\mathcal{R}$ over* ILC *and* $(\mathsf{Setup}, \mathsf{Commit})$ *is computationally (statistically) binding, then $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ as constructed above is computationally (statistically) knowledge sound for relation $\mathcal{R}$.*

Proof. We prove the computational case. The statistical case is similar.

In order to argue that $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is computationally knowledge sound, we will first show that for every DPT $\mathcal{P}^*$ there exists a deterministic (but not necessarily efficient) $\mathcal{P}^*_{\mathsf{ILC}}$ such that for all PPT $\mathcal{A}$ we have

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathcal{K}(1^\lambda); (pp_{\mathsf{ILC}}, \cdot) = pp; (u, s) \leftarrow \mathcal{A}(pp) : \\ \langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp, u; (\rho_{\mathsf{ILC}}, \rho)) \rangle = 1 \\ \wedge \ \ \langle \mathcal{P}^*_{\mathsf{ILC}}(s, pp, u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}}) \rangle = 0 \end{array}\right] \approx 0. \tag{1}$$

Note that the randomness ρ_{ILC} in $\mathcal{V}$ which comes from the internal $\mathcal{V}_{\mathsf{ILC}}$ in line two is the same as the randomness used by $\mathcal{V}_{\mathsf{ILC}}$ in line three.

Our constructed $\mathcal{P}^*_{\mathsf{ILC}}$ will run an internal copy of $\mathcal{P}^*$. When the internal $\mathcal{P}^*$ in round i sends a message $(\boldsymbol{c}_i, t_i)$, $\mathcal{P}^*_{\mathsf{ILC}}$ will simulate $\mathcal{P}^*$ on every possible continuation of the transcript, and for each $j = 1, \ldots, 2n$ find the most frequently occurring correct opening $((E_i)_j, (\boldsymbol{s}_i)_j)$ of $(\boldsymbol{c}_i)_j$. $\mathcal{P}^*_{\mathsf{ILC}}$ will then use this to get matrices E_i^*. For each row $\boldsymbol{e}^*_\tau$ of these matrices, $\mathcal{P}^*_{\mathsf{ILC}}$ finds a vector $\boldsymbol{v}_\tau$ and randomness $\boldsymbol{r}_\tau$ such that $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_\tau, \boldsymbol{r}_\tau), \boldsymbol{e}^*_\tau) < \frac{\mathsf{hd}_{\min}}{3}$ if such a vector exists. If for some τ no such vector $\boldsymbol{v}_\tau$ exists, then $\mathcal{P}^*_{\mathsf{ILC}}$ aborts. Otherwise we let V_i and R_i denote the matrices formed by the row vectors $\boldsymbol{v}_\tau$ and $\boldsymbol{r}_\tau$ in round i and $\mathcal{P}^*_{\mathsf{ILC}}$ sends V_i to the ILC. Notice that since the minimum distance of $\tilde{\mathsf{E}}_{\mathcal{C}}$ is at least $\mathsf{hd}_{\min}$, there is at most one such vector $\boldsymbol{v}_\tau$ for each $\boldsymbol{e}^*_\tau$.

The internal copy of $\mathcal{P}^*$ will expect to get two extra rounds, where in the first it should receive $\boldsymbol{\gamma}$ and Q and should respond with $\boldsymbol{v}^*_{(\boldsymbol{\gamma})}, \boldsymbol{r}^*_{(\boldsymbol{\gamma})}, V_{(Q)}$ and $R_{(Q)}$, and in the second it should receive J and send $E_{01}|_J, \boldsymbol{s}_1|_J, \ldots, E_\mu, \boldsymbol{s}_\mu|_J$. Since $\mathcal{P}^*_{\mathsf{ILC}}$ does not send and receive corresponding messages, $\mathcal{P}^*_{\mathsf{ILC}}$ does not have to run this part of $\mathcal{P}^*$. Of course, for each commitment sent by $\mathcal{P}^*$, these rounds are internally simulated many times to get the most frequent opening. Notice that a $\mathcal{V}_{\mathsf{ILC}}$ communicating over ILC with our constructed $\mathcal{P}^*_{\mathsf{ILC}}$ will, on challenge Q receive $V_{(Q)} = QV$ from the ILC.

The verifier $\mathcal{V}$ accepts only if its internal copy of $\mathcal{V}_{\mathsf{ILC}}$ accepts. Hence, the only three ways $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp, u; (\rho_{\mathsf{ILC}}, \rho))\rangle$ can accept without $\langle \mathcal{P}^*_{\mathsf{ILC}}(s, pp, u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}})\rangle$ being accepting are

1. if $\mathcal{P}^*$ makes an opening of a commitment that is not its most frequent opening of that commitment, or
2. if $\mathcal{P}^*_{\mathsf{ILC}}$ has an error because for some τ no $\boldsymbol{v}_\tau, \boldsymbol{r}_\tau$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_\tau, \boldsymbol{r}_\tau), \boldsymbol{e}^*_\tau) < \frac{\mathsf{hd}_{\min}}{3}$ exists, or
3. if $\mathcal{P}^*$ sends some $V^*_{(Q)} \neq V_{(Q)}$.

