Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle

Jens Groth[1]* and Yuval Ishai[2]**

[1] University College London
j.groth@ucl.ac.uk
[2] Technion and University of California Los Angeles
yuvali@cs.technion.ac.il

Abstract. A shuffle of a set of ciphertexts is a new set of ciphertexts with the same plaintexts in permuted order. Shuffles of homomorphic encryptions are a key component in mix-nets, which in turn are used in protocols for anonymization and voting. Since the plaintexts are encrypted it is not directly verifiable whether a shuffle is correct, and it is often necessary to prove the correctness of a shuffle using a zero-knowledge proof or argument.
In previous zero-knowledge shuffle arguments from the literature the communication complexity grows linearly with the number of ciphertexts in the shuffle. We suggest the first practical shuffle argument with sub-linear communication complexity. Our result stems from combining previous work on shuffle arguments with ideas taken from probabilistically checkable proofs.

Keywords: Shuffle, zero-knowledge argument, sub-linear communication, homomorphic encryption, mix-net.

1 Introduction

A shuffle of ciphertexts $e_1, \ldots, e_N$ is a new set of ciphertexts $E_1, \ldots, E_N$ with the same plaintexts in permuted order. Shuffles are used in many protocols for anonymous communication and voting. It is usually important to verify the correctness of the shuffle. Take for instance a voting protocol where the ciphertexts are encrypted votes; it is important to avoid that some of the ciphertexts in the shuffle are substituted with encryptions of other votes. There has therefore been much research on designing zero-knowledge arguments[3] for the correctness of a shuffle [37, 1, 2, 17, 30, 31, 21, 16, 33, 34, 32, 15, 24, 38].

When designing shuffle arguments, efficiency is a major concern. It is realistic to have elections with millions of encrypted votes, in which case the statement to be proven is very large. In this paper, our main goal is to get a *practical* shuffle argument with low

* Research done in part while visiting IPAM. Part of work done while at UCLA supported by NSF ITR/Cybertrust grant 0456717.
** Research done in part while visiting IPAM. Supported by BSF grant 2004361, ISF grant 1310/06, and NSF grants 0205594, 0430254, 0456717, 0627781, 0716835, 0716389.
[3] By zero-knowledge *arguments* [8] we refer to computationally-sound zero-knowledge proofs [20].

communication complexity. A theoretical solution to this problem would be to use Kilian's communication-efficient zero-knowledge argument [26] (see also Micali [29]). This method, however, requires a reduction to Circuit Satisfiability, a subsequent application of the PCP-theorem [4, 3, 12], and using a collision-free hash-function to build a hash-tree that includes the entire PCP. Even with the best PCP constructions known to date (cf. [7]), such an approach would be inefficient in practice.

OUR CONTRIBUTION. We present a sublinear-communication 7-move public coin perfect zero-knowledge argument of knowledge for the correctness of a shuffle of ElGamal ciphertexts [13]. (The protocol is presented in the common random string model, but can also be implemented in the plain model at the cost of a slightly higher constant number of rounds.) All shuffle arguments previously suggested in the literature have communication complexity $\Omega(N)\kappa$, where N is the number of ciphertexts in the shuffle and κ is a security parameter specifying the finite group over which the scheme works. Our shuffle argument has communication complexity $O(m^2 + n)\kappa$ for m and n such that $N = mn$. (The constant in the expression is low as well, see Section 8 for a more precise efficiency analysis.) With $m = N^{1/3}$ this would give a size of $O(N^{2/3})\kappa$ bits, but in practice a smaller choice of m will usually be better for computational reasons. Our shuffle argument moderately increases the prover's computational burden and reduces the amount of communication and the verifier's computational burden in comparison with previous work.

For practical purposes it will be natural to use the Fiat-Shamir heuristic [14] (i.e. compute the verifier's public-coin challenges using a cryptographic hash-function) to make our shuffle argument non-interactive. The Fiat-Shamir heuristic justifies reducing the communication and verifier computation at the cost of increased prover computation, since the non-interactive shuffle argument needs to be computed only once by the prover but may be distributed to and checked by many verifiers. Letting the prover do some extra work in order to reduce the communication and the computational burden of each verifier is therefore a good trade-off in practice. To the best of our knowledge, our protocol is the first practical instance of a sublinear-communication argument for any interesting nontrivial statement.

We have some further remarks on our result. Our technique also applies to other homomorphic cryptosystems, for instance Paillier encryption [35]; a more general treatment of a wider class of homomorphic encryptions can be obtained along the lines of [21]. For simplicity we focus just on ElGamal encryption in this paper. Similarly to previous shuffle arguments from the literature, we will present our protocol as an *honest verifier* zero-knowledge argument. There are very efficient standard techniques for converting honest verifier zero-knowledge arguments into fully zero-knowledge arguments [10, 18, 22].

TECHNIQUES. Our starting point is the honest verifier zero-knowledge shuffle argument by Groth [21], which builds on ideas by Neff [30]. Borrowing some of the ideas underlying the PCP theorem, namely the use of Hadamard codes and batch-verification techniques, we reduce the size of the shuffle argument. We note that unlike Kilian [26] we do not reduce the shuffle statement to an NP-complete language such as SAT; instead we work directly with the ciphertexts in the shuffle statement. Moreover, while we use ideas behind the PCP theorem we do not make use of a full-blown PCP. In particular, our argument avoids any use of linearity testing, low-degree testing, or other forms of code proximity testing that appear in all known PCPs.

RELATED WORK. Our work was inspired by the recent work of Ishai, Kushilevitz, and Ostrovsky [25], which introduced an approach for constructing sublinear-communication arguments using exponentially long but succinctly described PCPs. Similarly to [25] we use short *homomorphic* commitments as the main cryptographic building block. There are, however, several important differences between our techniques and those from [25]. In particular, the arguments obtained in [25] do not address our *zero-knowledge* requirement (and are only concerned with soundness), they inherently require the verifier to use *private coins* (which are undesirable in the context of our application), and they employ *linearity testing* that subsequently requires soundness amplification. Finally, the approach of [25] is generic and does not account for the special structure of the shuffle problem; this structure is crucial for avoiding an expensive reduction to SAT.

2 Preliminaries

2.1 Notation

We let Σ_N denote the symmetric group on $\{1, 2, \ldots, N\}$. 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 the function f is *negligible* when $f(\kappa) \approx 0$ and that it is *overwhelming* when $f(\kappa) \approx 1$.

Algorithms in our shuffle argument will get a security parameter κ as input, which specifies the size of the group we are working over. Sometimes we for notational simplicity avoid writing this explicitly, assuming κ can be deduced indirectly from other inputs given to the algorithms.

All our algorithms will be probabilistic polynomial time algorithms. We will assume that they can sample randomness from sets of the type $\mathbb{Z}_q$. We note that such randomness can be sampled from a source of uniform random bits in expected polynomial time (in $\log q$).

