A Commitment-Consistent Proof of a Shuffle[*]

Douglas Wikström
CSC KTH Stockholm, Sweden
dog@csc.kth.se

April 2, 2011

Abstract

We introduce a pre-computation technique that drastically reduces the online computational complexity of mix-nets based on homomorphic cryptosystems. More precisely, we show that there is a permutation commitment scheme that allows a mix-server to: (1) commit to a permutation and efficiently prove knowledge of doing so correctly in the offline phase, and (2) shuffle its input and give an extremely efficient commitment-consistent proof of a shuffle in the online phase.

We prove our result for a general class of shuffle maps that generalize all known types of shuffles, and even allows shuffling ciphertexts of different cryptosystems in parallel.

1 Introduction

Consider a situation where N senders $S_1, \ldots, S_N$ each have some input and wish to compute the sorted list of their inputs without revealing who submitted which message. A trusted party can do this by waiting until all senders have submitted some input, and then sort and output the list of all inputs. A protocol that emulates the trusted party is called a *mix-net* and the parties $M_1, \ldots, M_k$ that execute the protocol are referred to as *mix-servers*. As long as a certain fraction of the mix-servers are honest, the result should be correct and nobody should learn the correspondence between input ciphertexts and output messages. The obvious application for mix-nets is to conduct electronic elections, and this is also one of the applications Chaum [6] had in mind when he introduced mix-nets.

Many constructions of mix-nets are proposed in the literature, but few have provable security properties and many are actually flawed. The basic approach of all mix-nets with provable properties are based on ideas of Sako and Kilian [27]. The first rigorous definition of security was given by Abe and Imai [1], but they did not construct a scheme satisfying their construction. Wikström [29] gives the first definition of a universally composable (UC) mix-net, the first UC-secure construction, and also a more efficient UC-secure scheme [30]. An important building block in the construction of a mix-net is

[*]A conference version of this paper was presented at ACISP 2009.

a so called *proof of a shuffle* that allows the mix-servers to prove that they follow the protocol. The first efficient proofs of shuffles were given by Neff [21] and Furukawa and Sako [14].

1.1 Mix-Nets Based On Homomorphic Cryptosystems

Recall the mix-net of Sako and Kilian [27]. They present their scheme in terms of the El Gamal cryptosystem [15], but the idea works for any homomorphic cryptosystem.

A homomorphic cryptosystem $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ that allows threshold decryption is employed. A cryptosystem is said to be homomorphic if for every public key $pk \in \mathcal{PK}$, the plaintext space $\mathcal{M}_{pk}$, the randomness space $\mathcal{R}_{pk}$, and the ciphertext space $\mathcal{C}_{pk}$ are groups, and for every $m_0, m_1 \in \mathcal{M}_{pk}$ and $r_0, r_1 \in \mathcal{M}_{pk}$: $\mathsf{E}_{pk}(m_0, r_0)\mathsf{E}_{pk}(m_1, r_1) = \mathsf{E}_{pk}(m_0 m_1, r_0 r_1)$. A joint public key pk is generated somehow such that each mix-server holds a secret share of the corresponding secret key sk. Each sender S_i, holding a message m_i, computes a ciphertext $c_{0,i} = \mathsf{E}_{pk}(m_i)$, and then somehow submits it to the mix-servers. The mix-servers then take turns at re-encrypting and permuting these ciphertexts. Let $L_0 = (c_{0,1}, \ldots, c_{0,N})$ be the list of submitted ciphertexts. For $j = 1, \ldots, k$, M_j chooses a permutation π and $r_{j,i} \in \mathcal{R}_{pk}$ randomly, computes $c_{j,i} = c_{j-1,\pi(i)}\mathsf{E}_{pk}(1, r_{j,\pi(i)})$ for $i = 1, \ldots, N$, and then publishes $L_j = (c_{j,1}, \ldots, c_{j,N})$. In other words, each mix-server randomly re-encrypts each ciphertext and then outputs the resulting ciphertexts in random order. Then it proves, using a proof of a shuffle, that it formed L_j from L_{j-1} in this way. Finally, the mix-servers jointly threshold-decrypt L_k and output the resulting list of plaintexts. The idea is that since all mix-servers have randomly permuted the ciphertexts and the cryptosystem is assumed secure, it is infeasible to tell which plaintext corresponds to which original ciphertext in L_0.

The above description is simplified in that the senders submit homomorphic ciphertexts directly, which is not secure [25]. In a provably secure construction, the plaintexts of corrupted senders must be extractable by the simulator without the secret key of the cryptosystem. Until recently, all known submission schemes were either only heuristically secure, or involved costly interaction, but there is now a provably secure solution to this problem for several well known homomorphic cryptosystems [31].

Alternative Constructions. In the scheme of Furukawa et al. [13], each mix-server not only re-encrypts and permutes its input, but also partially decrypts it. As a result, the final list L_k essentially contains the plaintexts and no joint decryption step is needed. In the scheme of Wikström [30], re-encryption is also eliminated entirely, i.e., each mix-server only partially decrypts and permutes its input. In a preliminary unpublished version of Neff [21] a proof of a shuffle for the first type of mix-net is described as well [22]. These schemes have special advantages over the above, but do not lend themselves well to pre-computation, since partial decryption must be done sequentially.

Very few other approaches to constructing mix-nets have any provable security properties [19] and several are actually flawed [1, 10, 28].

1.2 Previous Work On Improving Efficiency

There are more or less obvious techniques that can be used to reduce the computational complexity of a mix-net. If a threshold below k is used for the decryption key, then all mix-servers do not need to take part in the mixing process. In the execution of a public-coin honest verifier proof of knowledge the random challenge of the honest verifier must be generated jointly by the mix-servers, which is costly. But if unpredictability suffices, then longer challenges can be extracted from a random seed using a PRG. Pre-computation can also be used in the coin-flipping protocols. The re-encryption factors can also be pre-computed and batch proof techniques [4] can be used to reduce the complexity of the proofs of correctness needed during joint decryption.

If such optimizations and pre-computations are used, the main computational cost lies in the proofs of shuffles. Thus, most previous work on reducing the complexity, e.g. [21, 14, 18, 13, 30], focus on reducing the complexity of a particular proof of a shuffle. Some parts of these proofs can easily be pre-computed as well.

An alternative approach is used by Adida and Wikström [3], who show that when the number of senders is relatively small, ideas from homomorphic election schemes [5] can be used to construct a mix-net where the online phase only requires decryption of a single ciphertext. The public-key obfuscated shuffle of Adida and Wikström [2] may also be viewed as a form of pre-computation, but their goal is not improved efficiency. In fact, their scheme is quite inefficient.

1.3 Our Contribution

We show how to split a proof of a shuffle into two protocols. The first protocol is used by a mix-server in the offline phase to prove knowledge of how to open a commitment to a permutation. The second protocol is used by a mix-server in the online phase to prove that it uses the permutation it committed to also during shuffling.

The first protocol is almost as efficient as the known proofs of shuffles; in fact it can be constructed from these, e.g., [21, 14, 18, 30]. Even without any standard optimization techniques such as simultaneous exponentiations, the computational complexity of the second protocol is half an exponentiation per sender in the El Gamal case and has similar properties for other cryptosystems. Thus, our pre-computation technique reduces the online computational complexity of virtually all mix-nets.

We also show that all known types of shuffles are instances of a generalized shuffle, where some homomorphic map $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$ is applied to each ciphertext and randomizer pair, and the resulting ciphertexts are permuted. In fact, we prove our results for this generalized shuffle. The generality of our result immediately gives that ciphertexts can be shuffled in parallel. Even ciphertexts of different cryptosystems can be shuffled in parallel, and distinct homomorphic maps can be used for ciphertexts of different cryptosystems.

The inspiration of this work comes from both Neff [21] and Furukawa and Sako [14]. Neff writes as follows about his "simple shuffle": "A single instance of this proof can be constructed to essentially 'commit' a particular permutation", but we are unable to derive our results starting from his "commitment". On the other hand, the Pedersen permutation commitment scheme used implicitly in the proof of a shuffle of Furukawa and Sako is perfectly suitable for constructing a fast commitment-consistent proof of a shuffle.

1.4 Notation

Natural numbers and integers are denoted by $\mathbb{N}$ and $\mathbb{Z}$ respectively. The ring of integers modulo n is denoted by $\mathbb{Z}_n$, $\mathbb{Z}_n^*$ denotes its multiplicative group, and SQ_n denotes the subgroup of squares in $\mathbb{Z}_n^*$. We use κ as the main security parameter, but also introduce several related parameters, e.g., the bit-size of challenges κ_c. We identify the set of κ-bit strings and the set of positive integers in $[0, 2^\kappa - 1]$ when convenient. A function $\epsilon(\kappa)$ is negligible if for every constant c and sufficiently large κ it holds that $\epsilon(\kappa) < \kappa^{-c}$. A function $f(\kappa)$ is overwhelming if $1 - f(\kappa)$ is negligible. We denote the set of N-permutations by $\mathbb{S}_N$. We denote the set $\{1, \ldots, l\}$ by $[l]$ and sometimes denote a list of elements $(a_1, \ldots, a_l)$ by $a_{[l]}$.

The Discrete Logarithm (DL) assumption for a group G_q with generator g states that given a random element $y \in G_q$, it is infeasible to compute x such that $y = g^x$. The decision Diffie-Hellman (DDH) assumption states that when $x, y, r \in \mathbb{Z}_q$ are randomly chosen, then it is infeasible to distinguish the distributions of (g^x, g^y, g^{xy}) and (g^x, g^y, g^r). See Appendix F for formal definitions.

We view a commitment scheme as consisting of a parameter generation algorithm Gen and a deterministic commitment algorithm Com. On input 1^κ, Gen outputs a parameter ck which defines a message set $\mathcal{M}_{ck}$, a polynomially sampleable randomness space $\mathcal{R}_{ck}$, and a commitment space $\mathcal{K}_{ck}$. We write $\mathcal{CK}$ for the set of commitment parameters. On input $ck \in \mathcal{CK}$, $m \in \mathcal{M}_{ck}$, and $r \in \mathcal{R}_{ck}$, Com outputs a commitment. To open a commitment the message and randomness is revealed.

We write $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ for a homomorphic cryptosystem and $\mathcal{M}_{pk}$, $\mathcal{R}_{pk}$, and $\mathcal{C}_{pk}$ for the abealian groups of messages, randomness, and ciphertexts defined by a public key pk. We let $\mathcal{PK}$ denote the set of all public keys. A homomorphic cryptosystem satisfies $\mathsf{E}_{pk}(m_1, r_1)\mathsf{E}_{pk}(m_2, r_2) = \mathsf{E}_{pk}(m_1 m_2, r_1 r_2)$ for every $pk \in \mathcal{PK}$, $m_1, m_2 \in \mathcal{M}_{pk}$, and $r_1, r_2 \in \mathcal{R}_{pk}$.

Throughout we assume that the order of the largest cyclic subgroup of $\mathcal{C}_{pk}$, and the order of any groups on which we base our commitment schemes, are bounded by 2^κ.

2 Background and Informal Description

Before we give details, it is worthwhile to recall some properties of batch proofs of discrete logarithms and proofs of shuffles. We also give a brief informal description of our commitment-consistent proof of a shuffle.

Batch Proofs. Consider a setting where many group elements $y_1, \ldots, y_N$ in some prime order group G_p with generator g are given, and the prover knows $x_i \in \mathbb{Z}_p$ such that $y_i = g_i^{x_i}$. It is expensive to prove knowledge of each logarithm x_i independently, but the use of batching [4] decreases this cost substantially as the following example shows, where $\mathcal{P}$ and $\mathcal{V}$ denotes the prover and verifier.

