Proofs of Restricted Shuffles

Full version*

Björn Terelius and Douglas Wikström

CSC KTH Stockholm, Sweden
{terelius,dog}@csc.kth.se

Abstract. A proof of a shuffle is a zero-knowledge proof that one list of ciphertexts is a permutation and re-encryption of another list of ciphertexts. We call a shuffle restricted if the permutation is chosen from a public subset of all permutations. In this paper, we introduce a general technique for constructing proofs of shuffles which restrict the permutation to a group that is characterized by a public polynomial. This generalizes previous work by Reiter and Wang [22], and de Hoogh et al. [7]. Our approach also gives a new efficient proof of an unrestricted shuffle that we think is conceptually simpler and allow a simpler analysis than all previous proofs of shuffles.

Keywords: cryptographic protocols, election schemes, mix-nets, proof of a shuffle

1 Introduction

Mix-Nets. Suppose that N senders $S_1, \ldots, S_N$ each wish to send a message, but remain anonymous within the group of senders, e.g., the senders could be voters in an election. Chaum [5] introduced the notion of a mix-net (or anonymous channel) to solve this problem. A mix-net is a cryptographic protocol executed by k mix-servers $M_1, \ldots, M_k$, where k typically is much smaller than N. All provably secure mix-nets proposed in the literature take as input a list L_0 of ciphertexts and order the mix-servers in a chain. Each mix-server M_j in the chain takes as input the output L_{j-1} of the previous mix-server in the chain. It processes each ciphertext in L_{j-1} by decrypting and/or re-encrypting it, and then forms its output L_j as the processed ciphertexts in random order. This operation is called a *shuffle*. If the ciphertexts are not decrypted during processing, the mix-servers then perform a joint verifiable decryption of the final list L_k. In some applications, the final output of the mix-net may be a list of the ciphertexts themselves, in which case no joint decryption takes place.

In Chaum's original construction a generic cryptosystem is used. A sender encrypts its message m_i as $\mathsf{E}_{pk_1}(\mathsf{E}_{pk_2}(\cdots \mathsf{E}_{pk_k}(m_i)\cdots))$, where pk_j is the public key of the jth mix-server, and the mix-servers use standard decryption when

* The original publication is available at http://www.springerlink.com/content/q84516u323h381j7/.

processing the ciphertexts. Park et al. [18] observed that if a homomorphic cryptosystem is used, the increase in the size of the submitted ciphertext can be avoided. Another, perhaps more important, consequence of using a homomorphic cryptosystem is that it simplifies the construction of a zero-knowledge proof that a mix-server shuffles its input correctly, a so called *proof of a shuffle*. We give a detailed account of previous work on such protocols later, but first we conclude the discussion on mix-nets.

Although the mix-servers use the homomorphic properties of the cryptosystem constructively, Pfitzmann [20] points out that a non-malleable cryptosystem must be used in the submission phase. There are several ways to achieve this in a provably secure way.

The security of a mix-net as a whole was first formalized by Abe and Imai [1] in a standalone model, but they did not provide a construction that satisfied their definition. The first definition of security in the UC framework [4] and the first mix-net provably secure as a whole was given by Wikström [25]. Later work [26] gave simpler and more efficient constructions.

Previous Work On Proofs of Shuffles. As far as we know the first proof of a shuffle appears in Sako and Kilian [24] based on a previous protocol by Park et al. [18]. They give a cut-and-choose based zero-knowledge proof of a shuffle with soundness 1/2. Its soundness can be increased using repetition, but this becomes computationally expensive if the number of ciphertexts is large. The first efficient proofs of shuffles were given independently by Neff [17] and Furukawa and Sako [11]. Both approaches were subsequently optimized and generalized, in particular by Groth [13] and Furukawa [9] respectively. Wikström [26] presented a proof of a shuffle based on an idea distinct from both that of Neff and that of Furukawa and Sako.

These shuffles have all been optimized and/or generalized in various ways, e.g., for proving re-encryption and partial decryption shuffling [10], for shuffling hybrid ciphertexts [12], or to reduce communication complexity [15].

A different approach to mix-nets was introduced by Adida and Wikström [3, 2]. They investigate to what extent a shuffle can be pre-computed in such a way that the shuffling can be done in public and the online processing of the mix-servers can be reduced to joint decryption. The construction in [2] is conceptually appealing, but inefficient, whereas the construction in [3] is very efficient as long as the number of senders is relatively small.

In a recent paper, Wikström [27] shows that any proof of a shuffle can be split in an offline part and an extremely efficient online part. The offline part is executed before, or at the same time, that senders submit their inputs and consists of a commitment scheme for permutations and a zero-knowledge proof of knowledge of correctly opening such a commitment. The online part is a commitment-consistent proof of a shuffle, where the prover shows that it correctly uses the committed permutation to process the input. This drastically reduces the online complexity of provably secure mix-nets.

Motivated by insider attacks, Reiter and Wang [22] construct a proof of a rotation (they use the term "fragile mixing"). A rotation is a shuffle, where the

permutation used is restricted to a random rotation of the ciphertexts. Their idea is to reduce the incentive of insiders to reveal any input/output-correspondence, since this would reveal the complete permutation used, which is assumed to be associated with a cost for the insider. Recently, a more efficient proof of a rotation is given by de Hoogh et al. [7]. In fact, they give two protocols: a general protocol for any homomorphic cryptosystem and rotation, and a more efficient solution which requires some mild assumptions on the homomorphic cryptosystem used.

De Hoogh et al. lists several possible applications of proofs of rotations beyond the classical application of proofs of shuffles for mix-nets, e.g., secure integer comparison [21], secure function evaluation [16], and submission schemes in electronic election schemes [23].

1.1 Our Contribution

We introduce a novel technique for restricting the class of permutations in a proof of a shuffle of N ciphertexts by showing that π is chosen such that $F(x_{\pi(1)}, \ldots, x_{\pi(N)}) = F(x_1, \ldots, x_N)$ for some public polynomial F. In particular, we can prove that the permutation is contained in the automorphism group of a (directed or undirected) graph on N elements.

A concrete general proof of rotation with efficiency comparable to that of de Hoogh et al. [7] is trivially derived from our technique, but several other natural concrete examples are derived just as easily, e.g. the list of ciphertexts may be viewed as a complete binary tree and the set of permutations restricted to isomorphisms of the tree.

Furthermore, the basic principle behind our technique can be used in a natural way to construct a novel efficient proof of an *unrestricted* shuffle with an exceptionally simple analysis. Given the large and unwieldy literature on how to construct efficient proofs of shuffles we think this conceptual simplification is of independent interest.

1.2 Informal Description of Our Technique

We briefly describe our results and the technique we use, but before we do so we recall the definition of Pedersen's perfectly hiding commitment scheme [19], or more precisely a well known generalization thereof. The commitment parameters consist of $N+1$ randomly chosen generators $g, g_1, \ldots, g_N$ in a group G_q of prime order q in which the discrete logarithm assumption holds. To commit to an array $(e_1, \ldots, e_N) \in \mathbb{Z}_q^N$, the committer forms $g^s \prod_{i=1}^N g_i^{e_i}$. Below we use the fact that sigma proofs can be used to efficiently prove any polynomial relation between the values $e_1, \ldots, e_N$.

A New Proof of a Shuffle. We first describe how to prove that a matrix $M \in \mathbb{Z}_q^{N \times N}$ over a finite field $\mathbb{Z}_q$ hidden in a Pedersen commitment is a permutation matrix. Let $\langle \cdot, \cdot \rangle$ denote the standard inner product in $\mathbb{Z}_q^N$ and let $\overline{x} = (x_1, \ldots, x_N)$ be a list of variables. If M does not have exactly one non-zero

element in each column and each row, then $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle \neq \prod_{i=1}^{N} x_i$ where $\overline{m}_i$ denotes the ith row of M. This can be tested efficiently by Schwarz-Zippel's lemma, which we recall states that if $f(x_1, \ldots, x_N)$ is a non-zero polynomial of degree d and we pick a point $\overline{e}$ from $\mathbb{Z}_q^N$ randomly, then the probability that $f(\overline{e}) = 0$ is at most d/q.

To prove that M is a permutation matrix, given that it has exactly one non-zero element in each row and column, is of course trivial; simply show that the elements of each row sum to one.

We observe that proving both these properties can be done efficiently using the matrix commitment scheme used in [11, 27]. The commitment parameters of this scheme consists of independently chosen generators $g, g_1, \ldots, g_N$ of a group G_q of prime order q. To commit to a matrix M, the committer computes $(a_1, \ldots, a_N) = (g^{s_1} \prod_{i=1}^{N} g_i^{m_{i,1}}, \ldots, g^{s_N} \prod_{i=1}^{N} g_i^{m_{i,N}})$. which allows publicly computing a commitment of all $\langle \overline{m}_i, \overline{e} \rangle$ as $\prod_{j=1}^{N} a_j^{e_j} = g^{\langle s, \overline{e} \rangle} \prod_{j=1}^{N} g_j^{\langle \overline{m}_j, \overline{e} \rangle}$. The prover may then use standard sigma proofs to show that $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle = \prod_{i=1}^{N} e_i$. To show that the sum of each row is one it suffices to prove that $\prod_{j=1}^{N} a_j / \prod_{j=1}^{N} g_j$ is on the form g^s for some s.