We write $A(x; r) = y$ when 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.

When defining security, we assume that there is an adversary attacking our scheme. This adversary is modeled as a non-uniform polynomial time stateful algorithm. By stateful, we mean that we do not need to give it the same input twice, it remembers from the last invocation what its state was. This makes the notation a little simpler, since we do not need to explicitly write out the transfer of state from one invocation to the next.

2.2 Group Generation

We will work over a group G_q of a prime order q. This could for instance be a subgroup of $\mathbb{Z}_p^*$, where p is a prime and $\gcd(q^2, p-1) = q$; or it could be an elliptic curve group or subgroup. We will assume the discrete logarithm problem is hard in G_q. More precisely, let $\mathcal{G}$ be a generating algorithm that takes a security parameter κ as input and outputs $gk := (q, G_q, g)$, where by G_q we denote a computationally efficient representation of the group and g is a random generator for G_q. The discrete logarithm assumption says that for any non-uniform polynomial time adversary $\mathcal{A}$:

$$\Pr\left[(q, G_q, g) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathbb{Z}_q; h := g^x : \mathcal{A}(q, G_q, g, h) = x\right] \approx 0.$$

(When the randomness of $\mathcal{G}$ is taken from a common random string, the above definition needs to be strengthened so that $\mathcal{A}$ is given the randomness used by $\mathcal{G}$.)

2.3 Generalized Pedersen Commitment

We will use a variant of the Pedersen commitment scheme [36] that permits making a commitment to a length-n vector in $\mathbb{Z}_q^n$ rather than a single element of $\mathbb{Z}_q$ as in Pedersen's original commitment. A crucial feature of this generalization is that the amount of communication it involves does not grow with n. The generalized scheme proceeds as follows. The key generation algorithm $K_{\rm com}$ takes (q, G_q, g) as input and outputs a commitment key $ck := (g_1, \ldots, g_n, h)$, where $g_1, \ldots, g_n, h$ are randomly chosen generators of G_q. The message space is $\mathcal{M}_{ck} := \mathbb{Z}_q^n$, the randomizer space is $\mathcal{R}_{ck} := \mathbb{Z}_q$ and the commitment space is $\mathcal{C}_{ck} := G_q$. (The parameter n will be given as an additional input to all algorithms; however, we prefer to keep it implicit in the notation.)

To commit to an n-tuple $(m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ we pick randomness $r \leftarrow \mathbb{Z}_q$ and compute the commitment $C := h^r \prod_{i=1}^n g_i^{m_i}$. The commitment is perfectly hiding since no matter what the messages are, the commitment is uniformly distributed in G_q. The commitment is computationally binding under the discrete logarithm assumption; we will skip the simple proof.

The commitment key ck will be part of the common random string in our shuffle argument. We remark that it can be sampled from a random string. We write $C := \mathrm{com}_{ck}(m_1, \ldots, m_n; r)$ for making a commitment to $m_1, \ldots, m_n$ using randomness r. The commitment scheme is homomorphic, i.e., for all $m_1, m_1', \ldots, m_n, m_n', r, r' \in Z_q$ we have

$$\mathrm{com}_{ck}(m_1, \ldots, m_n; r) \cdot \mathrm{com}_{ck}(m_1', \ldots, m_n'; r') = \mathrm{com}_{ck}(m_1 + m_1', \ldots, m_m + m_n'; r + r').$$

In some cases we will commit to less than n elements; this can be accomplished quite easily by setting the remaining messages to 0.

We will always assume that parties check that commitments are valid, meaning they check that $C \in G_q$. If G_q is a subgroup of $\mathbb{Z}_p^*$ this can be done by checking that $C^q = 1$, however, batch verification techniques can be used to lower this cost when we have multiple commitments to check.[4] If G_q is an elliptic curve of order q, then the validity check just consists of checking that C is a point on the curve, which is very inexpensive.

[4] See also [21] for a variant of the Pedersen commitment scheme over $\mathbb{Z}_p^*$ that makes it possible to completely eliminate the cost of verifying validity.

2.4 ElGamal Encryption

ElGamal encryption [13] in the group G_q works as follows. The public key is $pk := y = g^x$ with a random secret key $sk := x \leftarrow \mathbb{Z}_q^*$. The message space is $\mathcal{M}_{pk} := G_q$, the randomizer space is $\mathcal{R}_{pk} := \mathbb{Z}_q$ and the ciphertext space is $\mathcal{C}_{pk} := G_q \times G_q$. To encrypt a message $m \in G_q$ using randomness $R \in \mathbb{Z}_q$ we compute the ciphertext $E_{pk}(m;R) := (g^R, y^R m)$. To decrypt a ciphertext (u,v) we compute $m = vu^{-x}$.

The semantic security of ElGamal encryption is equivalent to the DDH assumption. Semantic security may be needed for the shuffle itself to be secure; however, the security of our shuffle argument will rely on the discrete logarithm assumption only. In particular, our shuffle argument is still sound and zero-knowledge even if the cryptosystem is insecure or the decryption key has been exposed.

ElGamal encryption is homomorphic with entry-wise multiplication in the ciphertext space. For all $(m,R),(m',R') \in \mathcal{M}_{pk} \times \mathcal{R}_{pk}$ we have

$$\begin{aligned} E_{pk}(mm';R+R') &= (g^{R+R'}, y^{R+R'}mm') \\ &= (g^R, y^R m) \cdot (g^{R'}, y^{R'}m') = E_{pk}(m;R) \cdot E_{pk}(m';R'). \end{aligned}$$

We will always assume that the ciphertexts in the shuffle are valid, i.e., $(u,v) \in G_q \times G_q$. Batch verification techniques can reduce the cost of verifying validity when we have multiple ciphertexts. To further reduce the cost of ciphertext verification, Groth [21] suggests a variant of ElGamal encryption that makes batch-checking ciphertext validity faster. Our shuffle argument works also for this variant of ElGamal encryption.

Our shuffle argument works with many types of cryptosystems; the choice of ElGamal encryption is made mostly for notational convenience. Our technique can be directly applied with any homomorphic cryptosystem that has a message space of order q. We are neither restricted to using the same underlying group (q, G_q, g) as the commitment scheme nor restricted to using ElGamal encryption or variants thereof. Using techniques from [21] it is also possible to generalize the shuffle argument to work for cryptosystems that do not have message spaces of order q. This latter application does require a few changes to the shuffle argument though and does increase the complexity of the shuffle argument, but the resulting protocol still has the same sub-linear asymptotic complexity.

