# A Sender Verifiable Mix-Net and a New Proof of a Shuffle

Douglas Wikström

Royal Institute of Technology (KTH)
KTH, Nada, SE-100 44 Stockholm, Sweden
dog@nada.kth.se

**Abstract.** We introduce the first El Gamal based mix-net in which each mix-server partially decrypts and permutes its input, i.e., no re-encryption is necessary. An interesting property of the construction is that a sender can verify non-interactively that its message is processed correctly. We call this *sender verifiability*.
The mix-net is provably UC-secure against static adversaries corrupting any minority of the mix-servers. The result holds under the decision Diffie-Hellman assumption, and assuming an ideal bulletin board and an ideal zero-knowledge proof of knowledge of a correct shuffle.
Then we construct the first proof of a decryption-permutation shuffle, and show how this can be transformed into a zero-knowledge proof of knowledge in the UC-framework. The protocol is sound under the strong RSA-assumption and the discrete logarithm assumption.
Our proof of a shuffle is not a variation of existing methods. It is based on a novel idea of independent interest, and we argue that it is at least as efficient as previous constructions.

## 1 Introduction

The notion of a mix-net was invented by Chaum [10]. Properly constructed a mix-net takes a list of cryptotexts and outputs the cleartexts permuted using a secret random permutation. Usually a mix-net is realized by a set of mix-servers organized in a chain that collectively execute a protocol. Each mix-server receives a list of encrypted messages from the previous mix-server, transforms them, using partial decryption and/or random re-encryption, reorders them, and outputs the result. The secret permutation is shared by the mix-servers.

### 1.1 Previous Work

Chaum's original "anonymous channel" [10,40] enables a sender to send mail anonymously. When constructing election schemes [10,17,42,47,39] a mix-net can be used to ensure that the vote of a given voter cannot be revealed. Abe gives an efficient construction of a general mix-net [2], and argues about its properties. Jakobsson has written (partly with Juels) more general papers on the topic of mixing [30,31,32] focusing on efficiency. There are two known approaches to proving a correct shuffle efficiently. These are introduced by Furukawa et al. [19,20,21], and Neff [37,38] respectively. Groth [27] generalizes Neff's protocol to form an abstract protocol for any homomorphic cryptosystem.

Desmedt and Kurosawa [13] describe an attack on a protocol by Jakobsson [30]. Similarly Mitomo and Kurosawa [36] exhibit a weakness in another protocol by Jakobsson [31]. Pfitzmann has given some general attacks on mix-nets [44,43], and Michels and Horster give additional attacks in [35]. Wikström [48] gives several attacks for a protocol by Golle et al. [26]. He also gives attacks for the protocols by Jakobsson [31] and Jakobsson and Juels [33]. Abe [3] has independently found related attacks.

Canetti [9], and independently Pfitzmann and Waidner [45] proposed security frameworks for reactive processes. We use the former universal composability (UC) framework. Both frameworks have composition theorems, and are based on older definitional work. The initial ideal-model based definitional approach for secure function evaluation is informally proposed by Goldreich, Micali, and Wigderson in [22]. The first formalizations appear in Goldwasser and Levin [24], Micali and Rogaway [34], and Beaver [5]. See [8,9] for an excellent background on these definitions.

Wikström [49] defines the notion of a mix-net in the UC-framework, and provides a construction that is provably secure against static adversaries under the decisional Diffie-Hellman assumption. The scheme is practical only when the number of mix-servers is small.

### 1.2 Contributions

We introduce a new type of El Gamal based mix-net in which each mix-server only decrypts and permutes its input. No re-encryption is necessary. This allows an individual sender to verify non-interactively that its message was processed correctly, i.e., the scheme is *sender verifiable*. Although some older constructions have this property, our is the first provably secure scheme.

Then we give the first proof of a decrypt-permutation shuffle of El Gamal cryptotexts. There are two known approaches, [37,27] and [19], to construct such a protocol, but our solution is based on a novel idea of independent interest, and we argue that it is at least as efficient as previous schemes.

We also give the first transformation of a proof of a shuffle into an efficient zero-knowledge proof of knowledge in the UC-framework. An important technical advantage of the new decrypt and permute construction is that witnesses are much smaller than for previous shuffle relations.

Combined, our results give a mix-net that is provably UC-secure against static adversaries corrupting any minority of the mix-servers. The mix-net is efficient for any number of mix-servers, improving the result in Wikström [49].

### 1.3 Outline of the Paper

The paper is organized as follows. Notation is introduced in Section 2. In Section 3 we define the ideal mix-net functionality. A partial result in this direction is given in Section 4, where we describe a sender verifiable mix-net and discuss sender verifiability. In Section 5 we describe a zero-knowledge proof of knowledge that a mix-server processes its input correctly. Then in Section 6 we transform this into a realization of an ideal zero-knowledge functionality in the UC-framework. Proofs of all claims are given in the full version [50] of this paper.

## 2 Notation

Throughout, $S_1, \ldots, S_N$ denote senders and $M_1, \ldots, M_k$ mix-servers. All participants are modeled as interactive Turing machines. We abuse notation and use $S_i$ and $M_j$ to denote both the machines themselves and their identity. We denote the set of permutations of $N$ elements by $\Sigma_N$. We use the term "randomly" instead of "uniformly and independently at random". A function $f : \mathbb{N} \rightarrow [0, 1]$ is said to be negligible if for each $c > 0$ there exists a $K_0 \in \mathbb{N}$ such that $f(K) < K^{-c}$ for $K > K_0 \in \mathbb{N}$. A probability $p(K)$ is overwhelming if $1 - p(K)$ is negligible.

We assume that $G_q$ is a group of prime order $q$ with generator $g$ for which the Decision Diffie-Hellman (DDH) Assumption holds. Informally, it means that it is infeasible to distinguish the distributions $(g^\alpha, g^\beta, g^{\alpha\beta})$ and $(g^\alpha, g^\beta, g^\gamma)$ when $\alpha, \beta, \gamma \in \mathbb{Z}_q$ are randomly chosen. This implies that also the Discrete Logarithm (DL) assumption holds, namely that it is infeasible to compute the logarithm in base $g$ of a random element in $G_q$. For concreteness we let $G_q$ be a subgroup of prime order $q$ of the multiplicative group $\mathbb{Z}_p^*$ for some prime $p$. When we say that an element in $\mathbb{Z}_q$ is prime, we mean that its representative in $\{0, \ldots, q-1\}$ is a prime when considered as an integer.

We review the El Gamal [14] cryptosystem employed in $G_q$. The private key $x$ is generated by choosing $x \in \mathbb{Z}_q$ randomly. The corresponding public key is $(g, y)$, where $y = g^x$. Encryption of a message $m \in G_q$ using the public key $(g, y)$ is given by $E_{(g,y)}(m, r) = (g^r, y^r m)$, where $r$ is chosen randomly from $\mathbb{Z}_q$, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key $x$ is given by $D_x(u, v) = u^{-x} v = m$.

We also use an RSA modulus $\mathbf{N} = \mathbf{pq}$, where $\mathbf{p}$ and $\mathbf{q}$ are safe primes. We denote by $\mathrm{QR}_\mathbf{N}$ the group of squares in $\mathbb{Z}_\mathbf{N}^*$ and adopt the convention that any element $\mathbf{b}$ in $\mathrm{QR}_\mathbf{N}$ is written in boldface. We assume that the strong RSA-assumption holds for such rings. Informally, it means that given random $(\mathbf{N}, \mathbf{h})$, where $\mathbf{h} \in \mathrm{QR}_\mathbf{N}$, it is infeasible to find a non-trivial $e$th root $\mathbf{b}$ of $\mathbf{h}$, i.e., an $e \neq \pm 1$ such that $\mathbf{b}^e = \mathbf{h}$. This differs from the RSA-assumption in that $e$ is not fixed.

The primary security parameter $K_1$ is the number of bits in $q$. Several other security parameters are introduced later in the paper. We denote by $\mathsf{PRG}$ a pseudo-random generator (cf. [23]). We denote by Sort the algorithm that given a list of strings as input outputs the same set of strings in lexicographical order.

### 2.1 The Universally Composable Security Framework

We analyze the security of our protocols in the Universally Composable (UC) security framework of Canetti [9]. There are several variants and extensions of this framework, but we consider a plain model with asynchronous authenticated communication. In the full version [50] we give a formal definition of this model. Here we only indicate how our notation differs from the standard [9].

The notion of a communication model, $\mathcal{C}_{\mathcal{I}}$, used below is not explicit in Canetti [9]. It works as a router between participants and between participants and ideal functionalities. Given the input $((A_1, B_1, C_1, \ldots), \ldots, (A_s, B_s, C_s, \ldots))$ it interprets $A_j$ as the receiver of $(B_j, C_j, \ldots)$. The adversary cannot read the correspondence with ideal functionalities, but it has full control over when a message is delivered.

Our results hold for both blocking and non-blocking adversaries, where a blocking adversary is allowed to block the delivery of a message indefinitely.

**Definition 1.** *We define $\mathcal{M}_l$ to be the set of* static adversaries *that corrupt less than $l$ out of $k$ participants of the mix-server type, and arbitrarily many participants of the sender type.*

Throughout we implicitly assume that a message handed to an ideal functionality that is not on the form prescribed in its definition is returned to the sender immediately. In particular this includes verifying membership in $G_q$ when appropriate. We use the same convention for definitions of protocols.