At this point we may invoke the commitment-consistent proof of a shuffle in [27] to extend the above to a complete proof of an unrestricted shuffle, but we also present a more direct construction.

Restricting the Set of Permutations. In the interest of describing the techniques, we consider how to restrict the set of permutations to the automorphism group of an undirected graph $\mathscr{G}$. Let the graph have vertices $V = \{1, 2, 3, \ldots, N\}$ and edges $E \subseteq V \times V$ and encode the graph as a polynomial $F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j) \in E} x_i x_j$ in $\mathbb{Z}_q[x_1, \ldots, x_N]$. Observe that π is contained in the automorphism group of $\mathscr{G}$ if and only if $F_{\mathscr{G}}(x_{\pi(1)}, \ldots, x_{\pi(N)}) = F_{\mathscr{G}}(x_1, \ldots, x_N)$.

Suppose that a prover has shown that a committed matrix $M \in \mathbb{Z}_q^{N \times N}$ is a permutation matrix corresponding to a permutation π. If π is not contained in the automorphism group of $\mathscr{G}$, it follows from Schwartz-Zippel's lemma that $\Pr\left[F_{\mathscr{G}}(e_{\pi(1)}, \ldots, e_{\pi(N)}) = F_{\mathscr{G}}(e_1, \ldots, e_N)\right]$ is exponentially small, when $\overline{e} \in \mathbb{Z}_q$ is randomly chosen by the verifier. Since $M\overline{e} = (e_{\pi(1)}, \ldots, e_{\pi(N)})$ the verifier can easily compute a commitment of the permuted random exponents as $\prod_{j=1}^{N} a_j^{e_j} = g^{\langle \overline{s}, \overline{e} \rangle} \prod_{j=1}^{N} g_i^{e_{\pi(i)}}$. The prover can then use a standard sigma proof to prove the equality, for example by proving correct computation of each gate in the arithmetic circuit for the polynomial.

Note that it is not important that the polynomial comes from a graph. Hence, we can apply the same technique to prove that π satisfies $F(e_{\pi(1)}, \ldots, e_{\pi(N)}) = F(e_1, \ldots, e_N)$ for any public polynomial F.

2 Notation and Tools

We use n as the security parameter and let q denote an n-bit prime integer. The field with q elements is denoted by $\mathbb{Z}_q$. Our protocols are employed in a group

G_q of prime order q with standard generator g, in which the discrete logarithm problem is assumed to be hard. We use bars over vectors to distinguish them from scalars. The angle bracket $\langle \overline{a}, \overline{b} \rangle$ denotes the inner product $\sum_{i=1}^{N} a_i b_i$ of two vectors $\overline{a}, \overline{b} \in \mathbb{Z}_q^N$. For a list $u = (u_1, \ldots, u_N) \in G_q^N$ and a vector $\overline{e} \in \mathbb{Z}_q^N$ we abuse notation and write $u^{\overline{e}} = \prod_{i=1}^{N} u_i^{e_i}$. Throughout, we use $S \subseteq \mathbb{Z}_q$ to denote the set from which the components of our random vectors are chosen.

If $\mathscr{R}_1$ and $\mathscr{R}_2$ are relations we denote by $\mathscr{R}_1 \vee \mathscr{R}_2$ the relation consisting of pairs $((x_1, x_2), w)$ such that $(x_1, w) \in \mathscr{R}_1$ or $(x_2, w) \in \mathscr{R}_2$.

Matrix Commitments. We use a variation of Pedersen commitments [19] in a group G_q, to commit to a matrix over $\mathbb{Z}_q$. The commitment parameter ck needed to commit to an $N \times 1$ matrix consists of a description of the group G_q with its standard generator g and randomly chosen group elements $g_1, \ldots, g_N \in G_q$. An $N \times 1$-matrix M over $\mathbb{Z}_q$ is committed to by computing $a = \mathcal{C}_{ck}(M, s) = g^s \prod_{i=1}^{N} g_i^{m_i}$, where $s \in \mathbb{Z}_q$ is chosen randomly. We abuse notation and omit the commitment parameter when it is clear from the context. An $N \times N$-matrix M is committed to column-wise, i.e., $\mathcal{C}(M, \overline{s}) = \left(\mathcal{C}\left((m_{i,1})_{i=1}^{N}, s_1\right), \ldots, \mathcal{C}\left((m_{i,N})_{i=1}^{N}, s_N\right)\right)$, where in this case $\overline{s}$ is chosen randomly in $\mathbb{Z}_q^N$. By committing to a matrix M column-wise we get the useful identity

$$\mathcal{C}(M, \overline{s})^{\overline{e}} = \prod_{j=1}^{N} g^{s_j e_j} \prod_{i=1}^{N} g_i^{m_{i,j} e_j} = g^{\langle \overline{s}, \overline{e} \rangle} \prod_{i=1}^{N} g_i^{\sum_{j=1}^{N} m_{i,j} e_j} = \mathcal{C}(M\overline{e}, \langle \overline{s}, \overline{e} \rangle) \quad .$$

This feature plays a central role in our approach. It is easy to see that the commitment scheme is perfectly hiding. The binding property is known to hold under the discrete logarithm assumption in G_q, see [11] for a proof.

We construct protocols that are sound under the assumption that the binding property of the above protocol is not violated. We define $\mathscr{R}_{com}$ to be the relation consisting of pairs $((ck, a), (M, \overline{s}, M', \overline{s}'))$ such that $a = \mathcal{C}_{ck}(M, \overline{s}) = \mathcal{C}_{ck}(M', \overline{s}')$, i.e., finding a witness corresponds to violating the binding property of the commitment scheme.

Σ-proofs. Recall that a sigma proof is a three-message protocol that is both special sound and special honest verifier zero-knowledge [6]. The first property means that the view of the verifier can be simulated for a given challenge, and the second property means that a witness can be computed from any pair of accepting transcripts with identical first messages and distinct challenges. It is well known that if a prover $\mathcal{P}^*$ convinces the verifier with probability δ, there exists an extractor running in expected time $\mathcal{O}\big(T/(\delta - \epsilon)\big)$ for some polynomial T and some negligible knowledge error ϵ. Given a statement x, we denote the execution of a sigma proof of knowledge of a witness w such that $(x, w) \in \mathscr{R}$ by $\Sigma\text{-}\mathsf{proof}[w \,|\, (x, w) \in \mathscr{R}]$.

We need to prove knowledge of how to open commitments such that the committed values satisfy a public polynomial relation, i.e. to construct a sigma proof

$$\Sigma\text{-}\mathsf{proof}\left[\overline{e} \in \mathbb{Z}_q^N, s \in \mathbb{Z}_q \,\middle|\, a = \mathcal{C}(\overline{e}, s) \wedge f(\overline{e}) = e'\right]$$

given a commitment $a = \mathcal{C}(\overline{e}, s)$, a polynomial $f \in \mathbb{Z}_q[x_1, \ldots, x_N]$, and a value $e' \in \mathbb{Z}_q$. We remind the reader how this can be done.

The parties agree on an arithmetic circuit over $\mathbb{Z}_q$ that evaluates the polynomial f. The prover constructs new commitments a_i to each individual value e_i hidden in a and proves that it knows how to open all commitments consistently with a proof of the form

$$\Sigma\text{-proof}\left[\overline{e} \in \mathbb{Z}_q^N, s, s_1, \ldots, s_N \in \mathbb{Z}_q \,\middle|\, a = \mathcal{C}(\overline{e}, s) \wedge \forall_{i=1}^N (a_i = \mathcal{C}(e_i, s_i))\right] \quad .$$

The resulting commitments $a_1, \ldots, a_N$ are considered the input of the arithmetic circuit. For each summation gate, the two input commitments of the gate are multiplied to form the output of the gate. For each product gate with input commitments $a_1 = \mathcal{C}(e_1, s_1)$ and $a_2 = \mathcal{C}(e_2, s_2)$, the prover forms a new commitment $a_3 = \mathcal{C}(e_3, s_3)$ and proves that $e_3 = e_1 e_2$ with a sigma protocol

$$\Sigma\text{-proof}\left[e_2, s_2, s_3' \in \mathbb{Z}_q \,\middle|\, a_3 = g^{s_3'} a_1^{e_2} \wedge a_2 = \mathcal{C}(e_2, s_2)\right] \quad .$$

Finally, the output a of the entire circuit is shown to be a commitment of e' using a protocol of the form

$$\Sigma\text{-proof}\left[s \in \mathbb{Z}_q \,\middle|\, a/g_1^{e'} = g^s\right] \quad .$$

Special soundness and special honest verifier zero-knowledge allow us to execute all these protocols in parallel using a single challenge from the verifier, thus forming a new sigma protocol. Together with the binding property of the commitment scheme, this implies that the prover knows $\overline{e} \in \mathbb{Z}_q^N$ and $s \in \mathbb{Z}_q$ such that $\mathcal{C}(\overline{e}, s) = a \wedge f(\overline{e}) = e'$.

