Efficient Zero-Knowledge Argument for Correctness of a Shuffle

Stephanie Bayer and Jens Groth*

University College London
{s.bayer,j.groth}@cs.ucl.ac.uk

Abstract. Mix-nets are used in e-voting schemes and other applications that require anonymity. Shuffles of homomorphic encryptions are often used in the construction of mix-nets. A shuffle permutes and re-encrypts a set of ciphertexts, but as the plaintexts are encrypted it is not possible to verify directly whether the shuffle operation was done correctly or not. Therefore, to prove the correctness of a shuffle it is often necessary to use zero-knowledge arguments.

We propose an honest verifier zero-knowledge argument for the correctness of a shuffle of homomorphic encryptions. The suggested argument has sublinear communication complexity that is much smaller than the size of the shuffle itself. In addition the suggested argument matches the lowest computation cost for the verifier compared to previous work and also has an efficient prover. As a result our scheme is significantly more efficient than previous zero-knowledge schemes in literature.

We give performance measures from an implementation where the correctness of a shuffle of 100,000 ElGamal ciphertexts is proved and verified in around 2 minutes.

Keywords: Shuffle, zero-knowledge, ElGamal encryption, mix-net, voting, anonymous broadcast.

1 Introduction

A mix-net [4] is a multi-party protocol which is used in e-voting or other applications which require anonymity. It allows a group of senders to input a number of encrypted messages to the mix-net, which then outputs the messages in random order. It is common to construct mix-nets from shuffles.

Informally, a shuffle of ciphertexts $C_1, \ldots, C_N$ is a set of ciphertexts $C'_1, \ldots, C'_N$ with the same plaintexts in permuted order. In our work we will examine shuffle protocols constructed from homomorphic encryption schemes. That means for a given public key pk, messages M_1, M_2, and randomness ρ_1, ρ_2 the encryption function satisfies $\mathcal{E}_{pk}(M_1 M_2; \rho_1 + \rho_2) = \mathcal{E}_{pk}(M_1; \rho_1)\mathcal{E}_{pk}(M_2; \rho_2)$. Thus, we may construct a shuffle of $C_1, \ldots, C_N$ by selecting a permutation $\pi \in \Sigma_N$ and randomizers $\rho_1, \ldots \rho_N$, and calculating $C'_1 = C_{\pi(1)}\mathcal{E}_{pk}(1; \rho_1), \ldots, C'_N = C_{\pi(N)}\mathcal{E}_{pk}(1; \rho_N)$.

A common construction of mix-nets is to let the mix-servers take turns in shuffling the ciphertexts. If the encryption scheme is semantically secure the shuffle $C'_1, \ldots, C'_N$

* Both authors are supported by EPSRC grant number EP/G013829/1.

D. Pointcheval and T. Johansson (Eds.): EUROCRYPT 2012, LNCS 7237, pp. 263–280, 2012.

output by a mix-server does not reveal the permutation or the messages. But this also means that a malicious mix-server in the mix-net could substitute some of the ciphertexts without being detected. In a voting protocol, it could for instance replace all ciphertexts with encrypted votes for candidate X. Therefore, our goal is to construct an interactive zero-knowledge argument that makes it possible to verify that the shuffle was done correctly (soundness), but reveals nothing about the permutation or the randomizers used (zero-knowledge).

Efficiency is a major concern in arguments for the correctness of a shuffle. In large elections it is realistic to end up shuffling millions of votes. This places considerable strain on the performance of the zero-knowledge argument both in terms of communication and computation. We will construct an honest verifier zero-knowledge argument for correctness of a shuffle that is highly efficient both in terms of communication and computation.

1.1 Related Work

The idea of a shuffle was introduced by Chaum [4] but he didn't give any method to guarantee the correctness. Many suggestions had been made how to build mix-nets or prove the correctness of a shuffle since then, but many of these approaches have been partially or fully broken, and the remaining schemes sometimes suffer from other drawbacks. None of these drawbacks are suffered by the shuffle scheme of Wikström [27] and approaches based on zero-knowledge arguments. Since zero-knowledge arguments achieve better efficiency they will be the focus of our paper.

Early contributions using zero-knowledge arguments were made by Sako and Killian [23], and Abe [1–3]. Furukawa and Sako [10] and Neff [20, 21] proposed the first shuffle arguments for ElGamal encryption with a complexity that depends linearly on the number of ciphertexts.

Furukawa and Sako's approach is based on permutation matrices and has been refined further [7, 16]. Furukawa, Miyachi, Mori, Obana, and Sako [8] presented an implementation of a shuffle argument based on permutation matrices and tested it on mix-nets handling 100,000 ElGamal ciphertexts. Recently, Furukawa and Sako [9] have reported on another implementation based on elliptic curve groups.

Wikström [28] also used the idea of permutation matrices and suggested a shuffle argument which splits in an offline and online phase. Furthermore, Terelius and Wikström [25] constructed conceptually simple shuffle arguments that allowed the restriction of the shuffles to certain classes of permutations. Both protocols are implemented in the Verificatum mix-net library [29].

Neff's approach [20] is based on the invariance of polynomials under permutation of the roots. This idea was picked up by Groth who suggested a perfect honest verifier zero-knowledge protocol [14]. Later Groth and Ishai [15] proposed the first shuffle argument where the communication complexity is sublinear in the number of ciphertexts.

1.2 Our Contribution

Results. We propose a practical efficient honest verifier zero-knowledge argument for the correctness of a shuffle. Our argument is very efficient; in particular we drastically

decrease the communication complexity compared to previous shuffle arguments. We cover the case of shuffles of ElGamal ciphertexts but it is possible to adapt our argument to other homomorphic cryptosystems as well.

Our argument has sublinear communication complexity. When shuffling N ciphertexts, arranged in an $m \times n$ matrix, our argument transmits $O(m+n)$ group elements giving a minimal communication complexity of $O(\sqrt{N})$ if we choose $m = n$. In comparison, Groth and Ishai's argument [15] communicates $\Theta(m^2+n)$ group elements and all other state of the art shuffle arguments communicate $\Theta(N)$ elements.

The disadvantage of Groth and Ishai's argument compared to the schemes with linear communication was that the prover's computational complexity was on the order of $O(Nm)$ exponentiations. It was therefore only possible to choose small m. In comparison, our prover's computational complexity is $O(N \log m)$ exponentiations for constant round arguments and $O(N)$ exponentiations if we allow a logarithmic number of rounds. In practice, we do not need to increase the round complexity until m gets quite large, so the speedup in the prover's computation is significant compared to Groth and Ishai's work and is comparable to the complexity seen in arguments with linear communication. Moreover, the verifier is fast in our argument making the entire process very light from the verifier's point of view.

In Sect. 6 we report on an implementation of our shuffle argument using shuffles of 100,000 ElGamal ciphertexts. We compare this implementation on the parameter setting for ElGamal encryption used in [8] and find significant improvements in both communication and computation. We also compare our implementation to the shuffle argument in the Verificatum mix-net [29] and find significant improvements in communication and moderate improvements in computation.

New Techniques. Groth [13] proposed efficient sublinear size arguments to be used in connection with linear algebra over a finite field. We combine these techniques with Groth and Ishai's sublinear size shuffle argument. The main problem in applying Groth's techniques to shuffling is that they were designed for use in finite fields and not for use with group elements or ciphertexts. It turns out though that the operations are mostly linear and therefore it is possible to carry them out "in the exponent"; somewhat similar to what is often done in threshold cryptography. Using this adaptation we are able to construct an efficient multi-exponentiation argument that a ciphertext C is the product of a set of known ciphertexts $C_1, \ldots, C_N$ raised to a set of hidden committed values $a_1, \ldots, a_N$. This is the main bottleneck in our shuffle argument and therefore gives us a significant performance improvement.