## 3 The Ideal Mix-Net

Although other definitions of security of mix-nets have been proposed, the most natural definition is given by Wikström [49] in the UC-framework. He formalizes a trusted party that waits for messages from senders, and then when a majority of the mix-servers request it, outputs these messages but in lexicographical order. For simplicity it accepts only one input from each sender. We prove security relative this functionality.

**Functionality 1 (Mix-Net).** The ideal functionality for a *mix-net*, $\mathcal{F}_{\mathrm{MN}}$, running with mix-servers $M_1, \ldots, M_k$, senders $S_1, \ldots, S_N$, and ideal adversary $\mathcal{S}$ proceeds as follows

1. Initialize a list $L = \emptyset$, and set $J_P = \emptyset$ and $J_M = \emptyset$.
2. Repeatedly wait for new inputs and do
   (a) Suppose $(S_i, \texttt{Send}, m_i)$, $m_i \in G_q$, is received from $\mathcal{C}_{\mathcal{I}}$. If $i \notin J_P$, set $J_P \leftarrow J_P \cup \{i\}$, and append $m_i$ to $L$. Then hand $(\mathcal{S}, S_i, \texttt{Send})$ to $\mathcal{C}_{\mathcal{I}}$.
   (b) Suppose $(M_j, \texttt{Run})$ is received from $\mathcal{C}_{\mathcal{I}}$. Set $J_M \leftarrow J_M \cup \{j\}$. If $|J_M| > k/2$, then sort the list $L$ lexicographically to form a list $L'$, hand $((\mathcal{S}, M_j, \texttt{Output}, L'), \{(M_l, \texttt{Output}, L')\}_{l=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$ and ignore further messages. Otherwise, hand $\mathcal{C}_{\mathcal{I}}$ the list $(\mathcal{S}, M_j, \texttt{Run})$.

## 4 A Sender Verifiable El Gamal Based Mix-Net

In recent El Gamal based mix-nets, e.g. [38,20,49], the mix-servers form a chain, and each mix-server randomly permutes, partially decrypts, and *re-encrypts* the output of the previous mix-server. In older constructions decryption is instead carried out jointly at the end of the chain. Our construction is different in that each mix-server *partially decrypts* and *sorts* the output of the previous mix-server. Thus, no cryptotext is re-encrypted and the permutation is not random, but determined by the lexicographical order of the cryptotexts.

Let us consider why re-encryption is often considered necessary. In several previous mix-nets each mix-server $M_j$ holds a secret key $x_j \in \mathbb{Z}_q$ corresponding to a public key $y_j = g^{x_j}$. A joint public key $y = \prod_{j=1}^{k} y_j$ is used by a sender $S_i$ to compute a cryptotext $(u_{0,i}, v_{0,i}) = (g^{r_i}, y^{r_i} m_i)$ of a message $m_i$ for a random $r_i \in \mathbb{Z}_q$. The mix-servers take turns and compute

$$(u_{j,i}, v_{j,i})_{i=1}^{N} = \left( g^{s_{j,i}} u_{j-1,\pi_j(i)}, \left( \prod_{l=j+1}^{k} y_l \right)^{s_{j,i}} v_{j-1,\pi_j(i)} / u_{j-1,\pi_j(i)}^{x_j} \right)_{i=1}^{N} ,$$

for random $s_{j,i} \in \mathbb{Z}_q$ and $\pi_j \in \Sigma_N$, i.e., each mix-server permutes, partially decrypts and re-encrypts its input. In the end $(v_{k,i})_{i=1}^{N} = (m_{\pi(i)})_{i=1}^{N}$ for some random joint permutation $\pi$. The reason that re-encryption is necessary with this type of scheme is that otherwise the first component $u_{0,i}$ of each cryptotext remains unchanged during the transformation, which allows anybody to break the anonymity of all senders. For the older type of construction it is obvious why re-encryption is necessary.

## 4.1 Our Modification

We modify the El Gamal cryptosystem to ensure that also the first component $u_{j-1,i}$ is changed during partial decryption. Each mix-server is given a secret key $(w_j, x_j) \in \mathbb{Z}_q^2$ and a corresponding public key $(z_j, y_j) = (g^{w_j}, g^{x_j})$. To partially decrypt and permute its input it computes

$$(u_{j-1,i}^{1/w_j}, v_{j-1,i} u_{j-1,i}^{-x_j/w_j})_{i=1}^{N} , \qquad (1)$$

from $L_{j-1}$, and sorts the result lexicographically. The result is denoted by $L_j = (u_{j,i}, v_{j,i})_{i=1}^{N}$. Note that both components of each cryptotext are transformed using the secret key of the mix-server. For this transformation to make any sense we must also modify the way the joint key is formed. We define

$$(Z_{k+1}, Y_{k+1}) = (g, 1) \quad \text{and} \quad (Z_j, Y_j) = (Z_{j+1}^{w_j}, Y_{j+1} Z_{j+1}^{x_j}) . \qquad (2)$$

The joint keys must be computed jointly by the mix-servers. A sender encrypts its message using the public key $(Z_1, Y_1)$, i.e., $(u_{0,i}, v_{0,i}) = (Z_1^{r_i}, Y_1^{r_i} m_i)$ for some random $r_i$. The structure of the keys are chosen such that a cryptotext on the form $(u_{j-1,i}, v_{j-1,i}) = (Z_j^{r_i}, Y_j^{r_i} m_i)$ given as input to mix-server $M_j$ satisfies

$$\begin{aligned}(u_{j-1,i}^{1/w_j}, v_{j-1,i} u_{j-1,i}^{-x_j/w_j}) &= (Z_j^{r_i/w_j}, Y_j^{r_i} Z_j^{-r_i x_j/w_j} m_i) \\ &= ((Z_j^{1/w_j})^{r_i}, (Y_j Z_j^{-x_j/w_j})^{r_i} m_i) = (Z_{j+1}^{r_i}, Y_{j+1}^{r_i} m_i) .\end{aligned}$$

Thus, each mix-server $M_j$ transforms a cryptotext $(u_{j-1,i}, v_{j-1,i})$ encrypted with the public key $(Z_j, Y_j)$ into a cryptotext $(u_{j,i}, v_{j,i})$ encrypted with the public key $(Z_{j+1}, Y_{j+1})$. Note that $\mathrm{Sort}(\{v_{k,i}\}_{i=1}^{N}) = \mathrm{Sort}(\{m_i\}_{i=1}^{N})$, since $Y_{k+1} = 1$.

There are several seemingly equivalent ways to set up the scheme, but some of these do not allow a reduction of the security of the mix-net to the DDH-assumption. The relation in Equation (1) is carefully chosen to allow a reduction.

## 4.2 Sender Verifiability

An important consequence of our modification is that a sender can compute $(Z_{j+1}^{r_i}, Y_{j+1}^{r_i} m_i)$ and verify that this pair is contained in $L_j$ for $j = 1, \ldots, k$. Furthermore, if this is not the case the sender can easily prove to any outsider which mix-server behaved incorrectly. We call this *sender verifiability*, since it allows a sender to verify that its cryptotext is processed correctly by the mix-servers. This is not a new property. In fact Chaum's original construction [10] has this property, but our construction is the first provably secure scheme with this property.

We think that sender verifiability is an important property that deserves more attention. The verification process is unconditional and easily explained to anybody with only a modest background in mathematics, and a verification program can be implemented with little skills in programming. This means that in the main application of mix-nets, electronic elections, a sender can convince herself that her vote was processed correctly. We stress that this verification does not guarantee anonymity or correct processing of any other cryptotext. Thus, a proof of the overall security of the mix-net is still required.

The reader may worry that sender verifiability allows a voter to point out its vote to a coercer. This is the case, but the sender can do this in previous mix-nets as well by pointing at its message in the original list $L_0$ of cryptotexts and revealing the randomness used during encryption, so this problem is not specific to our scheme. Furthermore, our scheme becomes coercion-free whenever the sender does not know the randomness of its cryptotext, as other El Gamal based mix-nets, but sender verifiability is then lost.

## 4.3 A Technical Advantage

There is also an important technical consequence of the lack of re-encryption in the mixing process. The witness of our shuffle relation consists of a pair $(w_j, x_j)$, which makes it easy to turn our proof of knowledge into a secure realization of the ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$. This should be contrasted with all previous shuffle relations, where the witness contains a long list of random exponents used to re-encrypt the input that must somehow be extracted by the ideal adversary in the UC-setting.

A potential alternative to our approach is to formalize the proof of a shuffle as a proof of membership [7] in the UC-framework. However, a proof of membership is not sufficient for the older constructions where decryption is carried out jointly at the end of the mixing chain. The problem is that the adversary could corrupt the last mix-server $M_k$ and instruct it to output $L_0$ instead of a re-encryption and permutation of $L_{k-1}$. This would obviously break the anonymity of all senders. The malicious behavior is not detected, since the ideal proof of membership only expects an element in the language and no witness from corrupted parties, and $L_0$ is a re-encryption and permutation of $L_{k-1}$. Interestingly, it seems that the adversary cannot attack the real protocol if the proof of membership of a correct shuffle is implemented using a proof of knowledge in the classical sense.