1. $\mathcal{V}$ picks $e_1, \ldots, e_N \in \mathbb{Z}_p$ randomly and hands them to prover.
2. Both parties compute $y = \prod_{i=1}^N y_i^{e_i}$.
3. $\mathcal{P}$ shows that it knows the logarithm w such that $y = g^w$ using a standard honest verifier zero-knowledge proof of knowledge.

The reason that this is a proof of knowledge is that the extractor may rewind the prover to the first step several times until it has found N linearly independent vectors $\overline{e}_j = (e_{j,1}, \ldots, e_{j,N})$ in $\mathbb{Z}_p^N$ for $j = 1, \ldots, N$ and extracted logarithms $w_1, \ldots, w_N$ such that $\prod_{i=1}^N y_i^{e_{j,i}} = g^{w_j}$. Note that linear independence imply that for every $l = 1, \ldots, N$ there are $d_{l,j}$ such that $\sum_{j=1}^N d_{l,j}\overline{e}_j$ is the lth standard unit vector in $\mathbb{Z}_p^N$. This gives

$$y_l = \prod_{j=1}^N \left(\prod_{i=1}^N y_i^{e_{j,i}}\right)^{d_{l,j}} = \prod_{j=1}^N (g^{w_j})^{d_{j,i}} = g^{\sum_{j=1}^N d_{l,j}w_j} \quad ,$$

which means that the logarithm of every individual element y_l can be computed as $x_l = \sum_{j=1}^N d_{l,j}w_j$. We remark that the components of the vectors can be chosen randomly in $\{0,1\}^{\kappa_e}$ for a κ_e much smaller than κ. From now on we use κ_e to denote the bit-size of components of random vectors as the above. Another important observation, used to reduce the need for jointly generated randomness when the honest verifier is implemented jointly by several parties, is that it suffices that the vectors are unpredictable, e.g., the verifier may instead choose a random seed z for a PRG, hand it to the prover, and define $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$.

Proofs of Shuffles. We do not go into the details of any particular proof of a shuffle, but we explain one of the ideas that appear in different forms in all known efficient schemes.

Consider a homomorphic cryptosystem such that the order of every non-trivial element in $\mathcal{C}_{pk}$ equals a prime p. Given are a public key pk and ciphertexts $(c_1, \ldots, c_N)$ and $(c'_1, \ldots, c'_N)$ that are related by $c'_i = c_{\pi(i)}\mathsf{E}_{pk}(1, r_{\pi(i)})$ for some permutation π and randomness $r_1, \ldots, r_N$.

A key observation, first made by Neff [21] and Furukawa and Sako [14], is that batch proofs are in some sense invariant under permutation and that this means that we can use batch techniques to construct an efficient proof of a shuffle. The idea can be described as follows, where we use a PRG to expand a seed into an unpredictable vector.

1. $\mathcal{V}$ picks a seed $z \in \{0,1\}^\kappa$ randomly and hands it to $\mathcal{P}$.
2. Both compute $c = \prod_{i=1}^N c_i^{e_i}$, where $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$ and $e_i \in [0, 2^{\kappa_e} - 1]$.

3. $\mathcal{P}$ computes $c' = \prod_{i=1}^{N} (c'_i)^{e_{\pi(i)}}$, hands it to $\mathcal{V}$, and convinces $\mathcal{V}$ that it is formed correctly.

4. $\mathcal{P}$ proves knowledge of $r \in \mathcal{R}_{pk}$ such that $c' = c\mathsf{E}_{pk}(1, r)$.

Note that the linear independence argument used in the basic batch proof above carries over to the shuffle setting, despite that some of the exponents are permuted (see Proposition 17 in Section A for details). The above description is simplified in that the prover must blind c' to avoid leaking knowledge. The problem of convincing the verifier that the original exponents, re-ordered using a fixed permutation π, are used to form c' is non-trivial, and solved differently in the various proofs of shuffles. If we ignore the cost of Step 3, then the above protocol is very efficient.

2.1 Commitment-Consistent Proofs of Shuffles

We observe that we can design Step 3 in such a way that almost all of it can be moved to the offline phase. Generators $g_1, \ldots, g_N$ of a group G_p of prime order p are given as part of the setup of the proof of a shuffle, and it is assumed to be infeasible to compute any non-trivial relations among these (this follows from the DL assumption).

Suppose that each mix-server commits to a permutation π using Pedersen commitments [24] $(a_1, \ldots, a_N) = (g^{r_1} g_{\pi^{-1}(1)}, \ldots, g^{r_N} g_{\pi^{-1}(N)})$ for random $r_1, \ldots, r_N \in \mathbb{Z}_p$, and also proves knowledge of the r_i and π such that $(a_1, \ldots, a_N)$ was formed in this way. Then in the online phase the verifier can choose, and hand to the prover, a random seed $z \in \{0,1\}^{\kappa}$, set $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$, and compute

$$a = \prod_{i=1}^{N} a_i^{e_i} = \prod_{i=1}^{N} g^{r_i e_i} g_{\pi^{-1}(i)}^{e_i} = g^r \prod_{i=1}^{N} g_i^{e_{\pi(i)}} \quad ,$$

where $r = \sum_{i=1}^{N} r_i e_i$. Note that a is of a perfect form for executing a standard proof of knowledge of equal exponents. More precisely, we may now replace Step 3 above in the online phase by:

- Prover computes $c' = \prod_{i=1}^{N} (c'_i)^{e_{\pi(i)}}$ and hands it to the verifier.
- Prover proves knowledge of $r' \in \mathbb{Z}_p$ and $e'_1, \ldots, e'_N \in \{0,1\}^{\kappa_e}$ with

$$a = g^{r'} \prod_{i=1}^{N} g_i^{e'_i} \quad \text{and} \quad c' = \prod_{i=1}^{N} (c'_i)^{e'_i} \quad .$$

The above is simplified in that some blinding factors are missing. The proof of knowledge of the exponents $r', e'_1, \ldots, e'_N$, combined with the computational binding property of multi-base Pedersen commitments implies that $e'_i = e_{\pi(i)}$ for some permutation $\pi(i)$. The computational complexity of the above protocol is very low, since almost all exponents have very few bits also in the proof of knowledge of equal exponents.

3 A Commitment-Consistent Proof of a Shuffle

In this section we first give more details of the commitment scheme and explain how any of the known proofs of shuffles can be used to prove knowledge of an opening of the commitment to a permutation. Then we present the commitment-consistent proof of a shuffle.

3.1 Permutation Commitments

We formalize the property we need from the Pedersen commitments above. A permutation commitment should allow the committer to compute a commitment $\mathsf{Com}^\star(\pi)$ of a permutation π, but obviously any string commitment can be used to commit to a permutation. The special property of a permutation commitment is that if the *receiver* holds a list $(e_1, \ldots, e_N)$, it can transform the permutation commitment into a commitment $\mathsf{Com}^e(e_{\pi(1)}, \ldots, e_{\pi(N)})$, of another type, of the the list elements, but in order defined by π. Here κ_{com} denotes the maximal bit size of each component of a list commitment.

Definition 1. Let $(\mathsf{Gen}^\star, \mathsf{Com}^\star)$ be a commitment scheme for $\mathbb{S}_N$ and let $(\mathsf{Gen}^e, \mathsf{Com}^e)$ be a commitment scheme for $[0, 2^{\kappa_{com}} - 1]^N$. The former is a κ_{com}-permutation commitment scheme of the latter if $\mathsf{Gen}^\star = \mathsf{Gen}^e$ and there exist deterministic polynomial time algorithms Map and Rand s.t. for every $ck \in \mathcal{CK}$, $r^\star \in \mathcal{R}^\star_{ck}$, $\pi \in \mathbb{S}_N$ and $e = (e_1, \ldots, e_N) \in [0, 2^{\kappa_{com}} - 1]^N$

$$\mathsf{Map}_{ck}(\mathsf{Com}^\star_{ck}(\pi, r^\star), e) = \mathsf{Com}^e_{ck}((e_{\pi(1)}, \ldots, e_{\pi(N)}), \mathsf{Rand}(r^\star, e)) \ .$$

Construction 2 (Pedersen Commitment)**.** The generation algorithm $\mathsf{Gen}^\star$ outputs random generators $g_1, \ldots, g_N \in G_q$, where G_q is a cyclic group of known order $q = \prod_{i=1}^t p_i$ with $p_i \geq 2^{\kappa_{com}}$. On input $\pi \in \mathbb{S}_N$ and $r_1, \ldots, r_N \in \mathbb{Z}_q$, the commitment algorithm $\mathsf{Com}^\star$ computes $a_i = g^{r_i} g_{\pi^{-1}(i)}$, and outputs $(a_1, \ldots, a_N)$. The parameter algorithm Gen^e is identical to $\mathsf{Gen}^\star$. On input $(e_1, \ldots, e_N) \in [0, 2^{\kappa_{com}} - 1]^N$ and $r \in \mathbb{Z}_q$, the algorithm Com^e computes $a = g^r \prod_{i=1}^N g_i^{e_i}$, and outputs a.

The idea of using (generalized) Pedersen commitments [24] to commit to permutations is not novel, e.g., it is used implicitly in [14], but the observation that a commitment of the first kind can be transformed into a commitment of the second kind seems new.

Proposition 3. *Both* $(\mathsf{Gen}^\star, \mathsf{Com}^\star)$ *and* $(\mathsf{Gen}^e, \mathsf{Com}^e)$ *of Construction 2 are perfectly hiding and computationally binding under the DL assumption. The former is a permutation commitment of the latter.*

The proof of the binding property is well known for prime order groups. A proof is given in Appendix A.

We will later make use of the following relation that corresponds to breaking a commitment scheme, i.e., finding two different ways to open a commitment.

Definition 4. The relation $\mathscr{R}^{twin}_{ck}$ consists of all pairs $\big(ck, (s_{[l]}, s_0, s'_{[l]}, s'_0)\big)$ such that $s_{[l]} \neq s'_{[l]}$ and $\mathsf{Com}^e_{ck}(s_{[l]}, s_0) = \mathsf{Com}^e_{ck}(s'_{[l]}, s'_0)$.

Suppose a committer produces a permutation commitment $a^\star$ and the verifier computes $a = \mathsf{Map}_{ck}(a^\star, (e_1, \ldots, e_N))$. Then we expect that the committer only can open a as $(e_{\pi(1)}, \ldots, e_{\pi(N)})$ for a *fixed* permutation π, i.e., if we repeat this procedure with different lists $(e_1, \ldots, e_N)$ the same permutation must be used by the committer every time. We can not prove this, but it is easy to see that if it also can open $a^\star$ to a permutation π, then it must use this permutation every time. Recall that in our application, each mix-server proves knowledge of how to open $a^\star$ during the offline phase. Thus, if a witness for the following relation can be extracted in the online phase we reach a contradiction. This suffices to prove the overall security of a mix-net.

Definition 5. The relation $\mathscr{R}_{ck}^{perm}$ consists of all pairs $\big(ck, (a^\star, s_{[N]}, s_0, s'_{[N]}, s'_0)\big)$ such that

$$\mathsf{Map}_{ck}(a^\star, s_{[N]}) = \mathsf{Com}^e_{ck}((s_{\pi(1)}, \ldots, s_{\pi(N)}), s_0) \enspace ,$$
$$\mathsf{Map}_{ck}(a^\star, s'_{[N]}) = \mathsf{Com}^e_{ck}((s'_{\pi'(1)}, \ldots, s'_{\pi'(N)}), s'_0) \enspace ,$$

$\pi \neq \pi'$, and $s_i \neq s_j$ and $s'_i \neq s'_j$ for all $i \neq j$.