We remark that it is sometimes possible to do better than the general technique above. In particular when proving that a shuffle is a rotation, one has to prove that a polynomial of the form $\sum_{i=1}^N x_i y_i$ has a certain value. This can be done efficiently by evaluating the polynomial as an inner product between the vectors $\overline{x}$ and $\overline{y}$ using a linear algebra protocol from [14].

Polynomial Equivalence Testing. We use the Schwartz-Zippel lemma to analyze the soundness of our protocols. The lemma gives an efficient, probabilistic test of whether a polynomial is identically zero.

Lemma 1 (Schwartz-Zippel). *Let $f \in \mathbb{Z}_q[x_1, \ldots, x_N]$ be a non-zero multivariate polynomial of total degree $d \geq 0$ over $\mathbb{Z}_q$, let $S \subseteq \mathbb{Z}_q$, and let $e_1, \ldots, e_N$ be chosen randomly from S. Then*

$$\Pr[f(e_1, \ldots, e_N) = 0] \leq \frac{d}{|S|} \quad .$$

3 Proof of Knowledge of Permutation Matrix

We show how to prove knowledge of how to open a Pedersen commitment of a matrix such that the matrix is a permutation matrix. Wikström [27] constructs a commitment-consistent proof of a shuffle for any shuffle-friendly map, based on the same permutation commitment we use here. Thus, it is trivial to construct a proof of a shuffle by combining the protocol below with the online protocol in [27].

Our protocol is based on a simple probabilistic test that accepts a non-permutation matrix with negligible probability.

Theorem 1 (Permutation Matrix). *Let $M = (m_{i,j})$ be an $N \times N$-matrix over $\mathbb{Z}_q$ and $\overline{x} = (x_1, \ldots, x_N)$ a vector of N independent variables. Then M is a permutation matrix if and only if $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle = \prod_{i=1}^{N} x_i$ and $M\overline{1} = \overline{1}$.*

Proof. Consider the polynomial $f(\overline{x}) = \prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle$ in the multivariate polynomial ring $R = \mathbb{Z}_q[x_1, \ldots, x_N]$, where $\overline{m}_i$ is the ith row of M. It is clear that $M\overline{1} = \overline{1}$ and $f(\overline{x}) = \prod_{i=1}^{N} x_i$ if M is a permutation matrix. Conversely, suppose that $M\overline{1} = \overline{1}$ and $f(\overline{x}) = \prod_{i=1}^{N} x_i$. If any row $\overline{m}_i$ were the zero vector, then f would be the zero polynomial. If all rows of M were non-zero, but some row $\overline{m}_i$ contained more than one non-zero element, then f would contain a factor of the form $\sum_{j \in J} m_{i,j} x_j$ with $|J| \geq 2$ and $m_{i,j} \neq 0$ for $j \in J$. Since R is a unique factorization domain, this contradicts the assumption that $f(\overline{x}) = \prod_{i=1}^{N} x_i$. If the jth column contained more than one non-zero element, then $\deg_{x_j} f \geq 2$, again contradicting $f = \prod_{j=1}^{N} x_j$. Thus there is exactly one non-zero element in each row and column and since $M\overline{1} = \overline{1}$, the non-zero element must equal one.

Protocol 1 (Permutation Matrix).
Common Input: Matrix commitment $a \in G_q^N$ and commitment parameters $g, g_1, \ldots, g_N \in G_q$.
Private Input: Permutation matrix $M \in \mathbb{Z}_q^{N \times N}$ and randomness $\overline{s} \in \mathbb{Z}_q^N$ such that $a = \mathcal{C}(M, \overline{s})$.

1. $\mathcal{V}$ chooses $\overline{e} \in S^N \subseteq \mathbb{Z}_q^N$ randomly and hands $\overline{e}$ to $\mathcal{P}$.
2. $\mathcal{P}$ defines $t = \langle \overline{1}, \overline{s} \rangle$ and $k = \langle \overline{s}, \overline{e} \rangle$. Then $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[\begin{matrix} t, k \in \mathbb{Z}_q \\ \overline{e}' \in \mathbb{Z}_q^N \end{matrix} \middle| \mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} \wedge \mathcal{C}\left(\overline{e}', k\right) = a^{\overline{e}} \wedge \prod_{i=1}^{N} e_i' = \prod_{i=1}^{N} e_i \right] .$$

We remark that $\mathcal{V}$ could instead hand a random seed to $\mathcal{P}$ and define $\overline{e}$ as the output of a PRG invoked on the seed. This reduces the amount of randomness needed by the verifier, which is important in applications where the role of $\mathcal{V}$ is played by a multiparty coin-flipping protocol. Since this trick is well known (cf. [27]) and complicates the exposition, we do not detail it here.

Proposition 1. *Protocol 1 is a perfectly complete, 4-message honest verifier zero-knowledge proof of knowledge of the relation $\mathscr{R}_\pi \vee \mathscr{R}_{com}$, where the relation $\mathscr{R}_\pi$ consists of pairs $((ck, a), (M, \overline{s}))$ such that M is a permutation matrix and $a = \mathcal{C}_{ck}(M, \overline{s})$.*

Under the discrete logarithm assumption, it is infeasible to find a witness of the relation $\mathscr{R}_{com}$ for the commitment parameter $(g, g_1, \ldots, g_N)$. Thus, for a randomly chosen commitment parameter, the proposition may be interpreted as a proof of knowledge of the relation $\mathscr{R}_\pi$.

3.1 Proof of Proposition 1

The completeness and zero-knowledge properties follows from the completeness and zero-knowledge properties of the sigma protocol and the hiding property of the commitment scheme. What remains is to show that the protocol is a proof of knowledge by creating an extractor. We do this by extracting witnesses $(\overline{e}', t, k)$ from the sigma proof for N linearly independent vectors $\overline{e}$ and use them to recover the matrix M. Finally, we show that if M is not a permutation matrix, then we are able to extract a witness of the commitment relation $\mathscr{R}_{com}$.

Three-Message Protocol. We first make a conceptual change that allows us to view our four-round prover as a particular three-round prover of a standard sigma-protocol. Given a prover $\mathcal{P}^*$, we denote by $\mathcal{P}_+$ the interactive machine that chooses $\overline{e} \in \mathbb{Z}_q^N$ randomly itself instead of letting $\mathcal{V}$ choose it, and then simulates $\mathcal{P}^*$. We denote by $\mathcal{V}_+$ the corresponding verifier that accepts $\overline{e}$ as part of the first message in the sigma proof.

Basic Extractor. We augment the common input with a list of linearly independent vectors $\overline{e}_1, \ldots, \overline{e}_l \in \mathbb{Z}_q^N$ where $l < N$, let $\mathcal{P}_\perp$ be identical to $\mathcal{P}_+$, and define $\mathcal{V}_\perp$ to be identical to $\mathcal{V}_+$ except that it only accepts if $\overline{e}$ is linearly independent of these. If $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability δ, then $\mathcal{P}_\perp$ clearly convinces $\mathcal{V}_\perp$ with probability at least $\delta - \frac{1}{|S|}$, since the probability that $\overline{e}$ is linearly dependent of $\overline{e}_1, \ldots, \overline{e}_l \in \mathbb{Z}_q^N$ is bounded by $\frac{1}{|S|}$.

It is well known that the sigma proof has an extractor $\mathcal{E}_\perp$ running $\mathcal{P}_\perp$ as a black-box that given linearly independent $\overline{e}_1, \ldots, \overline{e}_l \in \mathbb{Z}_q^N$ extracts an $\overline{e}$ that is linearly independent of the former vectors and a corresponding witness $(\overline{e}', t, k)$ of the sigma proof. Furthermore, $\mathcal{E}_\perp$ runs in expected time $T/(\delta - \frac{1}{|S|} - \epsilon)$ for some polynomial $T(n)$ in the security parameter n, where ϵ is the knowledge error of the sigma proof. Denote by $\mathcal{E}_N$ the extractor that computes witnesses $(\overline{e}_l, t_l, \overline{e}'_l, k_l) = \mathcal{E}_\perp(\overline{e}_1, \ldots, \overline{e}_{l-1}, a, g, g_1, \ldots, g_N)$ for $l = 1, \ldots, N$. Then $\mathcal{E}_N$ runs in expected time $\mathcal{O}\big(NT/(\delta - \frac{1}{|S|} - \epsilon)\big)$.