It is an open question if a proof of membership suffices for mix-nets where each mix-server *partially decrypts* and then re-encrypts and permutes its input.

## 4.4 Preliminaries

We describe the mix-net in a hybrid model as defined in the UC-framework. This means that the mix-servers and senders have access to a set of ideal functionalities introduced in this section. We assume the existence of an authenticated bulletin board. All parties can write to it, but no party can erase any message from it. A formal definition is given in [49,50]. We also assume an ideal functionality corresponding to the key set-up sketched in Section 4.1. This is given below.

**Functionality 2 (Special El Gamal Secret Key Sharing).** The ideal *Special El Gamal Secret Key Sharing over* $G_q$, $\mathcal{F}_{\mathrm{SKS}}$, with mix-servers $M_1, \ldots, M_k$, senders $S_1, \ldots, S_N$, and ideal adversary $\mathcal{S}$.

1. Initialize sets $J_j = \emptyset$ for $j = 0, \ldots, k$.
2. Until $|J_0| = k$, repeatedly wait for inputs. If $(M_j, \texttt{MyKey}, w_j, x_j)$ is received from $\mathcal{C_I}$ such that $w_j, x_j \in \mathbb{Z}_q$ and $j \notin J_0$. Set $J_0 \leftarrow J_0 \cup \{j\}$ compute $z_j = g^{w_j}$ and $y_j = g^{x_j}$, and hand $(\mathcal{S}, \texttt{PublicKey}, M_j, w_j, z_j)$ to $\mathcal{C_I}$.
3. Set $(Z_{k+1}, Y_{k+1}) = (g, 1)$ and $(Z_j, Y_j) = (Z_{j+1}^{w_j}, Y_{j+1} Z_{j+1}^{x_j})$. Then hand $((\mathcal{S}, \texttt{PublicKeys}, (Z_j, Y_j, z_j, y_j)_{j=1}^{k}), \{(S_i, \texttt{PublicKeys}, (Z_j, Y_j, z_j, y_j)_{j=1}^{k})\}_{i=1}^{N}$, $\{(M_l, \texttt{Keys}, w_l, x_l, (Z_j, Y_j, z_j, y_j)_{j=1}^{k})\}_{l=1}^{k})$ to $\mathcal{C_I}$.
4. Until $|J_0| = k$, repeatedly wait for inputs. If $(M_j, \texttt{Recover}, M_l)$ is received from $\mathcal{C_I}$, set $J_l \leftarrow J_l \cup \{j\}$. If $|J_l| > k/2$, then hand $((\mathcal{S}, \texttt{Recovered}, M_l, w_l, x_l)$, $\{(M_j, \texttt{Recovered}, M_l, w_l, x_l)\}_{j=1}^{k})$ to $\mathcal{C_I}$, and otherwise hand $(\mathcal{S}, M_j, \texttt{Recover}, M_l)$ to $\mathcal{C_I}$.

The above functionality can be securely realized by letting each mix-server secret share its secret key using Feldman's [15] verifiable secret sharing scheme. Note that the functionality explicitly allows corrupted mix-servers to choose their keys in a way that depends on the public keys of uncorrupted mix-servers. The special joint keys would then be computed iteratively using Equation (2), and during this process each mix-server would prove that it does this correctly using standard methods.

Each mix-server partially decrypts each cryptotext and sorts the resulting cryptotexts. Thus, proving correct behavior corresponds to proving knowledge of a secret key $(w, x)$ such that the cryptotexts $(u_i, v_i)$ input to a mix-server are related to the cryptotexts $(u_i', v_i')$ it outputs by the following relation.

**Definition 2 (Knowledge of Correct Decryption-Permutation).** *Define for each $N$ a relation $R_{\mathrm{DP}} \subset (G_q^3 \times G_q^{2N} \times G_q^{2N}) \times (\mathbb{Z}_q \times \mathbb{Z}_q)$, by*

$$((g, z, y, \{(u_i, v_i)\}_{i=1}^N, \{(u'_i, v'_i)\}_{i=1}^N), (w, x)) \in R_{\mathrm{DP}}$$

*precisely when $z = g^w$, $y = g^x$ and $(u'_i, v'_i) = (u_{\pi(i)}^{1/w}, v_{\pi(i)} u_{\pi(i)}^{-x/w})$ for $i = 1, \ldots, N$ and $\pi \in \Sigma_N$ such that the list $\{(u'_i, v'_i)\}_{i=1}^N$ is sorted lexicographically.*

To avoid a large class of "relation attacks" [44,43,48] no sender can be allowed to construct a cryptotext of a message related to the message encrypted by some other sender. Thus, each sender is required to prove knowledge of the randomness it uses to form its cryptotexts. This corresponds to the following relation.

**Definition 3 (Knowledge of Cleartext).** *Define a relation $R_{\mathrm{C}} \subset G_q^4 \times \mathbb{Z}_q$ by $((Z, Y, u, v), r) \in R_{\mathrm{C}}$ precisely when $\log_Z u = r$.*

Formally, we need a secure realization of the following functionality parameterized by the above relations.

**Functionality 3 (Zero-Knowledge Proof of Knowledge).** Let $\mathcal{L}$ be a language given by a binary relation $R$. The ideal *zero-knowledge proof of knowledge* functionality $\mathcal{F}_{\mathrm{ZK}}^R$ of a witness $w$ to an element $x \in \mathcal{L}$, running with parties $P_1, \ldots, P_k$

1. Upon receipt of $(P_i, \texttt{Prover}, x, w)$ from $\mathcal{C}_\mathcal{I}$, store $w$ under the tag $(P_i, x)$, and hand $(\mathcal{S}, P_i, \texttt{Prover}, x, R(x, w))$ to $\mathcal{C}_\mathcal{I}$.
2. Upon receipt of $(M_j, \texttt{Question}, P_i, x)$ from $\mathcal{C}_\mathcal{I}$, let $w$ be the string stored under the tag $(P_i, x)$ (the empty string if nothing is stored), and hand $((\mathcal{S}, M_j, \texttt{Verifier}, P_i, x, R(x, w)), (M_j, \texttt{Verifier}, P_i, R(x, w)))$ to $\mathcal{C}_\mathcal{I}$.

In [49] a secure realization $\pi_{\mathrm{C}}$ of $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ is given, under the DDH-assumption, which is secure against $\mathcal{M}_{k/2}$-adversaries.

The functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ is securely realized in Section 6.

### 4.5 The Mix-Net

We now give the details of our mix-net. It executes in a hybrid model with access to the ideal functionalities described above.

**Protocol 1 (Mix-Net).** The mix-net protocol $\pi_{\mathrm{MN}} = (S_1, \ldots, S_N, M_1, \ldots, M_k)$ consists of senders $S_i$, and mix-servers $M_j$.

SENDER $S_i$. Each sender $S_i$ proceeds as follows.

1. Wait for $(\texttt{PublicKeys}, (Z_j, Y_j, z_j, y_j)_{j=1}^k)$ from $\mathcal{F}_{\mathrm{SKS}}$.
2. Wait for an input $(\texttt{Send}, m_i)$, $m_i \in G_q$. Then choose $r_i \in \mathbb{Z}_q$ randomly and compute $(u_i, v_i) = E_{(Z_1, Y_1)}(m_i, r_i) = (Z_1^{r_i}, Y_1^{r_i} m_i)$. Then hand $(\texttt{Prover}, (Z_1, Y_1, u_i, v_i), r_i)$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$, and hand $(\texttt{Write}, (u_i, v_i))$ to $\mathcal{F}_{\mathrm{BB}}$.

MIX-SERVER $M_j$. Each mix-server $M_j$ proceeds as follows.

1. Choose $w_j, x_j \in \mathbb{Z}_q$ randomly and hand $(\texttt{MyKey}, w_j, x_j)$ to $\mathcal{F}_{\mathrm{SKS}}$.
2. Wait for $(\texttt{Keys}, (w_j, x_j), (Z_j, Y_j, z_j, y_j)_{j=1}^{k})$ from $\mathcal{F}_{\mathrm{SKS}}$, where $w_j, x_j \in \mathbb{Z}_q$ and $Z_j, Y_j, z_j, y_j \in G_q$.
3. Wait for an input $(\texttt{Run})$, and then hand $(\texttt{Write}, \texttt{Run})$ to $\mathcal{F}_{\mathrm{BB}}$.
4. Wait until more than $k/2$ different mix-servers have written $\texttt{Run}$ on $\mathcal{F}_{\mathrm{BB}}$, and let the last entry of this type be $(c_{\texttt{Run}}, M_i, \texttt{Run})$.
5. Form the list $L_* = \{(u_\gamma, v_\gamma)\}_{\gamma \in I_*}$, for some index set $I_*$, by choosing for $\gamma = 1, \ldots, N$ the entry $(c, S_\gamma, (u_\gamma, v_\gamma))$ on $\mathcal{F}_{\mathrm{BB}}$ with the smallest $c < c_{\mathrm{run}}$ such that $u_\gamma, v_\gamma \in G_q$, if present.
6. For each $\gamma \in I_*$ do the following,
   (a) Hand $(\texttt{Question}, S_\gamma, (Z_1, Y_1, u_\gamma, v_\gamma))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
   (b) Wait for $(\texttt{Verifier}, S_\gamma, b_\gamma)$ from $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.