3.2 Proof of Knowledge of Opening

We now explain how we can construct, from any proof of a shuffle of El Gamal ciphertexts over a prime order group G_p, a proof of knowledge that a Pedersen permutation commitment indeed is a commitment to a permutation.

Definition 6. The relation $\mathscr{R}_{ck}^{open}$ consists of all pairs $\big((ck, a^\star), (\pi, r^\star)\big)$ such that $a^\star = \mathsf{Com}^\star_{ck}(\pi, r^\star)$.

Protocol 7 (Proof of Knowledge of Correct Opening).
COMMON INPUT: Pedersen commitment parameters $g, g_1, \ldots, g_N \in G_p$ and a commitment $(a_1, \ldots, a_N) \in G_p^N$.
PRIVATE INPUT: Permutation $\pi \in \mathbb{S}_N$ and exponents $r_1, \ldots, r_N \in \mathbb{Z}_p$ such that $a_i = g^{r_i} g_{\pi^{-1}(i)}$.

1. $\mathcal{P}$ chooses $r'_i \in \mathbb{Z}_p$ and $h \in G_p$ randomly, computes $a'_i = g^{r'_i} a_i$ and $b_i = h^{r_i + r'_i}$, and hands $(a'_1, \ldots, a'_N)$ and $(h, b_1, \ldots, b_N)$ to $\mathcal{V}$.
2. $\mathcal{P}$ proves to $\mathcal{V}$ that it knows r'_i such that $a'_i = g^{r'_i} a_i$.
3. $\mathcal{P}$ and $\mathcal{V}$ view (h, g) as an El Gamal public key, and $\mathcal{P}$ uses its random commitment exponents $r_1 + r'_1, \ldots, r_N + r'_N$ to give a proof of a shuffle that the list $(b_1, a'_1), \ldots, (b_N, a'_N)$ is a re-encryption and permutation of the list of trivial ciphertexts $(1, g_1), \ldots, (1, g_N)$ using the public key (h, g), i.e., it proves that it knows some r''_i such that $(b_i, a_i) = (h^{r''_i}, g^{r''_i} g_{\pi^{-1}(i)})$.

Proposition 8. *The protocol inherits properties of the proof of a shuffle.*

1. *If the proof of a shuffle is public-coin, overwhelmingly (computationally) sound, and a proof of knowledge, then so is the protocol above.*

2. *If the proof of a shuffle is honest verifier (computationally under assumption A) zero-knowledge, then the above protocol is computationally zero-knowledge under the DDH assumption (and assumption A).*

A proof is given in Appendix A. Without the blinding exponent r'_i the protocol is not even computationally zero-knowledge, since the adversary could in principle know r_i. Some proofs of shuffles do not satisfy the standard computational versions of soundness, proof of knowledge, and zero-knowledge. In those cases the correspondingly more complicated security properties are also inherited, but we use the above proposition for simplicity. Readers with deeper understanding of proofs of shuffles should note that the basic principles of any proof of a shuffle can be used directly to construct a more efficient protocol, but this is not our focus here. We stress that the above simple solution is presented for completeness and ease of presentation. It is non-trivial to extend the above result to groups of *composite* order such as those considered in Construction 2.

3.3 Proof of Knowledge of Equal Exponents

Recall from our sketch in Section 2.1 that in our commitment-consistent proof of a shuffle, the prover essentially hands the product $\prod_{i=1}^{N}(c'_i)^{e_{\pi(i)}}$ to the verifier and shows that the exponents used are those committed to in a commitment $\mathsf{Com}^e(e_{\pi(1)}, \ldots, e_{\pi(N)})$. More precisely, we assume that: $\{h_1, \ldots, h_k\}$ is a generator set of the group $\mathcal{C}_{pk}$ of ciphertexts, ck is a commitment parameter, and that the prover hands $\prod_{i=1}^{N}(c'_i)^{e_{\pi(i)}}$ to the verifier in blinded form, i.e., it hands $\left(\mathsf{Com}^e_{ck}(s_{[k]}, s_0), \prod_{i=1}^{k} h_i^{s_i} \prod_{i=1}^{N}(c'_i)^{e_{\pi(i)}}\right)$ to the verifier for random exponents $s_{[k]}$ (and $s_0 \in \mathcal{R}_{ck}$), and then proves that it knows all of these exponents and that they are consistent with the exponents committed to in $\mathsf{Com}^e_{ck}((e_{\pi(1)}, \ldots, e_{\pi(N)}), e_0)$ for some $e_0 \in \mathcal{R}_{ck}$. Thus, we construct a protocol for the following relation.

Definition 9. From a scheme $(\mathsf{Gen}^e, \mathsf{Com}^e)$ for $[0, 2^{\kappa_{com}} - 1]^N$, a commitment parameter ck output by Gen^e, and a public key $pk \in \mathcal{PK}$ we define $\mathscr{R}^{eq}_{ck,pk}$ to consist of all $\left((pk, ck, h_{[k]}, c_{[N]}, a, b_1, b_2), (e_0, e_{[N]}, s_0, s_{[N]})\right)$ satisfying $a = \mathsf{Com}^e_{ck}(e_{[N]}, e_0)$, $b_1 = \mathsf{Com}^e_{ck}(s_{[k]}, s_0)$, and $b_2 = \prod_{i=1}^{k} h_i^{s_i} \prod_{i=1}^{N} c_i^{e_i}$.

If the largest cyclic subgroup of $\mathcal{C}_{pk}$ has order $q = \prod_{i=1}^{t} p_i$ with $p_i \geq 2^{\kappa_c}$, and a group G_q of order q is available for which the DL problem is hard, then a sigma protocol with the challenge chosen from $[0, 2^{\kappa_c} - 1]$, can be constructed using fairly standard methods. For completeness we give such a protocol in Appendix B.

Otherwise, we can either use Pedersen commitments over some prime order group G_p and use a proof of equal exponents over groups of different orders using a Fujisaki-Okamoto commitment [12] as a "bridge", or we can replace the permutation commitment by a corresponding Fujisaki-Okamoto commitment directly. It is not hard to derive a shuffle of such commitments from Wikström's shuffle [30]. The drawback of using Fujisaki-Okamoto commitments is that they are based on the use of an RSA modulus, and such moduli are costly to generate in a distributed setting. We detail both solutions in the appendix.

3.4 Shuffle-Friendly Maps

To *randomly shuffle* a list of homomorphic ciphertexts $(c_1, \ldots, c_N)$ usually means that each ciphertext is randomly re-encrypted and the resulting ciphertexts randomly permuted, but there are other possible shuffles. For the El Gamal cryptosystem, one can also partially decrypt during shuffling [13], or if a special key set-up is used one can avoid random re-encryption entirely [30]. There are also at least two types of shuffles of (variants of) Paillier [23] ciphertexts. A careful look at these shuffles reveal that they are all defined by evaluating a homomorphic map and permuting the result.

Definition 10. 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 $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$.

Example 11. Using the El Gamal cryptosystem over a group G_p with public key $pk = (g, y)$, where $y = g^x$ and x is the secret key, we have $\mathcal{M}_{pk} = G_p$, $\mathcal{R}_{pk} = \mathbb{Z}_p$, and $\mathcal{C}_{pk} = G_p \times G_p$. Then $\phi_{(g,y)}((u, v), r) = (g^r u, y^r v)$ describes re-encryption when $r \in \mathbb{Z}_p$ is randomly chosen. If $y_i = g^{x_i}$, $y = y_1 y_2 y_3$, and $x = x_1 + x_2 + x_3$, then $\phi^{x_1}_{(g,y)}((u, v), r) = (g^r u, (y/y_1)^r u^{-x_1} v)$ denotes partial decryption and re-encryption using the secret share x_1 and randomness r. The decryption shuffle in [30] can be described similarly.

Example 12. Using the Paillier cryptosystem with a public key $pk = n$ consisting of a random RSA modulus, we have $\mathcal{M}_{pk} = \mathbb{Z}_n$, $\mathcal{R}_{pk} = \mathbb{Z}^*_n$, and $\mathcal{C}_{pk} = \mathbb{Z}^*_{n^2}$ with encryption defined by $\mathsf{E}_{pk}(m, r) = (1 + n)^m r^n \bmod n^2$. Re-encryption is then defined by $\phi_n(c, r) = c r^n \bmod n^2$.

Suppose we wish to prove that a ciphertext c' is the result of invoking a particular shuffle-friendly map ϕ_{pk} on another ciphertext c. If the shuffle-friendly map ϕ_{pk} is public, e.g., it represents re-encryption, then what is needed is a proof that there exists some randomness r such that $\phi_{pk}(c, r) = c'$. If the shuffle-friendly map itself is not public, e.g., re-encryption and partial decryption, then the map ϕ_{pk} must then be defined by some hidden parameters. Without loss we assume that the map is defined by some relation to the public key. In the typical cases, the public key defines a secret key and the shuffle-friendly map is defined by the secret key. We consider a situation where the output ciphertext c' is committed to as $(\mathsf{Com}^e_{ck}((s_1, \ldots, s_k), s_0), c' \prod_{i=1}^k h_i^{s_i})$, and define a relation for a shuffle-friendly map as follows.

Definition 13 (Shuffle-Friendly Relation)**.** Let $pk \in \mathcal{PK}$, let ϕ_{pk} be a shuffle-friendly map for pk and let ck be a commitment parameter. We define $\mathscr{R}^{map}_{\phi_{pk}}$ to consist of all pairs $\big((pk, ck, h_{[k]}, c, b_1, b_2), (r, s_0, s_{[k]})\big)$ such that $b_1 = \mathsf{Com}^e_{ck}(s_{[k]}, s_0)$ and $b_2 = \phi_{pk}(c, r) \prod_{i=1}^k h_i^{s_i}$.

Example 14 (Example 11 continued)**.** Note that $\mathcal{C}_{pk} = G_p \times G_p$ is generated by $h_1 = (g, 1)$ and $h_2 = (1, g)$ with component-wise multiplication. If we consider a re-encryption and permutation shuffle and use Pedersen commitments over the group G_p with commitment parameter $ck = (g_1, g_2)$, then the relation consists of all pairs of the form $\big(((g,y),(g_1,g_2),(u,v),b_1,b_2),(r,s_0,s_1,s_2)\big)$ such that $b_1 = g^{s_0}g_1^{s_1}g_2^{s_2}$ and $b_2 = h_1^{s_1}h_2^{s_2}(g^r u, y^r v)$.

For the typical shuffle-friendly maps of the El Gamal and Paillier cryptosystems, it is well known how to construct sigma protocols [7] for the corresponding shuffle-friendly relation using standard methods. We give some examples in Appendix E.

3.5 Details of the Commitment-Consistent Proof of a Shuffle

Next we give a detailed description of the protocol that allows a mix-server to prove in the online phase that it re-encrypted and permuted its input and that the permutation used is the same permutation it committed to in the offline phase. We denote by κ_r a parameter that decides how well the commitments hide the committed values.

The two subprotocols can be executed in parallel and the second step of the protocol can be combined with the first move of the combined subprotocols.