2.5 Special Honest Verifier Zero-Knowledge Arguments of Knowledge

We will assume there is a setup algorithm $\mathcal{G}$ that generates some setup information gk. This setup information could for instance be a description of a group that we will be working in. Consider a pair of probabilistic polynomial time interactive algorithms (P,V) called the prover and the verifier. They may have access to a common random string σ generated by a probabilistic polynomial time key generation algorithm K. We consider a polynomial time decidable ternary relation R. For an element x we call w a witness if $(gk,x,w) \in R$. We define a corresponding group-dependent language L_{gk} consisting of elements x that have a witness w such that $(gk,x,w) \in R$. 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 together with the randomness used by V. This transcript ends with V either accepting or rejecting. We sometimes shorten the notation by saying $\langle P(x), V(y)\rangle = b$ if V ends by accepting, $b = 1$, or rejecting, $b = 0$.

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

Completeness:

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

Computational soundness:

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

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 and the setup parameters gk, σ.*

We define special honest verifier zero-knowledge (SHVZK) [9] 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 called a special honest verifier zero-knowledge argument for R with setup $\mathcal{G}$ if there exists a probabilistic polynomial time simulator S such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

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

We remark that there are efficient techniques to convert SHVZK arguments into zero-knowledge arguments for arbitrary verifiers in the common random string model [10, 18, 22]. In this paper, we will therefore for simplicity focus just on the special honest verifier zero-knowledge case.

WITNESS-EXTENDED EMULATION. We shall define an argument of knowledge[5] through witness-extended emulation, the name taken from Lindell [28]. Whereas Lindell's definition pertains to proofs of knowledge in the plain model, we will adapt his

[5] The standard definition of *proofs* of knowledge by Bellare and Goldreich [5] does not apply in our setting, since we work in the common random string model and are interested in *arguments* of knowledge. See Damgård and Fujisaki [11] for a discussion of this issue.

definition to the setting of public coin arguments in the common random string model. Informally, our definition says: given an adversary that produces an acceptable argument with probability ϵ, there exists an emulator that produces a similar argument with probability ϵ, 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*

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

where E has access to a transcript oracle $\langle P^*(gk, \sigma, x, s), V(gk, \sigma, x)\rangle$ 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 gk, σ, x, s gives us the view of both the prover and the verifier and at the same time gives us the witness.

Damgård and Fujisaki [11] have suggested an alternative definition of an argument of knowledge in the presence of a common random string. Witness-extended emulation as defined above implies knowledge soundness as defined by them [22].

THE FIAT-SHAMIR HEURISTIC. The Fiat-Shamir heuristic [14] can be used to make public coin SHVZK arguments non-interactive. In the Fiat-Shamir heuristic the verifier's challenges are computed by applying a cryptographic hash-function to the transcript of the protocol. Security can be formally argued in the random oracle model [6], in which the hash-function is modeled as a completely random function that returns a random string on each input it has not been queried before. While the Fiat-Shamir heuristic is not sound in general [19], it is still commonly believed to be a safe practice when applied to "natural" protocols.

2.6 Problem Specification and Setup

We will construct a 7-move public coin perfect SHVZK argument for the relation

$$\begin{aligned} R = \Big\{&(gk = (q, G_q, g), (pk = y, e_1, \ldots, e_N, E_1, \ldots, E_N), (\pi, R_1, \ldots, R_N)) \;\Big|\\ &y \in G_q \wedge \pi \in \Sigma_N \wedge R_1, \ldots, R_N \in \mathcal{R}_{pk} \wedge \forall i : E_i = e_{\pi^{-1}(i)} E_{pk}(1; R_i)\Big\}. \end{aligned}$$

In our SHVZK argument, the common random string σ will be generated as a public key $(g_1, \ldots, g_n, h)$ for the n-element Pedersen commitment scheme described in Section 2.3. Depending on the applications, there are many possible choices for who generates the commitment key and how this generation is done. For use in a mix-net, we could for instance imagine that there is a setup phase, where the mix-servers run a multi-party computation protocol to generate the setup and the commitment key. Another option is to let the verifier generate the common random string, since it is easy to verify whether a commitment key is valid or not. This option yields an 8-move (honest-verifier zero-knowledge) argument in the plain model.[6]

2.7 Polynomial Identity Testing

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

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

The Schwartz-Zippel lemma is frequently used in polynomial identity testing. Given two multi-variate polynomials p_1 and p_2 we can test whether $p_1(x_1, \ldots, x_\nu) - p_2(x_1, \ldots, x_\nu) = 0$ for random $x_1, \ldots, x_\nu \leftarrow \mathbb{Z}_q$. If the two polynomials are identical this will always be true, whereas if the two polynomials are different then there is only probability $\max(d_1, d_2)/q$ for the equality to hold.

3 Product of Committed Elements

Consider a sequence of commitments $A_1, \ldots, A_m$ and a value $a \in \mathbb{Z}_q$. We will give an SHVZK argument of knowledge of $\{a_{ij}\}_{i=1,j=1}^{m,n}$ and $\{r_i\}_{i=1}^{m}$ such that

$$\begin{aligned} A_1 &= \mathrm{com}_{ck}(a_{11}, a_{12}, \ldots, a_{1n}; r_1) \\ &\vdots \\ A_m &= \mathrm{com}_{ck}(a_{m1}, a_{m2}, \ldots, a_{mn}; r_m) \end{aligned} \qquad \text{and} \qquad a = \prod_{i=1}^{m}\prod_{j=1}^{n} a_{ij} \mod q.$$

The argument is of sub-linear size; the prover will send m^2 commitments and $2n$ elements from $\mathbb{Z}_q$, where $N = mn$ is the total number of committed elements a_{ij}. For $m = N^{1/3}$ this gives a size of $O(N^{2/3})\kappa$ bits.

[6] We can also get full zero-knowledge in the plain model. The verifier picks the common random string as above and also picks an additional key for a trapdoor commitment scheme. The verifier then makes engages in a zero-knowledge proof of knowledge of the trapdoor. We can now use the standard techniques for converting honest verifier zero-knowledge arguments to full zero-knowledge arguments [10, 18, 22]. By running the two proofs in parallel, the round complexity is only 8. Note, however, that since the verifier must know the secret trapdoor of the additional commitment scheme, the protocol is no longer public coin.

The argument is quite complex so let us first describe some of the ideas that go into it. In our argument, the prover will prove knowledge of the contents of the commitments. For the sake of simplicity we will first describe the argument assuming the prover knows the contents of the commitments and by the computational binding property of the commitment scheme is bound to these values. We will also for the sake of simplicity just focus on soundness and later when giving the full protocol add extra parts that will give us honest verifier zero-knowledge and witness-extended emulation. (Note that even completeness and soundness alone are nontrivial to achieve when considering *sublinear communication* arguments.)