Groth's sublinear size zero-knowledge arguments also suffered from a performance bottleneck in the prover's computation. At some juncture it is necessary to compute the sums of the diagonal strips in a product of two matrices. This problem is made even worse in our setting because when working with group elements we have to compute these sums in the exponents. By adapting techniques for polynomial multiplication such as Toom-Cook [5, 26] and the Fast Fourier Transform [6] we are able to reduce this computation. Moreover, we generalize the interactive technique of Groth [13] to further reduce the prover's computation.

2 Preliminaries

We use vector notation in the paper, and we write $\boldsymbol{xy} = (x_1y_1, \ldots, x_ny_n)$ for the entry-wise product and correspondingly $\boldsymbol{x}^z = (x_1^z, \ldots, x_n^z)$ for vectors of group elements. Similar, we write $\boldsymbol{x}_\pi$ if the entries of vector $\boldsymbol{x}$ are permuted by the permutation π, i.e., $\boldsymbol{x}_\pi = (x_{\pi(1)}, \ldots, x_{\pi(n)})$. We use the standard inner product $\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^n x_iy_i$ for vectors of field elements .

Our shuffle argument is constructed with homomorphic encryption. An encryption scheme is homomorphic if for a public key pk, messages M_1, M_2, and randomness ρ_1, ρ_2 the encryption function satisfies $\mathcal{E}_{pk}(M_1M_2; \rho_1 + \rho_2) = \mathcal{E}_{pk}(M_1; \rho_1)\mathcal{E}_{pk}(M_2; \rho_2)$. We will focus on ElGamal encryption, but our construction works with many different homomorphic encryption schemes where the message space has large prime order q. To simplify the presentation, we will use notation from linear algebra. We define $\boldsymbol{C}^{\boldsymbol{a}} = \prod_{i=1}^n C_i^{a_i}$ for vectors $(C_1, \ldots, C_n) \in \mathbb{H}^n$ and $(a_1, \ldots, a_n)^T \in \mathbb{Z}_q^n$, where $\mathbb{H}$ is the ciphertext space.

Likewise, we need a homomorphic commitment scheme in our protocol. Again informally, a commitment scheme is homomorphic if for a commitment key ck, messages a, b, and randomizers r, s it holds that $\mathrm{com}_{ck}(a + b; r + s) = \mathrm{com}_{ck}(a; r)\mathrm{com}_{ck}(b; s)$. We also demand that it is possible to commit to n elements in $\mathbb{Z}_q$, where q is a large prime, at the same time. I.e., given a vector $(a_1, \ldots, a_n)^T \in \mathbb{Z}_q^n$ we can compute a single short commitment $c = \mathrm{com}_{ck}(\boldsymbol{a}; r) \in \mathbb{G}$, where $\mathbb{G}$ is the commitment space. The length-reducing property of the commitment scheme mapping n elements to a single commitment is what allows us to get sublinear communication complexity. Many homomorphic commitment schemes with this property can be used, but for convenience we just focus on a generalization of the Pedersen commitment scheme [22]. To simplify notation, we write $\boldsymbol{c}_A = \mathrm{com}_{ck}(A; \boldsymbol{r})$ for the vector $(c_{A_1}, \ldots, c_{A_m}) = (\mathrm{com}_{ck}(\boldsymbol{a}_1; r_1), \ldots, \mathrm{com}_{ck}(\boldsymbol{a}_m; r_m)$ when A is a matrix with column vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$.

2.1 Special Honest Verifier Zero-Knowledge Argument of Knowledge

In the shuffle arguments we consider a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ both of which are probabilistic polynomial time interactive algorithms. We assume the existence of a probabilistic polynomial time setup algorithm $\mathcal{G}$ that when given a security parameter λ returns a common reference string σ.

The common reference string will be $\sigma = (pk, ck)$, where pk and ck are public keys for the ElGamal encryption scheme and the generalized Pedersen commitment scheme. The encryption scheme and the commitment scheme may use different underlying groups, but we require that they have the same prime order q. We will write $\mathbb{G}$ for the group used by the commitment scheme and write $\mathbb{H}$ for the ciphertext space.

The setup algorithm can also return some side-information that may be used by an adversary; however, we require that even with this side-information the commitment scheme should remain computationally binding. The side-information models that the keys may be set up using some multi-party computation protocol that leaks some information, the adversary may see some decryptions or even learn the decryption key, etc. Our protocol for verifying the correctness of a shuffle is secure in the presence of such leaks as long as the commitment scheme is computationally binding.

Let R be a polynomial time decidable ternary relation, we call w a witness for a statement x if $(\sigma, x, w) \in R$. We define the language

$$L_\sigma := \{x \mid \exists w : (\sigma, x, w) \in R\}$$

as the set of statements x that have a witness w for the relation R.

The public transcript produced by $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. The last part of the transcript is either accept or reject from the verifier. We write $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b,\ b \in \{0, 1\}$ for rejection or acceptance.

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

$$\Pr[(\sigma, \text{hist}) \leftarrow \mathcal{G}(1^\lambda); (x, w) \leftarrow \mathcal{A}(\sigma, \text{hist}) : (\sigma, x, w) \notin R \text{ or } \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x) \rangle = 1] = 1$$

Computational soundness:

$$\Pr[(\sigma, \text{hist}) \leftarrow \mathcal{G}(1^\lambda); x \leftarrow \mathcal{A}(\sigma, \text{hist}) : x \notin L_\sigma \text{ and } \langle \mathcal{A}, \mathcal{V}(\sigma, x) \rangle = 1] \approx 0$$

Definition 2 (Public coin). *An argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called* public coin *if the verifier chooses his messages uniformly at random and independently of the messages sent by the prover, i.e., the challenges correspond to the verifier's randomness ρ.*

Definition 3 (Special honest verifier zero-knowledge). *A public coin argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called a* perfect special honest verifier zero knowledge (SHVZK) *argument for R with common reference string generator $\mathcal{G}$ if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all non-uniform polynomial time interactive adversaries $\mathcal{A}$ we have*

$$\begin{aligned} &\Pr[(\sigma, \text{hist}) \leftarrow \mathcal{G}(1^\lambda); (x, w, \rho) \leftarrow \mathcal{A}(\sigma, \text{hist}); \\ &\quad tr \leftarrow \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x; \rho) \rangle : (\sigma, x, w) \in R \text{ and } \mathcal{A}(tr) = 1] \\ = &\Pr[(\sigma, \text{hist}) \leftarrow \mathcal{G}(1^\lambda); (x, w, \rho) \leftarrow \mathcal{A}(\sigma, \text{hist}); \\ &\quad tr \leftarrow \mathcal{S}(\sigma, x, \rho) : (\sigma, x, w) \in R \text{ and } \mathcal{A}(tr) = 1] \end{aligned}$$

To construct a fully zero-knowledge argument secure against *arbitrary* verifiers in the common reference string model one can first construct a SHVZK argument and then convert it into a fully zero-knowledge argument [11, 12]. This conversion has constant additive overhead, so it is very efficient and allows us to focus on the simpler problem of getting SHVZK against honest verifiers.

To define an argument of knowledge we follow the approach of Groth and Ishai [15] and do it through witness-extended emulation first introduced by Lindell [19]. This definition informally says that given an adversary that produces an acceptable argument with some probability, there exist an emulator that produces a similar argument with the same probability and at the same time provides a witness w.

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

$$\begin{aligned}&\Pr[(\sigma, \text{hist}) \leftarrow \mathcal{G}(1^\lambda); (x, s) \leftarrow \mathcal{A}(\sigma, \text{hist}); tr \leftarrow \langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x)\rangle : \mathcal{A}(tr) = 1] \\ \approx\ &\Pr[(\sigma, \text{hist}) \leftarrow \mathcal{G}(1^\lambda); (x, s) \leftarrow \mathcal{A}(\sigma, \text{hist}); (tr, w) \leftarrow \mathcal{X}^{\langle \mathcal{P}^*(\sigma,x,s), \mathcal{V}(\sigma,x)\rangle}(\sigma, x, \rho) : \\ &\quad \mathcal{A}(tr) = 1 \textit{ and if } tr \textit{ is accepting then } (\sigma, x, w) \in R].\end{aligned}$$

In the definition, s can be interpreted as the state of $\mathcal{P}^*$, including the randomness. So whenever $\mathcal{P}^*$ is able to make a convincing argument when in state s, the emulator can extract a witness at the same time giving us an argument of knowledge. This definition automatically implies soundness.