Protocol 15 (Commitment-Consistent Proof of a Shuffle)**.**
Common Input: A public key pk of a cryptosystem $\mathcal{CS}$, a generating set $\{h_1,\ldots,h_k\}$ of $\mathcal{C}_{pk}$, a commitment parameter ck, a permutation commitment $a^\star \in \mathcal{K}_{ck}^{\pi}$, ciphertexts $(c_1,\ldots,c_N) \in \mathcal{C}_{pk}^N$, and $(c'_1,\ldots,c'_N) \in \mathcal{C}_{pk}^N$.
Private Input: Permutation $\pi \in \mathbb{S}_N$, $s^\star \in \mathcal{R}_{ck}^\star$ and $r_1,\ldots,r_N \in \mathcal{R}_{pk}$ such that $a^\star = \mathsf{Com}_{ck}^\star(\pi, s^\star)$, and $c'_i = \phi_{pk}(c_{\pi(i)}, r_{\pi(i)})$.

1. $\mathcal{V}$ chooses a seed $z \in \{0,1\}^\kappa$ randomly and hands it to $\mathcal{P}$. Then both parties set $(e_1,\ldots,e_N) = \mathsf{PRG}(z)$, where $e_i \in \{0,1\}^{\kappa_e}$, and computes $a = \mathsf{Map}_{ck}(a^\star,(e_1,\ldots,e_N))$.
2. $\mathcal{P}$ chooses $t_0 \in \mathcal{R}_{ck}$ and $t_1,\ldots,t_k \in [0, 2^{\kappa+\kappa_r}-1]$ randomly, and computes and hands to $\mathcal{V}$
$$b_1 = \mathsf{Com}_{ck}^e((t_1,\ldots,t_k),t_0) \text{ and } b_2 = \prod_{i=1}^k h_i^{t_i} \prod_{i=1}^N (c'_i)^{e_{\pi(i)}} \enspace .$$
3. $\mathcal{P}$ proves, using a proof of equal exponents, that it knows exponents $t_0,\ldots,t_k$, $(e'_1,\ldots,e'_N)$ (computed as $(e_{\pi(1)},\ldots,e_{\pi(N)})$), and e_0 (computed as $\mathsf{Rand}(s^\star,(e_1,\ldots,e_N))$) such that
$$b_1 = \mathsf{Com}_{ck}^e((t_1,\ldots,t_k),t_0) \enspace , \enspace b_2 = \prod_{i=1}^k h_i^{t_i} \prod_{i=1}^N (c'_i)^{e'_i} \enspace , \text{ and}$$
$$a = \mathsf{Com}_{ck}^e((e'_1,\ldots,e'_N),e_0) \enspace .$$
4. $\mathcal{P}$ proves, using a proof of a shuffle map, that it knows exponents $t_0,\ldots,t_k$ and r (computed as $\prod_{i=1}^N r_i^{e_i}$) such that
$$b_1 = \mathsf{Com}_{ck}^e((t_1,\ldots,t_k),t_0) \text{ and } b_2 = \prod_{i=1}^k h_i^{t_i} \phi_{pk}\left(\prod_{i=1}^N c_i^{e_i}, r\right) \enspace .$$

Note that the protocol and the proposition below are quite general; they apply for all the usual homomorphic cryptosystems, any shuffle-friendly map, and any number of ciphertexts shuffled in parallel (this is considered as a separate case in [21]). It even applies to mixed settings where ciphertexts from different cryptosystems are shuffled in parallel. To state the security properties of the protocol we need to define a relation that captures a shuffle.

Definition 16. Let $pk \in \mathcal{PK}$, let ϕ_{pk} be a shuffle-friendly map for pk. Then we define the shuffle relation $\mathscr{R}^{shuf}_{\phi_{pk}}$ to consist of all pairs of the form $\big((pk, c_{[N]}, c'_{[N]}), (\pi, r_{[N]})\big)$ with $c'_i = \phi_{pk}(c_{\pi(i)}, r_{\pi(i)})$.

In the proposition we consider the relation $\mathscr{R}^{shuf}_{\phi_{pk}} \vee \mathscr{R}^{twin}_{ck} \vee \mathscr{R}^{perm}_{ck}$. In general, for two relations $\mathscr{R}_1$ and $\mathscr{R}_2$, the relation $\mathscr{R}_1 \vee \mathscr{R}_2$ denotes the relation consisting of all pairs $((x_1, x_2), w)$ such that $(x_1, w) \in \mathscr{R}_1$ or $(x_2, w) \in \mathscr{R}_2$.

Proposition 17. *Let the subprotocols be overwhelmingly complete sigma protocols for the relations* $\mathscr{R}^{eq}_{ck,pk} \vee \mathscr{R}^{twin}_{ck}$ *and* $\mathscr{R}^{map}_{\phi_{pk}}$ *respectively, and let the commitment scheme be statistically hiding.*

Then for every $pk \in \mathcal{PK}$ *and* $ck \in \mathcal{CK}$*, the protocol is a public-coin honest verifier statistical zero-knowledge proof of the relation* $\mathscr{R}^{shuf}_{\phi_{pk}} \vee \mathscr{R}^{twin}_{ck} \vee \mathscr{R}^{perm}_{ck}$ *with negligible soundness error, and it is overwhelmingly complete for witnesses of* $\mathscr{R}^{shuf}_{\phi_{pk}}$.

It is a proof of knowledge with negligible knowledge error of a string w *such that* $\mathscr{R}^{shuf}_{\phi_{pk}}\big((pk, c_{[N]}, c'_{[N]}), (w, r_{[N]})\big) = 1$, $\mathscr{R}^{twin}_{ck}(ck, w) = 1$*, or* $\mathscr{R}^{perm}_{ck}(ck, w) = 1$*, is satisfied for some randomness* $r_{[N]} \in \mathcal{R}_{pk}$*, where we use the notation for inputs to the protocol as defined above.*

Remark 18. It is observed in [30] that it does not suffice that a proof of a re-encryption and permutation shuffle is sound to be used in a provably secure mix-net. The permutation used by the mix-server to shuffle must also be extractable. However, extracting the *permutation* suffices.

A proof of the proposition is given in Appendix A. The basic idea is explained already in Section 2.1, except that in the general case the order q of the maximal cyclic subgroup of $\mathcal{C}_{pk}$ may not be prime or may even be unknown. Note that if q is not prime, then the "random vectors" are in fact defined over a ring and not over a field, and consequently they are not vectors at all. Thus, not all elements are invertible, which potentially is a problem when trying to find a linear combination of the "random vectors" equal to any standard unit vector, which is needed to extract a witness. Since we assume that all factors of the order of $\mathcal{C}_{pk}$ are large and all elements that must be inverted are random, this is not a problem and a witness can be extracted. However, if it is infeasible to compute the factorization of the order of $\mathcal{C}_{pk}$, or if the order itself is unknown, then this seems difficult. Fortunately, it suffices for the overall security of the mix-net that only the permutation can be extracted.

4 Application To Mix-Nets

At this point the reader should be comfortable with the idea that a proof of a shuffle can be split into a relatively costly offline part (Protocol 7) and a very efficient online part (Protocol 15), but how exactly do they fit into a mix-net?

Below we give a brief informal description of a mix-net based on the El Gamal cryptosystem over a group G_p of prime order p. This illustrates how our protocols are used and gives an idea of the complexity of a complete mix-net using our approach.

Offline Phase.

1. The mix-servers, $M_1, \ldots, M_k$, run a distributed key generation protocol to generate a joint public key (g, y) such that the corresponding secret key x, with $y = g^x$, is secret shared among the mix-servers.

2. M_j chooses $r_{j,i} \in \mathbb{Z}_p$ randomly and computes $(g^{r_{j,i}}, y^{r_{j,i}})$ for $i = 1, \ldots, N$.

3. M_j chooses a random permutation $\pi_j \in \mathbb{S}_N$, publishes a permutation commitment $a_j^\star = \mathsf{Com}^\star(\pi_j)$, and proves knowledge of the committed permutation using Protocol 7 (and verifies the proofs of knowledge of all other mix-servers).

Online Phase.

4. S_i chooses $r_i \in \mathbb{Z}_p$ randomly, computes $(u_{0,i}, v_{0,i}) = \mathsf{E}_{(g,y)}(m_i, r_i)$, where $m_i \in \mathbb{Z}_p$ is its message, and publishes this ciphertext.

5. Let $L_0 = (u_{0,i}, v_{0,i})_{i=1}^N$ be the input ciphertexts. For $l = 1, \ldots, k$:

   (a) If $l = j$, then M_j computes $(u_{j,i}, v_{j,i}) = (g^{r_{j,i}} u_{j-1,\pi_j(i)}, y^{r_{j,i}} v_{j-1,\pi_j(i)})$, publishes $L_j = (u_{j,i}, v_{j,i})_{i=1}^N$, and proves using Protocol 15 that L_{j-1} and L_j are consistent with $a_j^\star$.

   (b) If $l \neq j$, then M_j verifies the proof of M_l, i.e., that L_{l-1} and L_l are consistent with $a_l^\star$.

6. The mix-servers perform a threshold decryption of L_k using their shares of x and output the list of plaintexts $(m_{\pi(1)}, \ldots, m_{\pi(N)})$, where $\pi = \pi_k \cdots \pi_1$.

The random challenges needed in the subprotocols are generated jointly using a coin-flipping protocol over a broadcast channel or bulletin board. Thus, all verifiers jointly either accept or reject proofs. It is natural to ask why the security property of our commitment-consistent proof suffices, since it is sound for $\mathscr{R}_{\phi_{pk}}^{shuf} \vee \mathscr{R}_{ck}^{twin} \vee \mathscr{R}_{ck}^{perm}$ and not for $\mathscr{R}_{\phi_{pk}}^{shuf}$. This follows from the proof of knowledge property. For any successful prover there exists an extractor that outputs: a valid permutation π used to shuffle, a witness for $\mathscr{R}_{ck}^{twin}$, or a witness for $\mathscr{R}_{ck}^{perm}$. The second type of output directly contradicts the security of the commitment scheme. The third type of output combined with knowledge of how to open $a_j^\star$ (such an opening can be extracted during the offline phase), also contradicts the security of the commitment scheme. Thus, in a simulation the extractor outputs the permutation with overwhelming probability, which suffices to prove the overall security of the mix-net.

Depending on the secret sharing threshold, all mix-servers may not need to shuffle the ciphertexts, and there are obvious ways to avoid the assumption that all senders submit an input. Many details are of course missing from the above description, but in the El Gamal case all subprotocols missing from the description are available. Distributed key generation can be done using Feldman and Pedersen secret sharing [11, 24]. The submission of inputs must allow extraction of the plaintexts of corrupt senders without using the secret key of the cryptosystem. This can be done [31] based on the Cramer-Shoup cryptosystem [8] in such a way that each mix-server essentially pays the cost of checking the validity of N Cramer-Shoup ciphertexts. Batch techniques [4] can be used to reduce this further if most ciphertexts are expected to be valid, and validity checks can be done concurrently with receiving new ciphertexts. Random challenges can be generated using Pedersen verifiable secret sharing [24]. The sharing phase of many coins can be pre-computed, but since we only need a small number of bits in each challenge this type of optimization does not give much. Finally, during threshold decryption each mix-server must exponentiate N group elements to decrypt, but proving that this was done correctly can be done using batch proofs [4]. To summarize, the online running time of the mix-net is roughly the time to: validate N Cramer-Shoup ciphertexts, run the prover or verifier of k commitment-consistent proofs of shuffles of lists of ciphertexts of length N, decrypt N El Gamal ciphertexts, and prove or verify correctness of joint decryption, which is done using a batch proof.