   Then form $L_0 = \{(u_{0,i}, v_{0,i})\}_{i=1}^{N'}$ consisting of pairs $(u_\gamma, v_\gamma)$ such that $b_\gamma = 1$.
7. For $l = 1, \ldots, k$ do
   (a) If $l \neq j$, then do
      i. Wait until an entry $(c, M_l, (\texttt{List}, L_l))$ appears on $\mathcal{F}_{\mathrm{BB}}$, where $L_l$ is on the form $\{(u_{l,i}, v_{l,i})\}_{i=1}^{N'}$ for $u_{l,i}, v_{l,i} \in G_q$.
      ii. Hand $(\texttt{Question}, M_l, (g, z_l, y_l, L_{l-1}, L_l))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$, and wait for $(\texttt{Verifier}, M_l, b_l)$ from $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$.
      iii. If $b_l = 0$, then hand $(\texttt{Recover}, M_l)$ to $\mathcal{F}_{\mathrm{SKS}}$, and wait for $(\texttt{Recovered}, M_l, (w_l, x_l))$ from $\mathcal{F}_{\mathrm{SKS}}$. Then compute

$$L_l = \{(u_{l,i}, v_{l,i})\}_{i=1}^{N'} = \mathrm{Sort}(\{(u_{l-1,i}^{1/w_l}, v_{l-1,i} u_{l-1,i}^{-x_l/w_l})\}_{i=1}^{N'}) \enspace .$$

   (b) If $l = j$, then compute

$$L_j = \{(u_{j,i}, v_{j,i})\}_{i=1}^{N'} = \mathrm{Sort}(\{(u_{j-1,i}^{1/w_j}, v_{j-1,i} u_{j-1,i}^{-x_j/w_j})\}_{i=1}^{N'}) \enspace ,$$

   Finally hand $(\texttt{Prover}, (g, z_j, y_j, L_{j-1}, L_j), (w_j, x_j))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$, and hand $(\texttt{Write}, (\texttt{List}, L_j))$ to $\mathcal{F}_{\mathrm{BB}}$.
8. Output $(\texttt{Output}, \mathrm{Sort}(\{v_{k,i}\}_{i=1}^{N'}))$.

**Theorem 1.** *The ideal functionality $\mathcal{F}_{\mathrm{MN}}$ is securely realized by $\pi_{\mathrm{MN}}$ in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{SKS}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}})$-hybrid model with respect to $\mathcal{M}_{k/2}$-adversaries under the DDH-assumption in $G_q$.*

## 5 A New Efficient Proof of a Shuffle

We want to securely realize the ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$. It turns out that a useful step in this direction is to construct a statistical zero-knowledge proof for the relation $R_{\mathrm{DP}}$, i.e., a proof of the decryption-permutation shuffle. First we explain the key ideas in our approach. Then we give a detailed description of our protocol. Finally, we explain how it can be turned into a public coin protocol.

## 5.1 Our Approach

The protocol for proving the relation $R_{\text{DP}}$ is complex, but the underlying ideas are simple. To simplify the exposition we follow Neff [37,38] and consider the problem of proving that a list of elements in $G_q$ are exponentiated and permuted. More precisely, let $y, u_1, \ldots, u_N, u'_1, \ldots, u'_N \in G_q$ be defined by $y = g^x$ and $u'_i = u^x_{\pi(i)}$ for a permutation $\pi$. Only the prover knows $x$ and $\pi$ and it must show that the elements satisfy such a relation. We also omit numerous technical details. In particular we remove several blinding factors, hence the protocols are not zero-knowledge as sketched here.

**Extraction Using Linear Independence.** The verifier chooses a list $P = (p_i)_{i=1}^N \in \mathbb{Z}_q^N$ of random primes and computes $U = \prod_{i=1}^N u_i^{p_i}$. Then it requests that the prover computes $U' = \prod_{i=1}^N (u'_i)^{p_{\pi(i)}}$, proves that $U' = U^x$ and that it knows a permutation $\pi$ such that $U' = \prod_{i=1}^N (u'_i)^{p_{\pi(i)}}$.

The idea is then that if a prover succeeds in doing this it can be rewound and run several times with different random vectors $P_j$, giving different $U_j$ and $U'_j$, until a set $P_1, \ldots, P_N$ of linearly independent vectors in $\mathbb{Z}_q^N$ are found. Linear independence implies that there are coefficients $a_{l,j} \in \mathbb{Z}_q$ such that $\sum_{j=1}^N a_{l,j} P_j$ equals the $l$th unity vector $e_l$, i.e., the vector with a one in the $l$th position and all other elements zero. We would then like to conclude that

$$u_l^x = \left(\prod_{j=1}^N U_j^{a_{l,j}}\right)^x = \prod_{j=1}^N (U'_j)^{a_{l,j}} = \prod_{j=1}^N \left(\prod_{i=1}^N (u'_i)^{p_{j,\pi^{-1}(i)}}\right)^{a_{l,j}} = u'_{\pi(l)} \ , \tag{3}$$

since that would imply that the elements satisfy the shuffle-relation.

**Proving a Permutation of Prime Exponents.** The prover can use standard techniques to prove knowledge of integers $\rho_1, \ldots, \rho_N$ such that $U' = \prod_{i=1}^N (u'_i)^{\rho_i}$, but it must also prove that $\rho_i = p_{\pi(i)}$ for some permutation $\pi$.

Suppose that $\prod_{i=1}^N p_i = \prod_{i=1}^N \rho_i$ over $\mathbb{Z}$. Then unique factorization in $\mathbb{Z}$ implies that each $\rho_i$ equals some product of the $p_i$ and $-1$. If in addition we demand that $\rho_i \in [-2^K + 1, 2^K - 1]$, no such product can contain more than one factor. This implies that every product must contain exactly one factor. Thus, $\rho_i = \pm p_{\pi(i)}$ for some permutation $\pi$. If we also have $\sum_{i=1}^N p_i = \sum_{i=1}^N \rho_i$, then we must clearly have $\rho_i = p_{\pi(i)}$.

We observe that proving the above is relatively simple over a group of unknown order such as the group $\text{QR}_{\mathbf{N}}$ of squares modulo an RSA modulus $\mathbf{N}$. The prover forms commitments

$$\mathbf{b}_0 = \mathbf{g} \ , \quad (\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N = (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t'_i} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N \ ,$$

with random $t_i$ and $t'_i$ and proves, using standard methods, knowledge of $\rho_i, \tau_i, \tau'_i$ such that

$$U' = \prod_{i=1}^N (u'_i)^{\rho_i} \ , \quad \mathbf{b}_i = \mathbf{h}^{\tau_i} \mathbf{b}_{i-1}^{\rho_i} \ , \quad \text{and} \quad \mathbf{b}'_i = \mathbf{h}^{\tau'_i} \mathbf{g}^{\rho_i} \ . \tag{4}$$

Note that $\mathbf{b}_N = \mathbf{h}^{\tau}\mathbf{g}^{\prod_{i=1}^{N}\rho_i}$ for some $\tau$, so the verifier can check that $\prod_{i=1}^{N}\rho_i = \prod_{i=1}^{N}p_i$ by asking the prover to show that it knows $\tau$ such that $\mathbf{b}_N/\mathbf{g}^{\prod_{i=1}^{N}p_i} = \mathbf{h}^{\tau}$. We then note that a standard proof of knowledge over a group of unknown order also gives an upper bound on the bit-size of the exponents, i.e., it implicitly proves that $\rho_i \in [-2^K + 1, 2^K - 1]$. Finally, since $\prod_{i=1}^{N}\mathbf{b}'_i = \mathbf{h}^{\tau'}\mathbf{g}^{\sum_{i=1}^{N}\rho_i}$ for a $\tau' = \sum_{i=1}^{N}\tau'_i$, the verifier can check that $\sum_{i=1}^{N}\rho_i = \sum_{i=1}^{N}p_i$ by asking the prover to show that it knows $\tau'$ such that $\prod_{i=1}^{N}\mathbf{b}'_i/\mathbf{g}^{\sum_{i=1}^{N}p_i} = \mathbf{h}^{\tau'}$.

**Fixing a Permutation.** In Equation (3) above it is assumed that a fixed permutation $\pi$ is used for all prime vectors $P_1, \ldots, P_N$. Unfortunately, this is not necessarily the case, i.e., the permutation used in the $j$th proof may depend on $j$ and we should really write $\pi_j$.

To solve this technical problem we force the prover to commit to a fixed permutation $\pi$ before it receives the prime vector $P$. The commitment is on the form $(w_i)_{i=1}^{N} = (g^{r'_i}g_{\pi^{-1}(i)})_{i=1}^{N}$. The verifier then computes $W = \prod_{i=1}^{N} w_i^{p_i}$ and the prover proves that $W = g^{r'}\prod_{i=1}^{N} g_i^{\rho_i}$ in addition to Equations (4). The idea is that the prover must use $\pi$ to permute the $\rho_i$ or find a non-trivial representation of $1 \in G_q$ using $g, g_1, \ldots, g_N$, which is infeasible under the DL-assumption.