3 Shuffle Argument

We will give an argument of knowledge of a permutation $\pi \in \Sigma_N$ and randomness $\{\rho_i\}_{i=1}^N$ such that for given ciphertexts $\{C_i\}_{i=1}^N, \{C'_i\}_{i=1}^N$ we have $C'_i = C_{\pi(i)}\mathcal{E}_{pk}(1; \rho_i)$. The shuffle argument combines a multi-exponentiation argument, which allows us to prove that the product of a set of ciphertexts raised to a set of committed exponents yields a particular ciphertext, and a product argument, which allows us to prove that a set of committed values has a particular product. The multi-exponentiation argument is given in Sect. 4 and the product argument is given in Sect. 5. In this section, we will give an overview of the protocol and explain how a multi-exponentiation argument can be combined with a product argument to yield an argument for the correctness of a shuffle.

The first step for the prover is to commit to the permutation. This is done by committing to $\pi(1), \ldots, \pi(N)$. The prover will now receive a challenge x and commit to $x^{\pi(1)}, \ldots, x^{\pi(N)}$. The prover will give an argument of knowledge of openings of the commitments to permutations of $1, \ldots, N$ and $x^1, \ldots, x^N$ and demonstrate that the same permutation has been used in both cases. This means the prover has a commitment to $x^1, \ldots, x^N$ permuted in an order that was fixed before the prover saw x.

To check that the same permutation has been used in both commitments the verifier sends random challenges y and z. By using the homomorphic properties of the commitment scheme the prover can in a verifiable manner compute commitments to $d_1 - z = y\pi(1) + x^{\pi(1)} - z, \ldots, d_N - z = y\pi(N) + x^{\pi(N)} - z$. Using the product argument from Sect. 5 the prover shows that $\prod_{i=1}^N (d_i - z) = \prod_{i=1}^N (yi + x^i - z)$. Observe that we have two identical degree N polynomials in z since the only difference is that the roots have been permuted. The verifier does not know a priori that the two polynomials are identical but can by the Schwartz-Zippel lemma deduce that the prover has negligible chance over the choice of z of making a convincing argument unless indeed there is a permutation π such that $d_1 = y\pi(1) + x^{\pi(1)}, \ldots, d_N = y\pi(N) + x^{\pi(N)}$. Furthermore, there is negligible probability over the choice of y of this being true unless the first commitment contains $\pi(1), \ldots, \pi(N)$ and the second commitment contains $x^{\pi(1)}, \ldots, x^{\pi(N)}$.

The prover has commitments to $x^{\pi(1)}, \ldots, x^{\pi(N)}$ and uses the multi-exponentiation argument from Sect. 4 to demonstrate that there exists a ρ such that $\prod_{i=1}^{N} C_i^{x^i} = \mathcal{E}_{pk}(1;\rho)\prod_{i=1}^{N}(C'_i)^{x^{\pi(i)}}$. The verifier does not see the committed values and thus does not learn what the permutation is. However, from the homomorphic properties of the encryption scheme the verifier can deduce $\prod_{i=1}^{N} M_i^{x^i} = \prod_{i=1}^{N}(M'_i)^{x^{\pi(i)}}$ for some permutation π that was chosen before the challenge x was sent to the prover. Taking discrete logarithms we have the polynomial identity $\sum_{i=1}^{N}\log(M_i)x^i = \sum_{i=1}^{N}\log(M'_{\pi^{-1}(i)})x^i$. There is negligible probability over the choice of x of this equality holding true unless $M'_1 = M_{\pi(1)}, \ldots, M'_N = M_{\pi(N)}$. This shows that we have a correct shuffle.

Common reference string: pk, ck.
Statement: $\boldsymbol{C}, \boldsymbol{C'} \in \mathbb{H}^N$ with $N = mn$.
Prover's witness: $\pi \in \Sigma_N$ and $\boldsymbol{\rho} \in \mathbb{Z}_q^N$ such that $\boldsymbol{C'} = \mathcal{E}_{pk}(\mathbf{1};\boldsymbol{\rho})\boldsymbol{C}_\pi$.
Initial message: Pick $\boldsymbol{r} \leftarrow \mathbb{Z}_q^m$, set $\boldsymbol{a} = \{\pi(i)\}_{i=1}^N$ and compute $\boldsymbol{c}_A = \mathrm{com}_{ck}(\boldsymbol{a};\boldsymbol{r})$.
Send: $\boldsymbol{c}_A$
Challenge: $x \leftarrow \mathbb{Z}_q^*$.
Answer Pick $\boldsymbol{s} \in \mathbb{Z}_q^m$, set $\boldsymbol{b} = \{x^{\pi(i)}\}_{i=1}^N$ and compute $\boldsymbol{c}_B = \mathrm{com}_{ck}(\boldsymbol{b};\boldsymbol{s})$.
Send: $\boldsymbol{c}_B$
Challenge: $y, z \leftarrow \mathbb{Z}_q^*$.
Answer: Define $\boldsymbol{c}_{-z} = \mathrm{com}_{ck}(-z, \ldots, -z; \mathbf{0})$ and $\boldsymbol{c}_D = \boldsymbol{c}_A^{\,y}\boldsymbol{c}_B$. Compute $\boldsymbol{d} = y\boldsymbol{a} + \boldsymbol{b}$, and $\boldsymbol{t} = y\boldsymbol{r} + \boldsymbol{s}$. Engage in a product argument as described in Sect. 5 of openings $d_1 - z, \ldots, d_N - z$ and $\boldsymbol{t}$ such that

$$\boldsymbol{c}_D\boldsymbol{c}_{-z} = \mathrm{com}_{ck}(\boldsymbol{d} - \boldsymbol{z}; \boldsymbol{t}\,) \qquad \text{and} \qquad \prod_{i=1}^{N}(d_i - z) = \prod_{i=1}^{N}(yi + x^i - z)\,.$$

Compute $\rho = -\boldsymbol{\rho}\cdot\boldsymbol{b}$ and set $\boldsymbol{x} = (x, x^2, \ldots, x^N)^T$. Engage in a multi-exponentiation argument as described in Sect. 4 of $\boldsymbol{b}$, $\boldsymbol{s}$ and ρ such that

$$\boldsymbol{C}^{\boldsymbol{x}} = \mathcal{E}_{pk}(1;\rho)\boldsymbol{C'}^{\boldsymbol{b}} \qquad \text{and} \qquad \boldsymbol{c}_B = \mathrm{com}_{ck}(\boldsymbol{b};\boldsymbol{s})$$

The two arguments can be run in parallel. Furthermore, the multi-exponentiation argument can be started in round 3 after the computation of the commitments $\boldsymbol{c}_B$.
Verification: The verifier checks $\boldsymbol{c}_A, \boldsymbol{c}_B \in \mathbb{G}^m$ and computes $\boldsymbol{c}_{-z}, \boldsymbol{c}_D$ as described above and computes $\prod_{i=1}^{N}(yi + x^i - z)$ and $\boldsymbol{C}^{\boldsymbol{x}}$. The verifier accepts if the product and multi-exponentiation arguments both are valid.