We will now argue that for each of these three cases, the probability that they happen and $\mathcal{V}$ accepts is negligible.

Since $\mathcal{P}^*$ runs in polynomial time and the commitment scheme is computationally binding, there is only negligible probability that $\mathcal{P}^*$ sends a valid opening that is not the most frequent. Since $\mathcal{V}$ will reject any opening that is not valid, the probability of $\mathcal{V}$ accepting in case 1 is negligible.

Next, we consider the second case. To do so, define the event Err that E^* is such that for some $\boldsymbol{\gamma}^* \in \mathbb{F}^t$ we have $\mathsf{hd}(\tilde{\mathcal{C}}, \boldsymbol{\gamma}^* E^*) \geq \frac{\mathsf{hd}_{\min}}{3}$. Here $\tilde{\mathcal{C}}$ denotes the image of $\tilde{\mathsf{E}}_{\mathcal{C}}$, i.e. $\tilde{\mathcal{C}} = \{(c + r, r) : c \in \mathcal{C}, r \in \mathbb{F}^n\}$. Clearly, if $\mathcal{P}^*_{\mathsf{ILC}}$ returns an error because no $\boldsymbol{v}_i, \boldsymbol{r}_i$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_i, \boldsymbol{r}_i), \boldsymbol{e}^*_i) < \frac{\mathsf{hd}_{\min}}{3}$ exist then we have Err.

The proof of the following claim can be found in the full version of the paper.

Claim. Let $\boldsymbol{e}^*_0, \ldots, \boldsymbol{e}^*_t \in \mathbb{F}^{2n}$. If Err occurs, then for uniformly chosen $\boldsymbol{\gamma} \in \mathbb{F}^t$, there is probability at most $\frac{1}{|\mathbb{F}|}$ that $\mathsf{hd}(\tilde{\mathcal{C}}, \boldsymbol{e}^*_0 + \boldsymbol{\gamma} E^*) < \frac{\mathsf{hd}_{\min}}{6}$.

Thus, if Err then with probability at least $1 - \frac{1}{|\mathbb{F}|}$ the vector $\boldsymbol{\gamma}$ is going to be such that $\mathsf{hd}(\tilde{\mathcal{C}}, \boldsymbol{e}^*_0 + \boldsymbol{\gamma} E^*) \geq \frac{\mathsf{hd}_{\min}}{6}$. If this happens, then for the vectors $(\boldsymbol{v}^*_{(\boldsymbol{\gamma})}, \boldsymbol{r}^*_{(\boldsymbol{\gamma})})$ sent by $\mathcal{P}^*$, we must have $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}^*_{(\boldsymbol{\gamma})}, \boldsymbol{r}^*_{(\boldsymbol{\gamma})}), \boldsymbol{e}^*_0 + \boldsymbol{\gamma} E^*) \geq \frac{\mathsf{hd}_{\min}}{6}$. This means that either in the first half of the codeword $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}^*_{(\boldsymbol{\gamma})}, \boldsymbol{r}^*_{(\boldsymbol{\gamma})})$ or in the second half, there will be at least $\frac{\mathsf{hd}_{\min}}{12}$ values of j where it differs from $\boldsymbol{e}^*_0 + \boldsymbol{\gamma} E^*$. It is easy to see that the λ values of j in one half of $[2n]$ are chosen uniformly and independently at random conditioned on being different.

For each of these j, there is a probability at most $1 - \frac{\mathsf{hd}_{\min}}{12n}$ that $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\boldsymbol{\gamma})}, \boldsymbol{r}_{(\boldsymbol{\gamma})})_j = \boldsymbol{e}^*_{0,j} + \boldsymbol{\gamma} E^*|_j$, and since the j's are chosen uniformly under the condition that they are distinct, given that this holds for the first i values, the probability is even smaller for the $i+1$'th. Hence, the probability that it holds for all j in this half is negligible. This shows that the probability that Err happens and $\mathcal{V}$ accepts is negligible.

Now we turn to case 3, where Err does not happen but $\mathcal{P}^*$ sends a $V^*_{(Q)} \neq V_{(Q)}$. In this case, for all $\boldsymbol{\gamma}^* \in \mathbb{F}^t$, we have $\mathsf{hd}(\tilde{\mathcal{C}}, \sum_{\tau=1}^{t} \boldsymbol{\gamma}^*_\tau \boldsymbol{e}^*_\tau) < \frac{\mathsf{hd}_{\min}}{3}$. In particular, this holds for the vector $\boldsymbol{\gamma}$ given by $\boldsymbol{\gamma}_\tau = 1$ and $\boldsymbol{\gamma}_{\tau'} = 0$ for $\tau' \neq \tau$, so the $\boldsymbol{v}_\tau$'s are well-defined.

For two matrices A and B of the same dimensions, we define their Hamming distance $\mathsf{hd}_2(A, B)$ to be the number of j's such that the jth column of A and jth column of B are different. This agrees with the standard definition of Hamming distance, if we consider each matrix to be a string of column vectors. The proof of the following claim can be found in the full version of the paper.

Claim. Assume $\neg Err$ and let V and R be defined as above. Then for any $\boldsymbol{q} \in \mathbb{F}^t$ there exists an $\boldsymbol{r}_{(\boldsymbol{q})}$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{q}V, \boldsymbol{r}_{(\boldsymbol{q})}), \boldsymbol{q}E^*) < \frac{\mathsf{hd}_{\min}}{3}$.

