# Linear Algebra with Sub-linear Zero-Knowledge Arguments

Jens Groth*

University College London
E-mail: j.groth@ucl.ac.uk

**Abstract.** We suggest practical sub-linear size zero-knowledge arguments for statements involving linear algebra. Given commitments to matrices over a finite field, we give a sub-linear size zero-knowledge argument that one committed matrix is the product of two other committed matrices. We also offer a sub-linear size zero-knowledge argument for a committed matrix being equal to the Hadamard product of two other committed matrices. Armed with these tools we can give many other sub-linear size zero-knowledge arguments, for instance for a committed matrix being upper or lower triangular, a committed matrix being the inverse of another committed matrix, or a committed matrix being a permutation of another committed matrix.
A special case of what can be proved using our techniques is the satisfiability of an arithmetic circuit with $N$ gates. Our arithmetic circuit zero-knowledge argument has a communication complexity of $O(\sqrt{N})$ group elements. We give both a constant round variant and an $O(\log N)$ round variant of our zero-knowledge argument; the latter has a computation complexity of $O(N/\log N)$ exponentiations for the prover and $O(N)$ multiplications for the verifier making it efficient for the prover and very efficient for the verifier. In the case of a binary circuit consisting of NAND-gates we give a zero-knowledge argument of circuit satisfiability with a communication complexity of $O(\sqrt{N})$ group elements and a computation complexity of $O(N)$ multiplications for both the prover and the verifier.

**Keywords:** Sub-linear size zero-knowledge arguments, public-coin special honest verifier zero-knowledge, Pedersen commitments, linear algebra, circuit satisfiability.

## 1 Introduction

It has long been known [Kil92] that zero-knowledge arguments (with computational soundness) can have very low communication. However, known examples of communication-efficient zero-knowledge arguments tend to get their efficiency at the cost of increased computational complexity. Obtaining zero-knowledge arguments that are efficient with respect to *both* communication *and* computation is considered one of the important challenges in theoretical computer science [Joh00]. We address this challenge by constructing zero-knowledge arguments for statements related to linear algebra over finite fields that have sub-linear communication and at the same time also have low computational complexity.

### 1.1 Our Contribution

We use a variant of Pedersen commitments [Ped91] to give a public-coin zero-knowledge argument for the satisfiability of a binary circuit with $N$ gates that requires communicating $O(\sqrt{N})$ group elements. We give both a constant round zero-knowledge argument and an $O(\log N)$-round zero-knowledge argument. The latter has a computation complexity of $O(N)$ *multiplications* for both the prover and the verifier. In comparison, the natural Pedersen commitment based zero-knowledge argument for circuit satisfiability where one commits to each wire of the circuit and proves for each gate that the committed output-wire is correct has a communication complexity of $O(N)$ group elements and a computation complexity of $O(N)$ exponentiations for both the prover and the verifier.

The efficient zero-knowledge argument for circuit satisfiability is a special case of a more general result related to statements arising in linear algebra. We consider row vectors of elements from a finite field $\mathbb{Z}_p$, where $p$ is a large prime. Using a generalization of the Pedersen commitment we can commit to a vector of

* Part of this research was done while visiting IPAM, UCLA.

$n$ elements from $\mathbb{Z}_p$. Each commitment consists of a single group element. A set of $n$ commitments, can be considered a commitment to the $n$ rows of an $n \times n$ matrix. This paper is about zero-knowledge arguments for a set of committed vectors and matrices satisfying a set of linear algebra relations, for instance that a committed matrix is the product of two other committed matrices.

We give zero-knowledge arguments that communicate only $O(\sqrt{N})$ elements. In addition, the arguments are computationally efficient for both the prover and the verifier. The verifier is a public-coin verifier and does not need to take much action until the end of the argument, where the small size of the arguments makes it possible to verify the correctness using only little computation.

Our sub-linear size zero-knowledge arguments work for a wide range of linear algebra relations. We can commit to single field elements, vectors of field elements and square matrices of field elements. Our results also hold for non-square matrices, however, for simplicity we focus just on square matrices here. Given commitments to field elements, vectors and matrices we can prove relations such as a committed field element being the dot product of two committed vectors, a committed matrix being the product of two other committed matrices, or a committed vector being the Hadamard product (the entry-wise product) of two other vectors. Being able to prove such linear algebra relations makes it possible to address many other statements frequently arising in linear algebra. We can for instance prove that committed matrices are upper or lower triangular, have a particular trace, compute the sums of the rows or columns or prove that a committed matrix is the inverse of another committed matrix. We can also permute the entries of a matrix using either a public or a hidden permutation. Using the linear algebra relations, we also get sub-linear size zero-knowledge arguments for the satisfiability of arithmetic circuits and for the satisfiability of binary circuits demonstrating the generality of our results.

### 1.2 Related Work

Recent work on zero-knowledge proofs [IKOS07] give us proofs with a communication complexity that grows linearly in the size of the statement to be proven and [IKOS07,KR08,GKR08] also give us proofs with size that depend quasi-linearly on the witness-length. If we consider arguments, the communication complexity can be even lower and Kilian [Kil92] gave a zero-knowledge argument for circuit satisfiability with polylogarithmic communication. His argument goes through the PCP-theorem [AS98,ALM$^+$98,Din07] and uses a collision-free hash-function to build a hash-tree that includes the entire PCP though. Even with the best PCP constructions known to date [BSGH$^+$05] Kilian's argument has high computational complexity for practical parameters. In contrast, our goal is to get short zero-knowledge arguments that are simple and efficient enough for both prover and verifier to be used in practice and we give precise performance estimates in Section 8.

Groth and Ishai [GI08] gave a zero-knowledge argument for correctness of a shuffle [Cha81] of $N$ ElGamal ciphertexts. We rely on techniques developed by Groth and Ishai and as described below also develop several new techniques. For comparison, we believe it would be possible to modify their argument into an argument for circuit satisfiability with a sub-linear communication of $O(N^{2/3})$ group elements, but the corresponding computational complexity for the prover would be a super-linear number of exponentiations.

### 1.3 Our Techniques

The generality of our results relies on using randomization and batch-verification techniques to reduce all the linear algebra relations we talked above to equations of the form

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i,$$

where $\boldsymbol{x}_i, \boldsymbol{y}_i$ are committed vectors with $n$ elements in $\mathbb{Z}_p$, and $z$ is a committed field element, and $* : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p$ is a bilinear map. Besides greatly simplifying the task, the bilinear map also helps reduce computation because it maps pairs of $n$-element vectors into single field elements giving the prover less to commit to.

Groth and Ishai [GI08] gave a sub-linear size public-coin zero-knowledge argument for the correctness of a shuffle of $N$ ElGamal ciphertexts, with a communication complexity of $O(N^{2/3})$ group elements. We use similar techniques but by making more careful use of the public-coin challenges, we can reduce the communication complexity for our zero-knowledge arguments to $O(\sqrt{N})$ elements. The difference from Groth and Ishai's work is that they choose a set of challenges at random, whereas we let the prover process the verifier's challenges to get a more structured set of challenges. This processing consists of taking a challenge $e \in \mathbb{Z}_p$ from the verifier and using it to generate a set of challenges $(1, e, e^2, \ldots)$, which is a row of a Vandermonde matrix. In the zero-knowledge argument, we then arrange the challenges from the Vandermonde vector in such a way that it leads to many terms cancelling out with each other.

Groth and Ishai's shuffle argument suffered from an increase in the prover's computation complexity in comparison with shuffle arguments that do not have sub-linear size. The same effects apply to some extent to our zero-knowledge arguments when using a constant number of rounds, however, by allowing a logarithmic number of rounds we can eliminate the computational overhead. This is of interest in scenarios where round complexity matters less than computation, for instance in cases where the Fiat-Shamir heuristic is used to make the zero-knowledge argument non-interactive by letting the prover use a cryptographic hash-function to compute the verifier's challenges.

## 2 Preliminaries

Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(\kappa) \approx g(\kappa)$ when $|f(\kappa) - g(\kappa)| = O(\kappa^{-c})$ for every constant $c$. We say that $f$ is *negligible* when $f(\kappa) \approx 0$ and that it is *overwhelming* when $f(\kappa) \approx 1$.

We write $y = A(x; r)$ when the algorithm $A$, on input $x$ and randomness $r$, outputs $y$. We write $y \leftarrow A(x)$ for the process of picking randomness $r$ at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling $y$ uniformly at random from the set $S$. We write $(e_1, \ldots, e_m) \leftarrow \mathrm{Van}_m(\mathbb{Z}_p)$ when we pick $e \leftarrow \mathbb{Z}_p$ and define $e_1, \ldots, e_m$ by $e_i = e^{i-1} \bmod p$, corresponding to $(e_1, \ldots, e_m)$ being a row of a Vandermonde matrix.

## 3 Honest Verifier Zero-Knowledge Arguments

Consider a triple of probabilistic polynomial time interactive algorithms $(K, P, V)$ called the common reference string generator, the prover and the verifier. The common reference string generator takes the security parameter $\kappa$ as input in unary and generates a common reference string $\sigma$. In our case, the common reference string is a public key for a commitment scheme, see Section 3.1. In the examples we have in mind the commitment key can be chosen as a random string, placing us in the common random string model and it may even be possible to let the verifier choose it, placing us in the plain model. A natural generalization is to allow $K$ to take as input a description of a prime or a group and generate the common reference string on top of this input.

Let $R$ be a polynomial time decidable ternary relation. For a statement $x$ we call $w$ a witness if $(\sigma, x, w) \in R$. We define a corresponding common reference string dependent language $L_\sigma$ consisting of elements $x$ that have a witness $w$ such that $(\sigma, x, w) \in R$. This is a natural generalization of NP-languages; when $\sigma$ is empty we have the standard notion of an NP-language.

We write $\mathrm{tr} \leftarrow \langle P(x), V(y) \rangle$ for the public transcript produced by $P$ and $V$ when interacting on inputs $x$ and $y$. This transcript ends with $V$ either accepting or rejecting. We sometimes shorten the notation by saying $\langle P(x), V(y) \rangle = b$, where $b = 0$ corresponds to $V$ rejecting and $b = 1$ corresponds to $V$ accepting.

**Definition 1 (Argument).** *The triple $(K, P, V)$ is called an* argument *for relation $R$ with perfect completeness if for all non-uniform polynomial time interactive adversaries $\mathcal{A}$ we have*

**Perfect completeness:**

$$\Pr\left[\sigma \leftarrow K(1^\kappa); (x, w) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ or } \langle P(\sigma, x, w), V(\sigma, x)\rangle = 1\right] = 1.$$

**Soundness:**

$$\Pr\left[\sigma \leftarrow K(1^\kappa); x \leftarrow \mathcal{A}(\sigma) : x \notin L_\sigma \text{ and } \langle \mathcal{A}, V(\sigma, x)\rangle = 1\right] \approx 0.$$

**Definition 2 (Public coin argument).** *An argument $(K, P, V)$ is* public coin *if the verifier's messages are chosen uniformly at random independently of the messages sent by the prover.*

We define special honest verifier zero-knowledge (SHVZK) [CDS94] for a public coin argument as the ability to simulate the transcript for any set of challenges without access to the witness.