Computation of Committed Matrix. From linear independence follows that there exists $\alpha_{l,j} \in \mathbb{Z}_q$ such that $\sum_{j=1}^N \alpha_{l,j}\overline{e}_j$ is the lth standard unit vector in $\mathbb{Z}_q^N$. We

conclude that:

$$a_l = \prod_{j=1}^{N} a^{\alpha_{l,j}\overline{e}_j} = \prod_{j=1}^{N} \mathcal{C}\left(\overline{e}'_j, k_j\right)^{\alpha_{l,j}} = \mathcal{C}\left(\sum_{j=1}^{N} \alpha_{l,j}\overline{e}'_j, \sum_{j=1}^{N} \alpha_{l,j} k_j\right) \ .$$

Thus, we have $a = \mathcal{C}(M, \overline{s})$, where $\sum_{j=1}^{N} \alpha_{l,j}\overline{e}'_j$ is the lth column of a matrix $M \in \mathbb{Z}_q^{N\times N}$ and $\overline{s} = \left(\sum_{j=1}^{N} \alpha_{1,j} k_j, \ldots, \sum_{j=1}^{N} \alpha_{N,j} k_j\right) \in \mathbb{Z}_q^N$ is the random vector used to commit.

Product Inequality Extractor. We expect that the matrix M is a permutation matrix, but if this is not the case we must argue that we can find a non-trivial representation of 1 in G_q. We augment the original input with a non-permutation matrix M which we assume satisfy $M\overline{1} = \overline{1}$. We will see that had $M\overline{1} \neq \overline{1}$, we would have been able to extract a witness of $\mathscr{R}_{com}$. Let $\mathcal{P}_\pi$ be identical to $\mathcal{P}_+$, and define $\mathcal{V}_\pi$ to be identical to $\mathcal{V}_+$ except that it only accepts if

$$\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle \neq \prod_{i=1}^{N} e_i \ . \tag{1}$$

From Theorem 1 and Schwartz-Zippel's lemma follows that the probability that the additional requirement is not satisfied is at most $N/|S|$. Thus, if $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability δ, then $\mathcal{P}_\pi$ convinces $\mathcal{V}_\pi$ with probability at least $\delta - N/|S|$. Again, a standard argument implies that there exists an extractor $\mathcal{E}_\pi$ with black-box access to $\mathcal{P}^*$ running in time $\mathcal{O}\big(T'/(\delta - \frac{N}{|S|} - \epsilon)\big)$, which extracts $(\overline{e}, t, \overline{e}', k)$ such that $\mathcal{C}(\overline{e}', k) = a^{\overline{e}}$ and Equation (1) is satisfied.

Main Extractor. Denote by $\mathcal{E}$ the extractor that proceeds as follows:

1. It invokes $\mathcal{E}_N$ to find a matrix M and randomness $\overline{s}$ such that $a = \mathcal{C}(M, \overline{s})$. If M is a permutation matrix, then $\mathcal{E}$ has found the witness $(M, \overline{s})$ of relation $\mathscr{R}_\pi$.
2. If M does not satisfy $M\overline{1} = \overline{1}$, then set $\overline{e}'' = M\overline{1}$ and note that

   $$\overline{e}'' \neq \overline{1} \quad \text{and} \quad \mathcal{C}\left(\overline{1}, t_1\right) = a^{\overline{1}} = \mathcal{C}\left(\overline{e}'', \langle \overline{s}, \overline{1} \rangle\right) \ .$$

   Then $\mathcal{E}$ has found the witness $(a^{\overline{1}}, \overline{1}, t_1, \overline{e}'', \langle \overline{s}, \overline{1} \rangle)$ of the commitment relation $\mathscr{R}_{com}$.
3. If M satisfies $M\overline{1} = \overline{1}$, but is not a permutation matrix, then $\mathcal{E}$ invokes $\mathcal{E}_\pi$ with the additional input M to find $(\overline{e}, t, \overline{e}', k)$ such that $\mathcal{C}(\overline{e}', k) = a^{\overline{e}}$ and Equation (1) holds. Define $\overline{e}'' = M\overline{e}$ and note that

   $$\overline{e}'' \neq \overline{e}' \quad \text{and} \quad \mathcal{C}(\overline{e}', k) = a^{\overline{e}} = \mathcal{C}(\overline{e}'', \langle \overline{s}, \overline{e} \rangle) \ .$$

   The former holds, since $\prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i \neq \prod_{i=1}^{N} e''_i$. Then $\mathcal{E}$ has found the witness $(a^{\overline{e}}, \overline{e}', k, \overline{e}'', \langle \overline{s}, \overline{e} \rangle)$ of the commitment relation $\mathscr{R}_{com}$.

Note that the the expected running time of the extractor $\mathcal{E}$ is bounded by $\mathcal{O}\big((NT + T')/(\delta - \frac{N}{|S|} - \epsilon)\big)$ as required and that it always finds a witness of either $\mathscr{R}_\pi$ or $\mathscr{R}_{com}$. □

4 Proof of Knowledge of Restricted Permutation Matrix

We now detail how one can restrict π to the subset S_F of permutations that satisfies $F(\overline{x}'_1, \ldots, \overline{x}'_d) = F(\overline{x}_1, \ldots, \overline{x}_d)$ for a multivariate polynomial $F(\overline{x}_1, \ldots, \overline{x}_d)$ in $\mathbb{Z}_q[\overline{x}_1, \ldots, \overline{x}_d]$ where $\overline{x}'_i = (x_{i,\pi(1)}, \ldots, x_{i,\pi(N)})$. We remark that $d = 1$ in many instances, in which case the polynomial F will just depend on a single list of N variables.

The protocol below is an extension of Protocol 1. Thus, to simplify the exposition we denote by $\mathsf{P}_\pi(a, t, \overline{e}, \overline{e}', k)$ the predicate used to define the sigma proof of Protocol 1, i.e. $\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} \wedge \mathcal{C}\left(\overline{e}', k\right) = a^{\overline{e}} \wedge \prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i$.

Protocol 2 (Restricted Permutation Matrix).
Common Input: Matrix commitment $a \in G_q^N$, commitment parameters $g, g_1, \ldots, g_N \in G_q$, and a polynomial invariant F.
Private Input: Permutation matrix $M \in \mathbb{Z}_q^{N \times N}$ for a permutation $\pi \in S_F$ and randomness $\overline{s} \in \mathbb{Z}_q^N$ such that $a = \mathcal{C}(M, \overline{s})$.

1. $\mathcal{V}$ chooses $\overline{e}_1, \ldots, \overline{e}_d \in S^N \subseteq \mathbb{Z}_q^N$ randomly and hands $\overline{e}_1, \ldots, \overline{e}_d$ to $\mathcal{P}$.
2. $\mathcal{P}$ defines $t = \langle \overline{1}, \overline{s} \rangle$ and $k_\iota = \langle \overline{s}, \overline{e}_\iota \rangle$ for $\iota = 1, \ldots, d$. Then $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[\begin{array}{c|c} \overline{e}'_1 \in \mathbb{Z}_q^N, t, k_1 \in \mathbb{Z}_q & \mathsf{P}_\pi(a, t, \overline{e}_1, \overline{e}'_1, k_1) = 1 \\ \overline{e}'_2, \ldots, \overline{e}'_d \in \mathbb{Z}_q^N & \bigwedge_{j=1}^{d} \mathcal{C}\left(\overline{e}'_j, k_j\right) = a^{\overline{e}_j} \\ k_2, \ldots, k_d \in \mathbb{Z}_q & \bigwedge F(\overline{e}'_1, \ldots, \overline{e}'_d) = F(\overline{e}_1, \ldots, \overline{e}_d) \end{array}\right] .$$

Proposition 2. *Protocol 2 is a perfectly complete 4-message honest verifier zero-knowledge proof of knowledge of the relation $\mathscr{R}_{\mathscr{G}} \vee \mathscr{R}_{com}$, where the relation $\mathscr{R}_{\mathscr{G}}$ consists of pairs $((ck, a, F), (M, \overline{s}))$ such that M is a permutation matrix of $\pi \in S_F$ and $a = \mathcal{C}_{ck}(M, \overline{s})$.*

The proof, given in Appendix A.1, is a slight modification of the proof of Proposition 1.

4.1 Encoding Graphs As Polynomials

So far, we have not discussed where the polynomials would come from. In this section we describe how a graph can be encoded as a polynomial which is invariant under automorphisms of the graph.

The edge set E of an undirected graph $\mathscr{G}$ with N vertices can be encoded by the polynomial $F_{\mathscr{G}}(\overline{x}) = \sum_{(i,j)\in E} x_i x_j$ where $\overline{x}$ is a list of N independent variables. This encoding is generalized in the natural way to a hypergraph with edge set E by defining $F_{\mathscr{G}}(\overline{x}) = \sum_{e\in E} \prod_{i\in e} x_i$. Both encodings allow multiple edges and self-loops.