In particular, for any $V^*_{(Q)} \neq QV$, and any $R^*_{(Q)}$ we have

$$\mathsf{hd}_2\left(\tilde{\mathsf{E}}_{\mathcal{C}}\left(V^*_{(Q)}, R^*_{(Q)}\right), QE^*\right) \geq 2\frac{\mathsf{hd}_{\min}}{3}.$$

This means that if $\neg Err$ occurs and $\mathcal{P}^*$ attempts to open a $V^*_{(Q)} \neq V_{(Q)} = QV$ then

$$\mathsf{hd}_2\left(\tilde{\mathsf{E}}_{\mathcal{C}}\left(V^*_{(Q)}, R^*_{(Q)}\right), QE^*\right) \geq 2\frac{\mathsf{hd}_{\min}}{3}.$$

As argued above, if the distance between two strings of length $2n$ is at least $\frac{\mathsf{hd}_{\min}}{3}$, the probability that J will not contain a j such that the two strings differ in position j is negligible. Hence, the probability that $\tilde{\mathsf{E}}_{\mathcal{C}}\left(V^*_{(Q)}, R^*_{(Q)}\right)|_J = QE^*|_J$ is negligible. Thus, the probability that $\neg Err$ and $\mathcal{V}$ accepts while $\mathcal{V}_{\mathsf{ILC}}$ does not is negligible. This proves (1).

Next, we want to define a *transcript extractor* $\mathcal{T}$ that given rewindable access to $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp, u)\rangle$ outputs $\widetilde{\mathsf{trans}}_{\mathcal{P}_{\mathsf{ILC}}}$, which we would like to correspond to all messages sent between $\mathcal{P}^*_{\mathsf{ILC}}$ and the channel in $\langle \mathcal{P}^*_{\mathsf{ILC}}(s, pp, u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}})\rangle$. Here ρ_{ILC} is the randomness used by the $\mathcal{V}_{\mathsf{ILC}}$ inside $\mathcal{V}$ in the first execution of $\mathcal{T}$'s oracle $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp, u)\rangle$. However, we allow $\mathcal{T}$ to fail if $\mathcal{V}$ does not accept in this first transcript and further to fail with negligible probability. Formally, we want $\mathcal{T}$ to run in expected PPT such that for all PPT $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathcal{K}(1^\lambda); (pp_{\mathsf{ILC}}, \cdot) = pp; (u, s) \leftarrow \mathcal{A}(pp); \\ \widetilde{\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}} \leftarrow \mathcal{T}^{\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp,u)\rangle}(pp, u); \\ \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} \leftarrow \langle \mathcal{P}^*_{\mathsf{ILC}}(s, pp, u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}})\rangle : \\ b = 1 \quad \wedge \quad \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} \neq \widetilde{\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}} \end{array}\right] \approx 0. \tag{2}$$

Here b is the value output by $\mathcal{V}$ the first time $\mathcal{T}$'s oracle runs $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp, u)\rangle$, and the randomness ρ_{ILC} used by $\mathcal{V}_{\mathsf{ILC}}$ in the third line is identical to the random value used by the $\mathcal{V}_{\mathsf{ILC}}$ inside $\mathcal{V}$ in the first transcript. On input (pp, u), the transcript extractor $\mathcal{T}$ will first use its oracle to get a transcript of $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp, u; (\rho_{\mathsf{ILC}}, \rho))\rangle$. If $\mathcal{V}$ rejects, $\mathcal{T}$ will just abort. If $\mathcal{V}$ accepts, $\mathcal{T}$ will rewind the last message of $\mathcal{P}^*$ to get a transcript for a new random challenge J. $\mathcal{T}$ continues this way, until it has an accepting transcript for $2n$ independently chosen sets J. Notice that if there is only one choice of J that results in $\mathcal{V}$ accepting, $\mathcal{P}^*$ will likely have received each allowed challenge around $2n$ times and $\mathcal{T}$ will get the exact same transcript $2n$ times before it is done rewinding. Still, $\mathcal{T}$ runs in expected polynomial time: if a fraction p of all allowed set J results in accept, the expected number of rewindings *given* that the first transcripts accepts is $\frac{2n-1}{p}$. However, the probability that the first run accepts is p, and if it does not accept, $\mathcal{T}$ does not do any rewindings. In total, that gives $\frac{(2n-1)p}{p} = 2n - 1$ rewindings in expectation.