Consider first commitments $A_1, \ldots, A_m$ as described above. The verifier will pick a random challenge $s_1, \ldots, s_m$. By the homomorphic property

$$\prod_{i=1}^{m} A_i^{s_i} = \mathrm{com}_{ck}(\sum_{i=1}^{m} s_i a_{i1}, \ldots, \sum_{i=1}^{m} s_i a_{in}; \sum_{i=1}^{m} s_i r_i).$$

In our argument the prover will open this commitment multi-exponentiation as $f_1 := \sum_{i=1}^{m} s_i a_{i1}, \ldots, f_n := \sum_{i=1}^{m} s_i a_{in}, z := \sum_{i=1}^{m} s_i r_i$.

Consider now the case where we have three sets of commitments $\{A_i\}_{i=1}^{m}, \{B_\ell\}_{\ell=1}^{m}, \{C_{i\ell}\}_{i=1,\ell=1}^{m,m}$ containing respectively $m \times n$ matrices A, B and $m^2 \times n$ matrix C. The verifier will choose random challenges $s_1, \ldots, s_m, t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$. The prover can open the commitment products $\prod_{i=1}^{m} A_i^{s_i}$, $\prod_{\ell=1}^{m} B_\ell^{t_\ell}$, $\prod_{i=1}^{m} \prod_{\ell=1}^{m} C_{i\ell}^{s_i t_\ell}$ as described above. This gives us for each of the n columns

$$f_j := \sum_{i=1}^{m} s_i a_{ij} \quad , \quad F_j := \sum_{\ell=1}^{m} t_\ell b_{\ell j} \quad , \quad \phi_j := \sum_{i=1}^{m} \sum_{\ell=1}^{m} s_i t_\ell c_{i\ell j}.$$

In our proofs the verifier will check for each column that $\phi_j = f_j F_j$. These checks can be seen as quadratic equations in variables $s_1, \ldots, s_m, t_1, \ldots, t_m$ of the form

$$(\sum_{i=1}^{m} s_i a_{ij})(\sum_{\ell=1}^{m} t_\ell b_{\ell j}) = \sum_{i=1}^{m} \sum_{\ell=1}^{m} s_i t_\ell c_{i\ell j}.$$

If $c_{i\ell j} = a_{ij} b_{\ell j}$ for all i, ℓ, j the check will always pass, whereas if this is not the case, then by the Schwartz-Zippel lemma there is overwhelming probability over the choice of $s_1, \ldots, s_m, t_1, \ldots, t_m$ that the check will fail. (This type of checking is also used in the Hadamard-based PCP of Arora et al. [3].) We therefore have an argument for C_{ii} being a commitment to $\{a_{ij} b_{ij}\}_{j=1}^{n}$. The commitments $C_{i\ell}$ for $i \neq \ell$ are just fillers that make the argument work, we will not need them for anything else. In the argument we only reveal $O(n)$ elements in $\mathbb{Z}_q$ to simultaneously prove $N = mn$ equalities $c_{iij} = a_{ij} b_{ij}$; this is what will give us sub-linear communication complexity.

Let us now explain how we choose the matrix B. For $1 \leq I \leq m, 1 \leq J \leq n$ we set $b_{IJ} := \prod_{i=1}^{I-1} \prod_{j=1}^{n} a_{ij} \cdot \prod_{j=1}^{J} a_{Ij}$. This means that B is a matrix chosen such that b_{ij} is the previous element in the matrix B multiplied with a_{ij}. In particular, we have $b_{mn} = \prod_{i=1}^{m} \prod_{j=1}^{n} a_{ij} = a$. In addition, we will have an extra column with $b_{10} := 1$ and for $1 < i \leq m : b_{i0} := b_{i-1,n}$. In other words, the 0th column vector is the nth column vector of B shifted one step down. The prover will make a separate set of m commitments $B'_1, \ldots, B'_m$ to this column. Choosing $B'_1 := \mathrm{com}_{ck}(1; 0)$ it is straightforward to verify that $b_{10} = 1$. To show that the rest of the 0th column is correctly constructed the prover will open $\prod_{\ell=2}^{m-1}(B'_\ell)^{t_{\ell-1}}$ to the message $F_n - t_m a$. The linear equations give us $\sum_{\ell=2}^{m-1} t_{\ell-1} b_{\ell 0} + t_m a = \sum_{\ell=1}^{m} t_\ell b_{\ell n}$, which by the Schwartz-Zippel lemma has negligible probability of being true unless $b_{mn} = a$ and $b_{\ell+1,0} = b_{\ell n}$ for $1 \leq \ell < m$.

We have now described B extended with a 0th column vector. Write $\widetilde{B}$ for the matrix with the 0th column and the first $n-1$ columns of B. We will apply the A, B, C matrix argument we described before to the matrices $A, \widetilde{B}, C$, where we use commitments $C_{ii} := B_i$. This argument demonstrates for each $1 \leq j \leq n$ that $b_{ij} = a_{ij} b_{i,j-1}$. Putting everything together we now have: $b_{10} = 1, b_{ij} = a_{ij} b_{i,j-1}, b_{i0} = b_{i-1,n}$ and $b_{mn} = a$, which is sufficient to conclude that $a = \prod_{i=1}^{m} \prod_{j=1}^{n} a_{ij}$.

We will now describe the full protocol. The most significant change from the description given above is that we now add also elements a_{0j}, b_{0j} that are chosen at random to the matrices. The role of these elements is to give honest verifier zero-knowledge. The prover reveals elements of the form $f_j := a_{0j} + \sum_{i=1}^{m} s_i a_{ij}$ and $F_j := b_{0j} + \sum_{\ell=1}^{m} t_\ell b_{\ell j}$, which reveal nothing about $\sum_{i=1}^{m} s_i a_{ij}$ and $\sum_{\ell=1}^{m} t_\ell b_{\ell j}$ when a_{0j} and b_{0j} are random.

Initial message:

$a_{01}, \ldots, a_{0n} \leftarrow \mathbb{Z}_q \ ; \ r_0 \leftarrow \mathcal{R}_{ck} \ ; \ A_0 := \mathrm{com}_{ck}(a_{01}, a_{02}, \ldots, a_{0n}; r_0)$
For $1 \leq I \leq m, 1 \leq J \leq n \ : \ b_{IJ} := \prod_{i=1}^{I-1} \prod_{j=1}^{n} a_{ij} \ \cdot \ \prod_{j=1}^{J} a_{Ij}$
$b_{01}, \ldots, b_{0n} \leftarrow \mathbb{Z}_q \ ; \ r_{b0}, r_{b1} \ldots, r_{bm} \leftarrow \mathcal{R}_{ck}$

$$
\begin{aligned}
B_0 &:= \mathrm{com}_{ck}(b_{01}, b_{02}, \ldots, b_{0n}; r_{b0}) \\
B_1 &:= \mathrm{com}_{ck}(b_{11}, b_{12}, \ldots, b_{1n}; r_{b1}) \\
&\ \vdots \\
B_m &:= \mathrm{com}_{ck}(b_{m1}, b_{m2}, \ldots, b_{mn}; r_{bm})
\end{aligned}
$$