Theorem 5 (Full paper). *The protocol is a public coin perfect SHVZK argument of knowledge of $\pi \in \Sigma_N$ and $\boldsymbol{\rho} \in \mathbb{Z}_q^N$ such that $\boldsymbol{C'} = \mathcal{E}_{pk}(\mathbf{1};\boldsymbol{\rho})\boldsymbol{C}_\pi$.*

4 Multi-exponentiation Argument

Given ciphertexts $C_{11}, \ldots, C_{mn}$, and C we will in this section give an argument of knowledge of openings of commitments $\boldsymbol{c}_A$ to $A = \{a_{ij}\}_{i,j=1}^{n,m}$ such that

$$C = \mathcal{E}_{pk}(1;\rho)\prod_{i=1}^{m}\boldsymbol{C}_i^{\boldsymbol{a}_i} \qquad \text{and} \qquad \boldsymbol{c}_A = \mathrm{com}_{ck}(A;\boldsymbol{r})\,,$$

where $\boldsymbol{C}_i = (C_{i1}, \ldots, C_{in})$ and $\boldsymbol{a}_j = (a_{1j}, \ldots, a_{nj})^T$.

To explain the idea in the protocol let us for simplicity assume $\rho = 0$ and the prover knows the openings of $\boldsymbol{c}_A$, and leave the question of SHVZK for later. In other words, we will for now just explain how to convince the verifier in a communication-efficient manner that $C = \prod_{i=1}^{m} \boldsymbol{C}_i^{\boldsymbol{a}_i}$. The prover can calculate the ciphertexts

$$E_k = \prod_{\substack{1 \le i,j \le m \\ j=(k-m)+i}} \boldsymbol{C}_i^{\boldsymbol{a}_j} ,$$

where $E_m = C$. To visualize this consider the following matrix

$$\begin{array}{c} \begin{pmatrix} \boldsymbol{a}_1 & \dots & \boldsymbol{a}_m \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{C}_1 \\ \boldsymbol{C}_2 \\ \vdots \\ \boldsymbol{C}_m \end{pmatrix} \begin{pmatrix} \boldsymbol{C}_1^{\boldsymbol{a}_1} & \ddots & \boldsymbol{C}_1^{\boldsymbol{a}_m} \\ \boldsymbol{C}_2^{\boldsymbol{a}_1} & \ddots & \boldsymbol{C}_2^{\boldsymbol{a}_m} \\ \ddots & \ddots \quad \ddots & \ddots \\ \boldsymbol{C}_m^{\boldsymbol{a}_1} & \ddots & \boldsymbol{C}_m^{\boldsymbol{a}_m} \end{pmatrix} \begin{matrix} E_{2m-1} \\ \vdots \\ E_{m+1} \end{matrix} \\ \begin{matrix} E_1 & \dots & E_{m-1} & E_m \end{matrix} \end{array}$$

The prover sends the ciphertexts $E_1, \dots, E_{2m-1}$ to the verifier. The ciphertext $C = E_m$ is the product of the main diagonal and the other E_k's are the products of the other diagonals. The prover will use a batch-proof to simultaneously convince the verifier that all the diagonal products give their corresponding E_k.

The verifier selects a challenge $x \leftarrow \mathbb{Z}_q^*$. The prover sets $\boldsymbol{x} = (x, x^2, \dots, x^m)^T$, opens $\boldsymbol{c}_A^{\boldsymbol{x}}$ to $\boldsymbol{a} = \sum_{j=1}^{m} x^j \boldsymbol{a}_j$, and the verifier checks

$$C^{x^m} \prod_{\substack{k=1 \\ k \neq m}}^{2m-1} E_k^{x^k} = \prod_{i=1}^{m} \boldsymbol{C}_i^{(x^{m-i}\boldsymbol{a})} .$$

Since x is chosen at random, the prover has negligible probability of convincing the verifier unless the x^k-related terms match on each side of the equality for all k. In particular, since $\boldsymbol{a} = \sum_{j=1}^{m} x^j \boldsymbol{a}_j$ the x^m-related terms give us

$$C^{x^m} = \prod_{i=1}^{m} \boldsymbol{C}_i^{\,x^{m-i} \sum_{\substack{1 \le j \le m \\ m=m-i+j}} x^j \boldsymbol{a}_j} = \left(\prod_{i=1}^{m} \boldsymbol{C}_i^{\boldsymbol{a}_i} \right)^{x^m}$$

and allow the verifier to conclude $C = \prod_{i=1}^{m} \boldsymbol{C}_i^{\boldsymbol{a}_i}$.

Finally, to make the argument honest verifier zero-knowledge we have to avoid leaking information about the exponent vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_m$. The prover therefore commits to a random vector $\boldsymbol{a}_0 \leftarrow \mathbb{Z}_q^n$ and after she sees the challenge x she reveals $\boldsymbol{a} = \boldsymbol{a}_0 + \sum_{j=1}^{m} x^j \boldsymbol{a}_j$. Since $\boldsymbol{a}_0$ is chosen at random this vector does not leak any information about the exponents.

Another possible source of leakage is the products of the diagonals. The prover will therefore randomize each E_k by multiplying it with a random ciphertext $\mathcal{E}_{pk}(G^{b_k};\tau_k)$. Now each E_k is a uniformly random group element in $\mathbb{H}$ and will therefore not leak information about the exponents. Of course, this would make it possible to encrypt anything in the E_k and allow cheating. To get around this problem the prover has to commit to the b_k's used in the random encryptions and the verifier will check that the prover uses $b_m = 0$. The full argument that also covers the case $\rho \neq 0$ can be found below.

Common reference string: pk, ck.
Statement: $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_m \in \mathbb{H}^n$, $C \in \mathbb{H}$, and $\boldsymbol{c}_A \in \mathbb{G}^m$
Prover's witness: $A = \{\boldsymbol{a}_j\}_{j=1}^m \in \mathbb{Z}_q^{n\times m}$, $\boldsymbol{r} \in \mathbb{Z}_q^m$, and $\rho \in \mathbb{Z}_q$ such that

$$C = \mathcal{E}_{pk}(1;\rho)\prod_{i=1}^{m}\boldsymbol{C}_i^{\boldsymbol{a}_i} \qquad \text{and} \qquad \boldsymbol{c}_A = \mathrm{com}_{ck}(A;\boldsymbol{r})$$

Initial message: Pick $\boldsymbol{a}_0 \leftarrow \mathbb{Z}_q^n, r_0 \leftarrow \mathbb{Z}_q$, and $b_0, s_0, \tau_0 \ldots, b_{2m-1}, s_{2m-1}, \tau_{2m-1} \leftarrow \mathbb{Z}_q$ and set $b_m = 0, s_m = 0, \tau_m = \rho$. Compute for $k = 0, \ldots, 2m-1$

$$c_{A_0} = \mathrm{com}_{ck}(\boldsymbol{a}_0; r_0)\ ,\ c_{B_k} = \mathrm{com}_{ck}(b_k; s_k)\ ,\ E_k = \mathcal{E}_{pk}(G^{b_k};\tau_k)\prod_{\substack{i=1,j=0\\ j=(k-m)+i}}^{m,m}\boldsymbol{C}_i^{\boldsymbol{a}_j}$$

Send: $c_{A_0}, \{c_{B_k}\}_{k=0}^{2m-1}, \{E_k\}_{k=0}^{2m-1}$.
Challenge: $x \leftarrow \mathbb{Z}_q^*$.
Answer: Set $\boldsymbol{x} = (x, x^2, \ldots, x^m)^T$ and compute

$$\boldsymbol{a} = \boldsymbol{a}_0 + A\boldsymbol{x} \qquad r = r_0 + \boldsymbol{r}\cdot\boldsymbol{x} \qquad b = b_0 + \sum_{k=1}^{2m-1} b_k x^k$$
$$s = s_0 + \sum_{k=1}^{2m-1} s_k x^k \qquad \tau = \tau_0 + \sum_{k=1}^{2m-1}\tau_k x^k\ .$$