We let $J_1, \ldots, J_{2n}$ denote the set of challenges J in the accepting transcripts obtained by $\mathcal{T}$. If $\bigcup_{i=1}^{2n} J_i$ has less than $2n - \frac{\mathsf{hd}_{\min}}{3}$ elements, $\mathcal{T}$ terminates. Otherwise, $\mathcal{T}$ is defined similarly to $\mathcal{P}^*_{\mathsf{ILC}}$: it uses the values of the openings to get at least $2n - \frac{\mathsf{hd}_{\min}}{3}$ columns of each E_i. For each of the row vectors, $\boldsymbol{e}_\tau$, it computes $\boldsymbol{v}_\tau$ and $\boldsymbol{r}_\tau$ such that $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_\tau, \boldsymbol{r}_\tau)$ agrees with $\boldsymbol{e}_\tau$ in all entries $(\boldsymbol{e}_\tau)_j$ for which the j'th column have been revealed, if such $\boldsymbol{v}$ exists. Since $\mathcal{T}$ will not correct any errors, finding such $\boldsymbol{v}_\tau$ and $\boldsymbol{r}_\tau$ corresponds to solving a linear set of equations. Notice that since the minimum distance is more than $2\frac{\mathsf{hd}_{\min}}{3}$ there is at most one such $\boldsymbol{v}_\tau$ for each $\tau \in [t]$. If for some τ there is no such $\boldsymbol{v}_\tau$, then $\mathcal{T}$ aborts, otherwise $\mathcal{T}$ use the resulting vectors $\boldsymbol{v}_\tau$ as the prover messages to define $\widetilde{\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}}$.

If $|\bigcup_{i=1}^{\kappa} J_i| < 2n - \frac{\mathsf{hd}_{\min}}{3}$, there are at least $\frac{\mathsf{hd}_{\min}}{6}$ numbers in $[n] \setminus \bigcup_{i=1}^{\kappa} J_i$ or in $\{n+1, \ldots, 2n\} \setminus \bigcup_{i=1}^{\kappa} J_i$. In either case, a random allowed J has negligible probability of being contained in $\bigcup_{i=1}^{\kappa} J_i$. Since $\mathcal{T}$ runs in expected polynomial time, this implies by induction that there is only negligible probability that $|\bigcup_{i=1}^{\kappa} J_i| < \min(\kappa, 2n - \frac{\mathsf{hd}_{\min}}{3})$ and therefore $|\bigcup_{i=1}^{2n} J_i| < 2n - \frac{\mathsf{hd}_{\min}}{3}$.

Finally, we need to show

Claim. The probability that for some τ there are no $\boldsymbol{v}_\tau$ and $\boldsymbol{r}_\tau$ such that $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_\tau, \boldsymbol{r}_\tau)$ agrees with $\boldsymbol{e}_\tau$ on the opened $j \in \bigcup_{i=1}^{2n} J_i$ and $b = 1$ is negligible.

In particular, the probability that $b = 1$ but $\mathcal{T}$ does not extract the transcript of $\mathcal{P}^*_{\mathsf{ILC}}$ is negligible.

Proof. Since we can ignore events that happen with negligible probability, and the expected number of rewindings is polynomial, we can assume that in all the rewindings, $\mathcal{P}^*$ only makes openings to the most common openings. We showed that the probability that $b = 1$ but $\mathcal{P}^*$ sends a $V^*_{(Q)} \neq V$ is negligible and by the same argument the probability that $b = 1$ but $\mathcal{P}^*$ sends $\boldsymbol{v}^*_{(\gamma)} \neq \boldsymbol{v}_{(\gamma)}$ is negligible. Therefore, in the following, we will assume $\boldsymbol{v}^*_{(\gamma)} = \boldsymbol{v}_{(\gamma)}$.

Now suppose that there is some $\boldsymbol{e}_\tau$ such that the opened values are inconsistent with being $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_\tau, \boldsymbol{r}_\tau)$ for any $\boldsymbol{r}_\tau$. That is, there is some j such that $j, n+j \in \bigcup_{i=1}^{2n} J_i$ and $(\boldsymbol{e}_\tau)_j - (\boldsymbol{e}_\tau)_{n+j} \neq \mathsf{E}_{\mathcal{C}}(\boldsymbol{v})_j$. For uniformly chosen $\boldsymbol{\gamma}_\tau \in \mathbb{F}$, we get that $\boldsymbol{\gamma}_\tau((\boldsymbol{e}_\tau)_j - (\boldsymbol{e}_\tau)_{n+j} - \mathsf{E}_{\mathcal{C}}(\boldsymbol{v})_j)$ is uniformly distributed in $\mathbb{F}$. Hence for a random $\boldsymbol{\gamma} \in \mathbb{F}^t$, we have that $\boldsymbol{\gamma} \cdot ((\boldsymbol{e})_j - (\boldsymbol{e})_{n+j} - \mathsf{E}_{\mathcal{C}}(\boldsymbol{v})_j)$ is uniformly distributed. When $\mathcal{V}$ sends $\boldsymbol{\gamma}$, $\mathcal{P}^*$ will respond with $\boldsymbol{v}^*_{(\gamma)} = \boldsymbol{v}_{(\gamma)}$ and some $\boldsymbol{r}^*_{(\gamma)}$. $\mathcal{V}$ will only accept on a challenge J if for all $j \in J$ we have $(\boldsymbol{e}_0 + \boldsymbol{\gamma}\boldsymbol{e})_j = \tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\gamma)}, \boldsymbol{r}^*_{(\gamma)})_j$. Since $j, n+j \in \bigcup_{i=1}^{2n} J_i$ we have $(\boldsymbol{e}_0 + \boldsymbol{\gamma}\boldsymbol{e})_j = \tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\gamma)}, \boldsymbol{r}^*_{(\gamma)})_j$ and $(\boldsymbol{e}_0 + \boldsymbol{\gamma}\boldsymbol{e})_{n+j} = \tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\gamma)}, \boldsymbol{r}^*_{(\gamma)})_{n+j}$ so