Define $b_{10} := 1, b_{20} := b_{1n}, \ldots, b_{m0} := b_{m-1,n}$
$r'_2, \ldots, r'_m \leftarrow \mathcal{R}_{ck} \ ; \ B'_2 := \mathrm{com}_{ck}(b_{20}; r'_2), \ldots, B'_m := \mathrm{com}_{ck}(b_{m0}; r'_m)$
$b_{00} \leftarrow \mathbb{Z}_q \ ; \ r'_0 \leftarrow \mathcal{R}_{ck} \ ; \ B'_0 := \mathrm{com}_{ck}(b_{00}; r'_0)$
$\hat{r} \leftarrow \mathcal{R}_{ck} \ ; \ \widehat{B} := \mathrm{com}_{ck}(b_{0n}; \hat{r})$
For $0 \leq i, \ell \leq m \ : \ r_{i\ell} \leftarrow \mathcal{R}_{ck}$ and for $1 \leq i \leq m \ : \ r_{ii} := r_{bi}$.
For $0 \leq i, \ell \leq m$:

$$C_{i\ell} := \mathrm{com}_{ck}(a_{i1} b_{\ell 0}, \ldots, a_{in} b_{\ell, n-1}; r_{i\ell})$$

Since $b_{ij} = a_{ij} b_{i,j-1}$ and $r_{ii} = r_{bi}$ we have for $1 \leq i \leq m$ that $C_{ii} = B_i$.
Send $(A_0, B_0, B'_0, B'_2, \ldots, B'_m, \widehat{B}, C_{00}, \ldots, C_{mm})$ to the verifier

Challenge: $s_1, \ldots, s_m, t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$

Answer:

For $1 \leq j \leq n \ : \ f_j := a_{0j} + \sum_{i=1}^{m} s_i a_{ij} \ ; \ F_j := b_{0j} + \sum_{\ell=1}^{m} t_\ell b_{\ell j} \ ; \ F_0 := b_{00} + \sum_{\ell=1}^{m} t_\ell b_{\ell 0}$

$z := r_0 + \sum_{i=1}^m s_i r_i$; $z_b := r_{b0} + \sum_{\ell=1}^m t_\ell r_{b\ell}$; $z' := r'_0 + \sum_{\ell=2}^m t_\ell r'_\ell$; $\hat{z} := \hat{r} + \sum_{\ell=2}^m t_{\ell-1} r'_\ell$
$z_{ab} := r_{00} + \sum_{i=1}^m s_i r_{i0} + \sum_{\ell=1}^m t_\ell r_{0\ell} + \sum_{i=1}^m \sum_{\ell=1}^m s_i t_\ell r_{i\ell}$
Send $(f_1, \ldots, f_n, F_0, \ldots, F_n, z, z_b, z', \hat{z}, z_{ab})$ to the verifier

Verification:
Check $A_0 \prod_{i=1}^m A_i^{s_i} = \mathrm{com}_{ck}(f_1, \ldots, f_n; z)$
For $1 \leq \ell \leq m$ set $B_\ell := c_{\ell\ell}$. Check $B_0 \prod_{\ell=1}^m B_\ell^{t_\ell} = \mathrm{com}_{ck}(F_1, \ldots, F_n; z_b)$
Set $B'_1 := \mathrm{com}_{ck}(1; 0)$. Check $B'_0 \prod_{\ell=1}^m (B'_\ell)^{t_\ell} = \mathrm{com}_{ck}(F_0; z')$.
Check $\hat{B} \prod_{\ell=2}^m (B'_\ell)^{t_{\ell-1}} = \mathrm{com}_{ck}(F_n - t_m a; \hat{z})$
Check

$$C_{00} \cdot \prod_{i=1}^m C_{i0}^{s_i} \cdot \prod_{\ell=1}^m C_{0\ell}^{t_\ell} \cdot \prod_{i=1}^m \prod_{\ell=1}^m C_{i\ell}^{s_i t_\ell} = \mathrm{com}_{ck}(f_1 F_0, \ldots, f_n F_{n-1}; z_{ab})$$

Theorem 1. *The protocol described above is a 3-move public-coin perfect SHVZK argument of knowledge of a_{ij} and r_i such that $a = \prod_{i=1}^m \prod_{j=1}^n a_{ij}$ and for all i we have $A_i = \mathrm{com}_{ck}(a_{i1}, \ldots, a_{in}; r_i)$.*

The proof can be found in the full paper [23].

4 Committed Permutation of Known Elements

Consider a vector of commitments $B_1, \ldots, B_m$ and a set of values $\{a_{ij}\}_{i=1,j=1}^{m,n}$. In this section we will give an argument of knowledge of $\pi \in \Sigma_N$ and $\{r_i\}_{i=1}^m$ such that:

$$\begin{aligned} B_1 &= \mathrm{com}_{ck}(a_{\pi^{-1}(11)}, a_{\pi^{-1}(12)}, \ldots, a_{\pi^{-1}(1n)}; r_1) \\ &\vdots \\ B_m &= \mathrm{com}_{ck}(a_{\pi^{-1}(m1)}, a_{\pi^{-1}(m2)}, \ldots, a_{\pi^{-1}(mn)}; r_m) \end{aligned}$$

(Here we identify $[N]$ with $[m] \times [n]$.)

Our argument uses Neff's idea [30], which is to let the verifier pick a value x at random and let the prover argue that the committed values b_{ij} satisfy $\prod_{i=1}^m \prod_{j=1}^n (x - b_{ij}) = \prod_{i=1}^m \prod_{j=1}^n (x - a_{ij})$. If the committed b_{ij} are a permutation of a_{ij} this equation holds, since polynomials are invariant under permutation of their roots. On the other hand, if b_{ij} are not a permutation of a_{ij}, then by the Schwartz-Zippel lemma there is negligible chance over the choice of x for the equality to hold.

Initial challenge: $x \leftarrow \mathbb{Z}_q$
Answer: Define $B'_1 := \mathrm{com}_{ck}(x, \ldots, x; 0) B_1^{-1}, \ldots, B'_m := \mathrm{com}_{ck}(x, \ldots, x; 0) B_m^{-1}$ and $a := \prod_{i=1}^m \prod_{j=1}^n (x - a_{ij})$.
Make a 3-move argument of knowledge of openings of $B'_1, \ldots, B'_m$ such that the product of all the entries is a.

Theorem 2. *The protocol is a 4-move public coin perfect SHVZK argument of knowledge of a_{ij}, r_i, π such that $B_i := \mathrm{com}_{ck}(a_{\pi^{-1}(i1)}, \ldots, a_{\pi^{-1}(in)}; r_i)$.*

We refer to the full paper [23] for a proof.