**Definition 3 (Perfect special honest verifier zero-knowledge).** *The public coin argument $(K, P, V)$ is a perfect special honest verifier zero-knowledge argument for $R$ if there exists a probabilistic polynomial time simulator $S$ such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\begin{aligned}&\Pr\left[\sigma \leftarrow K(1^\kappa); (x, w, \rho) \leftarrow \mathcal{A}(\sigma); \mathrm{tr} \leftarrow \langle P(\sigma, x, w), V(\sigma, x; \rho)\rangle : (\sigma, x, w) \in R \text{ and } \mathcal{A}(\mathrm{tr}) = 1\right]\\ =\ &\Pr\left[\sigma \leftarrow K(1^\kappa); (x, w, \rho) \leftarrow \mathcal{A}(\sigma); \mathrm{tr} \leftarrow S(\sigma, x, \rho) : (\sigma, x, w) \in R \text{ and } \mathcal{A}(\mathrm{tr}) = 1\right].\end{aligned}$$

WITNESS-EXTENDED EMULATION. We shall define an argument of knowledge through witness-extended emulation [Gro04,Lin03]. Informally, the definition says: given an adversary that produces an acceptable argument with probability $\epsilon$, there exists an emulator that produces a similar argument with probability $\epsilon$, but at the same time provides a witness.

**Definition 4 (Witness-extended emulation).** *We say the public coin argument $(K, P, V)$ has witness-extended emulation if for all deterministic polynomial time $P^*$ there exists an expected polynomial time emulator $E$ such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\begin{aligned}&\Pr\left[\sigma \leftarrow K(1^\kappa); (x, s) \leftarrow \mathcal{A}(\sigma); \mathrm{tr} \leftarrow \langle P^*(\sigma, x, s), V(\sigma, x)\rangle : \mathcal{A}(\mathrm{tr}) = 1\right]\\ \approx\ &\Pr\Big[\sigma \leftarrow K(1^\kappa); (x, s) \leftarrow \mathcal{A}(\sigma); (\mathrm{tr}, w) \leftarrow E^{\langle P^*(\sigma,x,s), V(\sigma,x)\rangle}(\sigma, x) :\\ &\qquad \mathcal{A}(\mathrm{tr}) = 1 \text{ and if } \mathrm{tr} \text{ is accepting then } (\sigma, x, w) \in R\Big],\end{aligned}$$

*where $E$ has access to a transcript oracle $\langle P^*(\sigma, x, s), V(\sigma, x)\rangle$ that can be rewound to a particular round and run again with the verifier using fresh randomness.*

We think of $s$ as being the state of $P^*$, including the randomness. Then we have an argument of knowledge in the sense that the emulator can extract a witness whenever $P^*$ is able to make a convincing argument. This shows that the definition implies soundness. We remark that the verifier's randomness is part of the transcript and the prover is deterministic. So combining the emulated transcript with $\sigma, x, s$ gives us the view of both the prover and the verifier and at the same time gives us the witness.

We remark that the standard definition of *proofs* of knowledge by Bellare and Goldreich [BG92] does not apply in our setting, since we work in the common reference string model and are interested in *arguments* of knowledge; see Damgård and Fujisaki [DF02] for a discussion of this issue and an alternative definition of knowledge soundness. Witness-extended emulation implies knowledge soundness [Gro04].

FULL ZERO-KNOWLEDGE. For simplicity, we focus on SHVZK arguments in this paper. There are very efficient standard techniques [Dam00,GMY06,Gro04] to convert a SHVZK argument into a public-coin full zero-knowledge argument with a cheating verifier when a common reference string is available. If we work in the plain model and let the verifier choose the common reference string as described earlier, we can use coin-flipping techniques (for the full zero-knowledge property the coin-flips should be simulatable against a dishonest verifier) for the challenges to get private-coin[1] full zero-knowledge arguments against a cheating verifier. Challenges in our SHVZK arguments are just a few field elements so both in the case with and without a common reference string the overhead of getting full zero-knowledge is insignificant compared to the cost of the SHVZK arguments themselves.

### 3.1 Homomorphic Commitments

The central tool in our SHVZK arguments is a homomorphic commitment to $n$ elements in $\mathbb{Z}_p$, where $p$ is a $\kappa$-bit prime. Any homomorphic commitment scheme can be used, but for simplicity and for the sake of making a concrete efficiency analysis, we will in this paper use a generalization of Pedersen commitments [Ped91]. This commitment scheme is length-reducing; a commitment is a single group element no matter how large $n$ is. The length-reduction is crucial, by working on short commitments instead of long vectors we get SHVZK arguments with sub-linear communication complexity.

The generalized Pedersen commitment scheme works as follows. The key generation algorithm $K$ generates a commitment key $ck = (G, g_1, \ldots, g_n, h)$, where $g_1, \ldots, g_n, h$ are randomly chosen generators of a group $G$ of prime order $p$ with $|p| = \kappa$. The message space is $\mathbb{Z}_p^n$, the randomizer space is $\mathbb{Z}_p$ and the commitment space is $G$. We require $G$ to be a group where it is easy to determine membership and compute the binary operations and assume parties check that commitments are valid, by checking $c \in G$.[2]

To commit to a vector $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ we pick randomness $r \leftarrow \mathbb{Z}_p$ and compute the commitment $c = h^r \prod_{i=1}^n g_i^{x_i}$. As a matter of notation we will write $\mathrm{com}_{ck}(\boldsymbol{x}; r)$ when committing to a vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ using randomness $r$. In some cases we will commit to less than $n$ elements; this can be accomplished quite easily by setting the remaining messages to 0. When committing to a single element $x \in \mathbb{Z}_p$ using randomness $r$, we write $\mathrm{com}_{ck}(x; r)$. The generalized Pedersen commitment is perfectly hiding since no matter what the messages are, the commitment is uniformly distributed in $G$. The commitment is computationally binding under the discrete logarithm assumption; we will skip the simple proof.

The common reference string in our SHVZK arguments will be a commitment key $ck$. We remark that for typical choices of the group $G$, the commitment key can be easily sampled from a common random string and it is easy to verify that $ck$ is a valid commitment key. It may even be chosen by the verifier, provided the prover and verifier use a $p$ for which it is possible to generate groups where the discrete logarithm problem is hard.

The generalized Pedersen commitment is homomorphic. For all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}_p^n$ and $r, r' \in \mathbb{Z}_p$ we have

$$\mathrm{com}_{ck}(\boldsymbol{x}; r) \cdot \mathrm{com}_{ck}(\boldsymbol{x}'; r') = \mathrm{com}_{ck}(\boldsymbol{x} + \boldsymbol{x}'; r + r').$$

KNOWLEDGE OF CONTENT OF COMMITMENTS. There are standard techniques for proving knowledge of the opening of many commitments, see Appendix A. This can be done in 3 rounds and costs little in terms of communication and computation. Therefore, we will for simplicity and without loss of generality often assume without explicitly stating it that the prover knows the openings of the commitments that she sends to the verifier.

[1] Goldreich and Krawczyk [GK96] have shown that only languages in BPP have constant-round public-coin arguments.

[2] If the commitments belong to a group $\mathbb{Z}_q^*$ batch verification techniques can be used to lower the cost of checking group membership of many commitments. See also [Gro03] for a variant of the Pedersen commitment scheme over $\mathbb{Z}_q^*$ that makes it possible to almost eliminate the cost of verifying validity. If $G$ is an elliptic curve of order $p$, then the validity check just consists of checking that $c$ is a point on the curve, which is inexpensive.

### 3.2 Multi-exponentiation Techniques

Multi-exponentiation techniques allow computing products of the form $\prod_{i=1}^{n} g_i^{x_i}$ faster than computing $n$ single exponentiations. Multi-exponentiations appear frequently in the paper, for instance when computing the generalized Pedersen commitment described earlier. Pippenger [Pip80] developed a general theory of multi-exponentiations; we recommend Lim's presentation [Lim00] of concrete multi-exponentiation techniques with a complexity of less than $2n\kappa/\log n$ multiplications in $G$, when $n$ is large.

### 3.3 The Schwartz-Zippel Lemma

For completeness we state a variation of the well-known Schwartz-Zippel lemma that we will use several times in the paper.

**Lemma 1 (Schwartz-Zippel).** *Let* poly *be a non-zero multivariate polynomial of degree* $d$ *over* $\mathbb{Z}_p$*, then the probability of* $\mathrm{poly}(x_1, \ldots, x_m) = 0$ *for randomly chosen* $x_1, \ldots, x_m \leftarrow \mathbb{Z}_p$ *is at most* $d/p$*.*

The Schwartz-Zippel lemma is frequently used in polynomial identity testing. Given two multi-variate polynomials $\mathrm{poly}_1$ and $\mathrm{poly}_2$ we can test whether $\mathrm{poly}_1(x_1, \ldots, x_m) - \mathrm{poly}_2(x_1, \ldots, x_m) = 0$ for random $x_1, \ldots, x_m \leftarrow \mathbb{Z}_p$. If the two polynomials are identical this will always be true, whereas if the two polynomials are different then there is only probability $\max(d_1, d_2)/p$ for the equality to hold.

## 4 Equations with Matrices and Vectors

We wish to commit to matrices and vectors of elements from $\mathbb{Z}_p$ and make SHVZK arguments for them satisfying equations commonly arising in linear algebra. We first consider the following 6 types of equations over committed matrices $X_i, Y_i, Z \in \mathrm{Mat}_{n\times n}(\mathbb{Z}_p)$, committed row vectors $\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z} \in \mathbb{Z}_p^n$ and committed elements $z \in \mathbb{Z}_p$, with public $a_i \in \mathbb{Z}_p$.

$$\begin{array}{lll} \boldsymbol{z}^\top = \sum_{i=1}^{m} a_i X_i \boldsymbol{y}_i^\top & Z = \sum_{i=1}^{m} a_i X_i Y_i & Z = \sum_{i=1}^{m} a_i X_i \circ Y_i \\ z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top & \boldsymbol{z} = \sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i & \boldsymbol{z} = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \circ \boldsymbol{y}_i, \end{array}$$

where $\circ$ is the Hadamard product (entry-wise product). In this section, we will show how to reduce a set of such equations to a couple of equations of the form

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i,$$

where $* : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p$ is a bilinear map. One bilinear map we will use is the standard dot product of vectors $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^\top$. Another bilinear map we will use is given by $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}(\boldsymbol{y} \circ \boldsymbol{t})^\top$, where $\boldsymbol{t} \in \mathbb{Z}_p^n$ is a public vector chosen by the verifier.

The first step in the reduction is very simple. Since we have committed to row vectors, the three types of equations in the top involving matrices $X_i$ are actually just sets of $n$ equations of the types below. We can therefore focus on the three types of equations on the bottom.