Recall that κ_e denotes the bit-size of elements in random "vectors", κ_c denotes the bit-size of challenges, and κ_r decides the statistical error in simulations and also the completeness of our subprotocols. For practical security parameters, e.g., $\kappa = 1024$, $\kappa_e = \kappa_c = 80$ and $\kappa_r = 20$, we estimate the complexity of our protocol to $N/2$ square-and-multiply exponentiations. This can be reduced by a factor of $1/5$ if simultaneous exponentation [20] is used, giving a complexity corresponding to $N/10$ square-and-multiply exponentations (see Appendix C for details).

Thus, our commitment-consistent proof of a shuffle is several times faster in the online phase than any of the known proofs of shuffles. As far as we know this makes our mix-net faster in the online phase than any previous mix-net.

5 Acknowledgments

We thank Jun Furukawa and Andy Neff for helpful comments.

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] M. Bellare, J.A. Garay, and T. Rabin. Batch verification with applications to cryptography and checking. In *LATIN*, volume 1380 of *Lecture Notes in Computer Science*, pages 170–191. Springer Verlag, 1998.

[5] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *26th ACM Symposium on the Theory of Computing (STOC)*, pages 544–553. ACM Press, 1994.

[6] D. Chaum. Untraceable electronic mail, return addresses and digital pseudo-nyms. *Communications of the ACM*, 24(2):84–88, 1981.

[7] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – Crypto '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.

[8] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer Verlag, 1998.

[9] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer Verlag, 2002.

[10] Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 557–572. Springer Verlag, 2000.

[11] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 427–438. IEEE Computer Society Press, 1987.

[12] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 1997.

[13] 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.

[14] 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.

[15] T. El Gamal. A public key cryptosystem and a signiture scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[16] T. Granlund. Gnu multiple precision arithmetic library (GMP). Software available at `http://swox.com/gmp`, March 2005.

[17] T. Granlund. Private communication., March 2008.

[18] 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.

[19] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium*, pages 339–353. USENIX, 2002.

[20] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[21] 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.

[22] A. Neff. Private communication., May 2008.

[23] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Verlag, 1999.

[24] 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.

[25] 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.

[26] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[27] 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.

[28] D. Wikström. Five practical attacks for "optimistic mixing for exit-polls". In *Selected Areas in Cryptography – SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 160–174. Springer Verlag, 2003.

[29] 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.

[30] 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.

[31] Douglas Wikström. Simplified submission of inputs to protocols. Cryptology ePrint Archive, Report 2006/259, 2006. `http://eprint.iacr.org/`.

A Omitted Proofs

Proof of Proposition 3. In both schemes, if two distinct valid openings can be computed, then we can write $g^b \prod_{i=1}^N g_i^{b_i} = 1$ for some $b, b_1, \ldots, b_N \in \mathbb{Z}_p$, where not all are zero. It is well known and easy to see that this contradicts the DL assumption.

For a commitment $(a_1, \ldots, a_N) = \mathsf{Com}^{\star}_{ck}(\pi, (r_1, \ldots, r_N))$, where $ck = (g_1, \ldots, g_N)$, we have $\prod_{i=1}^N a_i^{e_i} = g^{\sum_{i=1}^N r_i e_i} \prod_{i=1}^N g_i^{e_{\pi(i)}}$ which equals a commitment computed as $\mathsf{Com}^e_{ck}((e_{\pi(1)}, \ldots, e_{\pi(N)}), \sum_{i=1}^N r_i e_i)$. This proves the second claim. □

Proof of Proposition 8. The first claim is obvious. The zero-knowledge simulator picks $b_1, \ldots, b_N \in G_p$ randomly and then invokes the simulators of the batch proof and the proof of a shuffle. A standard hybrid argument shows that the resulting distribution is polynomially indistinguishable from the distribution of a real interaction, under the DDH assumption. □

Proof of Proposition 17. The zero-knowledge simulator chooses z randomly as in the protocol, chooses $t_0 \in \mathcal{R}^e_{ck}$ and $b_1 \in \mathcal{C}_{pk}$ randomly, computes $b_1 = \mathsf{Com}^e_{ck}((0, \ldots, 0), t_0)$, and then invokes the statistical zero-knowledge simulators of the subprotocols. Due to the statistical zero-knowledge of the commitment scheme the resulting distribution is statistically close to the distribution of a real interaction.

Suppose that we, by repeatedly invoking the extractors of the subprotocols, manage to extract N different $\overline{e}_j = (e_{j,1}, \ldots, e_{j,N})$ with distinct components and exponents $t_{j,0}, \ldots, t_{j,k}$, and $e'_{j,0}, \ldots, e'_{j,N}$, satisfying

$$\begin{aligned} b_{j,1} &= \mathsf{Com}^e_{ck}((t_{j,1}, \ldots, t_{j,k}), t_{j,0}) \ , \quad b_{j,2} = \prod_{i=1}^k h_i^{t_{j,i}} \prod_{i=1}^N (c'_i)^{e'_{j,i}} \ , \text{ and} \\ a_j &= \mathsf{Com}^e_{ck}((e'_{j,1}, \ldots, e'_{j,N}), e'_{j,0}) \ , \end{aligned}$$

where $a_j = \mathsf{Map}_{ck}(a^\star, (e_{j,1}, \ldots, e_{j,N}))$, and exponents $t'_{j,0}, \ldots, t'_{j,k}$, and r_j satisfying

$$\begin{aligned} b_{j,1} &= \mathsf{Com}^e_{ck}((t'_{j,1}, \ldots, t'_{j,k}), t'_{j,0}) \text{ and} \\ b_{j,2} &= \prod_{i=1}^k h_i^{t'_{j,i}} \phi_{pk}\left(\prod_{i=1}^N c_i^{e_{j,i}}, r_j\right) \ , \end{aligned}$$

for $j = 1, \ldots, N$. The witness extracted from a subprotocol may be a witness of $\mathscr{R}^{twin}_{ck}$, in which case we are done. From now on we assume that this is not the case.

If it is not the case that $(t_{j,1}, \ldots, t_{j,k}) = (t'_{j,1}, \ldots, t'_{j,k})$ for all $j = 1, \ldots, N$, then we have found two distinct openings of a commitment $b_{j,1}$, i.e., a witness for ck with respect to the relation $\mathscr{R}^{twin}_{ck}$. Thus, we may drop the prime symbol and simply write $(t_{j,1}, \ldots, t_{j,k})$.

If it is not the case that $e'_{j,i} = e_{j,\pi(i)}$ for some *fixed* permutation π and all $j = 1, \ldots, N$ and $i = 1, \ldots, N$, then we have found a witness for ck with respect to the relation $\mathscr{R}^{perm}_{ck}$. Thus, we assume that the same permutation π appears for every list and simply write $(e_{j,\pi(1)}, \ldots, e_{j,\pi(N)})$ instead of $(e'_{j,1}, \ldots, e'_{j,N})$.

We conclude that

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

for $j = 1, \ldots, N$. We have now extracted a permutation π (the extractor only needs to extract a single list $(e_{j,\pi(1)}, \ldots, e_{j,\pi(N)})$ to find π). It remains to prove that with overwhelming probability there exists $(r_1, \ldots, r_N)$ such that $(\pi, r_1, \ldots, r_N)$ is a witness for $\mathscr{R}^{shuf}_{\phi_{pk}}$.

Suppose that for every $l = 1, \ldots, N$, there exist $x_{l,1}, \ldots, x_{l,N} \in \mathbb{Z}_q$, where q is the order of the maximal cyclic subgroup of $\mathcal{C}_{pk}$, such that $\sum_{j=1}^{N} x_{l,j}\bar{e}_j$ is the lth standard unit "vector" in $\mathbb{Z}_q^N$ (since $\mathbb{Z}_q$ is not a field, this is not a vector space). We conclude that

$$\begin{aligned} c'_{\pi^{-1}(l)} &= \prod_{j=1}^{t}\left(\prod_{i=1}^{N}(c'_i)^{e_{j,\pi(i)}}\right)^{x_{l,j}} = \prod_{j=1}^{t}\left(\phi_{pk}\left(\prod_{i=1}^{N} c_i^{e_{j,i}}, r_j\right)\right)^{x_{l,j}} \\ &= \phi_{pk}\left(\prod_{j=1}^{t}\left(\prod_{i=1}^{N} c_i^{e_{j,i}}\right)^{x_{l,j}}, \prod_{j=1}^{t} r_j^{x_{l,j}}\right) \\ &= \phi_{pk}\left(c_l, \prod_{j=1}^{t} r_j^{x_{l,j}}\right) \quad , \end{aligned}$$

where the third equality follows from the linearity of the shuffle map ϕ_{pk}. Thus, we have $c'_l = \phi_{pk}(c_{\pi(l)}, r'_{\pi(l)})$ for some $r'_{\pi(l)}$, $l = 1, \ldots, N$ as expected. We stress that we do not claim that the $x_{l,i}$ values can be computed. In fact, when the order q of the largest cyclic subgroup of $\mathcal{C}_{pk}$ is not known, then it seems very hard to compute such values.

To prove soundness and that the knowledge error is negligible, it remains to show that we can extract the values described at the beginning of the proof with high probability. First we note that without loss we can combine the two subprotocols into a single protocol by exploiting their special soundness and zero-knowledge properties, i.e., we may use the same challenge for both protocols.

Consider the following thought experiment. For any prover $\mathcal{P}^*$, we construct a new prover $\mathcal{P}^+$ that honestly chooses $(e_1, \ldots, e_N)$ instead of the verifier, and then invokes $\mathcal{P}^*$. The new prover obviously takes part in a different protocol, but we can still consider the problem of extracting two accepting interactions with distinct challenges from $\mathcal{P}^+$. A standard rewinding argument (see for example Lemma 8 in [26]), shows that there exists a polynomial $T(\kappa)$ and an extractor $\mathcal{X}_0$ such that if the success probability of $\mathcal{P}^*$ is at least $\delta(\kappa)$, then $\mathcal{X}_0$ outputs, in expected time $O(T(\kappa)/(\delta(\kappa) - 2^{-\kappa_c}))$, a list $(e_{j,1}, \ldots, e_{j,N})$, $t_{j,0}, \ldots, t_{j,k}$, and r_j, satisfying our requirements for $j = 1, \ldots, N$.

We let the extractor $\mathcal{X}_1$ invoke the extractor $\mathcal{X}_0$ exactly N times and output the extracted set of values. It is clear that $\mathcal{X}_1$ runs in expected time $O(N \cdot T(\kappa)/(\delta(\kappa) - 2^{-\kappa_c}))$. We claim that if $\delta(\kappa)$ is not too small, then with positive probability the lists $(e_{j,1}, \ldots, e_{j,N})$ output by $\mathcal{X}_0$ determines a witness of $\mathscr{R}^{map}_{\phi_{pk}}$. We remind the reader that we do not claim that such a witness can be extracted efficiently, only that it exists, and we prove that it exists by showing that there exists a non-efficient extraction procedure that outputs it with high probability.

In the protocol, the list $(e_1, \ldots, e_N)$ is derived from a small random seed z using a PRG. We first analyze an idealized setting where the output from the PRG is perfectly random and then show that using a PRG is almost as good.