$$\begin{aligned}(\boldsymbol{e}_0)_j - (\boldsymbol{e}_0)_{n+j} + \boldsymbol{\gamma}\boldsymbol{e}_j - \boldsymbol{\gamma}\boldsymbol{e}_{n+j} &= \tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\gamma)}, \boldsymbol{r}^*_{(\gamma)})_j - \tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\gamma)}, \boldsymbol{r}^*_{(\gamma)})_{n+j} \\ &= \mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_{(\gamma)})_j \\ &= (\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_0) + \boldsymbol{\gamma}\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}))_j\end{aligned}$$

that is,

$$\boldsymbol{\gamma}\boldsymbol{e}_j - \boldsymbol{\gamma}\boldsymbol{e}_{n+j} - \boldsymbol{\gamma}\mathsf{E}_{\mathcal{C}}(\boldsymbol{v})_j = \mathsf{E}_{\mathcal{C}}(\boldsymbol{v}_0)_j - (\boldsymbol{e}_0)_j + (\boldsymbol{e}_0)_{n+j}$$

For random $\boldsymbol{\gamma}$ the left-hand side is uniform and the right-hand side is fixed, hence equality only happens with negligible probability. That proves the claim. □

Since $\mathcal{E}_{\mathsf{ILC}}^{\langle \mathcal{P}^*_{\mathsf{ILC}}(s,pp,u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}},u)\rangle}(pp, u)$ is a straight-line extractor, we can simply assume that it gets the transcript as an input, and can be written as $\mathcal{E}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}})$. For any PPT $\mathcal{A}$ consider the following experiment.

$$\left[\begin{array}{c} pp \leftarrow \mathcal{K}(1^\lambda); (pp_{\mathsf{ILC}}, \cdot) = pp; (u, s) \leftarrow \mathcal{A}(pp); \\ \widetilde{\mathsf{trans}}_{\mathcal{P}_{\mathsf{ILC}}} \leftarrow \mathcal{T}^{\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp,u)\rangle}(pp, u); \\ \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} \leftarrow \langle \mathcal{P}^*_{\mathsf{ILC}}(s, pp, u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}})\rangle = b_{\mathsf{ILC}}; \\ w \leftarrow \mathcal{E}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}); \\ \widetilde{w} \leftarrow \mathcal{E}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, \widetilde{\mathsf{trans}}_{\mathcal{P}_{\mathsf{ILC}}}); \end{array}\right] \quad (3)$$

We have shown that when doing this experiment, the probability that $b = 1 \wedge b_{\mathsf{ILC}} = 0$ and the probability that $b = 1 \wedge \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} \neq \widetilde{\mathsf{trans}}_{\mathcal{P}_{\mathsf{ILC}}}$ are both negligible. By knowledge soundness of $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$, the probability that $b_{\mathsf{ILC}} = 1 \wedge (pp, u, w) \notin \mathcal{R}$ is also negligible. Finally, if $\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} = \widetilde{\mathsf{trans}}_{\mathcal{P}_{\mathsf{ILC}}}$ then clearly $w = \widetilde{w}$. Taken together this implies that the probability of $b = 1 \wedge (pp, u, \widetilde{w}) \notin R$ is negligible. We now define $\mathcal{E}^{\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp,u)\rangle}(pp, u)$ to compute $\mathcal{E}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, \mathcal{T}^{\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(pp,u)\rangle}(pp, u))$. The above experiment shows that $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is knowledge sound with $\mathcal{E}$ as extractor. □

Theorem 5 (SHVZK). *If* $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ *is perfect SHVZK and* (Setup, Commit) *is computationally (statistically) hiding then* $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ *is computationally (statistically) SHVZK.*

Proof To prove we have SHVZK we describe how the simulator $\mathcal{S}(pp, u, \rho)$ should simulate the view of $\mathcal{V}$. Along the way, we will argue why, the variables output by $\mathcal{S}$ have the correct joint distribution. To keep the proof readable, instead of saying that "the joint distribution of [random variable] and all previously defined random variables is identical to the distribution in the real view of $\mathcal{V}$ in $\langle\mathcal{P}(pp, u, w) \longleftrightarrow \mathcal{V}(pp, u)\rangle$" we will simply say that "[random variable] has the correct distribution".

Using the randomness ρ the simulator learns the queries $\rho_{\mathsf{ILC}} = (x_1, \ldots, x_{\mu-1}, Q)$ the internal $\mathcal{V}_{\mathsf{ILC}}$ run by the honest $\mathcal{V}$ will send. $\mathcal{S}$ can therefore run $\mathcal{S}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, \rho_{\mathsf{ILC}})$ to simulate the view of the internal $\mathcal{V}_{\mathsf{ILC}}$. This gives it $(t_1, \ldots, t_\mu, V_{(Q)})$. By the SHVZK property of $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ these random variables will all have the correct joint distribution.