5 Multi-exponentiation to Committed Exponents

Consider a set of commitments $A_1, \ldots, A_m$, a matrix of ciphertexts $E_{11}, \ldots, E_{mn}$ and a ciphertext E. In this section we will give an argument of knowledge of $\{a_{ij}\}_{i=1,j=1}^{m,n}$, $\{r_i\}_{i=1}^{m}$ and R such that:

$$\begin{aligned} A_1 &= \mathrm{com}_{ck}(a_{11}, a_{12}, \ldots, a_{1n}; r_1) \\ &\vdots \\ A_m &= \mathrm{com}_{ck}(a_{m1}, a_{m2}, \ldots, a_{mn}; r_m) \end{aligned} \qquad \text{and} \qquad E = E_{pk}(1;R)\prod_{i=1}^{m}\prod_{j=1}^{n} E_{ij}^{a_{ij}}.$$

The argument will contain m^2 commitments, m^2 ciphertexts and n elements in $\mathbb{Z}_q$, where $N = mn$. Choosing $m = N^{1/3}$ gives a communication complexity of $O(N^{2/3})\kappa$ bits.

When describing the idea, let us first just consider how to get soundness and ignore the issue of zero-knowledge for a moment. In the argument, the prover will prove knowledge of the committed exponents, so let us from now on assume the committed values are well-defined. The prover can compute m^2 ciphertexts

$$D_{i\ell} = \prod_{j=1}^{n} E_{\ell j}^{a_{ij}}.$$

We have $E = E_{pk}(1;R)\prod_{i=1}^{m} D_{ii} = E_{pk}(1;R)\prod_{i=1}^{m}\prod_{j=1}^{n} E_{ij}^{a_{ij}}$. Ignoring R that can be dealt with using standard zero-knowledge techniques all that remains is for the verifier to be convinced $D_{i\ell}$ have been correctly computed. For this purpose the verifier will select challenges $t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$ at random. The prover will open $\prod_{i=1}^{m} A_i^{t_i}$ to the values $f_1 := \sum_{i=1}^{m} t_i a_{i1}, \ldots, f_n := \sum_{i=1}^{m} t_i a_{in}$. The verifier now checks for each $1 \leq \ell \leq m$ that $\prod_{j=1}^{n} E_{\ell j}^{f_j} = \prod_{i=1}^{m} D_{i\ell}^{t_i}$. Writing this out we have $\prod_{i=1}^{m}(\prod_{j=1}^{n} E_{\ell j}^{a_{ij}})^{t_i} = \prod_{i=1}^{m} D_{i\ell}^{t_i}$. Since t_i are chosen at random, there is overwhelming probability for one of these checks to fail unless for all i, ℓ we have $D_{i\ell} = \prod_{j=1}^{n} E_{\ell j}^{a_{ij}}$.

In the argument, we wish to have honest verifier zero-knowledge. We will therefore multiply the $D_{i\ell}$ ciphertexts with random encryptions to avoid leaking information about the exponents. This, however, makes it possible to encrypt anything in $D_{i\ell}$, so to avoid cheating we commit to the plaintexts of those random encryptions and use the commitments to prove that they all cancel out against each other.

Initial message:

$a_{01}, \ldots, a_{0n} \leftarrow \mathbb{Z}_q$; $r_0 \leftarrow \mathcal{R}_{ck}$; $A_0 = \mathrm{com}_{ck}(a_{01}, a_{02}, \ldots, a_{0n}; r_0)$
$b_{01}, \ldots, b_{mm} \leftarrow \mathbb{Z}_q$; $r_{01}, \ldots, r_{mm} \leftarrow \mathcal{R}_{ck}$; $b_{mm} := -\sum_{i=1}^{m-1} b_{ii}$; $r_{mm} := -\sum_{i=1}^{m-1} r_{ii}$

$$\begin{array}{lcl} C_{01} := \mathrm{com}_{ck}(b_{01}; r_{01}) & \ldots & C_{0m} := \mathrm{com}_{ck}(b_{0m}; r_{0m}) \\ \vdots & & \vdots \\ C_{m1} := \mathrm{com}_{ck}(b_{m1}; r_{m1}) & \ldots & C_{mm} := \mathrm{com}_{ck}(b_{mm}; r_{mm}) \end{array}$$

$R_{01}, \ldots, R_{mm} \leftarrow \mathcal{R}_{pk}\ ; R_{mm} := R - \sum_{i=1}^{m-1} R_{ii}$

$$
\begin{array}{lll}
D_{01} := E_{pk}(g^{b_{01}}; R_{01}) \prod_{j=1}^{n} E_{1j}^{a_{0j}} & \cdots & D_{0m} := E_{pk}(g^{b_{0m}}; R_{0m}) \prod_{j=1}^{n} E_{mj}^{a_{0j}} \\
\vdots & & \vdots \\
D_{m1} := E_{pk}(g^{b_{m1}}; R_{m1}) \prod_{j=1}^{n} E_{1j}^{a_{mj}} & \cdots & D_{mm} := E_{pk}(g^{b_{mm}}; R_{mm}) \prod_{j=1}^{n} E_{mj}^{a_{mj}}
\end{array}
$$

Send $(A_0, C_{01}, \ldots, C_{mm}, D_{01}, \ldots, D_{mm})$ to the verifier

Challenge: $t_1, \ldots, t_m \leftarrow \mathbb{Z}_q$

Answer:

For $1 \le j \le n\ :\ f_j := a_{0j} + \sum_{i=1}^{m} t_i a_{ij}\ ;\ z := r_0 + \sum_{i=1}^{m} t_i r_i$

For $1 \le \ell \le m\ :\ F_\ell := b_{0\ell} + \sum_{i=1}^{m} t_i b_{i\ell}\ ;\ z_\ell := r_{0\ell} + \sum_{i=1}^{m} t_i r_{i\ell}\ ;\ Z_\ell := R_{0\ell} + \sum_{i=1}^{m} t_i R_{i\ell}$

Send $(f_1, \ldots, f_n, F_1, \ldots, F_m, z, z_1, \ldots, z_m, Z_1, \ldots, Z_m)$ to the verifier

Verification:

Check $A_0 \prod_{i=1}^{m} A_i^{t_i} = \mathrm{com}_{ck}(f_1, \ldots, f_n; z)$

For $1 \le \ell \le m$ check

$$
C_{0\ell} \prod_{i=1}^{m} C_{i\ell}^{t_i} = \mathrm{com}_{ck}(F_\ell; z_\ell) \qquad \text{and} \qquad E_{pk}(g^{F_\ell}; Z_\ell) \prod_{j=1}^{n} E_{\ell j}^{f_j} = D_{0\ell} \prod_{i=1}^{m} D_{i\ell}^{t_i}
$$

Check $\prod_{i=1}^{m} C_{ii} = \mathrm{com}_{ck}(0; 0)$

Check $E = \prod_{i=1}^{m} D_{ii}$

Theorem 3. *The protocol above is a 3-move public coin perfect SHVZK argument of knowledge of* $a_{11}, \ldots, a_{mn}, r_1, \ldots, r_m, R$ *so* $E = E_{pk}(1; R) \prod_{i=1}^{m} \prod_{j=1}^{n} E_{ij}^{a_{ij}}$ *and* $A_i = \mathrm{com}_{ck}(a_{i1}, \ldots, a_{in}; r_i)$.