Notice that the encoding above does not preserve information about the direction of the edges. For directed graphs, we instead introduce new variables $y_1, \ldots, y_N$ and use the polynomial $F_{\mathscr{G}}(\overline{x}, \overline{y}) = \sum_{(i,j)\in E} x_i y_j$, where x_i and y_i represent the origin and destination of a directed edge from and to vertex i

respectively. For example, the cyclic group C_N of rotations of N elements arise as the automorphism group of the directed cyclic graph on N vertices. This graph can be encoded by the polynomial $F_{\mathscr{G}}(\overline{x}, \overline{y}) = \sum_{(i,j)\in E} x_i y_j$ so we can use a proof of a restricted shuffle to show that one list of ciphertexts is a rotation of another.

This trick of adding more variables can be generalized to encode the order of the vertices in the edges of a hypergraph.

Theorem 2. *Let $F_{\mathscr{G}}(\overline{x}_1, \ldots, \overline{x}_d)$ be the encoding polynomial of a (directed or undirected) graph or hypergraph $\mathscr{G}$. A permutation π is an automorphism of $\mathscr{G}$ if and only if*

$$F_{\mathscr{G}}(\overline{x}_1, \ldots, \overline{x}_d) = F_{\mathscr{G}}(\overline{x}'_1, \ldots, \overline{x}'_d) \ ,$$

where $\overline{x}'_i = (x_{i,\pi(1)}, \ldots, x_{i,\pi(N)})$.

Proof. Recall that an automorphism is a permutation of the vertices which maps edges to edges. Since $F_{\mathscr{G}}$ is an encoding of the edge set and the edge sets are equal if and only if the permutation is an automorphism, it follows that the encoding polynomials are equal if and only if the permutation is an automorphism. □

5 Proofs of Restricted Shuffles

We immediately get an 8-message proof of a restricted shuffle for any shuffle-friendly map and any homomorphic cryptosystem by combining our result with the protocol in [27]. Here we give a 5-message proof of a restricted shuffle for the important special case where each element in the groups of ciphertexts and randomness have prime order q, e.g. El Gamal [8].

Recall the definition of shuffle-friendly maps from [27], where we use $\mathcal{C}_{pk}$ and $\mathcal{R}_{pk}$ to denote the groups of ciphertexts and randomness for a public key pk.

Definition 1. *A map ϕ_{pk} is shuffle-friendly for a public key $pk \in \mathcal{PK}$ of a homomorphic cryptosystem if it defines a homomorphic map $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$.*

For example, the shuffle-friendly map[1] of a shuffle where the ciphertexts are re-encrypted and permuted is defined by $\phi_{pk}(c, r) = c \cdot \mathsf{E}_{pk}(1, r)$. All the known shuffles of homomorphic ciphertexts or lists of ciphertexts can be expressed similarly (see [27] for more examples). We let $\mathsf{P}_F(a, t, \{\overline{e}_\iota, k_\iota, \overline{e}'_\iota\}_{\iota=1}^d)$ denote the predicate $\mathsf{P}_\pi(a, t, \overline{e}_1, \overline{e}'_1, k_1) \wedge F(\overline{e}'_1, \ldots, \overline{e}'_d) = F(\overline{e}_1, \ldots, \overline{e}_d) \wedge \mathcal{C}(\overline{e}'_j, k_j) = a^{\overline{e}_j}$ for all j, i.e the predicate that was used to define the sigma proof in Protocol 2.

Protocol 3 (Proof of Restricted Shuffle).
COMMON INPUT: Commitment parameters $g, g_1, \ldots, g_N \in G_q$, a polynomial F, public key pk, and ciphertexts $c_1, \ldots, c_N, c'_1, \ldots, c'_N \in \mathcal{C}_{pk}$.

[1] We remark that nothing prevents proving a shuffle of other objects than ciphertexts, i.e., any groups $\mathcal{C}_{pk}$ and $\mathcal{R}_{pk}$ of prime order q, and any homomorphic map ϕ_{pk} defined by some public parameter pk can be used.

PRIVATE INPUT: A permutation $\pi \in S_F$ and randomness $\overline{r} \in \mathcal{R}_{pk}^N$ such that $c'_i = \phi_{pk}\left(c_{\pi(i)}, r_{\pi(i)}\right)$.

1. Let $M \in \mathbb{Z}_q^{N \times N}$ be the permutation matrix representing π. $\mathcal{P}$ chooses $\overline{s} \in \mathbb{Z}_q^N$ randomly, computes $a = \mathcal{C}\left(M, \overline{s}\right)$, and hands a to $\mathcal{V}$.
2. $\mathcal{V}$ chooses $\overline{e}_1, \ldots, \overline{e}_d \in S^N \subseteq \mathbb{Z}_q^N$ randomly and hands $\overline{e}_1, \ldots, \overline{e}_d$ to $\mathcal{P}$.
3. $\mathcal{P}$ defines $\overline{e}'_\iota = M\overline{e}_\iota$, $t = \langle \overline{1}, \overline{s} \rangle$, $k_\iota = \langle \overline{s}, \overline{e}_\iota \rangle$ for $\iota = 1, \ldots, d$, and $u = \langle \overline{r}, \overline{e}_1 \rangle$. Then $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[\begin{array}{c} \{\overline{e}'_\iota \in \mathbb{Z}_q^N, k_\iota \in \mathbb{Z}_q\}_{\iota=1}^d \\ t \in \mathbb{Z}_q, u \in \mathcal{R}_{pk} \end{array} \middle| \begin{array}{c} \mathsf{P}_F(a, t, \{\overline{e}_\iota, k_\iota, \overline{e}'_\iota\}_{\iota=1}^d) = 1 \\ \prod_{i=1}^N (c'_i)^{e'_{1,i}} = \phi_{pk}\left(\prod_{i=1}^N c_i^{e_{1,i}}, u\right) \end{array} \right] .$$

In Appendix B we give a concrete instantiation of the above sigma proof for an unrestricted shuffle which has efficiency comparable to some of the most efficient proofs of a shuffle in the literature. We also explain how a shuffle of a complete binary tree can be derived with essentially no additional computational cost.

Proposition 3. *Protocol 3 is a perfectly complete 5-message honest verifier zero-knowledge proof of knowledge of the relation $\mathscr{R}_{\phi_{pk}} \vee \mathscr{R}_{com}$, where $\mathscr{R}_{\phi_{pk}}$ consists of pairs $((ck, a, F, pk, \overline{c}, \overline{c}'), (M, \overline{s}, \overline{r}))$ such that M is a permutation matrix of $\pi \in S_F$, $a = \mathcal{C}_{ck}\left(M, \overline{s}\right)$ and $c'_i = \phi_{pk}\left(c_{\pi(i)}, r_{\pi(i)}\right)$.*

A proof of the proposition is given in Appendix A.2

6 Variations and Generalizations

There are many natural variations and generalizations of our approach. Below we briefly mention some of these.

An alternative encoding of a graph is found by switching the roles of multiplication and addition in the encoding polynomial, e.g., the encoding polynomial of an undirected graph $\mathscr{G}$ could be defined as $F_{\mathscr{G}}(x_1, \ldots, x_N) = \prod_{(i,j)\in E}(x_i + x_j)$. Direction of edges can also be represented in an alternative way using powers to distinguish the ends of each edge, e.g, given a directed graph $\mathscr{G}$ the encoding polynomial could be defined by $F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j)\in E} x_i x_j^2$. We can combine these ideas, turning exponentiation into multiplication by a scalar, and get the encoding polynomial $F_{\mathscr{G}}(x_1, \ldots, x_N) = \prod_{(i,j)\in E}(x_i + 2x_j + 1)$ for our directed graph. The additive constant 1 is needed to fix the scalar multiple of each factor since factorization in the ring $\mathbb{Z}_q[x_1, \ldots, x_N]$ is only unique up to units. The same ideas can be extended to hypergraphs and oriented hypergraphs, i.e. hypergraphs where the edges are ordered tuples rather than sets of vertices.

It is easy to generalize the protocol to proving that $f(x_{\pi(1)}, \ldots, x_{\pi(N)}) = g(x_1, \ldots, x_N)$ for an arbitrary function g. In the body of the paper, we dealt with the important special case where $f = g$, but by choosing g different from f,

it is possible to prove that the permutation belong to a set that is not necessarily a group. For example, one can prove that the permutation is odd by choosing

$$f(x_1, \ldots, x_N) = \prod_{i<j}(x_i - x_j) \quad \text{and} \quad g(x_1, \ldots, x_N) = -\prod_{i<j}(x_i - x_j) \ .$$

However, it will generally not be possible to create a chain of mix-servers unless the permutation is restricted to a set that is closed under composition.