Then $\mathcal{S}$ reads the rest of ρ to learn also the challenges $\boldsymbol{\gamma}$ and J that $\mathcal{V}$ will send. The simulator picks uniformly at random $\boldsymbol{v}_{(\boldsymbol{\gamma})} \leftarrow \mathbb{F}^k$. Since in a real proof $\boldsymbol{v}_0$ is chosen at random, we see that the simulated $\boldsymbol{v}_{(\boldsymbol{\gamma})}$ has the correct distribution. Now $\mathcal{S}$ picks $E_{01}|_J, \ldots, E_\mu|_J$ uniformly at random. Recall that we defined $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}; \boldsymbol{r}) = (\mathsf{E}_{\mathcal{C}}(\boldsymbol{v}) + \boldsymbol{r}, \boldsymbol{r})$ and by definition of J being allowed, we have for all $j \in J$ that $j + n \notin J$. This means for any choice of $\boldsymbol{v}_0 \in \mathbb{F}^k$ and $V \in \mathbb{F}^{t \times k}$ that when we choose random $\boldsymbol{r}_0 \leftarrow \mathbb{F}^n$ and $R \leftarrow \mathbb{F}^{t \times n}$ we get uniformly random $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_0; \boldsymbol{r}_0)|_J$ and $\tilde{\mathsf{E}}_{\mathcal{C}}(V; R)$. Consequently, $E_{01}|_J, \ldots, E_\mu|_J$ have the correct distribution.

Next, the simulator picks $\boldsymbol{r}_{(\boldsymbol{\gamma})} \in \mathbb{F}^n$ and $R_{(Q)} \in \mathbb{F}^{t \times n}$ one entry and column at a time. For all j such that $j \notin J$ and $j + n \notin J$ the simulator picks random $(\boldsymbol{r}_{(\boldsymbol{\gamma})})_j \leftarrow \mathbb{F}$ and random $R_j \leftarrow \mathbb{F}^t$. For all j such that $j \in J$ or $j + n \in J$, the simulator then computes the unique $(\boldsymbol{r}_{(\boldsymbol{\gamma})})_j \in \mathbb{F}$ and $R_j \in \mathbb{F}^t$ such that we get $\tilde{\mathsf{E}}_{\mathcal{C}}(\boldsymbol{v}_{(\boldsymbol{\gamma})}; \boldsymbol{r}_{(\boldsymbol{\gamma})}) = \boldsymbol{e}_0|_J + \boldsymbol{\gamma} E|_J$ and $\tilde{\mathsf{E}}_{\mathcal{C}}(V_{(Q)}; R_{(Q)}) = QE|_J$.

Finally, $\mathcal{S}$ defines $E_{01}|_{\bar{J}}, \ldots, E_\mu|_{\bar{J}}$ to be 0 matrices. It then picks $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_\mu$ at random and makes the commitments $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_\mu$ as in the protocol. For $j \in J$ we see that all the $\boldsymbol{c}_i|_j$ commitments are computed as in the real execution from values that have the same distribution as in a real proof. Hence, they will have the correct distribution. The $\boldsymbol{c}_i|_j$s for $j \notin J$ are commitments to different values than in a real proof. However, by the computational (statistical) hiding property of the commitment scheme, they have a distribution that is computationally (statistically) indistinguishable from the correct distribution. □

4.3 Efficiency

We will now estimate the efficiency of a compiled proof of knowledge $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ for $(pp, u, w) \in \mathcal{R}$. Let μ be the number of rounds, $t = \sum_{i=1}^{\mu} t_i$, k, n given in $\mathsf{E}_{\mathcal{C}}$, and qc the query complexity, i.e., $Q \in \mathbb{F}^{\mathrm{qc} \times t}$. Let $T_{\mathcal{P}_{\mathsf{ILC}}}$ be the running time of $\mathcal{P}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u, w)$, $T_{\tilde{\mathsf{E}}_{\mathcal{C}}}(k)$ be the encoding time for a vector in $\mathbb{F}^k$, $T_{\mathsf{Commit}}(t_i)$ be the time to commit to t_i field elements, $T_{\mathrm{Mmul}}(\mathrm{qc}, t, b)$ be the time it takes to multiply matrices in $\mathbb{F}^{\mathrm{qc} \times t}$ and $\mathbb{F}^{t \times b}$, and $T_{\mathcal{V}_{\mathsf{ILC}}}$ is the running time of $\mathcal{V}_{\mathsf{ILC}}(pp_{\mathsf{ILC}}, u)$. Let furthermore C_{ILC} be the communication from the verifier to the prover in $\langle\mathcal{P}_{\mathsf{ILC}} \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}\rangle$, $C_{\mathsf{Commit}}(t_i)$ be the combined size of commitment and randomness for a message consisting of t_i field elements. We give the dominant factors of efficiency of the compiled proof in Fig. 7. The estimates presume $T_{\mathsf{Commit}}(t_1 + 1)$ is not too far from $T_{\mathsf{Commit}}(t_1)$.