Send: $\boldsymbol{a}, r, b, s, \tau$.
Verification: Check $c_{A_0}, c_{B_0}, \ldots, c_{B_{2m-1}} \in \mathbb{G}$, and $E_0, \ldots, E_{2m-1} \in \mathbb{H}$, and $\boldsymbol{a} \in \mathbb{Z}_q^n$, and $r, b, s, \tau \in \mathbb{Z}_q$, and accept if $c_{B_m} = \mathrm{com}_{ck}(0;0)$ and $E_m = C$, and

$$c_{A_0}\boldsymbol{c}_A^{\boldsymbol{x}} = \mathrm{com}_{ck}(\boldsymbol{a}; r) \qquad c_{B_0}\prod_{k=1}^{2m-1} c_{B_k}^{x^k} = \mathrm{com}_{ck}(b; s)$$
$$E_0\prod_{k=1}^{2m-1}E_k^{x^k} = \mathcal{E}_{pk}(G^b;\tau)\prod_{i=1}^{m}\boldsymbol{C}_i^{x^{m-i}\boldsymbol{a}}\ .$$

Theorem 6 (Full paper). *The protocol above is a public coin perfect SHVZK argument of knowledge of openings* $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m, \boldsymbol{r}$ *and randomness* ρ *such that* $C = \mathcal{E}_{pk}(1;\rho)\prod_{i=1}^{m} \boldsymbol{C}_i^{\boldsymbol{a}_i}$.

4.1 The Prover's Computation

The argument we just described has efficient verification and very low communication complexity, but the prover has to compute

$$E_0, \ldots, E_{2m-1} .$$

In this section we will for clarity ignore the randomization needed to get honest verifier zero-knowledge, which can be added in a straightforward manner at little extra computational cost. So let us say we need to compute for $k = 1, \ldots, 2m-1$ the elements

$$E_k = \prod_{\substack{i=1,j=1 \\ j=(k-m)+i}}^{m,m} \boldsymbol{C}_i^{\boldsymbol{a}_j} .$$

This can be done by first computing the m^2 products $\boldsymbol{C}_i^{\boldsymbol{a}_j}$ and then computing the E_k's as suitable products of some of these values. Since each product $\boldsymbol{C}_i^{\boldsymbol{a}_j}$ is of the form $\prod_{\ell=1}^{n} C_{i\ell}^{a_{j\ell}}$ this gives a total of $m^2 n$ exponentiations in $\mathbb{H}$. For large m this cost is prohibitive.

It turns out that we can do much better by using techniques inspired by multiplication of integers and polynomials, such as Karatsuba [17], Toom-Cook [5, 26] and using the Fast Fourier Transform [6]. A common theme in these techniques is to compute the coefficients of the product $p(x)q(x)$ of two degree $m-1$ polynomials $p(x)$ and $q(x)$ by evaluating $p(x)q(x)$ in $2m-1$ points $\omega_0, \ldots, \omega_{2m-2}$ and using polynomial interpolation to recover the coefficients of $p(x)q(x)$ from $p(\omega_0)q(\omega_0), \ldots, p(\omega_{2m-2})q(\omega_{2m-2})$.

If we pick $\omega \in \mathbb{Z}_q$ we can evaluate the vectors

$$\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega^{m-i}} \qquad \text{and} \qquad \sum_{j=1}^{m} \omega^{j-1}\boldsymbol{a}_j .$$

This gives us

$$\left(\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega^{m-i}}\right)^{\sum_{j=1}^{m}\omega^{j-1}\boldsymbol{a}_j} = \prod_{k=1}^{2m-1}\left(\prod_{\substack{i=1,j=1 \\ j=(k-m)+i}}^{m,m} \boldsymbol{C}_i^{\boldsymbol{a}_j}\right)^{\omega^{k-1}} = \prod_{k=1}^{2m-1} E_k^{\omega^{k-1}} .$$

Picking $2m-1$ different $\omega_0, \ldots, \omega_{2m-2} \in \mathbb{Z}_q$ we get the $2m-1$ ciphertexts

$$\prod_{k=1}^{2m-1} E_k^{\omega_0^{k-1}}, \ \ldots, \ \prod_{k=1}^{2m-1} E_k^{\omega_{2m-2}^{k-1}} .$$

The $\omega_0, \ldots, \omega_{2m-2}$ are different and therefore the transposed Vandermonde matrix

$$\begin{pmatrix} 1 & \ldots & 1 \\ \vdots & & \vdots \\ \omega_0^{2m-2} & \ldots & \omega_{2m-2}^{2m-2} \end{pmatrix}$$

is invertible. Let $\boldsymbol{y}_i = (y_0, \ldots, y_{2m-2})^T$ be the ith column of the inverse matrix. We can now compute E_i as

$$E_i = \prod_{\ell=0}^{2m-2} \left(\prod_{k=1}^{2m-1} E_k^{\omega_\ell^{k-1}} \right)^{y_\ell} = \prod_{\ell=0}^{2m-2} \left(\left(\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_\ell^{m-i}} \right)^{\sum_{j=1}^{m} \omega_\ell^{j-1} \boldsymbol{a}_j} \right)^{y_\ell}.$$

This means the prover can compute $E_1, \ldots, E_{2m-1}$ as linear combinations of

$$\left(\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_0^{m-i}} \right)^{\sum_{j=1}^{m} \omega_0^{j-1} \boldsymbol{a}_j} \quad \ldots \quad \left(\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_{2m-2}^{m-i}} \right)^{\sum_{j=1}^{m} \omega_{2m-2}^{j-1} \boldsymbol{a}_j}.$$

The expensive step in this computation is to compute $\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_0^{m-i}}, \ldots, \prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_{2m-2}^{m-i}}$.

If $2m-2$ is a power of 2 and $2m-2|q-1$ we can pick $\omega_1, \ldots, \omega_{2m-2}$ as roots of unity, i.e., $\omega_k^{2m-2} = 1$. This allows us to use the Fast Fourier Transformation "in the exponent" to simultaneously calculate $\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_k^{m-i}}$ in all of the roots of unity using only $O(mn \log m)$ exponentiations. This is asymptotically the fastest technique we know for computing $E_0, \ldots, E_{2m-2}$.

Unfortunately, the FFT is not well suited for being used in combination with multi-exponentiation techniques and in practice it therefore takes a while before the asymptotic behavior kicks in. For small m it is therefore useful to consider other strategies. Inspired by the Toom-Cook method for integer multiplication, we may for instance choose $\omega_0, \omega_1, \ldots, \omega_{2m-2}$ to be small integers. When m is small even the largest exponent ω_k^{2m-2} will remain small. For instance, if $m = 4$ we may choose $\omega_k \in \{0, -1, 1, -2, 2, -3, 3\}$, which makes the largest exponent $\omega_k^{m-1} = 3^3 = 27$. This makes it cheap to compute each $\prod_{i=1}^{m} \boldsymbol{C}_i^{\omega_k^{m-i}}$ because the exponents are very small.

The basic step of Toom-Cook sketched above can be optimized by choosing the evaluation points carefully. However, the performance degrades quickly as m grows. Using recursion it is possible to get subquadratic complexity also for large m, however, the cost still grows relatively fast. In the next section we will therefore describe an interactive technique for reducing the prover's computation. In our implementation, see Sect. 6, we have used a combination of the interactive technique and Toom-Cook as the two techniques work well together.