### 4.1 Reducing Many Equations of the Form $z = \sum_{i=1}^{m} a_i x_i y_i^\top$ to a Single Equation

Randomization can be used to reduce $Q$ equations of the form $z_q = \sum_{i=1}^{m_q} a_{qi} \boldsymbol{x}_{qi} \boldsymbol{y}_{qi}^\top$ to one single equation of the form $z = \sum_{i=1}^{m} \boldsymbol{z}_i \boldsymbol{y}_i^\top$, where $m = \sum_{q=1}^{Q} m_q$. The verifier selects $(r_1, \ldots, r_Q) \leftarrow \mathrm{Van}_Q(\mathbb{Z}_p)$ (observe this only requires the verifier to transmit one field element) and require the prover to demonstrate

$$\sum_{q=1}^{Q} r_q z_q = \sum_{q=1}^{Q} \sum_{i=1}^{m_q} (r_q a_{qi} \boldsymbol{x}_{qi}) \boldsymbol{y}_{qi}^\top.$$

This is a comparison of two degree $Q-1$ polynomials in the challenge consisting of a field element. By the Schwartz-Zippel lemma, there is probability at most $\frac{Q-1}{p}$ for the test to pass unless indeed all the equations hold. Setting $z = \sum_{q=1}^{Q} r_q z_q$ and $\boldsymbol{x}'_{qi} = r_q a_{qi} \boldsymbol{x}_{qi}$, whose commitments can easily be computed using the homomorphic property of the commitment scheme, we get the following equation of the desired form

$$z = \sum_{q=1}^{Q} \sum_{i=1}^{m_q} \boldsymbol{x}'_{qi} \boldsymbol{y}_{qi}^\top.$$

### 4.2 Reducing $\boldsymbol{z} = \sum_{i=1}^{m} \boldsymbol{a_i x_i Y_i}$ to the form $\boldsymbol{z} = \sum_{i=1}^{m} \boldsymbol{a_i x_i y_i^\top}$

We will now give a 3-move reduction of $\boldsymbol{z} = \sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i$ to the form $z = \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top$. The verifier picks $\boldsymbol{t} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and asks the prover to demonstrate

$$\boldsymbol{z}\boldsymbol{t}^\top = (\sum_{i=1}^{m} a_i \boldsymbol{x}_i Y_i)\boldsymbol{t}^\top = \sum_{i=1}^{m} a_i \boldsymbol{x}_i (Y_i \boldsymbol{t}^\top).$$

By the Schwartz-Zippel lemma, there is at most probability $\frac{n-1}{p}$ of this test passing unless indeed the values satisfy the equation. The problem is that the verifier does not have a straightforward way to compute commitments to $Y_i \boldsymbol{t}^\top$ since we have commitments to the rows of the matrices, but here the verifier is asking for a linear combination of the columns. Choosing $\boldsymbol{t}$ and sending it to the prover is therefore only the first round of the reduction; there will be two more rounds.

For each matrix $Y_i$ the prover creates a new commitment to $\boldsymbol{y}_i = \boldsymbol{t} Y_i^\top$ and sends it to the verifier. The equation can now be reduced to the form

$$\boldsymbol{z}\boldsymbol{t}^\top - \sum_{i=1}^{m} a_i \boldsymbol{x}_i \boldsymbol{y}_i^\top = 0,$$

which is of the desired form. In the process we have for each matrix $Y_i$ introduced an additional equation $\boldsymbol{y}_i = \boldsymbol{t} Y_i^\top$ that we need to prove too. We pick $\boldsymbol{s} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and ask the prover to demonstrate

$$\boldsymbol{y}_i \boldsymbol{s}^\top = (\boldsymbol{s} Y_i)\boldsymbol{t}^\top.$$

This is the key idea in this reduction, $\boldsymbol{s} Y_i$ is a combination of row vectors from $Y_i$ and thus easily computable. Using the homomorphic properties of the commitment scheme both the prover and the verifier can compute a commitment to $\boldsymbol{s} Y_i$.

We remark that since the last step in this reduction simply consists of the verifier picking a challenge $\boldsymbol{s}$, we can run the last round in parallel with the reduction in Section 4.1, so our reduction only costs 2 additional rounds. Further, we note that for all $Y_i$ in all equations, we can use the same $\boldsymbol{s}$ and $\boldsymbol{t}$. In the randomization step in the reduction in Section 4.1 we can use the homomorphic properties of the commitment scheme to combine all the vectors that we combine with respectively $\boldsymbol{s}$ and $\boldsymbol{t}$. The main cost of the reduction is therefore the computation of the $\boldsymbol{y}_i$'s and the $\boldsymbol{s} Y_i$'s and the commitments to $\boldsymbol{y}_i$, the rest has modest cost.

### 4.3 Reducing Equations with Hadamard Products to a Single Equation with a Bilinear Map

We will now reduce a set of $Q$ Hadamard equations of the form

$$\boldsymbol{z}_q = \sum_{i=1}^{m_q} a_{qi} \boldsymbol{x}_{qi} \circ \boldsymbol{y}_{qi}$$

to a single equation. The verifier picks $(r_1, \ldots, r_Q) \leftarrow \mathrm{Van}_Q(\mathbb{Z}_p)$ and requires the prover to give an argument for

$$\sum_{q=1}^{Q} r_q \boldsymbol{z}_q = \sum_{q=1}^{Q} \sum_{i=1}^{m_q} (r_q a_{qi} \boldsymbol{x}_{qi}) \circ \boldsymbol{y}_{qi}.$$

Setting $\boldsymbol{x}'_{qi} = r_q a_{qi} \boldsymbol{x}_{qi}$ and $\boldsymbol{z}' = \sum_{q=1}^{Q} r_q \boldsymbol{z}_q$, whose commitments can be computed using the homomorphic properties, this gives us the equation $\boldsymbol{z}' = \sum_{q=1}^{Q} \sum_{i=1}^{m_q} \boldsymbol{x}'_{qi} \circ \boldsymbol{y}_{qi}$.

Consider now a Hadamard equation of the form $\boldsymbol{z} = \sum_{i=1}^{m} \boldsymbol{x}_i \circ \boldsymbol{y}_i$. We can simplify this equation by picking $\boldsymbol{t} \leftarrow \mathrm{Van}_m(\mathbb{Z}_p)$ and requiring the prover to show

$$\boldsymbol{z}\boldsymbol{t}^\top = (\sum_{i=1}^{m} \boldsymbol{x}_i \circ \boldsymbol{y}_i)\boldsymbol{t}^\top = \sum_{i=1}^{m} \boldsymbol{x}_i (\boldsymbol{y}_i \circ \boldsymbol{t})^\top.$$

Defining the bilinear map

$$* : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p \qquad\qquad (\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x}(\boldsymbol{y} \circ \boldsymbol{t})^\top,$$

we have reduced the equation to

$$0 = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i - \boldsymbol{z} * \mathbf{1},$$

where $\mathbf{1}$ is the vector $(1, \ldots, 1)$.

## 5 SHVZK Arguments for a Vector Product Equation

We saw in the previous section that equations involving matrices and vectors could be efficiently reduced to an equation of the form

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i,$$

where $*$ is one of the two bilinear maps $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^\top$ or $\boldsymbol{x} * \boldsymbol{y} = \boldsymbol{x}(\boldsymbol{y} \circ \boldsymbol{t})^\top$. In this section we will give a SHVZK argument of knowledge of openings $z \in \mathbb{Z}_p$ and $\boldsymbol{x}_1, \boldsymbol{y}_1, \ldots, \ldots, \boldsymbol{x}_m, \boldsymbol{y}_m \in \mathbb{Z}_p^n$ satisfying such an equation.

### 5.1 The Minimal Case

We first consider the minimal case $m = 1$. We have three commitments $a, b, c$ to $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^n$ and $z \in \mathbb{Z}_p$ respectively and the prover wants to convince the verifier that $z = \boldsymbol{x} * \boldsymbol{y}$. The prover's private input in the argument consists of the openings $(\boldsymbol{x}, r), (\boldsymbol{y}, s)$ and $(z, t)$ of $a, b$ and $c$ respectively.

**P**$\rightarrow$ **V:** Pick $\boldsymbol{d}_x, \boldsymbol{d}_y \leftarrow \mathbb{Z}_p^n, d_z \leftarrow \mathbb{Z}_p$ and randomizers $r_d, s_d, t_1, t_0 \leftarrow \mathbb{Z}_p$.

Send to the verifier the following commitments

$$a_d = \mathrm{com}_{ck}(\boldsymbol{d}_x; r_d) \quad b_d = \mathrm{com}_{ck}(\boldsymbol{d}_y; s_d) \quad c_1 = \mathrm{com}_{ck}(\boldsymbol{x} * \boldsymbol{d}_y + \boldsymbol{d}_x * \boldsymbol{y}; t_1) \quad c_0 = \mathrm{com}_{ck}(\boldsymbol{d}_x * \boldsymbol{d}_y; t_0).$$

**P**$\leftarrow$ **V:** Send challenge $e \leftarrow \mathbb{Z}_p$ to the prover.
**P**$\rightarrow$ **V:** Send to the verifier the following answer

$$\boldsymbol{f}_x = e\boldsymbol{x} + \boldsymbol{d}_x \qquad \boldsymbol{f}_y = e\boldsymbol{y} + \boldsymbol{d}_y \qquad r_x = er + r_d \qquad s_y = es + s_d \qquad t_z = e^2 t + et_1 + t_0.$$

**V:** Accept the argument if

$$a^e a_d = \mathrm{com}_{ck}(\boldsymbol{f}_x; r_x) \quad \wedge \quad b^e b_d = \mathrm{com}_{ck}(\boldsymbol{f}_y; s_y) \quad \wedge \quad c^{e^2} c_1^e c_0 = \mathrm{com}_{ck}(\boldsymbol{f}_x * \boldsymbol{f}_y; t_z).$$

**Theorem 1.** *There is a* 3*-move public-coin argument of knowledge of committed values* $\boldsymbol{x}, \boldsymbol{y}, z$ *so* $z = \boldsymbol{x} * \boldsymbol{y}$*. The argument has perfect completeness, perfect SHVZK and witness-extended emulation.*

*Proof.* It is straightforward to verify that the protocol above is a 3-round public-coin protocol with perfect completeness.

To see that it has perfect SHVZK, consider a simulator $S$ that on input $a, b, c$ and challenge $e$ picks $\boldsymbol{f}_x, \boldsymbol{f}_y, r_x, s_y, t_z$ and $c_1 \leftarrow \mathrm{com}_{ck}(0)$ at random and computes $a_d = \mathrm{com}_{ck}(\boldsymbol{f}_x; r_x)a^{-e}, b_d = \mathrm{com}_{ck}(\boldsymbol{f}_y; s_y), c_0 = \mathrm{com}_{ck}(\boldsymbol{f}_x * \boldsymbol{f}_y; t_z)c_1^{-e}c^{-e^2}$. The simulated argument is $(a_d, b_d, c_1, c_0, e, \boldsymbol{f}_x, \boldsymbol{f}_y, r_x, s_y, t_z)$. To see that it is perfect SHVZK observe that also in a real argument, $\boldsymbol{f}_x, \boldsymbol{f}_y, r_x, s_y, t_z$ are random and $c_1$ is random because the commitment scheme is perfectly hiding. Conditioned on these values, the commitment $a_d, b_d, c_0$ are uniquely determined both in a real argument and in a simulated argument. We conclude, real arguments and simulated arguments have the same probability distribution.