## 5.2 An Honest Verifier Statistical Zero-Knowledge Computationally Convincing Proof of Knowledge of a Decryption-Permutation

In this section we describe our proof of a shuffle in detail. Although we consider a decrypt-permutation relation, our approach can be generalized to a proof of a shuffle for the other shuffle relations considered in the literature. In the full version [50] we detail such shuffles, including a shuffle of Paillier [41] cryptotexts.

We introduce several security parameters. We use $K_1$ to denote the number of bits in $q$, the order of the group $G_q$, and similarly $K_2$ to denote the number of bits in the RSA-modulus $\mathbf{N}$. We use $K_3$ to denote the number bits used in the random primes mentioned above. At some point in the protocol the verifier hands a challenge to the prover. We use $K_4$ to denote the number of bits in this challenge. At several points exponents must be padded with random bits to achieve statistical zero-knowledge. We use $K_5$ to denote the number of additional random bits used to do this. We assume that the security parameters are chosen such that $K_3 + K_4 + K_5 < K_1, K_2$, and $K_5 < K_3 - 2$. Below the protocol we explain how the informal description above relates to the different components of the protocol.

**Protocol 2 (Proof of Decryption-Permutation).** The common input consists of an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, generators $g, g_1, \ldots, g_N \in G_q$, a public key $(z, y) \in G_q^2$, and two lists $L = (u_i, v_i)_{i=1}^{N}$ and $L' = (u'_i, v'_i)_{i=1}^{N}$ in $G_q^{2N}$. The private input to the prover consists of $(w, x) \in \mathbb{Z}_q^2$ such that $(z, y) = (g^w, g^x)$ and $(u'_i, v'_i) = (u_{\pi(i)}^{1/w}, v_{\pi(i)}/u_{\pi(i)}^{x/w})$ for a permutation $\pi \in \Sigma_N$ such that $L'$ is lexicographically sorted.

1. The prover chooses $r'_i \in \mathbb{Z}_q$ randomly, computes $(w_i)_{i=1}^N = (g^{r'_i} g_{\pi^{-1}(i)})_{i=1}^N$, and hands $(w_i)_{i=1}^N$ to the verifier.
2. The verifier chooses random primes $p_1, \ldots, p_N \in [2^{K_3-1}, 2^{K_3}-1]$, and hands $(p_i)_{i=1}^N$ to the prover.
3. Both parties compute $(U, V, W) = (\prod_{i=1}^N u_i^{p_i}, \prod_{i=1}^N v_i^{p_i}, \prod_{i=1}^N w_i^{p_i})$.
4. The prover chooses the following elements randomly $k_1, k_2, k_3, k_4, k_5 \in \mathbb{Z}_q$, $l_1, \ldots, l_7, l_{r'}, l_{1/w}, l_{x/w}, l_w, l_x \in \mathbb{Z}_q$, $t_i, t'_i \in [0, 2^{K_2+K_5}-1]$, $s_i, s'_i \in [0, 2^{K_2+K_4+2K_5}-1]$, $r_i \in [0, 2^{K_3+K_4+K_5}-1]$ for $i = 1, \ldots, N$, $s \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N}-1]$, and $s' \in [0, 2^{K_2+K_5+\log_2 N}-1]$. Then the prover computes

$$(b_1, b_2) = (g^{k_1} U^{1/w}, g^{k_2} U^{x/w}) \tag{5}$$

$$(b_3, b_4, b_5) = (g_1^{k_3} g^{1/w}, g_1^{k_4} b_3^x, g_1^{k_5} b_3^w) \tag{6}$$

$$(\beta_1, \beta_2) = \left(g^{l_1} U^{l_{1/w}}, g^{l_2} U^{l_{x/w}}\right) \tag{7}$$

$$(\beta_3, \beta_4) = (g_1^{l_3} g^{l_{1/w}}, g_1^{l_6} g^{l_{x/w}}) \tag{8}$$

$$(\beta_5, \beta_6, \beta_7, \beta_8, \beta_9) = (g_1^{l_4} b_3^{l_x}, g^{l_x}, g_1^{l_5} b_3^{l_w}, g^{l_w}, g_1^{l_7}) \tag{9}$$

$$(\alpha_1, \alpha_2, \alpha_3) = \left(g^{l_1} \prod_{i=1}^N (u'_i)^{r_i}, g^{-l_2} \prod_{i=1}^N (v'_i)^{r_i}, g^{l_{r'}} \prod_{i=1}^N g_i^{r_i}\right) \tag{10}$$

$$\mathbf{b}_0 = \mathbf{g} \tag{11}$$

$$(\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N = (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t'_i} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N \tag{12}$$

$$(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)_{i=1}^N = (\mathbf{h}^{s_i} \mathbf{b}_{i-1}^{r_i}, \mathbf{h}^{s'_i} \mathbf{g}^{r_i})_{i=1}^N \tag{13}$$

$$(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = (\mathbf{h}^s, \mathbf{h}^{s'}) \ , \tag{14}$$

and $((b_i)_{i=1}^5, (\beta_i)_{i=1}^9, (\alpha_1, \alpha_2, \alpha_3), (\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N, (\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)_{i=1}^N, (\boldsymbol{\gamma}, \boldsymbol{\gamma}'))$ is handed to the verifier.
5. The verifier chooses $c \in [2^{K_4-1}, 2^{K_4}-1]$ randomly and hands $c$ to the prover.
6. Define $t = t_N + p_{\pi(N)}(t_{N-1} + p_{\pi(N-1)}(t_{N-2} + p_{\pi(N-2)}(t_{N-3} + p_{\pi(N-3)}(\ldots))))$, $t' = \sum_{i=1}^N t'_i$, $r' = \sum_{i=1}^N r'_i p_i$, $k_6 = k_4 + k_3 x$, and $k_7 = k_5 + k_3 w$. The prover computes

$$\begin{aligned}
(f_i)_{i=1}^7 &= (ck_i + l_i)_{i=1}^7 && \mod q \\
(f_{1/w}, f_{x/w}) &= (c/w + l_{1/w}, cx/w + l_{x/w}) && \mod q \\
(f_w, f_x) &= (cw + l_w, cx + l_x) && \mod q \\
f_{r'} &= cr' + l_{r'} && \mod q \\
(e_i, e'_i)_{i=1}^N &= (ct_i + s_i, ct'_i + s'_i)_{i=1}^N && \mod 2^{K_2+K_4+2K_5} \\
(d_i)_{i=1}^N &= (cp_{\pi(i)} + r_i)_{i=1}^N && \mod 2^{K_3+K_4+K_5} \\
e &= ct + s && \mod 2^{K_2+NK_3+K_4+K_5+\log_2 N} \\
e' &= ct' + s' && \mod 2^{K_2+K_5+\log_2 N}
\end{aligned}$$

Then it hands $(((f_i)_{i=1}^7, f_{1/w}, f_{x/w}, f_w, f_x, f_{r'}), (e_i, e'_i)_{i=1}^N, (d_i)_{i=1}^N, (e, e'))$ to the verifier.

7. The verifier checks that $b_i, \beta_i, \alpha_i \in G_q$, and that $L'$ is lexicographically sorted and that

$$(b_1^c \beta_1, b_2^c \beta_2) = (g^{f_1} U^{f_{1/w}}, g^{f_2} U^{f_{x/w}}) \tag{15}$$

$$(b_3^c \beta_3, b_4^c \beta_4) = (g_1^{f_3} g^{f_{1/w}}, g_1^{f_6} g^{f_{x/w}}) \tag{16}$$

$$(b_4^c \beta_5, y^c \beta_6) = (g_1^{f_4} b_3^{f_x}, g^{f_x}) \tag{17}$$

$$(b_5^c \beta_7, z^c \beta_8, (b_5/g)^c \beta_9) = (g_1^{f_5} b_3^{f_w}, g^{f_w}, g_1^{f_7}) \tag{18}$$

$$(b_1^c \alpha_1, (V/b_2)^c \alpha_2, W^c \alpha_3) = \left( g^{f_1} \prod_{i=1}^{N} (u_i')^{d_i}, g^{-f_2} \prod_{i=1}^{N} (v_i')^{d_i}, g^{f_{r'}} \prod_{i=1}^{N} g_i^{d_i} \right) \tag{19}$$

$$(\mathbf{b}_i^c \boldsymbol{\gamma}_i, (\mathbf{b}_i')^c \boldsymbol{\gamma}_i')_{i=1}^N = (\mathbf{h}^{e_i} \mathbf{b}_{i-1}^{d_i}, \mathbf{h}^{e_i'} \mathbf{g}^{d_i})_{i=1}^N \tag{20}$$

$$(\mathbf{g}^{-\prod_{i=1}^N p_i} \mathbf{b}_N)^c \boldsymbol{\gamma} = \mathbf{h}^e \tag{21}$$

$$\left( \mathbf{g}^{-\sum_{i=1}^N p_i} \prod_{i=1}^{N} \mathbf{b}_i' \right)^c \boldsymbol{\gamma}' = \mathbf{h}^{e'} \ . \tag{22}$$