4.2 Trading Computation for Interaction

We will present an interactive technique that can be used to reduce the prover's computation. The prover wants to show that C has the same plaintext as the product of the main diagonal of following matrix (here illustrated for $m = 16$).

$$\begin{pmatrix} \boldsymbol{C}_1^{\boldsymbol{a}_1}\ \boldsymbol{C}_1^{\boldsymbol{a}_2}\ \boldsymbol{C}_1^{\boldsymbol{a}_3}\ \boldsymbol{C}_1^{\boldsymbol{a}_4} & & & \\ \boldsymbol{C}_2^{\boldsymbol{a}_1}\ \boldsymbol{C}_2^{\boldsymbol{a}_2}\ \boldsymbol{C}_2^{\boldsymbol{a}_3}\ \boldsymbol{C}_2^{\boldsymbol{a}_4} & & \ddots & \\ \boldsymbol{C}_3^{\boldsymbol{a}_1}\ \boldsymbol{C}_3^{\boldsymbol{a}_2}\ \boldsymbol{C}_3^{\boldsymbol{a}_3}\ \boldsymbol{C}_3^{\boldsymbol{a}_4} & & & \\ \boldsymbol{C}_4^{\boldsymbol{a}_1}\ \boldsymbol{C}_4^{\boldsymbol{a}_2}\ \boldsymbol{C}_4^{\boldsymbol{a}_3}\ \boldsymbol{C}_4^{\boldsymbol{a}_4} & & & \\ & \ddots & & \\ & & \boldsymbol{C}_{13}^{\boldsymbol{a}_{13}}\ \boldsymbol{C}_{13}^{\boldsymbol{a}_{14}}\ \boldsymbol{C}_{13}^{\boldsymbol{a}_{15}}\ \boldsymbol{C}_{13}^{\boldsymbol{a}_{16}} & \\ & \ddots & \boldsymbol{C}_{14}^{\boldsymbol{a}_{13}}\ \boldsymbol{C}_{14}^{\boldsymbol{a}_{14}}\ \boldsymbol{C}_{14}^{\boldsymbol{a}_{15}}\ \boldsymbol{C}_{14}^{\boldsymbol{a}_{16}} & \\ & & \boldsymbol{C}_{15}^{\boldsymbol{a}_{13}}\ \boldsymbol{C}_{15}^{\boldsymbol{a}_{14}}\ \boldsymbol{C}_{15}^{\boldsymbol{a}_{15}}\ \boldsymbol{C}_{15}^{\boldsymbol{a}_{16}} & \\ & & \boldsymbol{C}_{16}^{\boldsymbol{a}_{13}}\ \boldsymbol{C}_{16}^{\boldsymbol{a}_{14}}\ \boldsymbol{C}_{16}^{\boldsymbol{a}_{15}}\ \boldsymbol{C}_{16}^{\boldsymbol{a}_{16}} & \end{pmatrix}$$

In the previous section the prover calculated all m^2 entries of the matrix. But we are only interested in the product along the diagonal so we can save computation by just focusing on the blocks close to the main diagonal.

Let us explain the idea in the case of $m = 16$. We can divide the matrix into 4×4 blocks and only use the four blocks that are on the main diagonal. Suppose the prover wants to demonstrate $C = \prod_{i=1}^{16} \boldsymbol{C}_i^{\boldsymbol{a}_i}$. Let us for now just focus on soundness and return to the question of honest verifier zero-knowledge later. The prover starts by sending $E_0, E_1, E_2, E_3, E_4, E_5, E_6$ that are the products along the diagonals of the elements in the blocks that we are interested in. I.e., $E_0 = \prod_{i=1}^{4} \boldsymbol{C}_{4i}^{\boldsymbol{a}_{4i-3}}, \ldots, E_6 = \prod_{i=1}^{4} \boldsymbol{C}_{4i-3}^{\boldsymbol{a}_{4i}}$ and $E_3 = C$. The verifier sends a random challenge x and using the homomorphic properties of the encryption scheme and of the commitment scheme both the prover and the verifier can compute $\boldsymbol{C}'_1, \ldots, \boldsymbol{C}'_4$ and $c_{A'_1}, \ldots, c_{A'_4}$ as

$$\boldsymbol{C}'_i = \boldsymbol{C}_{4i-3}^{x^3} \boldsymbol{C}_{4i-2}^{x^2} \boldsymbol{C}_{4i-1}^{x} \boldsymbol{C}_{4i} \qquad c_{A'_j} = c_{A_{4j-3}} c_{A_{4j-2}}^{x} c_{A_{4j-1}}^{x^2} c_{A_{4j}}^{x^3} .$$

They can also both compute $C' = \prod_{k=0}^{6} E_k^{x^k}$. The prover and the verifier now engage in an SHVZK argument for the smaller statement $C' = \prod_{i=1}^{4} \boldsymbol{C}_i'^{\boldsymbol{a}'_i}$. The prover can compute a witness for this statement with $\boldsymbol{a}'_i = \boldsymbol{a}_{4i-3} + x\boldsymbol{a}_{4i-2} + x^2\boldsymbol{a}_{4i-1} + x^3\boldsymbol{a}_{4i}$. This shows

$$C^{x^3} \prod_{\substack{k=0 \\ k \neq 3}}^{6} E_k^{x^k} = \prod_{i=1}^{4} (\boldsymbol{C}_{4i-3}^{x^3} \boldsymbol{C}_{4i-2}^{x^2} \boldsymbol{C}_{4i-1}^{x} \boldsymbol{C}_{4i})^{(\boldsymbol{a}_{4i-3} + x\boldsymbol{a}_{4i-2} + x^2\boldsymbol{a}_{4i-1} + x^3\boldsymbol{a}_{4i})} .$$

Looking at the x^3-related terms, we see this has negligible chance of holding for a random x unless $C = \prod_{i=1}^{16} \boldsymbol{C}_i^{\boldsymbol{a}_i}$, which is what the prover wanted to demonstrate.

We will generalize the technique to reduce a statement $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_m, C, c_{A_1}, \ldots, c_{A_m}$ with a factor μ to a statement $\boldsymbol{C}'_1, \ldots, \boldsymbol{C}'_{m'}, C', c_{A'_1}, \ldots, c_{A'_{m'}}$, where $m = \mu m'$. To add honest verifier zero-knowledge to the protocol, we have to prevent the E_k's from leaking information about $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$. We do this by randomizing each E_k with a random ciphertext $\mathcal{E}_{pk}(G^{b_k}; t_k)$. To prevent the prover to use the randomization to cheat she will have to commit the b_k's before seeing the challenge x.

Common Reference string: pk, ck.
Statement: $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_m \in \mathbb{H}^n$ and $C \in \mathbb{H}$ and $c_{A_1}, \ldots, c_{A_m} \in \mathbb{G}$ where $m = \mu m'$.
Prover's witness: $A \in \mathbb{Z}_q^{n \times m}, \boldsymbol{r} \in \mathbb{Z}_q^m$ and $\rho \in \mathbb{Z}_q$ such that

$$C = \mathcal{E}_{pk}(1; \rho) \prod_{i=1}^{m} \boldsymbol{C}_i^{\boldsymbol{a}_i} \qquad \text{and} \qquad \boldsymbol{c}_A = \mathrm{com}_{ck}(A; \boldsymbol{r}) \; .$$

Initial message: Pick $\boldsymbol{b} = (b_0, \ldots, b_{2\mu-2}), \boldsymbol{s}, \boldsymbol{\tau} \leftarrow \mathbb{Z}_q^{2\mu-1}$ and set $b_{\mu-1} = 0, s_{\mu-1} = 0, \tau_{\mu-1} = \rho$. Compute for $k = 0, \ldots, 2\mu - 2$

$$c_{b_k} = \mathrm{com}_{ck}(b_k; s_k) \qquad E_k = \mathcal{E}_{pk}(G^{b_k}; \tau_k) \prod_{\ell=0}^{m'-1} \prod_{\substack{i=1,j=1 \\ j=(k+1-\mu)+i}}^{\mu,\mu} \boldsymbol{C}_{\mu\ell+i}^{\boldsymbol{a}_{\mu\ell+j}} \; .$$