If a list $(e_1, \ldots, e_N)$ satisfies $e_i \neq e_{i'}$ for all $i \neq i'$, then we say that it is *internally unique*. It is easy to see that a randomly chosen list $(e_1, \ldots, e_N) \in [0, 2^{\kappa_e} - 1]^N$ is not internally unique with probability at most $N^2 2^{-\kappa_e}$.

We say that some lists $\overline{e}_1, \ldots, \overline{e}_j$ are in *general position*, if the set of linear combinations of these lists with coefficients in $\mathbb{Z}_q$ contains lists of the following form

$$\begin{aligned}
\overline{e}'_1 &= (1, 0, 0, \ldots, 0, e'_{1,j+1}, \ldots, e'_{1,N}) \\
\overline{e}'_2 &= (0, 1, 0, \ldots, 0, e'_{2,j+1}, \ldots, e'_{2,N}) \\
\vdots\ \vdots & \qquad\qquad\quad \vdots \\
\overline{e}'_j &= (0, 0, 0, \ldots, 1, e'_{j,j+1}, \ldots, e'_{2,N}) \ .
\end{aligned}$$

Suppose that some lists $\overline{e}_1, \ldots, \overline{e}_j$ in general position are given and consider the probability that a randomly chosen list $\overline{e}_{j+1}$ can be used to extend the set of lists in general position. By construction, $e_{j+1,j+1}$ is randomly chosen in $[0, 2^{\kappa_e} - 1]$, where 2^{κ_e} is smaller than any factor of q (the maximal order of any cyclic subgroup of $\mathcal{C}_{pk}$), and there are obviously at most κ/κ_e factors in q. Thus, from independence we conclude that the probability that $e_{j+1,j+1} - \sum_{l=1}^{j} e_{l,j+1} e_{j+1,l}$ is not invertible modulo q is bounded by $\frac{\kappa}{\kappa_e} 2^{-\kappa_e}$. Thus, the probability that a randomly chosen list can not be used to extend the sequence of lists in general position is bounded by $\frac{\kappa}{\kappa_e} 2^{-\kappa_e}$. In the protocol, the list $(e_1, \ldots, e_N)$ is derived from a random seed. If this would change the probabilities derived above more than by a negligible amount, then we could obviously break the PRG. Thus, using a PRG, the probability that the list $\overline{e}_{j+1}$ is not internally unique or does not extend the set of lists in general position is bounded by $N(N + \frac{\kappa}{\kappa_e}) 2^{-\kappa_e} + \epsilon_0(\kappa)$ for some negligible function $\epsilon_0(\cdot)$.

Suppose now that the first j calls to $\mathcal{X}_0$ resulted in internally unique lists in general position. Then during the next call to the extractor $\mathcal{X}_0$, the expected number of sampled lists is bounded by $\mathcal{X}_0$'s expected running time $O(T(\kappa)/(\delta(\kappa) - 2^{-\kappa_c}))$, which means that the expected number of generated lists that are either not internally unique or does not extend the already existing set of lists in general position is bounded by

$$O((N(N + \kappa/\kappa_e) 2^{-\kappa_e} + \epsilon_0(\kappa)) T(\kappa)/(\delta(\kappa) - 2^{-\kappa_c})) \ .$$

Thus, the probability that some list output by $\mathcal{X}_1$ is not internally unique, or that the lists are not in general position is upper bounded by

$$O((N(N + \kappa/\kappa_e) 2^{-\kappa_e} + \epsilon_0(\kappa)) 2^{-\kappa_e} T(\kappa)/(\delta(\kappa) - 2^{-\kappa_c})) \ .$$

This is negligible for a negligible $\delta(\kappa)$, which concludes the proof of soundness. □

B Proof of Equal Exponents For Pedersen Commitments

We provide a detailed description and analysis of the proof of equal exponents for the solution based on Pedersen commitments.

Protocol 19 (Proof of Equal Exponents)**.**
COMMON INPUT: A public key pk of a cryptosystem $\mathcal{CS}$, a generating set $\{h_1, \ldots, h_k\}$ of $\mathcal{C}_{pk}$, a cyclic group G_q that has the same order q as the largest cyclic subgroup of $\mathcal{C}_{pk}$, generators $g, g_1, \ldots, g_N$ of G_q, a commitment of exponents $a \in G_q$, ciphertexts $(c_1, \ldots, c_N) \in \mathcal{C}_{pk}$, a commitment (b_1, b_2), where $b_1 \in G_q$ and $b_2 \in \mathcal{C}_{pk}$.
PRIVATE INPUT: Exponents $e_1, \ldots, e_N \in [0, 2^{\kappa_e} - 1]$, $s_0, \ldots, s_k, e_0 \in \mathbb{Z}_q$ such that: $a = g^{e_0} \prod_{i=1}^N g_i^{e_i}$, $b_1 = g^{s_0} \prod_{i=1}^k g_i^{s_i}$, and $b_2 = \prod_{i=1}^k h_i^{s_i} \prod_{i=1}^N c_i^{e_i}$.

1. $\mathcal{P}$ chooses $t_1, \ldots, t_N \in [0, 2^{\kappa_e+\kappa_c+\kappa_r} - 1]$ and $t_0, l_0, \ldots, l_k \in \mathbb{Z}_q$, randomly, and computes and hands to $\mathcal{V}$ the following elements
$$\alpha = g^{t_0} \prod_{i=1}^N g_i^{t_i} \ , \ \ \beta_1 = g^{l_0} \prod_{i=1}^k g_i^{l_i} \ , \ \text{ and } \ \beta_2 = \prod_{i=1}^k h_i^{l_i} \prod_{i=1}^N c_i^{t_i} \ .$$
2. $\mathcal{V}$ chooses a challenge $c \in [0, 2^{\kappa_c} - 1]$ randomly and hands it to $\mathcal{P}$.
3. $\mathcal{P}$ computes and hands to $\mathcal{V}$ the following replies
$$\begin{aligned} d_0 &= ce_0 + t_0 \bmod q \ , \\ d_i &= ce_i + t_i \bmod 2^{\kappa_e+\kappa_c+\kappa_r} \ , \ \text{ for } i = 1, \ldots, N \ , \ \text{ and} \\ f_i &= cs_i + l_i \bmod q \ , \ \text{ for } i = 0, \ldots, k \end{aligned}$$
4. $\mathcal{V}$ checks that
$$g^{d_0} \prod_{i=1}^N g_i^{d_i} = a^c \alpha \ , \ \ g^{f_0} \prod_{i=1}^k g_i^{f_i} = b_1^c \beta_1 \ , \ \text{ and } \ \prod_{i=1}^k h_i^{f_i} \prod_{i=1}^N c_i^{d_i} = b_2^c \beta_2 \ . \quad (1)$$

Note that the prover can pre-compute α and the cost of computing β_1 may be ignored when N is much larger than k. Thus, the prover must essentially compute β_2 in the online phase. The verifier on the other hand must perform its verifications for both α and β_2 in the online phase.

Proposition 20. *Suppose that $q = \prod_{i=1}^t p_i$ with $p_i \geq 2^{\kappa_c}$. Then the protocol is an overwhelmingly complete sigma protocol for $\mathscr{R}_{ck,pk}^{eq}$.*

Proof. The zero-knowledge simulator chooses $c \in [0, 2^{\kappa_c} - 1]$, $d_0, f_i \in \mathbb{Z}_q$ for $i = 1, \ldots, k$, and $d_i \in [0, 2^{\kappa_e+\kappa_c+\kappa_r} - 1]$ for $i = 1, \ldots, N$ randomly and defines α, β_1 and β_2 by Equation (1). The resulting simulation is perfectly distributed, but due to the reduction modulo $2^{\kappa_e+\kappa_c+\kappa_r}$ the protocol only has overwhelming completeness. Suppose that accepting interactions: $(\alpha, \beta_1, \beta_2, c, d_0, \ldots, d_N, f_0, \ldots, f_k)$ and $(\alpha, \beta_1, \beta_2, c', d_0', \ldots, d_N', f_0', \ldots, f_k')$ with $c \neq c'$ are given. By construction $|c - c'| \leq 2^{\kappa_c} \leq p$ for each factor p of q. Thus, $c - c'$ is invertible modulo q. Furthermore, since q is the order of the largest cyclic subgroup of $\mathcal{C}_q$, the orders q_a, q_{b_1}, and q_{b_1} of a, b_1, and b_2 respectively divide q. Thus, $((c-c')^{-1} \bmod q) = (c-c')^{-1} \bmod q_a$ and correspondingly for q_{b_1}, and q_{b_1}. This implies that we may set $e_i = (c - c')^{-1}(d_i - d_i') \bmod q$ and $s_i = (c - c')^{-1}(f_i - f_i') \bmod q$, and conclude that $a = g^{e_0} \prod_{i=1}^N g_i^{e_i}$, $b_1 = g^{s_0} \prod_{i=1}^k g_i^{s_i}$, and $b_2 = \prod_{i=1}^k h_i^{s_i} \prod_{i=1}^N c_i^{e_i}$. □

B.1 A Special Pre-computation Trick For El Gamal

When we use the El Gamal cryptosystem the ciphertext space equals $G_q \times G_q$. In this case the verifier can use the following pre-computation technique based on batch proofs.

Let us write $c_i = (u_i, v_i)$. The verifier chooses $x \in [0, 2^{\kappa_e} - 1]$ randomly and computes $w_i = u_i g_i^x$. When it receives the reply to its challenge c it performs the verification as follows

$$g^{f_0} \prod_{i=1}^{k} g_i^{f_i} = b_1^c \beta_1 \ , \text{ and}$$
$$(g^{d_0}, 1) \prod_{i=1}^{k} h_i^{f_i} \prod_{i=1}^{N} (w_i, v_i)^{d_i} = (b_2(a, 1)^x)^c \beta_2(\alpha, 1)^x \ .$$

The reason for doing this is that the verifier can pre-compute all g_i^x, and that this reduces the online complexity by a factor of 2/3. A standard batch-proof argument implies that the above verification is essentially as sound as the original.

C Online Complexity Estimate In the El Gamal Case

In this section it is understood that we consider the *online* complexity of the protocol. For each of the prover and verifier, the complexity of the main part of Protocol 15 is roughly $2\frac{\kappa_e}{\kappa}N$ square-and-multiply exponentiations, and the complexity of Protocol 19 (the proof of knowledge of equal exponents in Appendix B) is roughly $2\frac{\kappa_e+\kappa_c+\kappa_r}{\kappa}N$ and for both the prover and verifier, using the special trick outlined above. Thus, the total complexity is $\frac{4\kappa_e+2\kappa_c+2\kappa_r}{\kappa}N$. For practical values, e.g., $\kappa = 1024$, $\kappa_e = \kappa_c = 80$ and $\kappa_r = 20$, this gives $\approx N/2$ exponentiations.

Essentially all exponentiations in the protocols are of the form $\prod_{i=1}^{N} u_i^{e_i}$ for some elements u_i and κ'-bit exponents e_i for some $\kappa' < \kappa$, which means that simultaneous exponentation [20] is directly applicable. The complexity of one square-and-multiply exponentiation with a κ'-bit exponent corresponds to κ' squarings and $\kappa'/2$ multiplications on average.

There are two ways the complexity is reduced in simultaneous exponentiation. Firstly, the squarings are done for all N elements at the same time, i.e., the cost for squaring is negligible when N is large. Secondly, the number of multiplications is reduced by pre-computing products of subsets of the elements u_i. One can only compute all possible combinations for a small number of elements at a time. Thus, the idea is to divide the elements into groups of w elements for some width w. For each such group $u_1, \ldots, u_w$ of elements and every $I \subset \{1, \ldots, w\}$ the product $\prod_{i \in I} u_i^{e_i}$ is then computed. These products can be computed using 2^w multiplications. Given all such products, multiplication by any combination of the elements $u_1, \ldots, u_w$ can be done at the cost of a single multiplication.