To see that we have witness-extended emulation, we consider an emulator that runs the prover and verifier with random challenges to get a transcript. If the transcript contains a rejecting argument, it simply outputs the transcript. If the transcript is accepting, it rewinds and runs the prover and verifier again with random challenges until it gets two more accepting transcripts.

Let us argue that this part of the emulation takes expected polynomial time. Consider the situation after the prover has sent the initial commitments to the verifier and has probability $\epsilon$ of successfully answering a random challenge. We only enter the extraction part if the first transcript is accepting, so with probability $1 - \epsilon$ we will not need to extract anything (in particular if $\epsilon = 0$ we will not spend time on extraction). If we do get an accepting transcript, we rewind an expected $\frac{2}{\epsilon}$ times to get two more accepting transcripts. Overall, we therefore expect $\epsilon \cdot \frac{2}{\epsilon} = 2$ rewinds when running the emulator, giving us an expected polynomial running time.

Since the emulator runs in expected polynomial time, there is overwhelming probability that it either fails to produce an accepting transcript or alternatively that it produces three accepting transcripts with different challenges (or in other words, there is negligible chance of ending with two or more transcripts that have identical challenges, in which case we would be stuck). In the latter case, it will extract the openings of $a, b, c, a_d, b_d, c_1, c_0$ as follows. Let the three accepting transcripts be $(a_d, b_d, c_1, c_0, e, \boldsymbol{f}_x, \boldsymbol{f}_y, r_x, s_y, t_z), (a_d, b_d, c_1, c_0, e', \boldsymbol{f}'_x, \boldsymbol{f}'_y, r'_x, s'_y, t'_z), (a_d, b_d, c_1, c_0, e', \boldsymbol{f}'_x, \boldsymbol{f}'_y, r'_x, s'_y, t'_z)$. Since the transcripts are accepting, we have $a^e a_d = \mathrm{com}_{ck}(\boldsymbol{f}_x; r_x)$ and $a^{e'} a_d = \mathrm{com}_{ck}(\boldsymbol{f}'_x; r'_x)$. The homomorphic property of the commitment scheme gives us $a^{e-e'} = \mathrm{com}_{ck}(\boldsymbol{f}_x - \boldsymbol{f}'_x; r_x - r'_x)$. Since the challenges are different, there exists $\eta$ so $\eta(e - e') = 1 \bmod p$, giving us $a = \mathrm{com}_{ck}(\eta(\boldsymbol{f}_x - \boldsymbol{f}'_x); \eta(r_x - r'_x))$. Having the opening of $a$ enables us to use the equation $a^e a_d = \mathrm{com}_{ck}(\boldsymbol{f}_x; r_x)$ to get an opening $\boldsymbol{d}_x, r_d$ of $a_d$. Similar methods give us openings of $b, b_d$ and $c, c_1, c_0$.

The remaining question is whether the extracted contents $\boldsymbol{x}, \boldsymbol{y}$ and $z$ of respectively $a, b$ and $c$ satisfy $z = \boldsymbol{x} * \boldsymbol{y}$. Consider what happens in a successful transcripts with a random challenge $e$. Since we know the contents of the commitments, by the binding property of the commitment scheme there is overwhelming probability that $\boldsymbol{f}_x = e\boldsymbol{x} + \boldsymbol{d}_x, \boldsymbol{f}_y = e\boldsymbol{y} + \boldsymbol{d}_y$ and $\boldsymbol{f}_x * \boldsymbol{f}_y = e^2 z + ez_1 + ez_0$, where $z_1, z_0$ are the contents of $c_1$ and $c_0$ respectively. Inserting the first two equations in the third gives us the following equation

$$(e\boldsymbol{x} + \boldsymbol{d}_x) * (e\boldsymbol{y} + \boldsymbol{d}_y) = e^2 z + ez_1 + z_0.$$

By the Schwartz-Zippel lemma the argument will fail with overwhelming probability over the random choice of $e$ unless the two polynomials are identical, which implies $\boldsymbol{x} * \boldsymbol{y} = z$. There is therefore only negligible probability of the emulator extracting $\boldsymbol{x}, \boldsymbol{y}, z$ so $\boldsymbol{x} * \boldsymbol{y} \neq z$.

Let us summarize the proof of witness-extended emulation. We have shown that there is an emulator running in expected polynomial time. This emulator first runs the prover and verifier as a black-box, giving it a transcript with exactly the same distribution as a transcript with a real prover and verifier. In case the transcript is rejecting, it takes no further action. In case the transcript is accepting, it tries to extract openings $\boldsymbol{x}, r, \boldsymbol{y}, s, z, t$ of $a, b$ and $c$. We have shown that there is negligible probability of the event that the transcript is accepting and the emulator fails to extract $\boldsymbol{x}, r, \boldsymbol{y}, s, z, t$. We have also shown that there is negligible probability of the event that the transcript is accepting and the emulator extracts $\boldsymbol{x}, \boldsymbol{y}, z$ so $\boldsymbol{x} * \boldsymbol{y} \neq z$. □

EFFICIENCY. The argument given above has a communication cost of 4 commitment and $2n + 3$ field elements, which for large $n$ is roughly $2n\kappa$ bits. The main computational cost for the prover is the computation of the commitments $a_d$ and $b_d$. Using multi-exponentiation techniques as in [Lim00], the cost of this corresponds to $4n\kappa/\log n$ multiplications in $G$. The main cost in the verification is also the computation of the commitments, corresponding to a cost of $4n\kappa/\log n$ multiplications. We can use randomization techniques to reduce the verifier's computation to $2n\kappa/\log n$ multiplications. The verifier picks $\alpha \leftarrow \mathbb{Z}_p$ and checks both commitments at the same time, by testing $(a^e a_d)^\alpha b^e b_d = \mathrm{com}_{ck}(\alpha \boldsymbol{f}_x + \boldsymbol{f}_y; \alpha r_x + s_y)$.

### 5.2 Constant-Round Reduction to the Minimal Case

Next, we give a SHVZK argument that uses a 2-round communication-efficient reduction to the minimal case $m = 1$.

**Common input:** Commitment key $ck$ and a statement consisting of commitments $a_1, b_1, \ldots, a_m, b_m, c$.
**Prover's input:** Openings of commitments $\boldsymbol{x}_1, r_1, \boldsymbol{y}_1, s_1, \ldots, \boldsymbol{x}_m, r_m, \boldsymbol{y}_m, s_m, z, t$ so $z = \sum_{i=1}^m \boldsymbol{x}_i * \boldsymbol{y}_i$.
**Argument:**

**P→ V:** Prover picks randomizers $t_\ell \leftarrow \mathbb{Z}_p$ for $0 \leq \ell \leq 2m-1$, setting $t_{m-1} = t$ though.
Prover computes $c_0, \ldots, c_{2m-2}$ as

$$c_\ell = \mathrm{com}_{ck}\left( \sum_{i,j\,:\,\ell = m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j \;;\; t_\ell \right).$$

Observe, by construction $c_{m-1} = c$.
Prover sends $c_0, \ldots, c_{2m-2}$ to verifier.

**P← V:** Verifier sends prover random challenge $e \leftarrow \mathbb{Z}_p$.

**P↔ V:** Define

$$a' = \prod_{i=1}^m a_i^{e^{i-1}} \qquad b' = \prod_{j=1}^m b_j^{e^{m-j}} \qquad c' = \prod_{\ell=0}^{2m-2} c_\ell^{e^\ell}.$$

Prover computes openings

$$\boldsymbol{x}' = \sum_{i=1}^m e^{i-1} \boldsymbol{x}_i \qquad r' = \sum_{i=1}^m e^{i-1} r_i \qquad \boldsymbol{y}' = \sum_{j=1}^m e^{m-j} \boldsymbol{y}_j \qquad s' = \sum_{j=1}^m e^{m-j} s_j$$

and

$$z' = \sum_{\ell=0}^{2m-2} e^\ell \sum_{i,j\,:\,\ell = m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j \qquad t' = \sum_{\ell=0}^{2m-2} e^\ell t_\ell.$$

Prover and verifier run the minimal case SHVZK argument from Section 5.1 on $a', b', c'$.

**Theorem 2.** *The argument above is a public-coin argument for knowledge of openings so $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$. The argument has perfect completeness, perfect SHVZK and computational witness-extended emulation.*

Before giving the proof, let us sketch the main ideas in the construction and why it works.

The important part in the reduction to the minimal case is to use the verifier's challenge in a way such that the prover only needs to send $2m-2$ commitments to the verifier. We do this by computing $a', b'$ as multi-exponentiations of $a_1, b_1, \ldots, a_m, b_m$ with exponents that are carefully chosen powers of the challenge $e$. The product of the openings of $a'$ and $b'$ is

$$\left(\sum_{i=1}^{m} e^{i-1}\boldsymbol{x}_i\right) * \left(\sum_{j=1}^{m} e^{m-j}\boldsymbol{y}_j\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} e^{m+i-j-1}\boldsymbol{x}_i * \boldsymbol{y}_j = \sum_{\ell=0}^{2m-2} e^{\ell}\left(\sum_{i,j\,:\,\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j\right).$$

This is the key observation to show that the argument is perfectly complete.

The part corresponding to $\ell = m-1$ gives us exactly the sum we are after, but we have some extra coefficients of the polynomial corresponding to $\ell \neq m-1$. To cancel them out, the prover makes $2m-2$ commitments to these values before seeing the challenge $e$. Suppose we know openings of all the commitments let us argue that there is negligible probability of correctly answering the challenge $e$ unless indeed $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$. Since all commitments are chosen by the prover before seeing the challenge $e$, by the binding property of the commitment scheme this shows

$$\sum_{\ell=0}^{2m-2} e^{\ell}\left(\sum_{\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j\right) = \sum_{\ell=0}^{2m-2} e^{\ell} z_{\ell},$$

for random $e$ where $z = z_{m-1}$ since $c = c_{m-1}$. But if $z \neq \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$ the Schwartz-Zippel lemma tells us this can happen with probability at most $\frac{2m-2}{p}$.

*Proof.* We first argue that the argument has perfect completeness. The product of the openings of $a'$ and $b'$ is

$$\left(\sum_{i=1}^{m} e^{i-1}\boldsymbol{x}_i\right) * \left(\sum_{j=1}^{m} e^{m-j}\boldsymbol{y}_j\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} e^{m+i-j-1}\boldsymbol{x}_i * \boldsymbol{y}_j = \sum_{\ell=0}^{2m-2} e^{\ell}\left(\sum_{i,j\,:\,\ell=m+i-j-1} \boldsymbol{x}_i * \boldsymbol{y}_j\right).$$

The latter is the content of $c'$ and the perfect completeness of the SHVZK argument for the minimal case therefore gives us perfect completeness.