We refer to the full paper [23] for the proof.

6 Shuffle Argument

Given ciphertexts $\{e_{ij}\}_{i=1,j=1}^{m,n}$ and $\{E_{ij}\}_{i=1,j=1}^{m,n}$ we will give an argument of knowledge of $\pi \in \Sigma_N$ and $\{R_{ij}\}_{i=1,j=1}^{m,n}$ such that for all i, j we have $E_{ij} = e_{\pi^{-1}(ij)} E_{pk}(1; R_{ij})$. The most expensive components of the argument will be a product of committed elements argument and a multi-exponentiation to committed elements argument described in the previous sections. The total size of the argument is therefore $O(m^2 + n)\kappa$ bits, where $N = mn$. With $m = N^{1/3}$ this gives an argument of size $O(N^{2/3})\kappa$ bits.

The argument proceeds in seven steps. First the prover commits to the permutation π, by making a commitment to $1, \ldots, N$ in permuted order. Then the verifier picks challenges $s_1, \ldots, s_m, t_1, \ldots, t_n$ at random. The prover commits to the challenges $s_i t_j$ in permuted order. The prover now proves that she has committed to $s_i t_j$ permuted in the same order as the permutation committed to in the initial commitment. The point of the argument is that since the permutation is committed before seeing the challenges, the prover has no choice in creating the commitment, the random challenges have already been assigned unique slots in the commitment.

The other part of the argument is to use the committed exponentiation technique to show that $\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j} = E_{pk}(1;R)\prod_{i=1}^{m}\prod_{j=1}^{n} E_{\pi(ij)}^{s_i t_j}$ for some known R. If we look at the plaintext, this implies $\prod_{i=1}^{m}\prod_{j=1}^{n} m_{ij}^{s_i t_j} = \prod_{i=1}^{m}\prod_{j=1}^{n} M_{\pi(ij)}^{s_i t_j}$. With the permutation fixed before the challenges are chosen at random there is overwhelming probability that the argument fails unless for all i, j we have $M_{ij} = m_{\pi^{-1}(ij)}$.

Initial message: The prover sets $a_{\pi(ij)} := m(i-1)+j$. The prover picks $r_{a1},\dots,r_{am} \leftarrow \mathcal{R}_{ck}$ and sets

$$\begin{aligned} A_1 &:= \mathrm{com}_{ck}(a_{11}, a_{12}, \dots, a_{1n}; r_{a1}) \\ &\vdots \\ A_m &:= \mathrm{com}_{ck}(a_{m1}, a_{m2}, \dots, a_{mn}; r_{am}) \end{aligned}$$

First challenge: $s_1,\dots,s_m,t_1,\dots,t_n \leftarrow \mathbb{Z}_q$
First answer: We define $b_{\pi(ij)} := s_i t_j$. The prover picks $r_{b1},\dots,r_{bn} \leftarrow \mathcal{R}_{ck}$ and sets

$$\begin{aligned} B_1 &:= \mathrm{com}_{ck}(b_{11}, b_{12}, \dots, b_{1n}; r_{b1}) \\ &\vdots \\ B_m &:= \mathrm{com}_{ck}(b_{m1}, b_{m2}, \dots, b_{mn}; r_{bm}) \end{aligned}$$

Second challenge: $\lambda \leftarrow \mathbb{Z}_q$
Answer: Make a 4-move argument of knowledge of $\pi \in \Sigma_N$ and openings of $A_1^{\lambda}B_1,\dots,A_m^{\lambda}B_m$ so they contain a permutation of the N values $\lambda(m(i-1)+j)+s_i t_j$. Observe, the first move of this argument can be made in parallel with the second challenge so we only use three additional moves.
Make a 3-move argument of knowledge of b_{ij}, r_{bi}, R so

$$\begin{aligned} B_1 &= \mathrm{com}_{ck}(b_{11}, b_{12}, \dots, b_{1n}; r_{b1}) \\ &\vdots \\ B_m &= \mathrm{com}_{ck}(b_{m1}, b_{m2}, \dots, b_{mn}; r_{bm}) \\ \text{and} \quad \prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j} &= E_{pk}(1;R)\prod_{i=1}^{m}\prod_{j=1}^{n} E_{ij}^{b_{ij}}. \end{aligned}$$

Theorem 4. *The protocol is a 7-move public coin perfect SHVZK argument of knowledge of $\pi \in \Sigma$ and $R_{ij} \in \mathcal{R}_{pk}$ so $E_{ij} = e_{\pi^{-1}(ij)}E_{pk}(1;R_{ij})$.*

We refer to the full paper [23] for the proof.

7 Efficient Verification

The small size of the argument gives a corresponding low cost of verification. There are, however, $2N$ ciphertexts that we must exponentiate in the verification. In this section we show that the verifier computation can be reduced to making multi-exponentiations of the ciphertexts to small exponents.

7.1 Prover-Assisted Multi-exponentiation

In our shuffle argument, the verifier has to compute

$$\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j}.$$

The prover can assist this computation by computing $D_1,\ldots,D_n$ as $D_j := \prod_{i=1}^{m} e_{ij}^{s_i}$. The verifier can then compute

$$\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j} = \prod_{j=1}^{m} D_j^{t_j}.$$

What remains is for the verifier to check that the ciphertexts are correct, which can be done by verifying

$$\prod_{j=1}^{n} D_j^{\alpha_j} = \prod_{i=1}^{m}(\prod_{j=1}^{n} e_{ij}^{\alpha_j})^{s_i}$$

for randomly chosen α_j. Since the check is done off-line, the verifier can use small exponents α_j, say, 32-bit exponents. This trick reduces the amount of verifier computation that is needed for computing $\prod_{i=1}^{m}\prod_{i=1}^{n} e_{ij}^{s_i t_j}$ to one m-exponentiation to exponents from $\mathbb{Z}_q$ and $m+1$ n-exponentiations to small exponents.

When m is small, this strategy may actually end up increasing the communication complexity of the shuffle. However, the exact same method can be employed when we let the verifier compute the t_j-values as products the n products of $\psi_1,\ldots,\psi_{n_1}$ and $\tau_1,\ldots,\tau_{n_2}$ where $n = n_1 n_2$. If we choose $n_2 = \sqrt{N}$ for instance, we get that the prover only sends $\sqrt{N}$ ciphertexts to the verifier. The verifier then makes $\sqrt{N}$-multi-exponentiations to small exponents $\alpha_1,\ldots,\alpha_{\sqrt{N}}$.

