A Verifiable Secret Shuffle of Homomorphic Encryptions

Jens Groth[1,2]

[1] Cryptomathic A/S, Denmark
www.cryptomathic.com
[2] BRICS, University of Aarhus, Denmark*
jg@brics.dk

Abstract. We show how to prove in honest verifier zero-knowledge the correctness of a shuffle of homomorphic encryptions (or homomorphic commitments.) A shuffle consists in a rearrangement of the input ciphertexts and a reencryption of them so that the permutation is not revealed. Our scheme is more efficient than previous schemes both in terms of communication complexity and computational complexity. Indeed, in the case of shuffling ElGamal encryptions, the proof of correctness is smaller than the encryptions themselves.

1 Introduction

A shuffle of ciphertexts $E_1, \ldots, E_n$ is a new set of ciphertexts $E'_1, \ldots, E'_n$ so that both sets of ciphertexts have the same plaintexts. If we are working with a homomorphic cryptosystem with encryption algorithm $E_{pk}(\cdot)$, we may shuffle $E_1, \ldots, E_n$ by selecting a permutation $\pi \in \Sigma_n$ and letting $E'_1 = E_{\pi(1)}E_{pk}(0), \ldots, E'_n = E_{\pi(n)}E_{pk}(0)$. If the cryptosystem is semantically secure it is not possible for somebody else to see which permutation we used in the shuffle. On the other hand this also means that somebody else cannot check directly if we did make a correct shuffle. Our goal in this paper is to construct a proof system that enables us to prove that indeed we have made a correct shuffle.

Shuffling encrypted elements and proving the correctness of the shuffle play an important part in mix-nets. A mix-net is a multi-party protocol to shuffle elements so that neither of the parties knows the permutation linking the input and output. To shuffle ciphertexts we may let the parties one after another make a shuffle with a random permutation and prove correctness of it. The proofs of correctness allow us to catch any cheater, and if at least one party is honest, it is impossible to link the input and output. In this indirect fashion, shuffling plays an important role in anonymization protocols and voting schemes.

Currently the two most efficient proof systems, both public coin honest verifier zero-knowledge, for proving the correctness of a shuffle of ElGamal encryptions are the schemes by Furukawa and Sako [4] and Neff [6]. The proof system

* Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.

Y.G. Desmedt (Ed.): PKC 2003, LNCS 2567, pp. 145–160, 2003.

in [4], the most efficient scheme of the two, requires the prover to make $8n$ exponentiations and the verifier to make $10n$ exponentiations. In the case where the ElGamal cryptosystem is based on a 1024-bit prime p with the operations taking place in a group of order q, where q is a 160-bit prime so that $q|p-1$, their proofs require the prover to send $5280n$ bits. Jakobsson, Juels and Rivest take a different approach in [5] and provide a global structure for creating an efficient mix-net. In their scheme, however, a cheating prover does have non-negligible chance of succeeding, the scheme only ensures that only a very small fraction of the ciphertexts may be substituted for something else. Furthermore, it does not hide the permutation completely.

Our scheme, unlike those of [4] and [6], is not restricted to proving shuffles of ElGamal ciphertexts but can be used with any homomorphic cryptosystem. However, for the sake of comparison with the previous schemes we consider what happens when we use our scheme on an ElGamal cryptosystem with the primes p, q chosen as above. The prover uses around $6n$ exponentiations to make a proof, the verifier uses roughly $6n$ exponentiations to verify a proof, and the proofs themselves are of size around $1184n$ bits. In some cases, for instance in connection with electronic voting, the encrypted data may have to remain uncompromised several years into the future and then a key length of 1024 bits cannot be considered adequate. We gain an additional advantage in comparison with [4] and [6] when the key size increases since the groups in which we do the exponentiations may be smaller than the groups used in the cryptosystem. Our proof system uses 7 rounds just as that of [6], whereas that of [4] only uses 3 rounds. The proof system we construct is public coin honest verifier zero-knowledge,[1] just as [4] and [6].

The main ideas in our proof system are quite general and, while we only consider homomorphic encryptions in this paper, can be used for proving the correctness of shuffles of both homomorphic encryptions and homomorphic commitments. When considering homomorphic commitments, where we may have only computational binding, the proof takes on the nature of a proof of knowledge of the permutation and the modifications done to the commitments, with knowledge error that can be the inverse of any polynomial in the security parameter. Indeed, also when working with encryptions, the proof system we present proves knowledge of the shuffle being correct. This does have some interest in its own right for achieving non-malleability of the secret shuffle. Consider for instance a party doing one of the shuffles in a mix-net. Surely, the global properties of the mix-net are unsatisfactory if a corrupt party can shuffle all the elements back again right after this shuffle! Some degree of non-malleability is therefore needed.