Next, let us show that we have perfect SHVZK. Consider a simulator that picks $c_0, \ldots, c_{2m-2}$ as random commitments to $\mathbf{0}$ except for $c_{m-1} = c$, and then runs the simulator for the SHVZK argument for the minimal case $m = 1$. Since the commitments are perfectly hiding, they have the same probability distribution as commitments to values that would be used in a real argument. The perfect SHVZK property of the minimal case $m = 1$ argument therefore gives us perfect SHVZK.

It remains to show that we have witness-extended emulation. The emulator first runs the prover and the verifier to get a transcript. This transcript has the same distribution as when a real prover and verifier are running and if it is rejecting the emulator stops here. If the transcript is accepting, however, the emulator will try to extract a witness consisting of openings of all the commitments. The emulator rewinds and runs the prover and verifier on random challenges and uses the emulator for the SHVZK argument for $m = 1$ in the end until it gets $2m-1$ accepting transcripts. Since we only run the extraction part of the emulation when the first transcript is rejecting, we have an overall expected polynomial running time for the emulator. By the witness-extended emulation property of the minimal case SHVZK argument, we get openings of all $a', b', c'$ that we feed to the final SHVZK argument and these openings satisfy $z' = \boldsymbol{x}'\boldsymbol{y}'^{\top}$. This gives us

$2m-1$ sets of equalities

$$a' = \prod_{i=1}^{m} a_i^{e^{i-1}} = \mathrm{com}_{ck}(\boldsymbol{x}'; r') \qquad b' = \prod_{j=1}^{m} b_j^{e^{m-j}} = \mathrm{com}_{ck}(\boldsymbol{y}'; s') \qquad c' = \prod_{\ell=0}^{2m-2} c_\ell^{e^\ell} = \mathrm{com}_{ck}(z'; t')$$

for random $e$'s. With overwhelming probability over the challenges $e$ the $2m-1$ vectors of the form $(1, e, \ldots, e^{2m-2})$ are linearly independent, which means we can extract openings of $c_0, \ldots, c_{2m-2}$. Similarly, the vectors $(1, e, \ldots, e^{m-1})$ are with overwhelming probability linearly independent, allowing us to extract openings of $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$.

To finalize the argument for having witness-extended emulation look at what happens on a random challenge $e$. By the witness-extended property of the final argument, with overwhelming probability we either get a rejecting transcript or we can extract openings $\boldsymbol{x}', \boldsymbol{y}', z'$ in the end so

$$z' = \boldsymbol{x}' * \boldsymbol{y}'.$$

By the binding property of the commitment scheme, these openings satisfy

$$\boldsymbol{x}' = \sum_{i=1}^{m} e^{i-1} \boldsymbol{x}_i \qquad \boldsymbol{y}' = \sum_{j=1}^{m} e^{m-j} \boldsymbol{y}_j \qquad z' = \sum_{\ell=0}^{2m-2} e^\ell z_\ell,$$

where all other parts are fixed before seeing the random challenge $e$. The Schwartz-Zippel theorem tells us this has negligible probability of succeeding unless the polynomials $z'$ and $\boldsymbol{x}' * \boldsymbol{y}'$ are identical. Looking at the $e^{m-1}$ part of the polynomials, this implies

$$z = z_{m-1} = \sum_{i,j : m-1 = i-1+m-j} \boldsymbol{x}_i * \boldsymbol{y}_j = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i.$$

□

EFFICIENCY. The prover sends $2m-2$ commitments to the verifier. Computing the commitments requires the prover to make $2m-2$ double-exponentiations and naïvely $m^2$ bilinear map evaluations to compute the entries to the commitments. Naïvely this requires $m^2 n$ multiplications, but using more advanced techniques such as organizing the vectors in $m \times n$ matrices and using Strassen's matrix multiplication algorithm to compute $XY^\top$ to get the $m^2$ dot products the cost can be further reduced. However, it is not known how to bring the cost down to $O(mn)$ multiplications. The prover also computes openings of $a', b'$ and $c'$, but this cost is small compared to the computation of all the dot products, so we will ignore it.

Combining the reduction and the minimal SHVZK argument we use 5 rounds. The communication is around $2m\kappa' + 2n\kappa$ bits. The prover's computation is less than $4n\kappa / \log n + 4m\kappa$ multiplications in $G$ and $m^2 n$ multiplications in $\mathbb{Z}_p$, where for large $m$ the latter part will dominate the cost. The verifier computes the commitments $a', b', c'$ and the computational cost of the final SHVZK argument, for a total of less than $8m\kappa / \log m + 4n\kappa / \log n$ multiplications in $G$, which as in the SHVZK for knowledge of openings can be reduced to $8m\kappa / \log m + 2n\kappa / \log n$ multiplications.

### 5.3 Trading Computation for Interaction

Let us again look at the equation

$$z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i.$$

When $m$ is large, the computational overhead of doing the multiplications in the SHVZK argument in the previous section may be prohibitive. In this section, we will trade computational complexity for round complexity by giving a $2\log m$-round reduction to the minimal case that only requires $4mn$ multiplications for the prover.

To illustrate the source of the gain, look at the matrix containing the $m^2$ products $\boldsymbol{x}_i * \boldsymbol{y}_j$. An example of an $8 \times 8$ matrix is given below.

$$\left(\begin{array}{cc|cc|cc|cc}
\boldsymbol{x}_1 * \boldsymbol{y}_1 & \boldsymbol{x}_1 * \boldsymbol{y}_2 & \boldsymbol{x}_1 * \boldsymbol{y}_3 & \boldsymbol{x}_1 * \boldsymbol{y}_4 & \boldsymbol{x}_1 * \boldsymbol{y}_5 & \boldsymbol{x}_1 * \boldsymbol{y}_6 & \boldsymbol{x}_1 * \boldsymbol{y}_7 & \boldsymbol{x}_1 * \boldsymbol{y}_8 \\
\boldsymbol{x}_2 * \boldsymbol{y}_1 & \boldsymbol{x}_2 * \boldsymbol{y}_2 & \boldsymbol{x}_2 * \boldsymbol{y}_3 & \boldsymbol{x}_2 * \boldsymbol{y}_4 & \boldsymbol{x}_2 * \boldsymbol{y}_5 & \boldsymbol{x}_2 * \boldsymbol{y}_6 & \boldsymbol{x}_2 * \boldsymbol{y}_7 & \boldsymbol{x}_2 * \boldsymbol{y}_8 \\
\hline
\boldsymbol{x}_3 * \boldsymbol{y}_1 & \boldsymbol{x}_3 * \boldsymbol{y}_2 & \boldsymbol{x}_3 * \boldsymbol{y}_3 & \boldsymbol{x}_3 * \boldsymbol{y}_4 & \boldsymbol{x}_3 * \boldsymbol{y}_5 & \boldsymbol{x}_3 * \boldsymbol{y}_6 & \boldsymbol{x}_3 * \boldsymbol{y}_7 & \boldsymbol{x}_3 * \boldsymbol{y}_8 \\
\boldsymbol{x}_4 * \boldsymbol{y}_1 & \boldsymbol{x}_4 * \boldsymbol{y}_2 & \boldsymbol{x}_4 * \boldsymbol{y}_3 & \boldsymbol{x}_4 * \boldsymbol{y}_4 & \boldsymbol{x}_4 * \boldsymbol{y}_5 & \boldsymbol{x}_4 * \boldsymbol{y}_6 & \boldsymbol{x}_4 * \boldsymbol{y}_7 & \boldsymbol{x}_4 * \boldsymbol{y}_8 \\
\hline
\boldsymbol{x}_5 * \boldsymbol{y}_1 & \boldsymbol{x}_5 * \boldsymbol{y}_2 & \boldsymbol{x}_5 * \boldsymbol{y}_3 & \boldsymbol{x}_5 * \boldsymbol{y}_4 & \boldsymbol{x}_5 * \boldsymbol{y}_5 & \boldsymbol{x}_5 * \boldsymbol{y}_6 & \boldsymbol{x}_5 * \boldsymbol{y}_7 & \boldsymbol{x}_5 * \boldsymbol{y}_8 \\
\boldsymbol{x}_6 * \boldsymbol{y}_1 & \boldsymbol{x}_6 * \boldsymbol{y}_2 & \boldsymbol{x}_6 * \boldsymbol{y}_3 & \boldsymbol{x}_6 * \boldsymbol{y}_4 & \boldsymbol{x}_6 * \boldsymbol{y}_5 & \boldsymbol{x}_6 * \boldsymbol{y}_6 & \boldsymbol{x}_6 * \boldsymbol{y}_7 & \boldsymbol{x}_6 * \boldsymbol{y}_8 \\
\hline
\boldsymbol{x}_7 * \boldsymbol{y}_1 & \boldsymbol{x}_7 * \boldsymbol{y}_2 & \boldsymbol{x}_7 * \boldsymbol{y}_3 & \boldsymbol{x}_7 * \boldsymbol{y}_4 & \boldsymbol{x}_7 * \boldsymbol{y}_5 & \boldsymbol{x}_7 * \boldsymbol{y}_6 & \boldsymbol{x}_7 * \boldsymbol{y}_7 & \boldsymbol{x}_7 * \boldsymbol{y}_8 \\
\boldsymbol{x}_8 * \boldsymbol{y}_1 & \boldsymbol{x}_8 * \boldsymbol{y}_2 & \boldsymbol{x}_8 * \boldsymbol{y}_3 & \boldsymbol{x}_8 * \boldsymbol{y}_4 & \boldsymbol{x}_8 * \boldsymbol{y}_5 & \boldsymbol{x}_8 * \boldsymbol{y}_6 & \boldsymbol{x}_8 * \boldsymbol{y}_7 & \boldsymbol{x}_8 * \boldsymbol{y}_8
\end{array}\right)$$

We want to argue knowledge of $c$ being a commitment to the trace of the matrix. In the SHVZK argument we gave in the previous section, all the $2m-1$ lines that are parallel with the diagonal correspond to entries that have the same degree in the polynomial in $e$. For instance the sum of the diagonal entries is the coefficient of $e^{m-1}$ whereas the sum of the entries with $i-j=1$ is the coefficient of $e^m$. In the SHVZK argument in the previous section, we computed all these $m^2$ products. Since they each cost $n$ multiplications to compute, we end up using $m^2 n$ multiplications. This cost may be reduced by using advanced matrix-multiplication techniques, but even with the best of those it is still significantly higher than $mn$ multiplications. As an example, in the $8 \times 8$ matrix above we end up computing $64$ vector products. We are only interested in the $8$ entries along the diagonal, so the remaining computation is just waste that we need to discard in the argument. We will devise a method that allows us to compute larger sub-matrices at once, instead of taking each individual entry at a time. Looking again at the example, if we can discard $2 \times 2$ matrices and $4 \times 4$ matrices, we only need to discard $14$ sub-matrices instead of the $56$ entries we need to discard in the reduction in the previous section.