For utmost generality, one can easily modify the protocol to prove that $f(x_1, \ldots, x_N, x_{\pi(1)}, \ldots, x_{\pi(N)}) = 0$ or even $f(x_1, \ldots, x_N, x_{\pi(1)}, \ldots, x_{\pi(N)}) \neq 0$ for any function f that can be computed verifiably. Given a commitment $y = \mathcal{C}(b, s)$, the prover can demonstrate that $b \neq 0$ by computing $t = 1/b$, $z = g^{s'} y^t$ and running a sigma proof of the form

$$\Sigma\text{-}\mathsf{proof}\left[r, t, s' \,\middle|\, z = g^{s'} y^t \wedge z/g_1 = g^r\right] \ .$$

As an example, one can prove that a permutation is a derangement, i.e. that $\pi(i) \neq i$ for all i by verifying $\prod_{i=1}^{N}(x_{\pi(i)} - x_i) \neq 0$.

In our exposition we assume for clarity that q is prime, but this is not essential. Composite q can be used, but this requires a more delicate analysis to handle the possibility of non-invertible elements and zero-divisors in the ring $\mathbb{Z}_q$, e.g., the random vectors are no longer vectors, but elements in a module. Even the case where q is unknown can be handled using an approach similar to that of [27].

7 Acknowledgements

We thank Johan Håstad for helpful discussions.

References

1. M. Abe and H. Imai. Flaws in some robust optimistic mix-nets. In *Australasian Conference on Information Security and Privacy – ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 39–50. Springer Verlag, 2003.
2. B. Adida and D. Wikström. How to shuffle in public. In *4th Theory of Cryptography Conference (TCC)*, volume 4392 of *Lecture Notes in Computer Science*, pages 555–574. Springer Verlag, 2007.
3. B. Adida and D. Wikström. Offline/online mixing. In *34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *Lecture Notes in Computer Science*, pages 484–495. Springer Verlag, 2007.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE Computer Society Press, 2001. (Full version at Cryptology ePrint Archive, Report 2000/067, `http://eprint.iacr.org`, October, 2001.).
5. D. Chaum. Untraceable electronic mail, return addresses and digital pseudo-nyms. *Communications of the ACM*, 24(2):84–88, 1981.

6. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer Verlag, 1997.
7. S. de Hoogh, B. Schoenmakers, B. Skoric, and J. Villegas. Verifiable rotation of homomorphic encryptions. In *Public Key Cryptography – PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 393–410. Springer Verlag, 2009.
8. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
9. J. Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE Transactions*, 88-A(1):172–188, 2005.
10. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 2002.
11. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer Verlag, 2001.
12. J. Furukawa and K. Sako. An efficient publicly verifiable mix-net for long inputs. In *Financial Cryptography 2006*, volume 4107 of *Lecture Notes in Computer Science*, pages 111–125. Springer Verlag, 2006.
13. J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160. Springer Verlag, 2003.
14. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In *Advances in Cryptology – Crypto2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 192–208. Springer Verlag, 2009.
15. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology – Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 379–396. Springer Verlag, 2008.
16. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *Advances in Cryptology – Asiacrypt 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177. Springer Verlag, 2000.
17. A. Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security (CCS)*, pages 116–125. ACM Press, 2001.
18. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology – Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 248–259. Springer Verlag, 1994.
19. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Verlag, 1992.
20. B. Pfitzmann. Breaking an efficient anonymous channel. In *Advances in Cryptology – Eurocrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 332–340. Springer Verlag, 1995.
21. T. I. Reistad and T. Toft. Secret sharing comparison by transformation and rotation. In *Information Theoretic Security (ICITS 2007)*, volume 4883 of *Lecture Notes in Computer Science*, pages 169–180. Springer Verlag, 2009.
22. M. K. Reiter and X. Wang. Fragile mixing. In *11th ACM Conference on Computer and Communications Security (CCS)*, pages 227–235. ACM Press, 2004.

23. P. Y. A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. In *11th European Symposium on Research in Computer Security (ESORICS)*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326. Springer Verlag, 2006.
24. K. Sako and J. Killian. Reciept-free mix-type voting scheme. In *Advances in Cryptology – Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer Verlag, 1995.
25. D. Wikström. A universally composable mix-net. In *1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 315–335. Springer Verlag, 2004.
26. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *Advances in Cryptology – Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 273–292. Springer Verlag, 2005.
27. D. Wikström. A commitment-consistent proof of a shuffle. In *ACISP*, volume 5594 of *Lecture Notes in Computer Science*, pages 407–421. Springer Verlag, 2009.

A Omitted Proofs

A.1 Proof of Proposition 2

The completeness and zero-knowledge properties follows immediately from the completeness and zero-knowledge property of the sigma proof and the hiding property of the commitment scheme.

The soundness and proof of knowledge properties follows by modifications of the proof of Proposition 1. We first replace the basic extractor. Then we consider an additional extractor for finding polynomial inequalities in the case where the extracted matrix M does not belong to the required set of permutation matrices. Finally, we add an additional step to the original main extractor.

Three-Message Protocol. We redefine the prover $\mathcal{P}_+$ such that it chooses all of $\overline{e}_1, \ldots, \overline{e}_d \in S^N$ (instead of letting the verifier choose them), and redefine $\mathcal{V}_+$ to accept $\overline{e}_1, \ldots, \overline{e}_d$ as part of the first message from the prover. We may then view the prover $\mathcal{P}_\pi$ and verifier $\mathcal{V}_\pi$ from the proof of Proposition 1 as modifications to the new $\mathcal{P}_+$ and $\mathcal{V}_+$ in the obvious way (where they use the components of the protocol already present in Protocol 1), and correspondingly for the extractor $\mathcal{E}_\pi$ which has the same properties as the original one (possibly by changing the polynomial $T(n)$ and the negligible error ϵ).

Basic Extractor. We augment the original input with several lists of linearly independent vectors $\overline{e}_{\iota,1}, \ldots, \overline{e}_{\iota,l} \in \mathbb{Z}_q^N$, for $\iota = 1, \ldots, d$ and $l < N$. Then we redefine $\mathcal{V}_\perp$ to only accept if the ιth random vector $\overline{e}_\iota$ chosen in an execution of the protocol is linearly independent of the vectors $\overline{e}_{\iota,1}, \ldots, \overline{e}_{\iota,l}$. We argue similarly as in the proof of Proposition 1. If $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability δ, then by the union bound $\mathcal{P}_\perp$ convinces $\mathcal{V}_\perp$ with probability at least $\delta - \frac{d+1}{|S|}$, and there exists an extractor $\mathcal{E}_\perp$ that given the lists of linearly independent vectors extracts:

1. vectors $\overline{e}_1, \ldots, \overline{e}_d$ such that every $\overline{e}_\iota$ is linearly independent of $\overline{e}_{\iota,1}, \ldots, \overline{e}_{\iota,l}$

2. a witness $(t, \overline{e}'_1, k_1)$ and $(\overline{e}'_\iota, k_\iota)$ for $\iota = 1, \ldots, d$ of the corresponding sigma proof.

Moreover, the extractor $\mathcal{E}_\perp$ runs in expected time $T/(\delta - \frac{d+1}{|S|} - \epsilon)$ for a polynomial $T(n)$. Then we redefine $\mathcal{E}_N$ such that it computes

$$(\{\overline{e}_{\iota,l}\}_{\iota=1}^d, t_l, \{\overline{e}'_{\iota,l}, k_{\iota,l}\}_{\iota=1}^d) = \mathcal{E}_\perp(\{\overline{e}_{\iota,1}, \ldots, \overline{e}_{\iota,l-1}\}_{\iota=1}^d, a, g, g_1, \ldots, g_N)$$

for $l = 1, \ldots, N$. Clearly, $\mathcal{E}_N$ runs in expected time $\mathcal{O}\big(NT/(\delta - \frac{d+1}{|S|} - \epsilon)\big)$.

Polynomial Inequality Extractor. Suppose that $M \in \mathbb{Z}_q^{N \times N}$ is a permutation matrix, but the permutation does not leave F invariant. We augment the common input with a permutation matrix M with this property, let $\mathcal{P}_F$ be identical to $\mathcal{P}_+$, and define a new verifier $\mathcal{V}_F$ to be identical to $\mathcal{V}_+$ except that it only accepts if

$$F(M\overline{e}_1, \ldots, M\overline{e}_d) \neq F(\overline{e}_1, \ldots, \overline{e}_d) \ . \tag{2}$$

From Schwartz-Zippel's lemma, we know that the probability that the additional requirement is not satisfied is at most $\Delta_F = \deg(F)/|S|$. Thus, if $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability δ, then $\mathcal{P}_F$ convinces $\mathcal{V}_F$ with probability $\delta - \Delta_F$. Again, this implies that there exists an extractor $\mathcal{E}_F$ running in expected time $T''/(\delta - \Delta_F - \epsilon)$ for some polynomial T'' that outputs a witness $(\{e_\iota\}_{\iota=1}^d, t, \{\overline{e}'_\iota, k_\iota\}_{\iota=1}^d)$ satisfying Equation (2).

Main Extractor. The new main extractor $\mathcal{E}$ is syntactically defined as the original main extractor in the proof of Proposition 1, except that it invokes the redefined extractors $\mathcal{E}_\perp$ and $\mathcal{E}_\pi$ described above, and the following step is added.