Let us give a brief introduction to our proof system and the tools we use. The shuffle can be done by selecting a permutation π at random, selecting randomizers $r'_1, \ldots, r'_n$ at random, and setting $E'_1 = E_{\pi(1)}E_{pk}(0; r'_1), \ldots, E'_n =$

[1] Actually it is special honest verifier zero-knowledge, see [1]. Special honest verifier knowledge says that given any challenges it is possible to simulate a proof with those challenges.

$E_{\pi(n)}E_{pk}(0;r'_n)$. The task for the prover is now to prove that some permutation π exists so that the plaintexts of $E'_1,\ldots,E'_n$ and $E_{\pi(1)},\ldots,E_{\pi(n)}$ are identical.

As a first step, we think of the following naïve proof system. The prover informs the verifier of the permutation π. The verifier picks at random $t_1,\ldots,t_n$, computes $E_1^{t_1}\cdots E_n^{t_n}$ and $(E'_1)^{t_{\pi(1)}}\cdots(E'_n)^{t_{\pi(n)}}$. Finally, the prover proves that the two resulting ciphertexts have the same plaintext in common. Unless π really corresponds to a pairing of ciphertexts with identical plaintexts the prover will be caught with overwhelming probability.

The obvious problem with the above scheme is the lack of zero-knowledge. We remedy this in the following way:

1. The prover commits to the permutation π by choosing different elements $s_1,\ldots,s_n$ and setting $c_{s,1}=\mathit{commit}(s_{\pi(1)}),\ldots,c_{s,n}=\mathit{commit}(s_{\pi(n)})$. He sends, in that order, $s_1,\ldots,s_n,c_{s,1},\ldots,c_{s,n}$ to the verifier. Additionally he proves that indeed $s_1,\ldots,s_n$ are inside the commitments. This fixes a permutation π.
2. The verifier selects at random $t_1,\ldots,t_n$ and the prover sends $c_{t,1}=\mathit{commit}(t_{\pi(1)}),\ldots,c_{t,n}=\mathit{commit}(t_{\pi(n)})$ to the verifier. He proves that he has committed to $t_1,\ldots,t_n$, and he proves that the pairs (s_i,t_i) match in the commitments, i.e., for all i the pair $(c_{s,i},c_{t,i})$ contains a pair (s_j,t_j), where $1\leq j\leq n$. This ensures that $t_1,\ldots,t_n$ are committed to in the sequence specified by π.
3. Finally, the prover uses multiplication proofs and equivalence proofs to show that the products $E_1^{t_1}\cdots E_n^{t_n}$ and $(E'_1)^{t_{\pi(1)}}\cdots(E'_n)^{t_{\pi(n)}}$ have equivalent contents without revealing anything else. This last step corresponds to carrying out the naïve proof system in zero-knowledge.

The remaining problem is to convince the verifier that $c_{s,1},\ldots,c_{s,n}$ contain a shuffle of $s_1,\ldots,s_n$, that $c_{t,1},\ldots,c_{t,n}$ contain a shuffle of $t_1,\ldots,t_n$, and finally that the two sequences of elements have been shuffled in the same way. It seems like we have just traded one shuffle problem with another. The difference is that the supposed contents of the commitments are known to both the prover and the verifier, whereas we cannot expect either to know the contents of the ciphertexts being shuffled. Following an idea from [6] we can prove efficiently a shuffle when the contents are known and we are using a homomorphic commitment scheme.[2]

To see that the pairs match we let the verifier pick λ at random, and let the prover demonstrate that $c_{s,1}c_{t,1}^{\lambda},\ldots,c_{s,n}c_{t,n}^{\lambda}$ contain a shuffle of $s_1+\lambda t_1,\ldots,s_n+\lambda t_n$. If a pair (s_i,t_i) is not contained in a pair of commitments as required, then with high likelihood over the choice of λ the shuffle proof will be impossible. We shall see later that we can combine the proofs associated with step 1 and 2 into one combined proof.

[2] See Section 3.

2 Homomorphic Commitments and Encryption

2.1 Commitments

We start by specifying the type of commitments we are using. First there is the key generation phase in which a public key, K, is generated. The key generation procedure is not of concern in this article so we will just assume whenever we use a key in the article that it has been appropriately generated and publicized. In general, we use the letter K to signify such a key. With each key there is an associated message space $\mathcal{M}_K$, a randomizer space $\mathcal{R}_K$, an opening space $\mathcal{B}_K \supset \mathcal{R}_K$, a commitment space $\mathcal{C}_K$, a commitment function $com_K(\cdot,\cdot) : \mathcal{M}_K \times \mathcal{R}_K \to \mathcal{C}_K$ and a verification function $ver_K(\cdot,\cdot,\cdot) : \mathcal{M}_K \times \mathcal{B}_K \times \mathcal{C}_K \to \{0,1\}$.