Below, we give a SHVZK argument that reduces the statement to the minimal case $m = 1$ through $\log m$ recursive calls to itself. For simplicity we assume that $m = 2^\mu$. We can do this without loss of generality, because we can always fill up with dummy elements consisting of zero-vectors and trivial commitments, which do not carry any computational overhead.

The idea in the recursive call is to handle the $2 \times 2$ matrices along the diagonal at once. We already have a commitment $c$ to the sum of the diagonal entries. In addition, the prover sends commitments $c_l, c_u$ to the verifier, containing respectively the sum of the lower-left corners of the sub-matrices and the sum of the upper-right corners of the sub-matrices along the diagonal.

The verifier responds with a random challenge $e \leftarrow \mathbb{Z}_p$. The prover now reduces her set of vectors to half, by computing

$$\boldsymbol{x}'_i = \boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i} \qquad\qquad \boldsymbol{y}'_i = e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i}.$$

The homomorphic properties of the commitments enables the verifier to compute commitments to these vectors as $a'_i = a_{2i-1} a_{2i}^e$ and $b'_i = b_{2i-1}^e b_{2i}$. We also compute $c' = c_l^{e^2} c^e c_u$, which is a commitment to the sum of the diagonal entries in the new matrix obtained from the vectors $\boldsymbol{x}'_1, \boldsymbol{y}'_1, \ldots, \boldsymbol{x}'_{m/2}, \boldsymbol{y}'_{m/2}$.

The prover and the verifier now engage in a SHVZK argument with these new commitments and vectors for $c'$ containing the sum of the diagonal elements of the matrix. The implication is that for random $e$ we have

$$\sum_{i=1}^{m/2} (\boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i}) * (e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i}) = e^2 z_l + ez + z_u,$$

where $z_l$ and $z_u$ are the contents of $c_l$ and $c_u$. By the Schwartz-Zippel lemma this implies with overwhelming probability $z = \sum_{i=1}^{m/2}(\boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i-1} + \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i})$, which is what we wanted to prove.

**Common input:** Commitment key $ck$ and commitments $a_1, b_1, \ldots, a_m, b_m, c$, with $m = 2^\mu$.
**Prover's input:** Openings of commitments $\boldsymbol{x}_1, r_1, \boldsymbol{y}_1, s_1, \ldots, \boldsymbol{x}_m, r_m, \boldsymbol{y}_m, s_m, z, t$ so $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$.
**Argument:**

**If** $m = 1$**:** Run the SHVZK argument from Section 5.1 with common input $ck, a_1, b_1, c$ and prover input $\boldsymbol{x}_1, r_1, \boldsymbol{y}_1, s_1, z, t$ to show $z = \boldsymbol{x}_1 * \boldsymbol{y}_1$.

**Else if** $m > 1$**:** Define $m' = m/2$ and do

**P**$\rightarrow$ **V:** Prover picks $t_l, t_u \leftarrow \mathbb{Z}_p$ and sends to verifier

$$c_l = \mathrm{com}_{ck}(\sum_{i=1}^{m'} \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i-1}; t_l) \qquad \text{and} \qquad c_u = \mathrm{com}_{ck}(\sum_{i=1}^{m'} \boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i}; t_u).$$

**P**$\leftarrow$ **V:** Verifier picks random challenge $e \leftarrow \mathbb{Z}_p$ and sends it to prover.

**P**$\leftrightarrow$ **V:** Recursively run argument with common input $ck, a'_1, b'_1, \ldots, a'_{m'}, b'_{m'}, c'$ given by

$$a'_i = a_{2i-1}a_{2i}^e \qquad b'_i = b_{2i-1}^e b_{2i} \qquad c' = c_l^{e^2} c^e c_u.$$

The prover's private input is $\boldsymbol{x}'_1, r'_1, \boldsymbol{y}'_1, s'_1, \ldots, \boldsymbol{x}'_{m'}, r_{m'}, \boldsymbol{y}'_{m'}, s_{m'}, z', t'$ given by

$$\boldsymbol{x}'_i = \boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i} \qquad r'_i = r_{2i-1} + er_{2i} \qquad \boldsymbol{y}'_i = e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i} \qquad s'_i = es_{2i-1} + s_{2i}$$

$$z' = e^2 \sum_{i=1}^{m'} \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i-1} + ez + \sum_{i=1}^{m'} \boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i} \qquad t' = e^2 t_l + et + t_u.$$

**Theorem 3.** *The argument above is a public-coin argument for knowledge of openings so $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$. The argument has perfect completeness, perfect SHVZK and computational witness-extended emulation.*

*Proof.* Obviously the argument has public-coin challenges and perfect completeness can be verified by inspection, since the SHVZK argument for $m = 1$ has perfect completeness.

To argue perfect SHVZK consider the case, where the challenges are known ahead of time. With $m > 1$ and challenge $e$ the simulator picks two random commitments $c_l, c_u$. Since the commitment scheme is perfectly hiding, these commitments are identically distributed to commitments $c_l, c_u$ that are computed as in the SHVZK argument using a real witness. By Theorem 1 the simulator can make a perfect simulation of the final SHVZK argument with $m = 1$.

It remains to show that we have witness-extended emulation. We will show this by induction over $m$, where we without loss of generality assume $m = 2^\mu$. Observe first that if $m = 1$, then by Theorem 4 we have witness-extended emulation as desired. Assume now we have a witness-extended emulator for $m'$, we will show how to use it to construct a witness-extended emulator for $m = 2m'$. We run the prover and the verifier 2 rounds and then use the witness-extended emulator for $m'$ to get the transcript. This transcript has the same probability distribution as a real transcript between a prover and verifier. If the transcript is rejecting, we do not need to take further action, but if the transcript is accepting we will try to extract openings of the commitments. We do that by rewinding and doing the same operation again with random challenges $e$ until we get two more accepting transcript. With overwhelming probability, these three accepting transcripts come with openings of all the commitments. We note that with overwhelming probability over the challenges $e$ the three challenge vectors $(1, e, e^2)$ are linearly independent. Look at a pair of commitments $a_{2i-1}, a_{2i}$ from the statement. From the witness-extended emulation, we get an opening of $a_{2i-1}a_{2i}^e$ for two different challenges $e$. This enables us to extract the content of both $a_{2i-1}$ and $a_{2i}$. In a similar fashion, we can extract

the contents of all commitments $a_1, b_1, \ldots, a_m, b_m$. We also get openings of $c_l^{e^2} c^e c_u$ for three different challenges $e$, giving us openings of $c_l$, $c$ and $c_u$. So we have openings of all commitments.

At this stage, we note that the witness-extended emulator for $m = 2m'$ runs the witness-extended emulator for $m'$ three times on an accepting transcript. This gives a blow-up in the expected running time of a factor 3 for each pair of rounds. Overall, we only have $2 \log m + 3$ rounds in the argument though, so we stay within expected polynomial time for the entire emulation.

Let us now look at the openings we have extracted. Since the commitments are computationally binding, in our recursion we call the SHVZK argument for $m'$ with $\boldsymbol{x}'_i = \boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i}$ and $\boldsymbol{y}'_i = e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i}$ and $z' = e^2 z_l + ez + z_u$ with random challenge $e$. From the witness-extended emulation property of the SHVZK with $m'$ we know

$$\sum_{i=1}^{m'} (\boldsymbol{x}_{2i-1} + e\boldsymbol{x}_{2i}) * (e\boldsymbol{y}_{2i-1} + \boldsymbol{y}_{2i}) = e^2 z_l + ez + z_u.$$

By the Schwartz-Zippel lemma there is overwhelming probability over $e$ that this fails unless $z = \sum_{i=1}^{m'} (\boldsymbol{x}_{2i-1} * \boldsymbol{y}_{2i-1} + \boldsymbol{x}_{2i} * \boldsymbol{y}_{2i}) = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$ as desired. Our proof by induction therefore shows that for all $m$ that are polynomial in the security parameter, we have witness-extended emulation. □

EFFICIENCY. Each recursive call to the SHVZK argument with $m > 1$ makes the prover send 2 commitments to the verifier. The main computational cost for the prover is the computation of $m = 2m'$ new vectors costing around $mn$ multiplications and $m$ bilinear map evaluations costing around $n$ multiplications each. Summing up over $\log m$ recursive calls, we get a total communication of $2 \log m$ commitments from the prover to the verifier and a computational cost for the prover of $4mn$ multiplications in $\mathbb{Z}_p$. The verifier can wait until the proof is over to compute anything; this permits the verifier to use multi-exponentiation techniques for computing the commitments $a, b, c$ that are used in the final call to the minimal case SHVZK argument where $m = 1$. As a consequence, the verifier uses around $4m\kappa / \log m$ multiplications. These costs do not count the cost of the minimal SHVZK argument itself. Including the final SHVZK argument, we get a cost of $2 \log m + 3$ rounds, $2n\kappa + 2 \log m \kappa'$ bits, $4n\kappa / \log n + 4mn$ multiplications for the prover, and $4n\kappa / \log n + 4m\kappa / \log m$ multiplications for the verifier, which with randomization techniques can be reduced to $2n\kappa / \log n + 4m\kappa / \log m$.

## 6 Zero-Knowledge Arguments for Linear Algebra Equations

We now have several tools to deal with committed matrices and vectors. We can add matrices and vectors using the homomorphic properties of the commitment scheme and we have SHVZK arguments for equations involving multiplications of matrices and vectors and Hadamard products of matrices and vectors. We will sketch how to use these tools to get sub-linear zero-knowledge arguments for equations often arising in linear algebra.

### 6.1 Inverse

To prove committed matrices satisfy $Y = X^{-1}$ or equivalently $XY = I$, we let the verifier pick $\boldsymbol{s} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and the prover give a SHVZK argument for $(\boldsymbol{s}X)Y = \boldsymbol{s}$.

### 6.2 Transpose

To prove that a committed matrices satisfy $Y = X^\top$, we let the verifier pick $\boldsymbol{s}, \boldsymbol{t} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and the prover give a SHVZK argument for $(\boldsymbol{s}X)\boldsymbol{t}^\top = (\boldsymbol{t}Y)\boldsymbol{s}^\top$.

### 6.3 Eigenvalues and Eigenvectors

To show that we have a commitment to an eigenvalue $\lambda$ and an eigenvector $\boldsymbol{y}^\top$ of $X$, we first commit to $\boldsymbol{z} = \lambda \boldsymbol{y}$. There are standard SHVZK arguments for $\boldsymbol{z} = \lambda \boldsymbol{y}$, so the prover can show the committed $\boldsymbol{z}$ is correct. Now the verifier picks $\boldsymbol{s} \leftarrow \mathrm{Van}_n(\mathbb{Z}_p)$ and we also give a SHVZK argument for $\boldsymbol{s}\boldsymbol{z}^\top = (\boldsymbol{s}X)\boldsymbol{y}^\top$.