Measure	Cost
Prover Computation	$T_{\mathcal{P}_{\mathsf{ILC}}} + t \cdot T_{\tilde{\mathsf{E}}_C}(k) + 2n \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i) + T_{\mathrm{Mmul}}(\mathrm{qc}+1, t, k+n)$
Verifier Computation	$T_{\mathcal{V}_{\mathsf{ILC}}} + (\mathrm{qc}+1) \cdot T_{\tilde{\mathsf{E}}_C}(k) + 2\lambda \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i) + T_{\mathrm{Mmul}}(\mathrm{qc}+1, t, 2\lambda)$
Communication	$C_{\mathsf{ILC}} + 2n \cdot \sum_{i=1}^{\mu} C_{\mathsf{Commit}}(t_i) + (\mathrm{qc}+1) \cdot (k+n) + (\mathrm{qc}+1) \cdot t + 2\lambda \cdot t$
Round Complexity	$\mu + 2$

Fig. 7. Efficiency of a compiled proof of knowledge $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ for $(pp, u, w) \in \mathcal{R}$. Communication is measured in field elements and computation in field operations.

5 Instantiations and Conclusion

Putting together the sequence of proofs and sub-proofs in the ILC model, compiling into the standard model using an error-correcting code and a commitment scheme, and finally instantiating the commitment scheme yields special honest-verifier zero-knowledge proofs for arithmetic circuit satisfiability.

Let us now analyze the efficiency of the compilation we get from Fig. 7. If the error-correcting code is linear-time computable, we get $T_{\tilde{\mathsf{E}}_C}(k) = \mathcal{O}(k)$ operations in $\mathbb{F}$, and with the code from Druk and Ishai [DI14] is will actually be $\mathcal{O}(k)$ additions in $\mathbb{F}$.

Let us now plug in the efficiency of our ILC proof given in Fig. 4 into the efficiency formulas in Fig. 7. We use $k \approx \sqrt{N}$, $n = \mathcal{O}(k)$, $t = \mathcal{O}(\sqrt{N})$, $\mu = \mathcal{O}(\log\log N)$, $\mathrm{qc} = 20 = \mathcal{O}(1)$ and assume $k \gg \lambda$. We then get prover computation $T_{\mathcal{P}} = \mathcal{O}(N)$ multiplications $+ 2n \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i)$, verifier computation $T_{\mathcal{V}} = \mathcal{O}(N)$ additions $+ 2\lambda \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i)$, communication $C = 2n \cdot \sum_{i=1}^{\mu} C_{\mathsf{Commit}}(t_i) + \mathcal{O}(\lambda\sqrt{N})$ field elements, and round complexity $\mu = \mathcal{O}(\log\log N)$.

Instantiating with the commitment scheme from Applebaum et al. [AHI+17] we get computational knowledge soundness and statistical SHVZK. The commitments are compact, a commitment has size $C_{\mathsf{Commit}}(t_i) = \mathrm{poly}(\lambda)$ regardless of the message size, giving us sub-linear communication. The commitments can be computed in linear time at a cost of $T_{\mathsf{Commit}}(t_i) = \mathrm{poly}(\lambda) + \mathcal{O}(t_i)$ additions., giving us linear time computation for prover and verifier.

Instantiating with the commitment from Ishai et al. [IKOS08] we get statistical knowledge soundness and computational SHVZK. The commitments have linear size $C_{\mathsf{Commit}}(t_i) = \mathrm{poly}\lambda + t_i$ giving us linear communication overall. The commitments can be computed in linear time at a cost of $T_{\mathsf{Commit}}(t_i) = \mathrm{poly}(\lambda) + \mathcal{O}(t_i)$ additions, again giving us linear time computation for prover and verifier.

We summarize the costs in Fig. 8 below and conclude that we now have SHVZK proof systems for arithmetic circuit satisfiability where the prover computation only has constant overhead compared to direct computation of the

Measure\Instantiation	Using [AHI$^+$17]	Using [IKOS08]
Prover Computation	$\mathcal{O}(N)$ multiplications in $\mathbb{F}$	$\mathcal{O}(N)$ multiplications in $\mathbb{F}$
Verifier Computation	$\mathcal{O}(N)$ additions in $\mathbb{F}$	$\mathcal{O}(N)$ additions in $\mathbb{F}$
Communication	poly$(\lambda)\sqrt{N}$ field elements	$\mathcal{O}(N)$ field elements
Round Complexity	$\mathcal{O}(\log\log N)$	$\mathcal{O}(\log\log N)$
Completeness	Perfect	Perfect
Knowledge Soundness	Computational	Statistical
SHVZK	Statistical	Computational

Fig. 8. Efficiency of two instantiations of our SHVZK proofs.

arithmetic circuit given the witness. Moreover, the verifier computation is a linear number of *additions*, which is proportional to the time it takes simply to read the instance.