Given the key we can commit to an element $m \in \mathcal{M}_K$ by selecting at random, according to some distribution specified by the commitment scheme and the key, $r \in \mathcal{R}_K$, and letting the commitment be $c = com_K(m;r) \in \mathcal{C}_K$. This (m,r,c)-triple satisfies $ver_K(m,r,c) = 1$.

To open a commitment we reveal $m \in \mathcal{M}_K, r \in \mathcal{B}_K$ such that $ver_K(m,r,c) = 1$. Note that we do allow openings not corresponding to correctly formed commitments since the opening space and the randomizer space do not need to be identical.[3] However, we still require the binding property to be satisfied, i.e., nobody can find a commitment in $\mathcal{C}_K$ and two correct openings of it with different messages m_1 and m_2.

The spaces associated with the commitment scheme shall be abelian groups[4]. We write the commitment space with multiplicative notation and the other groups with additive notation, so we have groups $(\mathcal{M}_K,+),(\mathcal{R}_K,+) \leq (\mathcal{B}_K,+)$ and $(\mathcal{C}_K,\cdot)$. In this paper the message space will be $\mathbf{Z}_N$ for some suitable large integer N, and the randomizer space will be some finite group where elements are chosen uniformly at random. See, however, Section 5 for a possible use of integer commitments.

Homomorphic property: The commitment schemes we look at must be homomorphic, meaning that for all $m_1, m_2 \in \mathcal{M}_K$ and all $r_1, r_2 \in \mathcal{B}_K$:

$$com_K(m_1;r_1)com_K(m_2;r_2) = com_K(m_1+m_2;r_1+r_2) \ .$$

Root opening: We demand that if we can find $c \in \mathcal{C}_K$, $e \neq 0$ with $\gcd(e,|\mathcal{M}_K|) = 1$[5] and $m \in \mathcal{M}_K, z \in \mathcal{B}_K$ such that $com_K(m;z) = c^e$ then we can compute an opening of c.

[3] See [2] for an example of an integer commitment scheme where the randomizer space and the opening space are different.

[4] Throughout the paper we assume that both the groups and the elements in the groups we work with can be represented in a suitable manner, the binary operations and inversions can be computed efficiently, and that we can recognize whether an element belongs to a particular group.

[5] If $\mathcal{M}_K$ is infinite we define $\gcd(e,|\mathcal{M}_K|) = 1$ for all e.

Few polynomial roots: For any non-zero polynomial $p(T) \in \mathcal{M}_K[T]$ of low degree[6], the probability that a randomly picked element from $\mathcal{M}_K$ is a root is negligible.

For future use, we note that the few polynomial roots assumption implies that the order of $\mathcal{M}_K$, if finite, does not have small divisors. This in turn means that when picking an integer at random from a sufficiently large interval it is with overwhelming probability prime to $|\mathcal{M}_K|$, a useful fact to bear in mind in connection with the root opening assumption.

We give the following example of a homomorphic commitment scheme. The key is $K = (p, q, g, h)$, where p, q are large primes, $q|p-1$, g, h are generators of a group $G \leq \mathbf{Z}_p^*$ and $\log_g h, \log_h g$ are both unknown. We have $\mathcal{M}_K = \mathbf{Z}_q, \mathcal{C}_K = <g>$ and $\mathcal{B}_K = \mathcal{R}_K = \mathbf{Z}_q$. Commitment to m is done by selecting $r \in \mathbf{Z}_q$ at random and letting $com_K(m; r) = g^r h^m \bmod p$. Verifying a commitment opening (m, r) of $c \in \mathcal{C}_K$ is done by letting $ver_K(m, r, c) = 1$ if and only if $m, r \in \mathbf{Z}_q$ and $c = g^r h^m \bmod p$.

2.2 Multicommitments

In the commitment scheme described above the elements in the message space have a natural description as integers. However, there are circumstances where we do not really need this property but we just need the message space to have a structure as an abelian group. Potentially, having fewer restrictions on the commitment scheme allows for faster generation of commitments.

In our case, what we will need is a way to commit to a tuple of elements. We therefore describe what we in the paper shall call multicommitments. A multicommitment scheme works just as a commitment scheme, except we may commit to a k-tuple from the message space $\mathcal{M}_{MK}$, where k is specified in the key MK.