### 6.4 Sums of Rows and Columns

Computing the sum of all row vectors or all column vectors of a matrix corresponds to computing $X\mathbf{1}^\top$ and $\mathbf{1}X$ respectively, where $\mathbf{1} = (1, \ldots, 1)$. The sum of all entries in a matrix can be computed as $\mathbf{1}A\mathbf{1}^\top$. With our techniques we get efficient SHVZK arguments for the correctness of these computations.

### 6.5 Hadamard Products of Rows and Columns

Let us give a SHVZK argument for a committed vector $\boldsymbol{z}$ containing the Hadamard product of all the rows $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ of committed matrix. The prover commits to vectors $\boldsymbol{y}_i = \boldsymbol{x}_1 \circ \cdots \circ \boldsymbol{x}_i$, using $\boldsymbol{y}_1 = \boldsymbol{x}_1$ and $\boldsymbol{y}_n = \boldsymbol{z}$. By demonstrating for $1 \leq i < n$ that $\boldsymbol{y}_{i+1} = \boldsymbol{y}_i \circ \boldsymbol{x}_{i+1}$ we convince the verifier that $\boldsymbol{z}$ is the Hadamard product of the row vectors in $X$. We remark that it is easy to get a SHVZK argument for $z = \prod_{i=1}^{n} z_i$, where $\boldsymbol{z} = (z_1, \ldots, z_n)$, so we can extend our SHVZK argument to prove $z$ is the product of all entries in the matrix. In case we want to show $\boldsymbol{z}^\top$ is the Hadamard product of all the columns, we can commit to $X^\top$, using the SHVZK argument from Section 6.2 to prove correctness, and show $\boldsymbol{z}$ is the Hadamard product of all the rows using the SHVZK argument from Section 6.5.

### 6.6 Triangularity

The Hadamard product enables us to prove that a subset of the entries in a committed matrix $X$ consists of all zeroes. Let $S$ be the matrix that has 1 in all entries belong to the subset and has 0 in all other entries. We give a SHVZK argument for $S \circ X = 0$. This SHVZK argument can for instance be used to demonstrate that a committed matrix is lower triangular, upper triangular or diagonal.

### 6.7 Trace

To show committed values satisfying $z = \mathrm{trace}(X)$ we give a SHVZK argument for $z = \sum_{i=1}^{n} \boldsymbol{s}_i \boldsymbol{x}_i^\top$, where $\boldsymbol{s}_i$ is the $i$th row vector of $I$.

### 6.8 Absolute Value of Determinant

We can commit to an $LUP$ factorization of a matrix $X$. Proving lower and upper triangularity we already know how to do. It is also easy to prove that we have a committed permutation matrix $P$, for instance by showing that the matrix is a hidden permutation of $I$ (see Section 6.10) and that $\mathbf{1}P = \mathbf{1}$ and $P\mathbf{1}^\top = \mathbf{1}^\top$. Since we can single out the diagonal elements of $L$ and $U$ we can compute the determinants of these matrices. We know that $P$ has determinant $-1$ or 1. We therefore get the determinant up to the sign. We leave it as an open problem to give a sub-linear zero-knowledge argument for the permutation matrix $P$ having determinant $-1$ or $+1$.

### 6.9 Known Permutation of a Matrix

Consider a publicly known permutation $\pi$ over $\mathbb{Z}_n \times \mathbb{Z}_n$ and two committed matrices $Y = \pi(X)$, meaning for all pairs $(i, j)$ we have $y_{ij} = x_{\pi(ij)}$. To give a SHVZK argument for this, the verifier first picks $R \leftarrow \mathrm{Van}_{n^2}(\mathbb{Z}_p)$ and we ask the prover to show

$$\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} r_{\pi(ij)}y_{ij},$$

which by the Schwartz-Zippel fails with probability $\frac{n^2-1}{p}$ unless indeed $Y = \pi(X)$. Define $S = \pi(R)$ and call the row vectors of the matrices respectively $\boldsymbol{r}_i, \boldsymbol{s}_i, \boldsymbol{x}_i, \boldsymbol{y}_i$. The statement above is equivalent to

$$\sum_{i=1}^{n} \boldsymbol{r}_i \boldsymbol{x}_i^\top = \sum_{i=1}^{n} \boldsymbol{s}_i \boldsymbol{y}_i^\top,$$

for which we already know how to give a SHVZK argument.

### 6.10 Hidden Permutation of a Matrix

To show that there is a secret permutation $\pi$ so $Y = \pi(X)$, we use the fact that polynomials are identical under permutation of the roots; an idea that stems from Neff [Nef01]. The verifier picks $r \leftarrow \mathbb{Z}_p$ at random and we let $R$ be the matrix that has $r$ in all entries. We then use the SHVZK argument from Section 6.5 to show that the product of the entries in $X - R$ equals the product of the entries in $Y - R$. In other words, we show for a random $r$ that

$$\prod_{i=1}^{n}\prod_{j=1}^{n}(x_{ij} - r) = \prod_{i=1}^{n}\prod_{j=1}^{n}(y_{ij} - r),$$

which by the Schwartz-Zippel lemma demonstrates that the two polynomials are identical and thus there exists a permutation $\pi$ so $x_{ij} = y_{\pi(ij)}$.

This type of SHVZK argument is useful in the context of shuffling [Cha81] and one of the main contributions of Groth and Ishai [GI08] was to show how to give an argument with sub-linear communication. Their SHVZK argument had a communication complexity that could be brought down to $\Theta(n^{4/3})$ group elements at the cost of a super-linear computation complexity of $\Theta(n^{8/3})$ exponentiations.[3] In comparison, our SHVZK argument has a communication complexity of $\Theta(n)$ field elements and even for the constant round protocol we get a much better computation complexity of $n^3$ *multiplications*. Using a logarithmic number of rounds, we can bring that down to $2n^2\kappa/\log n$ multiplications, which is more efficient than the best non-sublinear shuffle argument [Gro03].

## 7 Circuit Satisfiability

Let us consider an arithmetic circuit built from $N$ addition and multiplication gates over a field $\mathbb{Z}_p$. We want to give a SHVZK argument for the existence of input values to the circuit that makes it evaluate to 1. All gates have two input wires and one output, some of which may be known constants. By introducing dummy gates we can without loss of generality assume $3N = n^2$ and that the number of multiplication gates $M$ and the number of addition gates $A$ are multiples of $n$.

[3] The computational complexity of Groth and Ishai's shuffle argument can be reduced at the cost of increasing communication.

1. We number the addition gates 1 through $A$ and the multiplication gates $A+1$ through $A+M$. We arrange the inputs and outputs such that the first two rows contain input values to the first $n$ addition gates and the third row contains the corresponding output values, then follows another two rows of input gates and one row of output gates, *etc*. Arranging the circuit in this way, the first $3A$ rows are used for addition gates, while the last $3M$ rows are used for multiplication gates. The prover commits to all these values.
2. For the addition gates, we create the commitment to row $3i$ as the product of the commitments to row $3i-2$ and $3i-1$. By the homomorphic properties of the commitment scheme, this shows that the addition gates are satisfied by the committed wires.
3. For the multiplication gates we can use the SHVZK argument for Hadamard products, to show that the commitment to row $3i$ is the Hadamard product of the commitments to rows $3i-2$ and $3i-1$. This shows that all the multiplication gates are satisfied by the committed values.
4. Some of the values in the matrix may be publicly known constants. By introducing dummy gates and organizing the matrix such that constants appear in the same row, we can without loss of generality assume that we have entire rows that have publicly known constants. We can make these commitments with trivial randomness so the verifier easily can check that the right constants appear in the right places.
5. Finally, we need to demonstrate that all wires appearing many places in the matrix have the same value assigned to them. The output wire of one gate, might for instance appear elsewhere in matrix as an input wire of another gate; we need to give a SHVZK argument for them having the same value. Let us first look at just one wire that appears many places, say coordinates $(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)$. We can create a directed Hamiltonian cycle on this set of indices. Let now $\pi$ be a permutation that contains directed Hamiltonian cycles for all wires in the circuit. We use our SHVZK argument for known permutations to show that $X = \pi(X)$. This proves that the committed values are consistent, giving the same value to the same wire everywhere in the matrix.

### 7.1 Binary Circuits

We have given a SHVZK argument for arithmetic circuit satisfiability, demonstrating the generality of our techniques. The argument consists of committing to a matrix and using some of the SHVZK arguments we have developed in the paper, so it inherits the low communication complexity from the previous sections. The computational complexity of the arithmetic circuit is dominated by the commitment to the wires, costing the prover $O(N\kappa/\log N)$ multiplications.

If we look instead at a binary circuit, where the wires can be 0 or 1, we can reduce the computational complexity. Committing to a binary matrix requires only $O(N/\log N)$ *multiplications* of group elements. Giving a satisfiability argument for a binary circuit requires demonstrating that we have committed to binary values only. This can be done quite easily by demonstrating the committed matrix satisfies $X = X \circ X$.

## 8 Efficiency

In the following table, we give efficiency estimates for the SHVZK arguments we have considered in the paper. We use the parameters $\kappa, \kappa'$ and $n$ to represent respectively the size of a field element, the size of a group element and the number of elements in a vector. We assume $n$ is large, since this is where efficient zero-knowledge arguments are most needed and ignore small terms. We measure communication in bits and computation in multiplications in $\mathbb{Z}_p$. We let $\rho, \epsilon$ be the costs of respectively a multiplication in $G$ and an addition in $\mathbb{Z}_p$ measured in multiplications in $\mathbb{Z}_p$.