References

[AHI+17] Applebaum, B., Haramaty, N., Ishai, Y., Kushilevitz, E., Vaikuntanathan, V.: Low-complexity cryptographic hash functions. Cryptology ePrint Archive, Report 2017/036 (2017). http://eprint.iacr.org/2017/036

[BCC+16] Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49896-5_12

[BCCT12] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Innovations in Theoretical Computer Science Conference-ITCS. ACM (2012)

[BCCT13] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: ACM Symposium on Theory of Computing-STOC. ACM (2013)

[BCG+13] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_6

[BCI+13] Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Erratum: succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, p. E1. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36594-2_41

[BFM88] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: ACM Symposium on Theory of Computing-STOC. ACM (1988)

[BJY97] Bellare, M., Jakobsson, M., Yung, M.: Round-optimal zero-knowledge arguments based on any one-way function. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 280–305. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_20

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security (ACM CCS). ACM (1993)

[BSCG+16] Ben-Sasson, E., Chiesa, A., Gabizon, A., Riabzev, M., Spooner, N.: Short interactive oracle proofs with constant query complexity, via composition and sumcheck. In: Electronic Colloquium on Computational Complexity (ECCC) (2016)

[BSCGV16] Ben-Sasson, E., Chiesa, A., Gabizon, A., Virza, M.: Quasi-linear size zero knowledge from linear-algebraic PCPs. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 33–64. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_2

[BSCS16] Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53644-5_2

[CD98] Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998). doi:10.1007/BFb0055745

[CDD+16] Cascudo, I., Damgård, I., David, B., Döttling, N., Nielsen, J.B.: Rate-1, linear time and additively homomorphic UC commitments. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 179–207. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53015-3_7

[CDP12] Cramer, R., Damgård, I., Pastro, V.: On the amortized complexity of zero knowledge protocols for multiplicative relations. In: Smith, A. (ed.) ICITS 2012. LNCS, vol. 7412, pp. 62–79. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32284-6_4

[CDS94] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). doi:10.1007/3-540-48658-5_19

[CGM16] Chase, M., Ganesh, C., Mohassel, P.: Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. Cryptology ePrint Archive, Report 2016/583 (2016). http://eprint.iacr.org/2016/583

[Dam00] Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_30

[DI06] Damgård, I., Ishai, Y.: Scalable secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006). doi:10.1007/11818175_30

[DI14] Druk, E., Ishai, Y.: Linear-time encodable codes meeting the Gilbert-Varshamov bound and their cryptographic applications. In: Innovations in Theoretical Computer Science Conference-ITCS. ACM (2014)

[DIK10] Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_23

[FNO15] Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 191–219. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_7

[FS86] Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12

[Gal62] Gallager, R.G.: Low-density parity-check codes. IRE Trans. Inf. Theory **8**(1), 21 (1962)

[GGI+14] Gentry, C., Groth, J., Ishai, Y., Peikert, C., Sahai, A., Smith, A.: Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. J. Cryptol. (2014)

[GGPR13] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_37

[GI01] Guruswami, V., Indyk, P.: Expander-based constructions of efficiently decodable codes. In: Symposium on Foundations of Computer Science-FOCS. IEEE Computer Society (2001)

[GI02] Guruswami, V., Indyk, P.: Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In: ACM Symposium on Theory of Computing-STOC. ACM (2002)

[GI03] Guruswami, V., Indyk, P.: Linear time encodable and list decodable codes. In: ACM Symposium on Theory of Computing-STOC. ACM (2003)

[GI05] Guruswami, V., Indyk, P.: Linear-time encodable/decodable codes with near-optimal rate. IEEE Trans. Inf. Theory **51**(10), 3393 (2005)

[GKR08] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: ACM Symposium on Theory of Computing-STOC. ACM (2008)

[GMR85] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: ACM Symposium on Theory of Computing-STOC. ACM (1985)

[GQ88] Guillou, L.C., Quisquater, J.-J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Barstow, D., et al. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988). doi:10.1007/3-540-45961-8_11

[Gro04] Groth, J.: Honest verifier zero-knowledge arguments applied. BRICS (2004)

[Gro09] Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_12

[Gro10] Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17373-8_19

[Gro16] Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49896-5_11

[GSV98] Goldreich, O., Sahai, A., Vadhan, S.: Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In: ACM Symposium on Theory of Computing-STOC. ACM (1998)

[HM96] Halevi, S., Micali, S.: Practical and provably-secure commitment schemes from collision-free hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 201–215. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_16

[HMR15] Hu, Z., Mohassel, P., Rosulek, M.: Efficient zero-knowledge proofs of non-algebraic statements with sublinear amortized cost. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 150–169. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_8

[IKOS08] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: ACM Symposium on Theory of Computing-STOC. ACM (2008)

[IKOS09] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. **39**(3), 1121 (2009)

[JKO13] Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: ACM Conference on Computer and Communications Security (ACM CCS). ACM (2013)

[Kil92] Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: ACM Symposium on Theory of Computing-STOC. ACM (1992)

[KR08] Kalai, Y.T., Raz, R.: Interactive PCP. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 536–547. Springer, Heidelberg (2008). doi:10.1007/978-3-540-70583-3_44

[MP03] Micciancio, D., Petrank, E.: Simulatable commitments and efficient concurrent zero-knowledge. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 140–159. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_9

[MRS17] Mohassel, P., Rosulek, M., Scafuro, A.: Sublinear zero-knowledge arguments for RAM programs. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 501–531. Springer, Cham (2017). doi:10.1007/978-3-319-56620-7_18

[PHGR13] Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: IEEE Symposium on Security and Privacy. IEEE Computer Society (2013)

[Sch91] Schnorr, C.-P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161 (1991)

[Spi95] Spielman, D.A.: Linear-time encodable and decodable error-correcting codes. In: ACM Symposium on Theory of Computing-STOC. ACM (1995)