A Commitment-Consistent Proof of a Shuffle

Douglas Wikström

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

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 [23]. The first rigorous definition of security was given by Abe and Imai [1], but they did not construct a scheme satisfying their construction. Wikström [25] gives the first definition of a universally composable (UC) mix-net, the first UC-secure construction, and also a more efficient UC-secure scheme [26]. An important building block in the construction of a mix-net is 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 [18] and Furukawa and Sako [13].

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 407–421, 2009.

1.1 Mix-Nets Based on Homomorphic Cryptosystems

Recall the mix-net of Sako and Kilian [23]. They present their scheme in terms of the El Gamal cryptosystem [14], 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 [22]. 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 [27].

Alternative Constructions. In the scheme of Furukawa et al. [12], 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 [26], 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 [18] a proof of a shuffle for the first type of mix-net is described as well [19]. 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 [16] and several are actually flawed [1,9,24].

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. [12,13,18,15,26], 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., [13,15,18,26]. 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 $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \rightarrow \mathcal{C}_{pk}$ 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 [18] and Furukawa and Sako [13]. 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 $\mathbb{S}_N$.

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 full version 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 $r_1, r_2 \in \mathcal{R}_{pk}$.

We denote the set $\{1, \ldots, l\}$ by $[l]$ and sometimes denote a list of elements $(a_1, \ldots, a_l)$ by $a_{[l]}$.

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.

1. Verifier 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. Prover 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. Due to space restrictions, we can not go into the details of any particular proof of a shuffle, but we can 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 [18] and Furukawa and Sako [13], 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 parties 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 3 in full version 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 [21] $(a_1, \ldots, a_N) = (g^{r_1} g_{\pi^{-1}(1)}, \ldots, g^{r_N} g_{\pi^{-1}(N)})$ 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 1 (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 [21] to commit to permutations is not novel, e.g., it is used implicitly in [13], but the observation that a commitment of the first kind can be transformed into a commitment of the second kind seems new.

Proposition 1. *Both* $(\mathsf{Gen}^\star, \mathsf{Com}^\star)$ *and* $(\mathsf{Gen}^e, \mathsf{Com}^e)$ *of Construction 1 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 the full version.

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 2. *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 3. *The relation* $\mathscr{R}_{ck}^{perm}$ *consists of all* $\big(ck, (a^\star, s_{[N]}, s_0, s'_{[N]}, s'_0)\big)$ *such that* $\mathsf{Map}_{ck}(a^\star, s_{[N]}) = \mathsf{Com}^e_{ck}((s_{\pi(1)}, \ldots, s_{\pi(N)}), s_0)$, $\mathsf{Map}_{ck}(a^\star, s'_{[N]}) = \mathsf{Com}^e_{ck}((s'_{\pi'(1)}, \ldots, s'_{\pi'(N)}), s'_0)$, $\pi \neq \pi'$, *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 4. *The relation* $\mathscr{R}_{ck}^{open}$ *consists of all* $\big((ck, a^\star), (\pi, r^\star)\big)$ *such that* $a^\star = \mathsf{Com}^\star_{ck}(\pi, r^\star)$.

Protocol 1 (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 2. *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 the full version. 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 1.

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 5. *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 the full version.

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 [11] 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 [26]. 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 the full version.

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 [12], or if a special key

set-up is used one can avoid random re-encryption entirely [26]. There are also at least two types of shuffles of (variants of) Paillier [20] ciphertexts. A careful look at these shuffles reveal that they are all defined by evaluating a homomorphic map and permuting the result.

Definition 6. *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 1. 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 [26] can be described similarly.

Example 2. 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 $\phi_n(c, r) = cr^n \bmod n^2$.

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 7 (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 3 (Example 1 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 parameter $ck = (g_1, g_2)$, then the relation consists of all pairs $\big(((g, y), (g_1, g_2), (u, v), b_1, b_2), (r, s_0, s_1, s_2)\big)$ 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 the full version.