| SHVZK argument | Rounds | Communication | Prover computation | Verifier computation |
|---|---|---|---|---|
| $z = \boldsymbol{x} * \boldsymbol{y}$ | 3 | $2n\kappa$ | $4n\frac{\kappa\rho}{\log n}$ | $2n\frac{\kappa\rho}{\log n}$ |
| $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$ | 5 | $2n\kappa + 2m\kappa'$ | $m^2n + 2mn + 4m\frac{\kappa\rho}{\log m} + 4n\frac{\kappa\rho}{\log n}$ | $8m\frac{\kappa\rho}{\log m} + 2n\frac{\kappa\rho}{\log n}$ |
| $z = \sum_{i=1}^{m} \boldsymbol{x}_i * \boldsymbol{y}_i$ | $2\log m + 3$ | $2n\kappa$ | $4mn + 4n\frac{\kappa\rho}{\log n}$ | $4m\frac{\kappa\rho}{\log m} + 2n\frac{\kappa\rho}{\log n}$ |
| Inverse $Y = X^{-1}$ | 4 | $2n\kappa$ | $n^2 + 4n\frac{\kappa\rho}{\log n}$ | $4n\frac{\kappa\rho}{\log n}$ |
| Transpose $Y = X^{\top}$ | 6 | $2n\kappa$ | $2n^2 + 4n\kappa\rho/\log n$ | $6n\frac{\kappa\rho}{\log n}$ |
| Eigenv. $\lambda\boldsymbol{y}^{\top} = X\boldsymbol{y}^{\top}$ | 5 | $5n\kappa$ | $n^2 + 12n\frac{\kappa\rho}{\log n}$ | $6n\frac{\kappa\rho}{\log n}$ |
| Triangularity | 6 | $2n\kappa + 2n\kappa'$ | $n^3\epsilon + 4n^2 + 8n\frac{\kappa\rho}{\log n}$ | $10n\frac{\kappa\rho}{\log n}$ |
| Triangularity | $2\log m + 4$ | $2n\kappa$ | $6n^2 + 4n\frac{\kappa\rho}{\log n}$ | $6n\frac{\kappa\rho}{\log n}$ |
| Trace$(X)$ | 5 | $2n\kappa + 2n\kappa'$ | $n^3\epsilon + 2n^2 + 8n\frac{\kappa\rho}{\log n}$ | $10n\frac{\kappa\rho}{\log n}$ |
| Trace$(X)$ | $2\log n + 3$ | $2n\kappa$ | $4n^2 + 4n\frac{\kappa\rho}{\log n}$ | $6n\frac{\kappa\rho}{\log n}$ |
| Hadamard of rows | 7 | $2n\kappa + 2n\kappa'$ | $n^3 + 2n^2\frac{\kappa\rho}{\log n}$ | $10n\frac{\kappa\rho}{\log n}$ |
| Hadamard of rows | $2\log n + 5$ | $2n\kappa$ | $2n^2\frac{\kappa\rho}{\log n}$ | $6n\frac{\kappa\rho}{\log n}$ |
| Known $Y = \pi(X)$ | 6 | $2n\kappa + 4n\kappa'$ | $4n^3 + 12n\frac{\kappa\rho}{\log n}$ | $3n^2 + 14n\frac{\kappa\rho}{\log n}$ |
| Known $Y = \pi(X)$ | $2\log n + 4$ | $2n\kappa$ | $9n^2 + 4n\frac{\kappa\rho}{\log n}$ | $3n^2 + 6n\frac{\kappa\rho}{\log n}$ |
| Hidden $Y = \pi(X)$ | 8 | $2n\kappa + 2n\kappa'$ | $n^3 + 2n^2\frac{\kappa\rho}{\log n}$ | $10n\frac{\kappa\rho}{\log n}$ |
| Hidden $Y = \pi(X)$ | $2\log n + 6$ | $2n\kappa$ | $2n^2\frac{\kappa\rho}{\log n}$ | $6n\frac{\kappa\rho}{\log n}$ |
| Arithmetic circuit | 7 | $O(\sqrt{N}(\kappa + \kappa'))$ | $O(N^{3/2} + N\frac{\kappa\rho}{\log N})$ | $O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$ |
| Arithmetic circuit | $\log N + 5$ | $O(\sqrt{N}\kappa)$ | $O(N\frac{\kappa\rho}{\log N})$ | $O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$ |
| Binary circuit | 7 | $O(\sqrt{N}(\kappa + \kappa'))$ | $O(N^{3/2}\epsilon + N + \sqrt{N}\frac{\kappa\rho}{\log N})$ | $O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$ |
| Binary circuit | $\log N + 5$ | $O(\sqrt{N}\kappa)$ | $O(N)$ | $O(N + \sqrt{N}\frac{\kappa\rho}{\log N})$ |

## 9 Acknowledgment

Yuval Ishai was involved at an early stage of this research and we greatly appreciate the fruitful discussions and his insightful comments.

## References

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, 1992.

[BSGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short PCPs verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, pages 120–134, 2005.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, volume 893 of *Lecture Notes in Computer Science*, pages 174–187, 1994.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, 2000.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, 2002.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 379–396, 2008. Full paper available at `http://www.daimi.au.dk/~jg/PCPShuffle.pdf`.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal of Computing*, 25(1):169–192, 1996.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.

[Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *PKC*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160, 2003. Full paper available at `http://eprint.iacr.org/2005/246`.

[Gro04] Jens Groth. Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, 2004. PhD thesis. xii+119 pp.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30, 2007.

[Joh00] David Johnson. Challenges for theoretical computer science, 2000. Available at `http://www.research.att.com/~dsj/nsflist.html#Crypto`.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732, 1992.

[KR08] Yael Tauman Kalai and Ran Raz. Interactive pcp. In *ICALP*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547, 2008.

[Lim00] Chae Hoon Lim. Efficient multi-exponentiation and application to batch verification of digital signatures, 2000. `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps`.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–125, 2001.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, 1991.

[Pip80] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal of Computing*, 9(2):230–250, 1980.

## A Argument of Knowledge of Commitment Openings

Consider a set of commitments $c_1, \ldots, c_m$ given by $c_i = \mathrm{com}(\boldsymbol{x}_i; r_i)$. We will now give a 3-move public coin perfect SHVZK argument of knowledge of the committed vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{Z}_p^n$.

**P$\rightarrow$ V:** The prover on input of a commitment key $ck$, a statement consisting of $m$ commitments $c_1, \ldots, c_m$ and openings of the commitments $\{(\boldsymbol{x}_i, r_i)\}_{i=1}^m$ chooses $\boldsymbol{x}_0 \leftarrow \mathbb{Z}_p^n, r_0 \leftarrow \mathbb{Z}_p$ and sends $c_0 = \mathrm{com}_{ck}(\boldsymbol{x}_0; r_0)$ to the verifier.
**P$\leftarrow$ V:** The verifier sends the prover a random challenge $e \leftarrow \mathbb{Z}_p$.
**P$\rightarrow$ V:** The prover answers the challenge with $\boldsymbol{z}, s$ computed as $\boldsymbol{z} = \sum_{i=0}^m e^i \boldsymbol{x}_i$ and $s = \sum_{i=0}^m e^i r_i$.
**V:** The verifier accepts the argument if $\prod_{i=0}^m c_i^{e^i} = \mathrm{com}_{ck}(\boldsymbol{z}; s)$.

**Theorem 4.** *There is a 3-move public coin argument with witness extended emulation for knowledge of the openings of a set of commitments. The argument has perfect completeness and perfect SHVZK.*

*Proof.* The protocol above consists of 3 moves and is clearly public-coin. It is straightforward to verify that it has perfect completeness.

To show that it is perfect SHVZK consider a simulator $S$ that has $c_1, \ldots, c_m$ and knows the challenge $e$ ahead of time. The simulator picks $\boldsymbol{z} \leftarrow \mathbb{Z}_p^n$ and $s \leftarrow \mathbb{Z}_p$ at random and computes $c_0 = \mathrm{com}_{ck}(\boldsymbol{z}; s) \prod_{i=1}^m c_i^{-e^i}$. The simulated argument is $(c_0, e, \boldsymbol{z}, s)$. To see that the simulation is good, observe that we get randomly distributed $\boldsymbol{z}, s$ just as in a real argument. Conditioned on $\boldsymbol{z}, s$ and $e$ the remaining part of the proof $c_0$ is uniquely determined whether we look at a real argument or a simulated argument. This means real arguments and simulated arguments have identical probability distributions.

It remains to show that the binding property of the commitment scheme implies witness-extended emulation. The emulator starts by running the prover and verifier to get a transcript $\mathrm{tr}$. If the transcript ends with the argument being rejected, it simply outputs this transcript without extracting a witness. On the other hand, if the transcript ends with the verifier accepting, then the emulator will try to extract a witness. It rewinds the prover and the verifier and runs them again, with random challenges, until it collects an additional $m$ accepting transcripts. We can organize the challenge vectors as rows of an $m+1 \times m+1$ Vandermonde matrix. If there are two transcripts with identical challenges the emulator will fail to extract a witness, but we will later argue that this happens only with negligible probability. If there are no challenges that are alike, the rows of the Vandermonde matrix are different and it follows from the properties of Vandermonde matrices that the determinant is non-zero, so the matrix is invertible. Let $A$ be the inverse of the challenge matrix. This means $\sum_{j=0}^m a_{hj} e_j^i = \delta_{hi}$, where $e_j$ is the challenge in the $j$'th successful accepting transcript and where $\delta_{hi}$ is Kronecker's delta given by $\delta_{hi} = 1$ if $h = i$ and $\delta_{hi} = 0$ else. The transcripts give us $m+1$ different challenges $e_j$ and corresponding answers $\boldsymbol{z}_j, s_j$. The verification equations tell us that for all $j$ we have $\prod_{i=0}^m c_i^{e_j^i} = \mathrm{com}_{ck}(\boldsymbol{z}_j; s_j)$. Therefore, for all $h$ we have

$$c_h = \prod_{i=0}^m c_i^{\delta_{hi}} = \prod_{i=0}^m c_i^{\sum_{j=0}^m a_{hj} e_j^i} = \prod_{i=0}^m \prod_{j=0}^m c_i^{a_{hj} e_j^i} = \prod_{j=0}^m (\prod_{i=0}^m c_i^{e_j^i})^{a_{hj}} = \mathrm{com}_{ck}(\sum_{j=0}^m a_{hj} \boldsymbol{z}_j; \sum_{j=0}^m a_{hj} s_j),$$

giving us an opening of $c_h$.

To see that the emulator does a successful witness-extraction, we first observe that we only extract $m$ additional accepting transcripts when the first transcript is accepting. Consider the conditional probability of the prover that it successfully answers a random challenge $e$ after having sent $c_0$. In case $\epsilon = 0$ it will fail to produce an accepting transcript and we do not have to extract openings of the commitments. With probability $1 - \epsilon$ the first transcript is not accepting and we do not need to spend time rewinding. If the first transcript is accepting, we have $\epsilon > 0$ and using an expected number of $m/\epsilon$ rewinds we obtain $m$ additional

accepting transcripts. Overall, we expect to rewind $\epsilon \times \frac{m}{\epsilon} = m$ times, giving us an expected polynomial time for the verifier.

Consider now the possibility of the emulator getting a valid first transcript but failing to produce a witness, which will happen if the $m + 1$ transcripts have two or more transcripts with identical challenges. Since the emulator is running in expected polynomial time, the event of an accepting first transcript and subsequently in the extraction phase getting two or more identical challenges is negligible since the challenges are chosen at random from $\mathbb{Z}_p$. We conclude that there is only negligible risk of having an emulation with an accepting first transcript and not extracting openings of the commitments. □

EFFICIENCY. The argument given above has a communication cost of 1 commitment and $m + 2$ field elements. The prover computes one commitment and computes a linear combination of the vectors $\boldsymbol{x}_i$ and a linear combination of the randomizers $r_i$. Using multi-exponentiation techniques the prover uses around $2n\kappa/\log n$ multiplications of group elements and $mn$ multiplications of field elements. The verifier does a multi-exponentiation when computing the commitment and also forms a multi-exponentiation of $m$ commitments, costing the equivalent of $2m\kappa/\log m + 2n\kappa/\log n$ multiplications in the group. The argument is therefore cheap both in terms of communication and computation.