Equations (5)-(9) are used to prove that $(b_1, V/b_2) = (g^{\kappa_1} U^{1/w}, g^{-\kappa_2} V/U^{x/w})$ using standard Schnorr-like proofs of knowledge of logarithms. Equations (12) contain commitments corresponding to those in the outline of our approach. Equations (13) are used to prove knowledge of exponents $\tau_i, \tau_i', \rho_i$ such that $(\mathbf{b}_i, \mathbf{b}_i') = (\mathbf{h}^{\tau_i} \mathbf{b}_{i-1}^{\rho_i}, \mathbf{h}^{\tau_i'} \mathbf{g}^{\rho_i})$. We remark that the verifier need not check that $\mathbf{b}_i, \mathbf{b}_i', \boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i', \boldsymbol{\gamma}, \boldsymbol{\gamma}' \in \mathrm{QR}_{\mathbf{N}}$ for our analysis to go through. Equations (14) are used to prove that $\prod_{i=1}^N \rho_i = \prod_{i=1}^N p_i$ and $\sum_{i=1}^N \rho_i = \sum_{i=1}^N p_i$, i.e., that $\rho_i$ in fact equals $p_{\pi(i)}$ for some permutation $\pi$. Equation (10) is used to prove that $(b_1, V/b_2)$ also equals $(g^{k_1} \prod_{i=1}^N (u_i^{1/w_j})^{p_i}, g^{-k_2} \prod_{i=1}^N (v_i/u_i^{x_j/w_j})^{p_i})$. If the two ways of writing $b_1$ and $b_2$ are combined we have

$$(U^{1/w}, V/U^{x/w}) = \left( \prod_{i=1}^{N} (u_i^{1/w_j})^{p_i}, \prod_{i=1}^{N} (v_i/u_i^{x_j/w_j})^{p_i} \right) \ ,$$

which by the argument in Section 5.1 implies that $((g, z, y, L, L'), (w, x)) \in R_{\mathrm{DP}}$.

## 5.3 Security Properties

Formally, the security properties of our protocol are captured by the following.

**Proposition 1 (Zero-Knowledge).** *Protocol 2 is honest verifier statistical zero-knowledge.*

The protocol could be modified by adding a first step, where the verifier chooses $(\mathbf{N}, \mathbf{g}, \mathbf{h})$ and $(g_1, \ldots, g_N)$. This would give a computationally sound proof of knowledge. However, in our application we wish to choose these parameters jointly and only once, and then let the mix-servers execute the proof with these parameters as common inputs. Thus, there may be a negligible portion of the parameters on which the prover can convince the verifier of false statements. Because of this we cannot hope to prove that the protocol is a proof of knowledge in the formal sense. Damgård and Fujisaki [12] introduce the notion of a computationally convincing proof of knowledge to deal with situations like these. We do not use the notion of "computationally convincing proofs" explicitly in our security analysis, but the proposition below implies that our protocol satisfies their definition.

We consider a malicious prover $A$ which is given $\boldsymbol{\Gamma} = (\mathbf{N}, \mathbf{g}, \mathbf{h})$ and $\overline{g} = (g, g_1, \ldots, g_N)$ as input and run with internal randomness $r_\mathrm{p}$. The prover outputs an instance $I_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$, i.e., public keys $z, y \in G_q$ and two lists $L, L' \in G_q^{2N}$ and then interacts with the honest verifier on the common input consisting of $(\boldsymbol{\Gamma}, \overline{g}, z, y, L, L')$. Denote by $T_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, r_\mathrm{v})$ the transcript of such an interaction when the verifier runs with internal randomness $r_\mathrm{v}$. Let Acc be the predicate taking a transcript $T$ as input that outputs 1 if the transcript is accepting and 0 otherwise. Let $L_{R_\mathrm{DP}}$ be the language corresponding to the decryption-permutation relation $R_\mathrm{DP}$. We prove the following proposition.

**Proposition 2 (Soundness).** *Suppose the strong RSA-assumption and the DL-assumption are true. Then for all polynomial-size circuit families $A = \{A_K\}$ it holds that $\forall c > 0$, $\exists K_0$, such that for $K_1 \geq K_0$*

$$\Pr_{\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, r_\mathrm{v}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, r_\mathrm{v})) = 1 \wedge I_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) \notin L_{R_\mathrm{DP}}] < \frac{1}{{K_1}^c} \enspace .$$

## 5.4 Generation of Primes From a Small Number of Public Coins

In our protocol the verifier must generate vectors in $\mathbb{Z}_q^N$ such that each component is a "randomly" chosen prime in $[2^{K_3-1}, 2^{K_3}-1]$. We define a generator $\mathsf{PGen}$ that generates prime vectors from public coins. Let $p(n)$ be the smallest prime at least as large as $n$. Our generator $\mathsf{PGen}$ takes as input $N$ random integers $n_1, \ldots, n_N \in [2^{K_3-1}, 2^{K_3}-1]$ and internal randomness $r$, and defines $p_i = p(n_i)$. To find $p_i$ it first redefines $n_i$ such that it is odd by incrementing by one if necessary. Then it executes the Miller-Rabin primality test for $n_i, n_i+2, n_i+4, \ldots$ until it finds a prime. We put an explicit bound on the running time of the generator by bounding the number of integers it considers and the number of iterations of the Miller-Rabin test it performs in total. If the generator stops due to one of these bounds it outputs $\perp$. If $N \geq K_3$, the bound corresponds to $\frac{6{K_3}^4}{{K_1}^3}N$ exponentiations modulo a $K_1$-bit integer. The generator can be used in the obvious way to turn the protocol above into a public-coin protocol. The verifier sends $(n_1, \ldots, n_N, r)$ to the prover instead of $p_1, \ldots, p_N$ and the prover and verifier generates the primes by computing $(p_1, \ldots, p_N) = \mathsf{PGen}(n_1, \ldots, n_N, r)$. A result by Baker and Harman [4] implies that the resulting distribution is close to uniform.

**Theorem 2 (cf. [4]).** *For large integers $n$ there exists a prime in $[n - n^{0.535}, n]$.*

**Corollary 1.** *For all primes $p \in [2^{K_3-1}, 2^{K_3}-1]$, $\Pr[p(n) = p] \leq 2^{-0.465(K_3-1)}$, where the probability is taken over a random choice of $n \in [2^{K_3-1}, 2^{K_3}-1]$*

The corollary gives a very pessimistic bound. It is commonly believed that the theorem is true with 0.465 replaced by any constant less than one. Furthermore, Cramér argues probabilistically that there is a prime in every interval $[n - \log^2 n, n]$. See Ribenboim [46] for a discussion on this.

We must argue that the generator fails with negligible probability. There are two ways the generator can fail. Either it outputs $p_1, \ldots, p_N$, where $p_i \neq p(n_i)$ for some $i$, or it outputs $\perp$.

**Lemma 1.** *The probability that* $\mathsf{PGen}(n_1, \ldots, n_N, r) \neq (p(n_1), \ldots, p(n_N))$ *conditioned on* $\mathsf{PGen}(n_1, \ldots, n_N, r) \neq \perp$ *is negligible.*

Unfortunately, the current understanding of the distribution of the primes does not allow a strict analysis of the probability that $\mathsf{PGen}(n_1, \ldots, n_N, r) = \perp$. Instead we give a heuristic analysis in Cramér's probabilistic model of the primes.

**Definition 4 (Cramér's Model).** *For each integer $n$, let $X_n$ be an independent binary random variable such that* $\Pr[X_n = 1] = 1/\ln n$. *An integer $n$ is said to be prime* if $X_n = 1$.*

The idea is to consider the primality of the integers as a typical outcome of the sequence $(X_n)_{n \in \mathbb{Z}}$. Thus, when we analyze the generator we assume that the primality of an integer $n$ is given by $X_n$, and our analysis is both over the internal randomness of $\mathsf{PGen}$ and the randomness of $X_n$.

**Lemma 2.** *In Cramér's model the probability that* $\mathsf{PGen}(n_1, \ldots, n_N, r) = \perp$ *is negligible.*

We stress that zero-knowledge and soundness of the modified protocol are not heuristic. The zero-knowledge property holds for arbitrarily distributed *integers* $p_i$. Soundness follows from Lemma 1. It is only completeness that is argued heuristically. Although this is not always clear, similar heuristic arguments are common in the literature, e.g. to generate safe primes and to encode arbitrary messages in $G_q$. We assume that Lemma 2 is true from now on.

Although we now have a public-coin protocol it requires many random bits. This can be avoided by use of a pseudo-random generator $\mathsf{PRG}$ as suggested by Groth [28]. Instead of choosing $n_1, \ldots, n_N$ randomly and sending these integers to the prover, the verifier chooses a random seed $s \in [0, 2^{K_1} - 1]$ and hands this to the prover. The prover and verifier then computes $(n_1, \ldots, n_N) = \mathsf{PRG}(s)$ and computes the primes from the integers as described above. The output $(p_1, \ldots, p_N)$ may not appear to the prover as random, since he holds the seed $s$. However, we prove in the full version [50] that if we define $P_j = \mathsf{PGen}(\mathsf{PRG}(s))$ and let $P_1, \ldots, P_{j-1} \in \mathbb{Z}_q^N$ be any linearly independent vectors, the probability that $P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})$ or $p_{j,i} = p_{j,l}$ for some $i \neq l$ is negligible for all $1 \leq j \leq N$. This is all we need in our application.