4. If M is a permutation matrix, but the permutation does not leave F invariant, then $\mathcal{E}$ invokes $\mathcal{E}_F$ with the additional input M to find $(\{e_\iota\}_{\iota=1}^d, t, \{\overline{e}'_\iota, k_\iota\}_{\iota=1}^d)$ satisfying Equation (2). Define $\overline{e}''_\iota = M\overline{e}'_\iota$ and note that there exist a $1 \leq \iota \leq d$ such that

$$\overline{e}''_\iota \neq \overline{e}'_\iota \quad \text{and} \quad \mathcal{C}\left(\overline{e}'_\iota, k_\iota\right) = a^{\overline{e}_\iota} = \mathcal{C}\left(\overline{e}''_\iota, \langle \overline{s}, \overline{e}'_\iota \rangle\right) \ .$$

The former holds, since $\prod_{i=1}^N e'_{\iota,i} = \prod_{i=1}^N e_{\iota,i} \neq \prod_{i=1}^N e''_{\iota,i}$. Then $\mathcal{E}$ has found the witness $(a^{\overline{e}_\iota}, \overline{e}'_\iota, k_\iota, \overline{e}''_\iota, \langle \overline{s}, \overline{e}'_\iota \rangle)$ of the commitment relation $\mathscr{R}_{com}$.

Note that the extractor always finds a witness of either $\mathscr{R}_{\mathcal{G}}$ or $\mathscr{R}_{com}$ and that the expected running time of $\mathcal{E}$ is bounded by $\mathcal{O}\big((NT + T' + T'')/(\delta - \Delta_F - \frac{N}{|S|} - \epsilon)\big)$ as required.

A.2 Proof of Proposition 3

The completeness follows from the completeness of the sigma proof. The zero-knowledge property follows from the perfect hiding property of the commitment

scheme and from the zero-knowledge property of the sigma proof. We consider a prover $\mathcal{P}^*$ which convinces $\mathcal{V}$ with probability δ. First note that Step 2-3 is an execution of Protocol 2, except that there is an additional requirement involving the ciphertexts. It turns out that, when a witness of $\mathscr{R}_\pi$ is found, all of the extracted values used to compute the matrix M are useful and not only the matrix M and the randomness $\overline{s}$. To simplify notation in the next section we write $\overline{w} = \overline{e}_1$ and $\overline{w}' = \overline{e}'_1$. Thus, we assume that there is an extended extractor $\mathcal{E}_{ext}$ with some negligible knowledge error ϵ that outputs linearly independent vectors $\overline{w}_1, \ldots, \overline{w}_N$ in $\mathbb{Z}_q^N$ along with the matrix M and the randomness $\overline{s}$. We may also assume that $\overline{w}'_j = M\overline{w}_j$, i.e., that $w'_{j,i} = w_{j,\pi(i)}$, where M is the permutation matrix of π. Let T_{ext} be a polynomial such that $\frac{T_{ext}}{\delta(a)-\epsilon}$ bounds the expected running time of $\mathcal{E}_{ext}$, when the prover convinces the verifier with probability $\delta(a)$ depending on the commitment a. Let δ be the probability that the prover $\mathcal{P}^*$ convinces $\mathcal{V}$ in the full protocol.

Extractor. The extractor $\mathcal{E}$ of Protocol 3 proceeds as follows:

1. Sample at most $\frac{2}{\delta}$ interactions between $\mathcal{P}^*$ and $\mathcal{V}$ and halt if no accepting transcript is found. Otherwise let a denote the commitment used in Step 1 of the accepting interaction.
2. Fix the commitment a and run the extractor $\mathcal{E}_{ext}$ for at most $\frac{4T_{ext}}{\delta-2\epsilon}$ time steps, or until it outputs:
   (a) a permutation matrix M leaving the polynomial F invariant,
   (b) randomness $\overline{s}$ such that $a = \mathcal{C}(M, \overline{s})$, and
   (c) linearly independent vectors $\overline{w}_1, \ldots, \overline{w}_N$ in $\mathbb{Z}_q^N$ such that for $j = 1, \ldots, N$

$$\prod_{i=1}^N (c'_i)^{w_{j,\pi(i)}} = \phi_{pk}\left(\prod_{i=1}^N c_i^{w_{j,i}}, u_j\right) \ .$$

If the extractor $\mathcal{E}_{ext}$ succeeds, we know from linear independence that there exist $\alpha_{l,j}$ such that $\sum_{j=1}^N \alpha_{l,j}\overline{w}_j$ is the lth standard unit vector in $\mathbb{Z}_q^N$. We conclude that

$$\begin{aligned} c'_{\pi^{-1}(l)} &= \prod_{j=1}^N \left(\prod_{i=1}^N (c'_i)^{w_{j,\pi(i)}}\right)^{\alpha_{l,j}} = \prod_{j=1}^N \left(\phi_{pk}\left(\prod_{i=1}^N c_i^{w_{j,i}}, u_j\right)\right)^{\alpha_{l,j}} \\ &= \phi_{pk}\left(\prod_{j=1}^N \left(\prod_{i=1}^N c_i^{w_{j,i}}\right)^{\alpha_{l,j}}, \sum_{j=1}^N \alpha_{l,j}u_j\right) = \phi_{pk}\left(c_l, \sum_{j=1}^N \alpha_{l,j}u_j\right) \ , \end{aligned}$$

where we use the homomorphic property of ϕ_{pk} in the third equality. Thus, the extractor has found a permutation π satisfying the required polynomial identity, and randomness $r_l = \sum_{j=1}^N \alpha_{l,j}u_j$ such that $c'_l = \phi_{pk}\left(c_{\pi(l)}, r_{\pi(l)}\right)$.

Analysis of the extractor. It is clear that the extractor runs in time

$$\frac{2T_{prot}}{\delta} + \frac{4T_{ext}}{\delta - 2\epsilon} \leq \frac{2T_{prot} + 4T_{ext}}{\delta - 2\epsilon}$$

where T_{prot} is the time it takes to run the protocol once.

We must now investigate the probability with which the extractor succeeds. Observe that the expected number of interactions to find the first accepting transcript is δ^{-1}. By Markov's inequality, one will be found with probability $\frac{1}{2}$ in $\frac{2}{\delta}$ interactions.

Let *Accept* denote the event that the verifier accepts, and define a set S of good first messages by

$$S = \{a : \Pr[Accept \,|a] \geq \delta/2\}$$

Note that $\Pr[a \in S \,|Accept] \geq \frac{1}{2}$, so if an accepting transcript is found in Step 1 we get a good a with probability at least $\frac{1}{2}$. If we obtain a good first message, then the extractor $\mathcal{E}_{ext}$ will find the desired accepting inputs in expected time at most $\frac{T_{ext}}{\delta/2-\epsilon}$, so by Markov's inequality, it succeeds with probability $\frac{1}{2}$ in time $\frac{4T_{ext}}{\delta-2\epsilon}$. To summarize, the extractor runs in time $\frac{2T_{prot}+4T_{ext}}{\delta-2\epsilon}$ and extracts the desired quantities with probability $\frac{1}{8}$. This can be converted to an extractor that finds the permutation and randomness in expected time

$$\frac{16T_{prot} + 32T_{ext}}{\delta - 2\epsilon}$$

by repeating until it succeeds.

B A Concrete Proof of a Shuffle

To make our proof of a shuffle concrete and allow us to estimate its complexity we instantiate the Sigma-proof we need. We first consider the Sigma-proof we need in Protocol 1.

B.1 Concrete Proof of Knowledge of Permutation Matrix

For concreteness we let n_v, n_c, and n_r denote the bitsize of components in random vectors, challenges, and random paddings respectively. For these additional security parameters 2^{-n_v}, 2^{-n_c}, and 2^{-n_r} must be negligible in n. In practice, n_v and n_c govern the soundness of the protocols below and n_r govern the statistical zero-knowledge property. Given are $a = \mathcal{C}(M, \overline{s})$, $\overline{s} \in \mathbb{Z}_q^N$, and $\overline{e} \in S^N$ and we instantiate the needed sigma-proof below.