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 2 (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 $\mathcal{V}$
$$b_1 = \mathsf{Com}_{ck}^e((t_1, \ldots, t_k), t_0) \text{ and } b_2 = \prod\nolimits_{i=1}^{k} h_i^{t_i} \prod\nolimits_{i=1}^{N} (c'_i)^{e_{\pi(i)}} \ .$$
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
$$b_1 = \mathsf{Com}_{ck}^e((t_1, \ldots, t_k), t_0) \ , \ b_2 = \prod\nolimits_{i=1}^{k} h_i^{t_i} \prod\nolimits_{i=1}^{N} (c'_i)^{e'_i} \ , \text{ and}$$
$$a = \mathsf{Com}_{ck}^e((e'_1, \ldots, e'_N), e_0) \ .$$
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
$$b_1 = \mathsf{Com}_{ck}^e((t_1, \ldots, t_k), t_0) \text{ and } b_2 = \prod\nolimits_{i=1}^{k} h_i^{t_i} \phi_{pk}\left(\prod\nolimits_{i=1}^{N} c_i^{e_i}, r\right) \ .$$

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 [18]). 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 8. *Let $pk \in \mathcal{PK}$, let ϕ_{pk} be a shuffle-friendly map for pk. Then we define the shuffle relation $\mathscr{R}_{\phi_{pk}}^{shuf}$ 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 3. *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 an sound public-coin honest verifier statistical zero-knowledge proof of the relation $\mathscr{R}^{shuf}_{\phi_{pk}} \vee \mathscr{R}^{twin}_{ck} \vee \mathscr{R}^{perm}_{ck}$, and 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 1. It is observed in [26] 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 the full version. 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 1) and a very efficient online part (Protocol 2), 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 1 (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 2 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 [10,21]. 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 [27] 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 [21]. 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 [17] is used, giving a complexity corresponding to $N/10$ square-and-multiply exponentations (see full version 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.

Acknowledgments

We thank Jun Furukawa and Andy Neff for helpful comments.

References

1. Abe, M., Imai, H.: Flaws in some robust optimistic mix-nets. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 39–50. Springer, Heidelberg (2003)
2. Adida, B., Wikström, D.: How to shuffle in public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
3. Adida, B., Wikström, D.: Offline/online mixing. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 484–495. Springer, Heidelberg (2007)
4. Bellare, M., Garay, J.A., Rabin, T.: Batch verification with applications to cryptography and checking. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 170–191. Springer, Heidelberg (1998)
5. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: 26th ACM Symposium on the Theory of Computing (STOC), pp. 544–553. ACM Press, New York (1994)
6. Chaum, D.: Untraceable electronic mail, return addresses and digital pseudo-nyms. Communications of the ACM 24(2), 84–88 (1981)
7. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

8. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
9. Desmedt, Y., Kurosawa, K.: How to break a practical MIX and design a new one. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 557–572. Springer, Heidelberg (2000)
10. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 427–438. IEEE Computer Society Press, Los Alamitos (1987)
11. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
12. Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An implementation of a universally verifiable electronic voting scheme based on shuffling. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 16–30. Springer, Heidelberg (2003)
13. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
14. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
15. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2003)
16. Jakobsson, M., Juels, A., Rivest, R.: Making mix nets robust for electronic voting by randomized partial checking. In: 11th USENIX Security Symposium, pp. 339–353. USENIX (2002)
17. Menezes, A., Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
18. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: 8th ACM Conference on Computer and Communications Security (CCS), pp. 116–125. ACM Press, New York (2001)
19. Neff, A.: Private communication (May 2008)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
22. Pfitzmann, B.: Breaking an efficient anonymous channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)
23. Sako, K., Killian, J.: Reciept-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
24. Wikström, D.: Five practical attacks for "optimistic mixing for exit-polls". In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 160–174. Springer, Heidelberg (2004)
25. Wikström, D.: A universally composable mix-net. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 315–335. Springer, Heidelberg (2004)
26. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)
27. Wikström, D.: Simplified submission of inputs to protocols. Cryptology ePrint Archive, Report 2006/259 (2006), `http://eprint.iacr.org/`