**Universal Verifiability and Random Oracles.** If the Fiat-Shamir heuristic is applied to a proof of a shuffle, any outsider can check, non-interactively, that a mix-server behaves correctly. If the verification involves no trusted parameters the resulting mix-net is called "universally verifiable". In our protocol the RSA parameters $(\mathbf{N}, \mathbf{g}, \mathbf{h})$ must be trusted by the verifier and we do not see how these can be generated from public coins. Thus, if the Fiat-Shamir heuristic is applied to our protocol the result is not really universally verifiable.

However, we can achieve universal verifiability under the root assumption in class groups with prime discriminant. A class group is defined by its discriminant $\Delta$. It is conjectured that finding non-trivial roots in a class group with discriminant $\Delta = -p$ for a prime $p$ is infeasible (cf. [29]). The idea would be to generate a prime $p$ of suitable size from random coins handed to the prover by the verifier in the first round. Then the integer part of the protocol would be executed in the class group defined by $\Delta = -p$. With this modification the protocol gives a universally verifiable mix-net.

### 5.5 Complexity

Comparing the complexity of protocols is tricky, since any comparison must take place for equal security rather than for equal security parameters. The only rigorous method to do this is to perform an exact security analysis of each protocol and choose the security parameters accordingly. Various optimization and pre-computing techniques are also applicable to different degrees in different protocols and in different applications. Despite this we argue informally, but carefully, in the full paper [50] that the complexity of our protocol is at least as good as that of the most efficient previous proofs of a shuffle.

More precisely, our protocol requires 5 rounds as the previously known most round efficient proof of a shuffle [21] involving decryption. Furthermore, for practical parameters, e.g. $K_1 = 2048$, $K_2 = 1024$, $K_4 = 160$, $K_3 = 100$, and $K_5 = 50$, the complexity is less than $2.5N$ and $1.6N$ general exponentiations in $G_q$ for the prover and verifier. With optimizations as in [21] this corresponds to $0.5N$ and $0.8N$ general exponentiations in $G_q$, which indicates that the protocol is at least as fast as that in [21].

## 6 Secure Realization of $\mathcal{F}_{\mathbf{ZK}}^{R_{\mathbf{DP}}}$

In this section we transform the proof of a shuffle into a secure realization of $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ in a $(\mathcal{F}_{\mathrm{RSA}}, \mathcal{F}_{\mathrm{CF}}, \mathcal{F}_{\mathrm{BB}})$-hybrid model, where $\mathcal{F}_{\mathrm{RSA}}$ is an RSA common reference string functionality, and $\mathcal{F}_{\mathrm{CF}}$ is a coin flipping functionality.

**Functionality 4 (RSA Common Reference String).** The ideal *RSA Common Reference String*, $\mathcal{F}_{\mathrm{RSA}}$, with mix-servers $M_1, \ldots, M_k$ and ideal adversary $\mathcal{S}$ proceeds as follows.

1. Generate two random $K_2/2$-bit primes $\mathbf{p}$ and $\mathbf{q}$ such that $(\mathbf{p}-1)/2$ and $(\mathbf{q}-1)/2$ are prime and compute $\mathbf{N} = \mathbf{pq}$. Then choose $\mathbf{g}$ and $\mathbf{h}$ randomly in $\mathrm{QR}_{\mathbf{N}}$. Finally, hand $((\mathcal{S}, \mathtt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h}), \{(M_j, \mathtt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h})\}_{j=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$.

There are protocols [6,16] for generating a joint RSA modulus, but these are not analyzed in the UC-framework, so for technical reasons we cannot apply these directly. If these protocols cannot be used to give a UC-secure protocol, general methods [11] can be used since this need only be done once.

**Functionality 5 (Coin-Flipping).** The ideal *Coin-Flipping functionality*, $\mathcal{F}_{\mathrm{CF}}$, with mix-servers $M_1, \ldots, M_k$, and adversary $\mathcal{S}$ proceeds as follows.

1. Set $J_{K,} = \emptyset$ for all $K$.
2. On reception of $(M_j, \mathtt{GenerateCoins}, K)$ from $\mathcal{C_I}$, set $J_K \leftarrow J_K \cup \{j\}$. If $|J_K| > k/2$, then set $J_K \leftarrow \emptyset$ choose $c \in \{0,1\}^K$ and hand $((\mathcal{S}, \mathtt{Coins}, c), \{(M_j, \mathtt{Coins}, c)\}_{j=1}^{k})$ to $\mathcal{C_I}$.

It is not hard to securely realize the coin-flipping functionality using a UC-secure verifiable secret sharing scheme (cf. [1]). Each mix-server $M_j$ chooses a random string $c_j$ of $K$ bits and secretly shares it. Then all secrets are reconstructed and $c$ is defined as $\oplus_{j=1}^{k} c_j$.

Finally, we give the protocol which securely realizes $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$. This is essentially a translation of Protocol 2 into a multiparty protocol in the UC-setting.

**Protocol 3 (Zero-Knowledge Proof of Decryption-Permutation).** The protocol $\pi_{\mathrm{DP}} = (M_1, \ldots, M_k)$ consists of mix-servers $M_j$ and proceeds as follows.

Mix-Server $M_j$. Each mix-server $M_j$ proceeds as follows.

1. Wait for $(\mathtt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h})$ from $\mathcal{F}_{\mathrm{RSA}}$. Then hand $(\mathtt{GenerateCoins}, NK_1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, (g_1', \ldots, g_N'))$. Then map these strings to elements in $G_q$ by $g_i = (g_i')^{(p-1)/q} \mod p$ (recall that $G_q \subset \mathbb{Z}_p^*$).
2. On input $(\mathtt{Prover}, (g, z, y, L, L'), (w, x))$, where $((g, z, y, L, L'), (w, x)) \in L_{R_{\mathrm{DP}}}$
   (a) Hand $(\mathtt{Prover}, (g, z, 1, 1), w)$ and $(\mathtt{Prover}, (g, y, 1, 1), x)$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
   (b) Denote by $W$ the first message of the prover in Protocol 2. Then hand $(\mathtt{Write}, \mathtt{W}, W)$ to $\mathcal{F}_{\mathrm{BB}}$.
   (c) Then hand $(\mathtt{GenerateCoins}, K_1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, s)$. Then set $P = \mathsf{PGen}(\mathsf{PRG}(s))$. If $P = \perp$ go to Step 2c, otherwise let $P$ be the primes used by the prover in Protocol 2.
   (d) Denote by $C$ the second message of the prover in Protocol 2. Hand $(\mathtt{Write}, \mathtt{C}, C)$ to $\mathcal{F}_{\mathrm{BB}}$. Then hand $(\mathtt{GenerateCoins}, K_4 - 1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, c')$. Let $c = c' + 2^{K_4 - 1}$ be the final challenge in Protocol 2.
   (e) Denote by $R$ the third message of the prover in Protocol 2. Hand $(\mathtt{Write}, \mathtt{R}, R)$ to $\mathcal{F}_{\mathrm{BB}}$.
3. On input $(\mathtt{Question}, M_l, (g, z, y, L, L'))$, where $L, L' \in G_q^{2N}$ and $(z, y) \in G_q$
   (a) Hand $(\mathtt{Question}, M_l, (g, z, 1, 1))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ and wait until it returns $(\mathtt{Verifier}, M_l, b_{z,l})$. Then hand $(\mathtt{Question}, M_l, (g, y, 1, 1))$ and wait until it returns $(\mathtt{Verifier}, M_l, b_{y,l})$. If $b_{z,l} b_{y,l} = 0$ output $(\mathtt{Verifier}, M_l, 0)$.
   (b) Then wait until $(M_l, \mathtt{W}, W)$ appears on $\mathcal{F}_{\mathrm{BB}}$. Hand $(\mathtt{GenerateCoins}, K_1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, s)$. Then set $P = \mathsf{PGen}(\mathsf{PRG}(s))$. If $P = \perp$ go to Step 3b, otherwise let $P$ be the primes used by the verifier in Protocol 2.

(c) Wait until $(M_l, \mathtt{C}, C)$ appears on $\mathcal{F}_{\mathrm{BB}}$. Then hand $(\mathtt{GenerateCoins}, K_4 - 1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, c')$, and until $(M_l, \mathtt{R}, R)$ appears on $\mathcal{F}_{\mathrm{BB}}$. Let $c = c' + 2^{K_4 - 1}$ be the final challenge in Protocol 2. Then verify $(W, P, C, c, R)$ as in Protocol 2 and set $b_j = 1$ or $b_j = 0$ depending on the result.

(d) Hand $(\mathtt{Write}, \mathtt{Judgement}, M_l, b_j)$ to $\mathcal{F}_{\mathrm{BB}}$ and wait until $(M_{l'}, \mathtt{Judgement}, M_l, b_{l'})$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l' \neq j$. Set $b = 1$ if $|\{b_{l'} \mid b_{l'} = 1\}| > k/2$ and otherwise $b = 0$. Output $(\mathtt{Verifier}, M_l, L, L', b)$.