Protocol 4. An instantiation of the following sigma-proof.

$$\Sigma\text{-proof}\left[\begin{array}{c} \overline{e}' \in \mathbb{Z}_q^N \\ t, k, z \in \mathbb{Z}_q \end{array} \middle| \begin{array}{c} \left(\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} \wedge \mathcal{C}\left(\overline{e}', k\right) = a^{\overline{e}} \wedge \prod_{i=1}^{N} e_i' = \prod_{i=1}^{N} e_i\right) \\ \vee g_1 = g^z \end{array}\right]$$

1. $\mathcal{P}$ chooses $\overline{r}, \overline{s} \in \mathbb{Z}_q^N$, $s_\alpha \in \mathbb{Z}_q$, $s' \in [0, 2^{n_v+n_r+n_c}-1]^N$, $s_\gamma, s_\delta \in \mathbb{Z}_q$ and hands the following values to $\mathcal{V}$, where we set $B_0 = g_1$,

$$B_i = g^{r_i} B_{i-1}^{e_i'} \ , \ \alpha = g^{s_\alpha} \prod_{i=1}^{N} g_i^{s_i'} \ , \ \beta_i = g^{s_i} B_{i-1}^{s_i'} \ , \ \gamma = g^{s_\gamma} \ , \ \delta = g^{s_\delta} \ .$$

2. $\mathcal{V}$ chooses a challenge $c \in [0, 2^{n_c}-1]$ at random and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ responds with

$$d_\alpha = ck + s_\alpha \bmod q \ , \ d_i' = ce_i' + s_i' \ , \ d_i = cr_i + s_i \bmod q \ ,$$
$$d_\gamma = c\langle \overline{s}, \overline{1}\rangle + s_\gamma \bmod q \ , \text{ and } \ d_\delta = ce_N'' + s_\delta \bmod q \ ,$$

   where $e_1'' = s_1$ and $e_i'' = e_{i-1}'' e_i' + s_i \bmod q$ for $i > 1$.
4. $\mathcal{V}$ accepts if and only if

$$(a^{\overline{e}})^c \alpha = g^{d_\alpha} \prod_{i=1}^{N} g_i^{d_i'} \ , \ B_i^c \beta_i = g^{d_i} B_{i-1}^{d_i'} \ , \tag{3}$$

$$\left(a^{\overline{1}} \Big/ \prod_{i=1}^{N} g_i\right)^c \gamma = g^{d_\gamma} \ , \text{ and } \ \left(B_N \Big/ g_1^{\prod_{i=1}^{N} e_i}\right)^c \delta = g^{d_\delta} \ . \tag{4}$$

Proposition 4. *Protocol 4 is perfectly complete, special sound, and statistical special zero-knowledge.*

Proof. The proof is standard. The special zero-knowledge simulator chooses $B_1, \ldots, B_N \in G_q$, and $\overline{d}, \overline{d}' \in \mathbb{Z}_q^N$, and $d_\alpha, d_\gamma, d_\delta \in \mathbb{Z}_q$ randomly and define α, β_i, γ, and δ by Equation (3) and (4). The statistical distance between a simulated transcript and a real transcript is $\mathcal{O}(2^{-n_r})$, which is small if n_r is chosen appropriately. From two accepting transcripts

$$\left((B_i)_{i=1}^N, \alpha, (\beta_i)_{i=1}^N, \gamma, \delta, c, \overline{d}, \overline{d}', d_\alpha, d_\gamma, d_\delta\right) \text{ and}$$
$$\left((B_i)_{i=1}^N, \alpha, (\beta_i)_{i=1}^N, \gamma, \delta, c^*, \overline{d}^*, \overline{d}'^*, d_\alpha^*, d_\gamma^*, d_\delta^*\right) \text{ and}$$

with $c \neq c^*$ we define $\overline{e}' = (\overline{d}' - \overline{d}'^*)/(c - c^*)$, $\overline{r} = (\overline{d} - \overline{d}^*)/(c - c^*)$, $k = (d_\alpha - d_\alpha^*)/(c - c^*)$, $t = (d_\gamma - d_\gamma^*)/(c - c^*)$, $k_\delta = (d_\delta - d_\delta^*)/(c - c^*)$, and conclude that

$$\mathcal{C}\left(\overline{e}', k\right) = a^{\overline{e}} \ , \ B_i = g^{r_i} B_{i-1}^{e_i'} \ , \ \mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} \ , \text{ and } \ B_N = g^{k_\delta} g_1^{\prod_{i=1}^{N} e_i} \ .$$

By solving the recursive equation involving B_i, we conclude that either $\prod_{i=1}^{N} e_i' = \prod_{i=1}^{N} e_i$, or we have found the discrete logarithm z such that $g_1 = g^z$.

The prover computes roughly $2N$ exponentiations with exponents in $\mathbb{Z}_q$, but they all use the same basis element g. Fixed base exponentiation techniques can be applied directly to reduce this by a fairly large constant factor compared to square-and-multiply exponentiation, say a factor $1/8$, provided that $N > 1000$. The prover also computes N exponentiations with exponents of bitsize n_v and $2N$ exponentiations with bitsize $n_v + n_c + n_r$ respectively. This corresponds to $\frac{3n_v+2n_c+2n_r}{n}$ exponentiations with exponents in $\mathbb{Z}_q$. If we use $n_v = n_c = n_r = 80$ and $n = 1024$, this corresponds to computing roughly $N/2$ exponentiations. This can be further reduced at least by a factor of 3 using simultaneous exponentiation. Thus, we estimate the complexity of the prover to correspond to at most $\frac{1}{6}N$ square-and-multiply exponentiations for practical parameters. The complexity of the verifier is estimated to $\frac{1}{6}N$ as well, by noting that, provided $n_c = n_v$, we may match exponentiations by s'_i, s_i, and e'_i performed by the prover by exponentiations by d'_i, d_i, and c performed by the verifier.

B.2 Concrete Proof of a Shuffle

To turn Protocol 4 into a complete proof of a shuffle, we can apply the protocol of [27] directly or interlace it with Protocol 4 to avoid increasing the number of rounds. From special statistical zero-knowledge, it is easy to see that we may also reuse the random vector and challenge. The complexity of the prover and verifier in the protocol in [27] amounts to N exponentiations by $(n_v + n_r + n_c)$-bit exponents in the group G_q and $\mathcal{M}_{pk}$ respectively. Thus, the complexity depends on the underlying cryptosystem. For El Gamal over G_q, with the parameters above, and using simultaneous exponentiation, this amounts to roughly $\frac{1}{4}N$ square-and-multiply exponentiations. Thus, the total complexity with these parameters is roughly $\frac{2}{5}N$ square-and-multiply exponentiations, which is almost as fast as the protocol in [27] which requires precomputation. Below we sketch the protocol given in [27]. Let $\{h_1, \ldots, h_t\}$ be a generator set of the ciphertext space $\mathcal{C}_{pk}$, e.g., for El Gamal over G_q, this would be $\{(1, g), (g, 1)\}$. The prover hands

$$C_0 = \mathcal{C}\left(l_1, \ldots, l_t, l\right) \text{ and } C_1 = \prod_{j=1}^{t} h_j^{l_j} \prod_{i=1}^{N} (c'_i)^{e'_i}$$

and proves that these were formed correctly using $\overline{e}'$. This is done using a standard proof of knowledge of multi-logarithms. Then it also proves that it knows some u such that $C_1 = \prod_{j=1}^{t} h_j^{l_j} \phi_{pk}\left(\prod_{i=1}^{N} c_i^{e_{1,i}}, u\right)$. From this we may conclude that $\prod_{i=1}^{N} (c'_i)^{e'_{1,i}} = \phi_{pk}\left(\prod_{i=1}^{N} c_i^{e_{1,i}}, u\right)$, or a witness of the relation $\mathscr{R}_{com}$ can be extracted.

B.3 Proof of Tree Shuffle Without Additional Cost

Note that B_i is a commitment of $\prod_{j=1}^{i} e'_j$ in Protocol 4. In other words, the roles of the B_i's is to aggregate the product $\prod_{j=1}^{N} e'_j$. When proving restricted shuffles

it may be advantageous to aggregate the product in a different order, since the partial results can then be reused. Consider the following. Let $T = (V, E_T)$ be a (rooted) complete binary tree on $N = 2^{N_0}$ leaves $L = \{1, \ldots, N\}$ and let S denote the the set of permutations of the leaves L that are consistent with some automorphism of T. Suppose that we wish to shuffle a list of N ciphertexts using a permutation from S. One way to do this is to consider the alternative encoding of S as the automorphism group of a hypergraph $G = (L, E_G)$ with

$$E_G = \{e \subseteq 2^L : e = L(T') \text{ for a complete subtree } T' \text{ of } T\} \ ,$$

where $L(T')$ denotes the set of leaves of T'. Recall that the encoding polynomial of G is then simply $F_{\mathscr{G}}(\overline{x}) = \sum_{f \in E_G} m_f(\overline{x})$, where we write $m_f(\overline{x}) = \prod_{i \in f} x_i$ for the monomial encoding of an edge f. Suppose now that we aggregate the product $\prod_{i=1}^{N} e'_i$ in a tree-like fashion by forming commitments of $m_f(\overline{e})$ for all m_f with $\deg m_f = 2^i$ for $i = 1, \ldots, N_0$. Then we have commitments of $m_f(\overline{e})$ for all monomials of $F_{\mathscr{G}}(\overline{x})$ for free, and we can easily compute a commitment of $F_{\mathscr{G}}(\overline{e})$ by simply multiplying these commitments. Thus, proving such a shuffle can be done essentially without additional cost.