Thus, the average cost of computing $\prod_{i=1}^{N} u_i^{e_i}$ is roughly $\frac{N}{w} \cdot 2^w + \frac{N}{w}\kappa'$ multiplications, where κ' is the bitsize of the exponents. We conclude that the complexity of Protocol 15, using the parameters above, corresponds to $2(\frac{2^3}{3} + \frac{80}{3})N = 59N$ multiplications if we set $w = 3$. Similarly, the complexity of Protocol 15 corresponds to $2(\frac{2^4}{4} + \frac{80+80+20}{4})N = 98N$ if we set $w = 4$. This sums to $157N$ multiplications.

We may estimate the number of multiplications computed using square-and-multiply exponentiations by $1.5(4 \cdot 80 + 2 \cdot 80 + 2 \cdot 20)N \leq 780N$. Thus, if simultaneous exponentiation is used the complexity can be reduced by a factor of $154/780 \approx 1/5$. This gives a total complexity corresponding to $N/10$ square-and-multiply exponentiations.

However, optimized code such as GMP [16] already uses other tricks, such as sliding window exponentiation, which are to some extent in conflict with using simultaneous exponentiation [17]. Thus, one can not expect a reduction in complexity by a factor of $1/5$ starting from an optimized implementation.

D Solution Based On Fujisaki-Okamoto Commitments

An alternative to using Pedersen commitments is to use Fujisaki-Okamoto commitments [12]. This is less practical, since such commitments assume the availability of an RSA modulus (which is costly to generate jointly), but it allows construction of commitment-consistent proofs of shuffles for cryptosystems where the order of the group of ciphertexts is unknown.

D.1 Permutation Commitment

Construction 21 (Fujisaki-Okamoto Commitment)**.** The parameter generation algorithm $\mathsf{Gen}^\star$ outputs a random RSA modulus $n = pq$, where p and q are random $\kappa/2$-bit safe primes, and random generators $g, g_1, \ldots, g_N \in SQ_n$. On input $\pi \in \mathbb{S}_N$ and $r_1, \ldots, r_N \in [0, 2^{\kappa+\kappa_r} - 1]$, the commitment algorithm $\mathsf{Com}^\star$ computes $a_i = g^{r_i} g_{\pi^{-1}(i)}$, and outputs $(a_1, \ldots, a_N)$. The parameter algorithm Gen^e is identical to $\mathsf{Gen}^\star$. On input $(e_1, \ldots, e_N) \in [0, 2^{\kappa_e} - 1]$ and $r \in [0, 2^{\kappa+\kappa_r} - 1]$ the commitment algorithm Com^e computes $a = g^r \prod_{i=1}^N g_i^{e_i}$, and outputs $(a_1, \ldots, a_N)$.

It is technically convenient to consider a non-zero tuple $(n_0, \ldots, n_N)$ as an opening of any message if $g^{n_0} \prod_{i=1}^N g_i^{n_i} = 1$. Similarly, a tuple $(\beta, \eta, n_0, \ldots, n_N)$ such that $g^{n_0} \prod_{i=1}^N g_i^{n_i} = \beta^\eta$, $\eta \neq 0$, and $\eta \nmid n_i$ for some i, is also viewed as a valid opening of any message.

Proposition 22. *Both* $(\mathsf{Gen}^\star, \mathsf{Com}^\star)$ *and* $(\mathsf{Gen}^e, \mathsf{Com}^e)$ *of Construction 21 are statistically hiding and computationally binding under the strong RSA-assumption. The former is a permutation commitment of the latter.*

Proof. The statistical hiding property follows since the random exponents r_i and r are chosen randomly in $[0, 2^{\kappa+\kappa_r} - 1]$ and all generators generate the group of squares with overwhelming probability, where κ is an upper bound on the orders of g and g_i. In both schemes it holds that if two distinct valid openings can be computed, then we can write $g^{n_0} \prod_{i=1}^N g_i^{n_0} = 1$ for some $n_0, \ldots, n_N \in \mathbb{Z}$, where not all are zero. In Corollary 34 we extend known results [9] and show that this contradicts the strong RSA assumption.

For any Fujisaki-Okamoto commitment $(a_1, \ldots, a_N) = \mathsf{Com}^\star_{ck}(\pi, r_{[N]})$ we have $\prod_{i=1}^N a_i^{e_i} = g^{\sum_{i=1}^N r_i e_i} \prod_{i=1}^N g_i^{e_{\pi(i)}}$ which equals a commitment $\mathsf{Com}^e_{ck}(e_{\pi(1)}, \ldots, e_{\pi(N)}, \sum_{i=1}^N r_i e_i)$, which proves the second claim. The only potential issue is that the bit-size of $\sum_{i=1}^N r_i e_i$ is larger than the bit-size of any individual r_i. Thus, formally speaking we need to consider two variants of the cryptosystem with different randomness spaces, but we do not make this explicit. □

The above forms of (generalized) Fujisaki-Okamoto commitments are used implicitly in Wikström [30].

D.2 Proof of Equal Exponents For Fujisaki-Okamoto Commitments

The next protocol can be used to provide a proof of equal exponents both in the case where Pedersen commitments are used and when Fujisaki-Okamoto commitments are used. In the first case it is useful as a "bridge", when the order of $\mathcal{C}_{pk}$ is unknown or if it is infeasible to construct a group G_q for which the DL assumption holds, where q is the order of the largest cyclic subgroup of $\mathcal{C}_{pk}$. In the second case it is used directly in the natural way.

Protocol 23 (Generic Proof of Equal Exponents).
COMMON INPUT: Fujisaki-Okamoto parameters n and $h_i \in SQ_n$ for $i = 1, \ldots, n_0$, $z_1, z_2 \in \mathbb{Z}_n^*$, and group elements $y_j \in G_j$, $g_{j,i} \in G_j$ for $i = 0, \ldots, n_j$ and $j = 1, 2, 3$ and with $n_3 = n_1 + n_2$.
PRIVATE INPUT: Exponents $w_1, w_2, x_{1,0}, x_{2,0} \in [0, 2^{\kappa+\kappa_r} - 1]$ and $x_{j,i} \in [0, 2^{\kappa_j} - 1]$ for $i = 1, \ldots, n_j$ and $j = 1, 2$ such that $y_1 = \prod_{i=0}^{n_1} g_{1,i}^{x_{1,i}}$, $y_2 = \prod_{i=0}^{n_2} g_{2,i}^{x_{2,i}}$, and $y_3 = \prod_{i=1}^{n_1} g_{3,i}^{x_{1,i}} \prod_{i=n_1+1}^{n_1+n_2} g_{3,i}^{x_{2,i}}$, $z_1 = h_0^{w_1} \prod_{i=1}^{n_1} h_i^{x_{1,i}}$, and $z_2 = h_0^{w_2} \prod_{i=1}^{n_2} h_i^{x_{2,i}}$.

1. $\mathcal{P}$ chooses $s_1, s_2, r_{1,0}, r_{2,0} \in [0, 2^{\kappa+2\kappa_r+\kappa_c} - 1]$, $r_{j,i} \in [0, 2^{\kappa_j+\kappa_c+\kappa_r} - 1]$ randomly and computes and hands to $\mathcal{V}$ the following elements
$$\gamma_1 = \prod_{i=0}^{n_1} g_{1,i}^{r_{1,i}} \ , \ \gamma_2 = \prod_{i=0}^{n_2} g_{2,i}^{r_{2,i}} \ , \ \gamma_3 = \prod_{i=1}^{n_1} g_{3,i}^{r_{1,i}} \prod_{i=n_1+1}^{n_1+n_2} g_{3,i}^{r_{2,i}}$$
$$\zeta_1 = h_0^{s_1} \prod_{i=1}^{n_1} h_i^{x_{1,i}} \ , \ \text{and} \ \zeta_2 = h_0^{s_2} \prod_{i=1}^{n_2} h_i^{x_{2,i}} \ .$$
2. $\mathcal{V}$ chooses a challenge $c \in [0, 2^{\kappa_c} - 1]$ randomly and hands it to $\mathcal{P}$.
3. $\mathcal{P}$ computes and hands to $\mathcal{V}$ the following replies
$$\begin{aligned} d_{j,0} &= c x_{j,i} + r_{j,i} \bmod 2^{\kappa+2\kappa_r+\kappa_c} \ , \\ d_{j,i} &= c x_{j,i} + r_{j,i} \bmod 2^{\kappa_j+\kappa_c+\kappa_r} \ \text{for } i = 1, \ldots, N \\ f_j &= c w_j + s_j \bmod 2^{\kappa+2\kappa_r+\kappa_c} \ \text{for } j = 1, 2 \ . \end{aligned}$$
4. $\mathcal{V}$ checks that
$$\prod_{i=0}^{n_1} g_{1,i}^{d_{1,i}} = y_1^c \gamma_1 \ , \ \prod_{i=0}^{n_2} g_{2,i}^{d_{2,i}} = y_2^c \gamma_2 \ , \ \prod_{i=1}^{n_1} g_{3,i}^{d_{1,i}} \prod_{i=n_1+1}^{n_1+n_2} g_{3,i}^{d_{2,i}} = y_3^c \gamma_3 \ , \tag{2}$$
$$h_0^{f_1} \prod_{i=1}^{n_1} h_i^{d_{1,i}} = z_1^c \zeta_1 \ , \ \text{and} \ h_0^{f_2} \prod_{i=1}^{n_2} h_i^{d_{2,i}} = z_2^c \zeta_2 \ . \tag{3}$$

Definition 24. Let $ck = (n, (h_i)_{i=1}^{n_0})$ be defined as in the protocol. Then we define the relation $\mathscr{R}_{ck}^{geq}$ to consist of all

$$\big((ck, z_0, z_1, y_1, y_2, y_3, (g_{1,i})_{i=1}^{n_1}, (g_{2,i})_{i=1}^{n_2}, (g_{3,i})_{i=1}^{n_3}), (w_1, w_2, (x_{1,i})_{i=0}^{N}, (x_{2,i})_{i=0}^{N})\big) \ ,$$

such that

$$y_1 = \prod_{i=0}^{n_1} g_{1,i}^{x_{1,i}} \ , \quad y_2 = \prod_{i=0}^{n_2} g_{2,i}^{x_{2,i}} \ , \quad y_3 = \prod_{i=1}^{n_1} g_{3,i}^{x_{1,i}} \prod_{i=n_1+1}^{n_1+n_2} g_{3,i}^{x_{2,i}}$$

$$z_1 = h_0^{y_1} \prod_{i=1}^{n_1} h_i^{x_{1,i}} \ , \quad \text{and} \quad z_2 = h_0^{y_2} \prod_{i=1}^{n_2} h_i^{x_{2,i}} \ .$$

Proposition 25. *For every $pk \in \mathcal{PK}$ and $ck \in \mathcal{CK}$, the protocol is a sigma protocol with overwhelming completeness for the relation $\mathscr{R}_{ck,pk}^{geq} \vee \mathscr{R}_{ck}^{twin}$.*

Proof. The zero-knowledge simulator chooses $c \in [0, 2^{\kappa_c} - 1]$, $d_{j,0} \in [0, 2^{\kappa + 2\kappa_r + \kappa_c} - 1]$, $d_{j,i} \in [0, 2^{\kappa_j + \kappa_c + \kappa_r} - 1]$, and $f_j \in [0, 2^{\kappa + 2\kappa_r + \kappa_c} - 1]$, randomly and defines γ_1, γ_2, γ_3, ζ_1, and ζ_2 by Equations (2)-(3). This gives a perfectly distributed interaction.