Send: $\boldsymbol{c}_b = (c_{b_0}, \ldots, c_{b_{2\mu-2}})$ and $\boldsymbol{E} = (E_0, \ldots, E_{2\mu-2})$.
Challenge: $x \leftarrow \mathbb{Z}_q^*$.
Answer: Set $\boldsymbol{x} = (1, x, \ldots, x^{2\mu-2})^T$ and send $b = \boldsymbol{b} \cdot \boldsymbol{x}$ and $s = \boldsymbol{s} \cdot \boldsymbol{x}$ to the verifier. Compute for $\ell = 1, \ldots, m'$

$$\boldsymbol{a}'_\ell = \sum_{j=1}^{\mu} x^{j-1} \boldsymbol{a}_{\mu(\ell-1)+j} \qquad r'_\ell = \sum_{j=1}^{\mu} x^{j-1} r_{\mu(\ell-1)+j} \qquad \rho' = \boldsymbol{\tau} \cdot \boldsymbol{x} \; .$$

Define $\boldsymbol{C}'_1, \ldots, \boldsymbol{C}'_{m'}$ and $c_{A'_1}, \ldots, c_{A'_{m'}}$ and C' by

$$\boldsymbol{C}'_\ell = \prod_{i=1}^{\mu} \boldsymbol{C}_{\mu(\ell-1)+i}^{x^{\mu-i}} \qquad c_{A'_\ell} = \prod_{j=1}^{\mu} c_{A_{\mu(\ell-1)+j}}^{x^{j-1}} \qquad C' = \mathcal{E}_{pk}(G^{-b}; 0) \boldsymbol{E}^{\boldsymbol{x}}.$$

Engage in an SHVZK argument of openings $\boldsymbol{a}'_1, \ldots, \boldsymbol{a}'_{m'}, \boldsymbol{r'}$, and ρ' such that $C' = \mathcal{E}_{pk}(1; \rho') \prod_{\ell=1}^{m'} \boldsymbol{C}'^{\boldsymbol{a}'_\ell}_\ell$.
Verification: Check $\boldsymbol{c}_b \in \mathbb{G}^{2\mu-1}$ and $E_0, \ldots, E_{2\mu-2} \in \mathbb{H}$ and $b, s \in \mathbb{Z}_q$. Accept if

$$c_{b_{\mu-1}} = \mathrm{com}_{ck}(0; 0) \qquad E_{\mu-1} = C \qquad \boldsymbol{c}_b^{\boldsymbol{x}} = \mathrm{com}_{ck}(b; s)$$

and if the SHVZK argument for $\boldsymbol{C}'_1, \ldots, \boldsymbol{C}'_{m'}, C', c_{A'_1}, \ldots, c_{A'_{m'}}$ is valid.

Theorem 7 (Full paper). *The protocol above is a public coin perfect SHVZK argument of knowledge of $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m, \boldsymbol{r}$ such that $C = \mathcal{E}_{pk}(1; \rho) \prod_{i=1}^{m} \boldsymbol{C}_i^{\boldsymbol{a}_i}$*

5 Product Argument

We will sketch an argument that a set of committed values have a particular product. More precisely, given commitments $A_1, \ldots, A_m$ to $a_{11}, \ldots, a_{mn}$ and a value b we want to give an argument of knowledge for $\prod_{i=1}^{m} \prod_{j=1}^{n} a_{ij} = b$. Our strategy is to compute a commitment

$$B = \mathrm{com}_{ck}(\prod_{i=1}^{m} a_{i1}, \ldots, \prod_{i=1}^{m} a_{in}; s) \; .$$

We give an argument of knowledge that B is true, i.e., it contains $\prod_{i=1}^{m} a_{i1}, \ldots, \prod_{i=1}^{m} a_{in}$. Groth [13] described how to do this efficiently. Next, we give an argument of knowledge that b is the product of the values inside B. This can be done using an argument given in [14]. Here, we just give an overview of the protocol.

Common reference string: pk, ck.
Statement: $A_1, \ldots, A_m \in \mathbb{G}$ and $b \in \mathbb{Z}_q$.
Prover's witness: $a_{11}, \ldots, a_{mn}, r_1, \ldots, r_m \in \mathbb{Z}_q$ such that

$$\begin{aligned} A_1 &= \mathrm{com}_{ck}(a_{11}, \ldots, a_{1n}; r_1) \\ &\vdots \qquad\qquad \vdots \\ A_m &= \mathrm{com}_{ck}(a_{m1}, \ldots, a_{mn}; r_m)\,, \end{aligned}$$

$$\text{and} \qquad \prod_{i=1}^{m}\prod_{j=1}^{n} a_{ij} = b\,.$$

Initial message: Pick $s \leftarrow \mathbb{Z}_q$ and compute $B = \mathrm{com}_{ck}(\prod_{i=1}^{m} a_{i1}, \ldots, \prod_{i=1}^{m} a_{in}; s)$. Send B to the verifier. Engage in an SHVZK argument of knowledge as described in [13] of $B = \mathrm{com}_{ck}(\prod_{i=1}^{m} a_{i1}, \ldots, \prod_{i=1}^{m} a_{in}; s)$, where $a_{11}, \ldots, a_{mn}$ are the committed values in $A_1, \ldots, A_m$. Engage (in parallel) in an SHVZK argument of knowledge as described in [14] of b being the product of the values in B.
Verification: The verifier accepts if $B \in \mathbb{G}$ and both SHVZK arguments are valid.

Theorem 8. *The protocol is a public coin perfect SHVZK argument of knowledge of openings $a_{11}, \ldots, a_{mn}, r_1, \ldots, r_m \in \mathbb{Z}_q$ such that $b = \prod_{i=1}^{m}\prod_{i=1}^{n} a_{ij}$.*

The proof along with details of the underlying arguments can be found in the full paper.

6 Implementation and Comparison

We will now compare our protocol with the most efficient shuffle arguments for ElGamal encryption. First, we compare the theoretical performance of the schemes without any optimization. Second, we compare an implementation of our protocol with the implementation by Furukawa et al. [8] and with the implementation in the Verificatum mix-net library [29].

Theoretical Comparison. Previous work in the literature mainly investigated the case where we use ElGamal encryption and commitments over the same group $\mathbb{G}$, i.e., $\mathbb{H} = \mathbb{G} \times \mathbb{G}$. Table 1 gives the asymptotic behavior of these protocols compared to our protocol for $N = mn$ as m and n grows.

In our protocol, we may as detailed in Sect. 4.1 use FFT techniques to reduce the prover's computation to $O(N \log m)$ exponentiations as listed in Table 1. Furthermore, by increasing the round complexity as in Sect. 4.2 we could even get a linear complexity of $O(N)$ exponentiations. These techniques do not apply to the other shuffle arguments; in particular it is not possible to use FFT techniques to reduce the factor m in the shuffle by Groth and Ishai [15].

Table 1. Comparison of the protocols with ElGamal encryption

SHVZK argument	Rounds	Time $\mathcal{P}$ Expos	Time $\mathcal{V}$ Expos	Size Elements
[10]	3	$8N$	$10N$	$5N\ \mathbb{G}$+ $N\ \mathbb{Z}_q$
[8]	5	$9N$	$10N$	$5N\ \mathbb{G}$ + $N\ \mathbb{Z}_q$
[14]	7	$6N$	$6N$	$3N\ \mathbb{Z}_q$
[7]	3	$7N$	$8N$	$N\ \mathbb{G}$+ $2N\ \mathbb{Z}_q$
[25]	5	$9N$	$11N$	$3N\ \mathbb{G} + 4N\ \mathbb{Z}_q$
[15]	7	$3mN$	$4N$	$3m^2\ \mathbb{G}$+ $3n\ \mathbb{Z}_q$
This paper	9	$2\log(m)N$	$4N$	$11m\ \mathbb{G}$+ $5n\ \mathbb{Z}_q$

As the multi-exponentiation argument, which is the most expensive step, already starts in round 3 we can insert two rounds of interactive reduction as described in Sect. 4.2 without increasing the round complexity above 9 rounds. For practical parameters this would give us enough of a reduction to make the prover's computation comparable to the schemes with linear $O(N)$ computation.