**Theorem 3.** *The ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ is securely realized by $\pi_{\mathrm{DP}}$ in the $(\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{CF}}, \mathcal{F}_{\mathrm{RSA}}, \mathcal{F}_{\mathrm{BB}})$-hybrid model with respect to $\mathcal{M}_{k/2}$-adversaries under the DL-assumption and the strong RSA-assumption.*

**Corollary 2.** *The composition of $\pi_{\mathrm{MN}}$, $\pi_{\mathrm{C}}$, $\pi_{\mathrm{DP}}$, securely realizes $\mathcal{F}_{\mathrm{MN}}$ in the $(\mathcal{F}_{\mathrm{SKS}}, \mathcal{F}_{\mathrm{CF}}, \mathcal{F}_{\mathrm{RSA}}, \mathcal{F}_{\mathrm{BB}})$-hybrid model with respect to $\mathcal{M}_{k/2}$-adversaries under the DDH-assumption and the strong RSA-assumption.*

As indicated in the body of the paper all assumptions except the assumption of a bulletin board can be eliminated. The assumption of a bulletin board can only be eliminated for blocking adversaries (cf. [49]).

## 7 Conclusion

We have introduced a novel way to construct a mix-net, and given the first provably secure sender verifiable mix-net. We have also introduced a novel approach to construct a proof of a shuffle, and shown how this can be used to securely realize the ideal zero-knowledge proof of knowledge functionality for a decrypt-permutation relation. Combined, this gives the first universally composable mix-net that is efficient for any number of mix-servers.

## 8 Acknowledgments

I thank Johan Håstad for excellent advise, in particular for discussing efficient generation of the primes. I also thank Mårten Trolin for discussions.

## References

1. M. Abe, S. Fehr, *Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography*, to appear at Crypto 2004. (full version at Cryptology ePrint Archive, Report 2004/118, http://eprint.iacr.org/, May, 2004).
2. M. Abe, *Universally Verifiable mix-net with Verification Work Independent of the Number of Mix-centers*, Eurocrypt '98, pp. 437-447, LNCS 1403, 1998.
3. M. Abe, *Flaws in Some Robust Optimistic Mix-Nets*, In Proceedings of Information Security and Privacy, 8th Australasian Conference, LNCS 2727, pp. 39-50, 2003.
4. R. C. Baker and G. Harman, *The difference between consecutive primes*, Proc. Lond. Math. Soc., series 3, 72 (1996) 261–280.

5. D. Beaver, *Foundations of secure interactive computation*, Crypto '91, LNCS 576, pp. 377-391, 1991.
6. D. Boneh, and M. Franklin, *Efficient generation of shared RSA keys*, Crypto' 97, LNCS 1233, pp. 425-439, 1997.
7. J. Buus Nielsen, *Universally Composable Zero-Knowledge Proof of Membership*, manuscript, http://www.brics.dk/ buus/, April, 2005.
8. R. Canetti, *Security and composition of multi-party cryptographic protocols*, Journal of Cryptology, Vol. 13, No. 1, winter 2000.
9. R. Canetti, *Universally Composable Security: A New Paradigm for Cryptographic Protocols*, http://eprint.iacr.org/2000/067 and ECCC TR 01-24. Extended abstract appears in 42nd FOCS, IEEE Computer Society, 2001.
10. D. Chaum, *Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms*, Communications of the ACM - CACM '81, Vol. 24, No. 2, pp. 84-88, 1981.
11. R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, *Universally Composable Two-Party and Multi-Party Secure Computation*, 34th STOC, pp. 494-503, 2002.
12. I. Damgård, E. Fujisaki, *A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order*, Asiacrypt 2002, LNCS 2501, pp. 125-142, 2002.
13. Y. Desmedt, K. Kurosawa, *How to break a practical MIX and design a new one*, Eurocrypt 2000, pp. 557-572, LNCS 1807, 2000.
14. T. El Gamal, *A Public Key Cryptosystem and a Signiture Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, No. 4, pp. 469-472, 1985.
15. P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, 28th FOCS, pp. 427-438, 1987.
16. P. Fouque and J. Stern, *Fully Distributed Threshold RSA under Standard Assumptions*, Cryptology ePrint Archive, Report 2001/008, 2001.
17. A. Fujioka, T. Okamoto and K. Ohta, *A practical secret voting scheme for large scale elections*, Auscrypt '92, LNCS 718, pp. 244-251, 1992.
18. E. Fujisaki, T. Okamoto, *Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations*, Crypto 97, LNCS 1294, pp. 16-30, 1997.
19. J. Furukawa, K. Sako, *An efficient scheme for proving a shuffle*, Crypto 2001, LNCS 2139, pp. 368-387, 2001.
20. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, K. Sako, *An implementation of a universally verifiable electronic voting scheme based on shuffling*, Financial Cryptography '02, 2002.
21. J. Furukawa, *Efficient, Verifiable Shuffle Decryption and its Requirements of Unlinkability*, PKC 2004, LNCS 2947, pp. 319-332, 2004.
22. O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, 19th STOC, pp. 218-229, 1987.
23. O. Goldreich, *Foundations of Cryptography*, Cambridge University Press, 2001.
24. S. Goldwasser, L. Levin, *Fair computation of general functions in presence of immoral majority*, Crypto '90, LNCS 537, pp. 77-93, 1990.
25. S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences (JCSS), Vol. 28, No. 2, pp. 270-299, 1984.
26. P. Golle, S. Zhong, D. Boneh, M. Jakobsson, A. Juels, *Optimistic Mixing for Exit-Polls*, Asiacrypt 2002, LNCS, 2002.
27. N. Groth, *A Verifiable Secret Shuffle of Homomorphic Encryptions*, PKC 2003, pp. 145-160, LNCS 2567, 2003.
28. N. Groth, *Personal Communication*, 2004.
29. J. Buchmann, S. Hamdy, *A Survey on IQ Cryptography*, In Public-Key Cryptography and Computational Number Theory, Walter de Gruyter, pp. 1-15, 2001.

30. M. Jakobsson, *A Practical Mix*, Eurocrypt '98, LNCS 1403, pp. 448-461, 1998.
31. M. Jakobsson, *Flash Mixing*, In Proceedings of the 18th ACM Symposium on Principles of Distributed Computing - PODC '98, pp. 83-89, 1998.
32. M. Jakobsson, A. Juels, *Millimix: Mixing in small batches*, DIMACS Techical report 99-33, June 1999.
33. M. Jakobsson, A. Juels, *An optimally robust hybrid mix network*, In Proceedings of the 20th ACM Symposium on Principles of Distributed Computing - PODC '01, pp. 284-292, 2001.
34. S. Micali, P. Rogaway, *Secure Computation*, Crypto '91, LNCS 576, pp. 392-404, 1991.
35. M. Michels, P. Horster, *Some remarks on a reciept-free and universally verifiable Mix-type voting scheme*, Asiacrypt '96, pp. 125-132, LNCS 1163, 1996.
36. M. Mitomo, K. Kurosawa, *Attack for Flash MIX*, Asiacrypt 2000, pp. 192-204, LNCS 1976, 2000.
37. A. Neff, *A verifiable secret shuffle and its application to E-Voting*, In Proceedings of the 8th ACM Conference on Computer and Communications Security - CCS 2001, pp. 116-125, 2001.
38. A. Neff, *Verifiable Mixing (Shuffling) of ElGamal Pairs*, preliminary full version of [37], http://www.votehere.com/documents.html, Mars, 2005.
39. V. Niemi, A. Renvall, *How to prevent buying of votes in computer elections*, Asiacrypt'94, LNCS 917, pp. 164-170, 1994.
40. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, *Fault Tolerant Anonymous Channel*, Information and Communications Security - ICICS '97, pp. 440-444, LNCS 1334, 1997.
41. P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Eurocrypt '99, LNCS 1592, pp. 223-238, 1999.
42. C. Park, K. Itoh, K. Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, Eurocrypt '93, LNCS 765, pp. 248-259, 1994.
43. B. Pfitzmann, *Breaking an Efficient Anonymous Channel*, Eurocrypt '94, LNCS 950, pp. 332-340, 1995.
44. B. Pfitzmann, A. Pfitzmann, *How to break the direct RSA-implementation of mixes*, Eurocrypt '89, LNCS 434, pp. 373-381, 1990.
45. B. Pfitzmann, M. Waidner, *Composition and Integrity Preservation of Secure Reactive Systems*, 7th Conference on Computer and Communications Security of the ACM, pp. 245-254, 2000.
46. P. Ribenboim, *The new book of prime number records*, 3rd ed., ISBN 0-38794457-5, Springer-Verlag, 1996.
47. K. Sako, J. Killian, *Reciept-free Mix-Type Voting Scheme*, Eurocrypt '95, LNCS 921, pp. 393-403, 1995.
48. D. Wikström, *Five Practical Attacks for "Optimistic Mixing for Exit-Polls"*, In proceedings of Selected Areas of Cryptography (SAC), LNCS 3006, pp. 160-174, 2003.
49. D. Wikström, *A Universally Composable Mix-Net*, Proceedings of First Theory of Cryptography Conference (TCC '04), LNCS 2951, pp. 315-335, 2004.
50. D. Wikström, *A Sender Verifiable Mix-Net and a New Proof of a Shuffle*, Cryptology ePrint Archive, Report 2005/137, 2005, http://eprint.iacr.org/.