Suppose have accepting interactions

$$(\gamma_1, \gamma_2, \gamma_3, \zeta_1, \zeta_2, c, (d_{1,i})_{i=0}^{N}, (d_{2,i})_{i=0}^{N}, f_1, f_2) \text{ and}$$
$$(\gamma_1, \gamma_2, \gamma_3, \zeta_1, \zeta_2, c', (d'_{1,i})_{i=0}^{N}, (d'_{2,i})_{i=0}^{N}, f'_1, f'_2) \ ,$$

with $c \neq c'$. If $c - c'$ divides $f_j - f'_j$ and $d_{j,i} - d'_{j,i}$ for $i = 0, \ldots, N$, then we may define $s_j = (f_j - f'_j)/(c - c')$ and $r_{j,i} = (d_{j,i} - d'_{j,i})/(c - c')$, which gives a witness for $\mathscr{R}_{ck,pk}^{geq}$. Otherwise we have found a witness of $\mathscr{R}_{ck}^{twin}$. □

Below we show how a proof of equal exponents for $\mathscr{R}_{ck,pk}^{eq} \vee \mathscr{R}_{ck}^{twin}$ can be derived from the above protocol when Pedersen commitments are used by the mix-servers to commit to their permutations.

Protocol 26 (Proof of Equal Exponents).
COMMON INPUT: A public key pk of a cryptosystem $\mathcal{CS}$, a generating set $\{h_1, \ldots, h_k\}$ of $\mathcal{C}_{pk}$, generators $g, g_1, \ldots, g_N$ of a group G_q of order q, a commitment of exponents $a \in G_q$, ciphertexts $(c_1, \ldots, c_N) \in \mathcal{C}_{pk}$, a commitment (b_1, b_2), where $b_1 \in G_q$ and $b_2 \in \mathcal{C}_{pk}$.
PRIVATE INPUT: Exponents $e_1, \ldots, e_N \in [0, 2^{\kappa_e} - 1]$, $s_0, \ldots, s_k, e_0 \in \mathbb{Z}_q$ such that: $a = g^{e_0} \prod_{i=1}^{N} g_i^{e_i}$, $b_1 = g^{s_0} \prod_{i=1}^{k} g_i^{s_i}$, and $b_2 = \prod_{i=1}^{k} h_i^{s_i} \prod_{i=1}^{N} c_i^{e_i}$.

1. $\mathcal{P}$ chooses $y_1, y_2 \in [0, 2^{\kappa + \kappa_r} - 1]$ randomly and computes and hands to $\mathcal{V}$

$$z_1 = g_0^{y_1} \prod_{i=1}^{N} g_i^{e_i} \text{ and } z_2 = g_0^{y_2} \prod_{i=1}^{k} g_i^{s_i} \ .$$

2. Invoke Protocol 23 on the above elements and exponents.

Finally, we show how Protocol 23 can be invoked directly, when Fujisaki-Okamoto commitments are used by the mix-servers to commit to their permutations.

Protocol 27 (Proof of Equal Exponents)**.**
COMMON INPUT: A public key pk of a cryptosystem $\mathcal{CS}$, a generating set $\{h_1, \ldots, h_k\}$ of $\mathcal{C}_{pk}$, Fujisaki-Okamoto parameters $n, g, g_1, \ldots, g_N$, a commitment of exponents $a \in \mathbb{Z}_n^*$, ciphertexts $(c_1, \ldots, c_N) \in \mathcal{C}_{pk}$, a commitment (b_1, b_2), where $b_1 \in \mathbb{Z}_n^*$ and $b_2 \in \mathcal{C}_{pk}$.
PRIVATE INPUT: Exponents $e_1, \ldots, e_N \in [0, 2^{\kappa_e} - 1]$, $s_0, \ldots, s_k, e_0 \in [0, 2^{\kappa+\kappa_r} - 1]$ such that: $a = g^{e_0} \prod_{i=1}^N g_i^{e_i}$, $b_1 = g^{s_0} \prod_{i=1}^k g_i^{s_i}$, and $b_2 = \prod_{i=1}^k h_i^{s_i} \prod_{i=1}^N c_i^{e_i}$.

Invoke Protocol 23 on the above elements and exponents (some of the elements are simply eliminated).

E Proofs of Knowledge of Shuffle-Friendly Maps

Protocol 28 (Proof of Re-encryption For El Gamal)**.**

COMMON INPUT: Prime order group G_p, El Gamal public key (y, g), and ciphertexts (u, v) and (u', v').
PRIVATE INPUT: Exponent $r \in \mathbb{Z}_p$ such that $(u', v') = (g^r u, y^r v)$.

1. $\mathcal{P}$ chooses $s \in \mathbb{Z}_p$ randomly, computes $(\mu, \nu) = (g^s, y^s)$, and hands (μ, ν) to $\mathcal{V}$.
2. $\mathcal{V}$ chooses a challenge $c \in \mathbb{Z}_p$ randomly and hands it to $\mathcal{P}$.
3. $\mathcal{P}$ computes $d = cr + s$ and hands d to $\mathcal{V}$.
4. $\mathcal{V}$ verifies that $(g^d, y^d) = \big((u'/u)^c \mu, (v'/v)^c \nu\big)$.

Protocol 29 (Proof of Partial Decryption and Re-encryption For El Gamal)**.**

COMMON INPUT: Prime order group G_p, El Gamal public keys (y, g) and z, and ciphertexts (u, v) and (u', v').
PRIVATE INPUT: Exponents $w \in \mathbb{Z}_p$ and $r \in \mathbb{Z}_p$ such that $z = g^w$ and $(u', v') = (g^r u, (y/z)^r u^{-w} v)$.

1. $\mathcal{P}$ chooses $s, t \in \mathbb{Z}_p$ randomly, computes $(\zeta, \mu, \nu) = (g^t, g^s, (y/z)^s u^{-t})$, and hands (ζ, μ, ν) to $\mathcal{V}$.
2. $\mathcal{V}$ chooses a challenge $c \in \mathbb{Z}_p$ randomly and hands it to $\mathcal{P}$.
3. $\mathcal{P}$ computes $d = cr + s$ and $f = cw + t$ and hands (d, f) to $\mathcal{V}$.
4. $\mathcal{V}$ verifies that $(g^f, g^d, (y/z)^d u^{-f}) = \big(z^c \zeta, (u'/u)^c \mu, (v'/v)^c \nu\big)$.

F Assumptions

F.1 Discrete Logarithm Assumptions

Definition 30. The discrete logarithm assumption for a cyclic group G_q of order q, with generator g, states that if $x \in \mathbb{Z}_q$ is randomly chosen, then for every polynomial time adversary A, $\Pr[A(g, g^x) = x]$ is negligible.

Definition 31. The decision Diffie-Hellman assumption for a cyclic group G_q of order q, with generator g, states that if $x, y, r \in \mathbb{Z}_q$ are randomly chosen, then for every polynomial time adversary A: $|\Pr[A(g, g^x, g^y, g^{xy}) = 1] - \Pr[A(g, g^x, g^y, g^r) = 1]|$ is negligible.

F.2 Strong RSA Assumption

Definition 32. The strong RSA assumption states that if p and q are randomly chosen $\kappa/2$-bit safe primes, $n = pq$, and $g \in SQ_n$ is randomly chosen, then for every polynomial time adversary A, $\Pr[A(n, g) = (\beta, \eta) \wedge \eta \neq \pm 1 \wedge \beta^\eta = g \bmod n]$ is negligible.

The following lemma is proved in [9].

Lemma 33. *Let (n, g) be randomly chosen RSA parameters as defined in Definition 32, and let $h \in SQ_n$ be randomly chosen. Then for every polynomial time adversary A*

$$\begin{aligned}
&\Pr[A(n, g, h) = (\beta, \eta_0, \eta_1, \eta_2) \wedge |\eta_0| \in [1, 2^{\kappa/2-1}] \\
&\quad \wedge (\eta_0 \nmid \eta_1 \vee \eta_0 \nmid \eta_2) \wedge \beta^{\eta_0} = g^{\eta_1} h^{\eta_2} \bmod n] \\
&\Pr[A(n, g, h) = (\beta, \eta_1, \eta_2) \wedge (\eta_1, \eta_2) \neq (0, 0) \wedge g^{\eta_1} = h^{\eta_2} \bmod n]
\end{aligned}$$

are negligible in κ under the strong RSA assumption.

Corollary 34. *Let (n, g) be randomly chosen RSA parameters as defined in Definition 32, and let $g_1, \ldots, g_N \in SQ_n$ be randomly chosen. Then for every polynomial N and polynomial time adversary A*

$$\begin{aligned}
&\Pr\Big[A(n, (g_i)_{i=1}^N) = (\beta, (\eta_i)_{i=0}^N) \wedge |\eta_0| \in [1, 2^{\kappa/2-1}] \wedge (\exists i > 0 : \eta_0 \nmid \eta_i) \\
&\qquad \wedge \beta^{\eta_0} = \prod\nolimits_{i=1}^N g_i^{\eta_i} \bmod n\Big] \\
&\Pr\Big[A(n, (g_i)_{i=1}^N) = (\eta_i)_{i=1}^N \wedge (\exists i : \eta_i \neq 0) \wedge \prod\nolimits_{i=1}^N g_i^{\eta_i} = 1 \bmod n\Big]
\end{aligned}$$

are negligible in κ under the strong RSA assumption.

Proof. Suppose that A contradicts the first claim. Then we can construct an adversary A' that contradicts the first claim of Lemma 33 as follows. It chooses an index $t \in [1, N]$ randomly and defines $g_t = h$ and $g_i = g^{r_i}$ for $i \neq t$, where $r_i \in [0, 2^{\kappa+\kappa_r} - 1]$ is chosen randomly for $i \neq t$. Then it invokes A on n and these generators. When it returns

$(\beta, (\eta_i)_{i=0}^N)$, A' defines $\beta' = \beta$, $\eta_0' = \eta_0$, $\eta_2' = \eta_t$, and $\eta_1' = \sum_{i \neq t} \eta_i r_i$, and returns $(\beta', \eta_0', \eta_1', \eta_2')$. Conditioned on $(\beta, (\eta_i)_{i=0}^N)$ satisfying the requirements in the lemma, the probability that $\eta_0 \nmid \eta_t$ is $1/N$, and by construction we have $\beta^{\eta_0} = g^{\eta_1} h^{\eta_2} \bmod n$. Thus, A' contradicts the first claim of Lemma 33.

Suppose that A contradicts the second claim. Then the following adversary A'' contradicts the second claim of Lemma 33. It defines $g_i = g^{r_i}$ for randomly chosen $r_i \in [0, 2^{\kappa + \kappa_r} - 1]$ and invokes A on n and these generators. When A returns $(\eta_i)_{i=1}^N$, it defines $\eta_1' = \sum_{i=1}^N r_i \eta_i$ and $\eta_2' = 0$. Then it returns (η_1', η_2'). It is easy to see that conditioned on $\eta_i \neq 0$ for some i, η_1' is non-zero with overwhelming probability, since there are exponentially many r_i consistent with g_i for every $i = 1, \ldots, N$.

□