The figures in Table 1 are for non-optimized versions of the schemes. All of the schemes may for instance benefit from the use of multi-exponentiation techniques, see e.g. Lim [18] for how to compute a product of n exponentiations using only $O(\frac{n}{\log n})$ multiplications. The schemes may also benefit from randomization techniques, where the verifier does a batch verification of all the equations it has to check.

Experimental Results. We implemented our shuffle argument in C++ using the NTL library by Shoup [24] for the underlying modular arithmetic. We experimented with five different implementations to compare their relative merit:

1. Without any optimizations at all.
2. Using multi-exponentiation techniques.
3. Using multi-exponentiation and the Fast Fourier transform.
4. Using multi-exponentiation and a round of the interactive technique with $\mu = 4$ and Toom-Cook for $m' = 4$ giving $m = \mu m' = 16$.
5. Using multi-exponentiation and two rounds of the interactive technique first with $\mu = 4$ and Toom-Cook for $m' = 4$ giving $m = \mu^2 m' = 64$.

In our experiments we used ElGamal encryption and commitments over the same group $\mathbb{G}$, which was chosen as an order q subgroup of $\mathbb{Z}_p^*$, where $|q| = 160$ and $|p| = 1024$. These security parameters are on the low end for present day use but facilitate comparison with earlier work. The results can be found in Table 2 for $N = 100,000$, $m = 8, 16, 64$ on our machine. We see that the plain multi-exponentiation techniques yield better results than the FFT method for small m; the better asymptotic behavior of the FFT only kicks in for $m > 16$. As expected the Toom-Cook inspired version with added interaction has the best running time and communication cost.

Table 2. Run time of the shuffle arguments in seconds on a Core2Duo 2.53 GHz, 3 MB L2-Cache, 4 GB Ram machine for $N = 100,000$ and $m = 8, 16, 64$

	Optimization	Total time	Time $\mathcal{P}$	Time $\mathcal{V}$	Size
$m = 8$	Unoptimized	570	462	108	4.3 MB
	Multi-expo	162	125	37	
	FFT	228	190	38	
$m = 16$	Unoptimized	900	803	97	2.2 MB
	Multi-expo	193	169	24	
	FFT	245	221	24	
	Toom-Cook	139	101	38	
$m = 64$	Multi-expo	615	594	21	0.7MB
	FFT	328	307	20	
	Toom-Cook	128	91	18	

Comparison with Other Implementations. Furukawa, Miyauchi, Mori, Obana, and Sako [8] gave performance results for a mix-net using a version of the Furukawa-Sako [10] shuffle arguments. They optimized the mix-net by combining the shuffling and decryption operations into one. They used three shuffle centers communicating with each other and their results included both the process of shuffling and the cost of the arguments. So, to compare the values we multiply our shuffle argument times with 3 and add the cost of our shuffling operation on top of that. The comparison can be found in Table 3.

Table 3. Runtime comparison of [8] (CPU: 1 GHz, 256 MB) to our shuffle argument (Toom-Cook with $m = 64$, CPU: 1.4 GHz, 256 MB)

$N = 100,000$	[8]	This paper
Single argument	51 min	15 min
Argument size	66 MB	0.7 MB
Total mix-net time	3 hrs 44 min	53 min

We expected to get better performance than they did and indeed we see that our argument is faster and the communication is a factor 100 smaller. When adding the cost of shuffling and decryption to our argument we still have a speedup of a factor 3 in Table 3 when comparing the two mix-net implementations and taking the difference in the machines into account.

Recently, Furukawa et al. [9] announced a new implementation based on elliptic curve groups. Due to the speed of using elliptic curves this gave them a speedup of a factor 3. A similar speedup can be expected for our shuffle argument if we switch to using elliptic curves in our implementation.

Recently Wikström made a complete implementation of a mix-net in Java in [29] called Verificatum, which is based on the shuffle argument in [25]. To produce comparable data, we ran the demo file with only one mix party in the non-interactive mode

Table 4. Runtime comparison of [25] to our shuffle argument on our machine (CPU: 2.53 GHz, 4 GB)

$N = 100,000$	[25]	This paper Toom-Cook
Single argument	5 min	2 min
Argument size	37.7 MB	0.7 MB

using the same modular group as in our protocol. Verificatum is a full mix-net implementation; for fairness in the comparison we only counted the time of the relevant parts for the shuffle argument. As described in Table 1 the theoretical performance of Verificatum's shuffle argument is $20N$ exponentiations, while our prover with Toom-Cook and 2 extra rounds of interaction uses $12N$ exponentiations and our verifier $4N$, so in total $16N$ exponentiations. So we expect a similar running time for the Verificatum mix-net. As shown in Table 4 we perform better, but due to the different programming languages used and different levels of optimization in the code we will not draw any conclusion except that both protocols are efficient and usable in current applications. In terms of size it is clear that our arguments leave a much smaller footprint than Verificatum; we save a factor 50 in the communication.

Acknowledgment. We would like to thank Douglas Wikström for discussions and help regarding our comparison with the shuffle argument used in Verificatum [29].

References

1. Abe, M.: Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 437–447. Springer, Heidelberg (1998)
2. Abe, M.: Mix-Networks on Permutation Networks. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (1999)
3. Abe, M., Hoshino, F.: Remarks on Mix-Network Based on Permutation Networks. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 317–324. Springer, Heidelberg (2001)
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)
5. Cook, S.: On the minimum computation time of functions. PhD thesis, Department of Mathematics, Harvard University (1966), `http://cr.yp.to/bib/1966/cook.html`
6. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Math. Comp. 19, 297–301 (1965)
7. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Transactions 88-A(1), 172–188 (2005)
8. Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An Implementation of a Universally Verifiable Electronic Voting Scheme Based on Shuffling. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 16–30. Springer, Heidelberg (2003)
9. Furukawa, J., Mori, K., Sako, K.: An Implementation of a Mix-Net Based Network Voting Scheme and Its Use in a Private Organization. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 141–154. Springer, Heidelberg (2010)
10. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)

11. Garay, J., MacKenzie, P., Yang, K.: Strengthening zero-knowledge protocols using signatures. J. Cryptology 19(2), 169–209 (2006)
12. Groth, J.: Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, 2004. PhD thesis. xii+119 (2004)
13. 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)
14. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. J. Cryptology 23(4), 546–579 (2010)
15. Groth, J., Ishai, Y.: Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008)
16. Groth, J., Lu, S.: Verifiable Shuffle of Large Size Ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Heidelberg (2007)
17. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics Dokl. 7, 595–596 (1963)
18. Lim, C.: Efficient multi-exponentiation and application to batch verification of digital signatures (2000), http://dasan.sejong.ac.kr/~chlim/pub/multiexp.ps
19. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology 16(3), 143–184 (2003)
20. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM CCS, pp. 116–125 (2001)
21. Neff, C.A.: Verifiable mixing (shuffling) of elgamal pairs (2003), http://people.csail.mit.edu/rivest/voting/papers/Neff-2004-04-21-ElGamalShuffles.pdf
22. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
23. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
24. Shoup, V.: Ntl library (2009), http://www.shoup.net/ntl/
25. Terelius, B., Wikström, D.: Proofs of Restricted Shuffles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 100–113. Springer, Heidelberg (2010)
26. Toom, A.: The complexity of a scheme of functional elements realizing the multiplication of integers (2000), http://www.de.ufpe.br/~toom/my_articles/engmat/MULT-E.PDF
27. Wikström, D.: The Security of a Mix-Center Based on a Semantically Secure Cryptosystem. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 368–381. Springer, Heidelberg (2002)
28. Wikström, D.: A Commitment-Consistent Proof of a Shuffle. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 407–421. Springer, Heidelberg (2009)
29. Wikström, D.: Verificatum (2010), http://www.verificatum.com/