7.2 Randomized Verification

In the argument for multi-exponentiation to committed exponents, the verifier must check m equalities of the form

$$E_{pk}(g^{F_\ell};Z_\ell)\prod_{j=1}^{n} E_{\ell j}^{f_j} = D_{0\ell}\prod_{i=1}^{m} D_{i\ell}^{t_i}.$$

This can be done off-line in a randomized way by picking $\alpha_1,\ldots,\alpha_m$ at random and testing whether

$$\begin{aligned} E_{pk}(g^{\sum_{\ell=1}^{m}\alpha_\ell F_\ell};\sum_{\ell=1}^{m}\alpha_\ell Z_\ell)\prod_{j=1}^{n}\left(\prod_{\ell=1}^{m} E_{\ell j}^{\alpha_\ell}\right)^{f_j} &= \prod_{\ell=1}^{m}\left(E_{pk}(g^{F_\ell};Z_\ell)\prod_{j=1}^{n} E_{\ell j}^{f_j}\right)^{\alpha_\ell} \\ &= \prod_{\ell=1}^{m} D_{0\ell}^{\alpha_\ell}\ \prod_{i=1}^{m}\left(\prod_{\ell=1}^{m} D_{i\ell}^{\alpha_\ell}\right)^{t_i}. \end{aligned}$$

This way, we make n m-multi-exponentiations to small exponents α_ℓ and one n-multi-exponentiation to larger exponents f_j.

8 Comparison

Let us compare our shuffle argument with the most efficient arguments for correctness of a shuffle of ElGamal ciphertexts in the literature. Furukawa and Sako [17] suggested an efficient argument for correctness of a shuffle based on committing to a permutation matrix. This scheme was further refined by Furukawa [15]. We will use Groth and Lu's [24] estimates for the complexity of Furukawa's scheme. Neff [30, 31] gave an efficient interactive proof for correctness of a shuffle. Building on those ideas Groth [21] suggested a perfect SHVZK argument for correctness of a shuffle. Our shuffle argument builds on Neff's and Groth's schemes.

We will compare the schemes using an elliptic curve of prime order q. We use $|q| = 256$ so SHA256 can be used to choose the public coin challenges. We measure the communication complexity in bits and measure the prover and verifier computation in single exponentiations. By this we mean that in all schemes, we count the cost of a multi-exponentiation to n exponents as n single exponentiations. We compare the most efficient shuffle arguments in Table 1. Section 7 offer a couple of speedup techniques.

Elliptic curve Group order: $\|q\| = 256$	Furukawa-Sako [17]	Groth [21]	Furukawa [15, 24]	proposed
Prover (single expo.)	$8N$	$6N$	$7N$	$3mN + 5N$
Verifier (single expo.)	$10N$	$6N$	$8N$	$4N + 3n$
Prover's communication (bits)	$1280N$	$768N$	$768N$	$768m^2 + 768n$
Rounds	3	7	3	7

Table 1. Comparison of shuffle arguments for $N = mn$ ElGamal ciphertexts.

If we employ the randomization techniques from Section 7 then the prover's cost increases by $2N$ exponentiations, whereas the verifier's complexity reduces to $4N$ small exponentiations and $m^2 + 3n$ exponentiations to full size exponents from $\mathbb{Z}_q$.

For all schemes it holds that multi-exponentiation techniques can reduce their cost, see e.g. Lim [27]. We refer to the full paper of Groth [21] for a discussion of randomization techniques and other tricks that can be used to reduce the computational complexity of all the shuffle arguments. An additional improvement of our scheme is to let the prover assist the verifier in computing the multi-exponentiation $\prod_{i=1}^{m}\prod_{j=1}^{n} e_{ij}^{s_i t_j}$, see Section 7. Table 2 has back-of-the-envelope estimates when we compare an optimized version of our scheme to that of Groth [21]. We assume that we are shuffling $N = 100,000$ ElGamal ciphertexts with parameters $m = 10, n = 10,000$ so $N = mn$. We count the computational cost in the number of multiplications. In parenthesis we are giving timing estimates assuming the use of equipment where a multiplication takes $1\mu s$, which is conservative given today's equipment. We only count the cost of the shuffle argument in Table 2, not the cost of computing the shuffle or the size of the shuffle (51 Mbits).

	Groth [21]	proposed
Prover's computation	$18 \cdot 10^6$ mults (18 sec.)	$143 \cdot 10^6$ mults (143 sec.)
Verifier's computation	$14 \cdot 10^6$ mults (14 sec.)	$5 \cdot 10^6$ mults (5 sec.)
Prover's communication	77 Mbits	8 Mbits

Table 2. Comparison of shuffle arguments for $100,000$ ElGamal ciphertexts.

References

1. Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 437–447, 1998.
2. Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *PKC*, volume 1992 of *Lecture Notes in Computer Science*, pages 317–324, 2001.
3. 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.
4. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.
5. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, 1992.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73, 1993.
7. 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.
8. Gilles Brassard, David Chaum, and Claude Crèpeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.
9. 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.
10. 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.
11. 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.
12. Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.
13. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
14. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1986.
15. Jun Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 88-A(1):172–188, 2005.
16. Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 16–30, 2002.

17. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, 2001.
18. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.
19. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, pages 102–113, 2003. Full paper available at `http://eprint.iacr.org/2003/034`.
20. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. *SIAM Journal of Computing*, 18(1):186–208, 1989.
21. 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 ePrint Archive: `http://eprint.iacr.org/2005/246`.
22. Jens Groth. Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, 2004. PhD thesis. xii+119 pp.
23. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle, 2008. Available at `http://www.brics.dk/~jg/PCPShuffle.pdf`.
24. Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In *PKC*, volume 4450 of *Lecture Notes in Computer Science*, pages 377–392, 2007.
25. Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *CCC*, pages 278–291, 2007.
26. Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732, 1992.
27. 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`.
28. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.
29. Silvio Micali. Computationally sound proofs. *SIAM Journal of Computing*, 30(4):1253–1298, 2000.
30. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–125, 2001.
31. C. Andrew Neff. Verifiable mixing (shuffling) of ElGamal pairs, 2003. `http://www.votehere.net/vhti/documentation/egshuf.pdf`.
32. Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. A provably secure and effcient verifiable shuffle based on a variant of the Paillier cryptosystem. *Journal of Universal Computer Science*, 11(6):986–1010, 2005.
33. Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: a formal model and a Paillier-based three-round construction with provable security. *International Journal of Information Security*, 5(4):241–255, 2006.
34. Takao Onodera and Keisuke Tanaka. Shufle for Paillier's encryption scheme. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E88-A(5):1241–1248, 2005.
35. Pascal Paillier. Public-key cryptosystems based on composite residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–239, 1999.
36. 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.
37. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403, 1995.
38. Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 273–292, 2005.