An example of a multicommitment scheme is the following variant of the commitment scheme mentioned previously. The key consists of large primes p, q so that $q|p-1$, and randomly chosen generators $g, h_1, \ldots, h_k$ of a group $G \leq \mathbf{Z}_p^*$ of order q. The sender of the commitment does not have knowledge about the generation of the keys, in particular, he does not know discrete logarithms of the generators with respect to other generators. The message space is $\mathcal{M}_{(p,q,g,h_1,\ldots,h_k)} = \mathbf{Z}_q$, the randomizer and opening spaces are $\mathcal{R}_{(p,q,g,h_1,\ldots,h_k)} = \mathcal{B}_{(p,q,g,h_1,\ldots,h_k)} = \mathbf{Z}_q$ and the commitment space is $\mathcal{C}_{(p,q,g,h_1,\ldots,h_k)} = < g >$. The commitment function is defined as follows: $mcom_{(p,q,g,h_1,\ldots,h_k)}(m_1, \ldots, m_k; r) = g^r h_1^{m_1} \cdots h_k^{m_k} \bmod p$. Opening of the commitment is done by revealing $m_1, \ldots, m_k, r$.

2.3 Encryptions

In this paper, we use pk to denote a public key for a semantically secure homomorphic public key cryptosystem. Associated with such a key is a message

[6] More precisely of degree less than n, the number of elements we want to shuffle.

space $\mathcal{M}_{pk}$, a randomizer space $\mathcal{R}_{pk}$ and a ciphertext space $\mathcal{C}_{pk}$. Furthermore, we have an encryption function $E_{pk}(\cdot;\cdot) : \mathcal{M}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$. We restrict ourselves to cryptosystems with errorless decryption, i.e., it is infeasible to find $m \in \mathcal{M}_{pk}, r \in \mathcal{R}_{pk}$ so that $E_{pk}(m;r)$ does not decrypt to m. We also require that we have stable decryption in the sense that we cannot find an element $E \in \mathcal{C}_{pk}$ that decrypts to anything but a message in the message space or can decrypt to two different messages with significant[7] probability.

We require that the message space, the randomizer space and the ciphertext space are large finite groups. In other words we have groups $(\mathcal{M}_{pk}, +), (\mathcal{R}_{pk}, +)$ and $(\mathcal{C}_{pk}, \cdot)$. We will also assume a couple of properties that correspond to those of the commitments:

Homomorphic properties: The cryptosystem is homomorphic, meaning that for all $m_1, m_2 \in \mathcal{M}_{pk}$ and for all $r_1, r_2 \in \mathcal{R}_{pk}$:

$$E_{pk}(m_1; r_1)E_{pk}(m_2; r_2) = E_{pk}(m_1 + m_2; r_1 + r_2) \ .$$

In addition, if $E_1, E_2 \in \mathcal{C}_{pk}$ are two ciphertexts decrypting to m_1, m_2 respectively then E_1E_2 decrypts to $m_1 + m_2$.

When dealing with a shuffle of ciphertexts $E_1, \ldots, E_n$ and $E'_1, \ldots, E'_n$ we will typically not be interested in whether they have been formed correctly, i.e., there exists a permutation π and $r'_1, \ldots, r'_n$ so that $E'_1 = E_{\pi(1)}E_{pk}(0; r'_1), \ldots, E'_n = E_{\pi(n)}E_{pk}(0; r'_n)$. What we really want to know is whether they decrypt correctly, i.e., there exists a permutation π so that for all i we have that E'_i and $E_{\pi(i)}$ decrypt to the same.

This difference means that when dealing with commitments we typically want to prove knowledge of openings of commitments satisfying some property. When dealing with ciphertexts we typically want to prove knowledge that their contents satisfy some property. Accordingly, we modify the root opening assumption to something that will be more suitable when talking about ciphertexts later in the article.

Root decryption: We demand that if we can find $E \in \mathcal{C}_{pk}$, $e \neq 0$ so that $\gcd(e, |\mathcal{M}_{pk}|) = 1$ and $m \in \mathcal{M}_{pk}, z \in \mathcal{R}_{pk}$ so that $E_{pk}(m; z) = E^e$, then m can be written uniquely as em', where $m' \in \mathcal{M}_{pk}$, and E decrypts to m'.

We mention two examples of cryptosystems that satisfy our requirements. The ElGamal cryptosystem may be set up with the same kind of key as in the commitment scheme example. This time the message space is $\mathcal{M}_{(p,q,g,h)} = <g>$. We encrypt by letting $E_{(p,q,g,h)}(m; r) = (g^r \bmod p, h^r m \bmod p)$. Decryption can be done if the secret exponent x so that $h = g^x \bmod p$ is known. The ciphertext (u, v) decrypts to $m = vu^{-x} \bmod p$.

Another example of a homomorphic cryptosystem is the generalization of Paillier's cryptosystem [7] invented by Damgård and Jurik [3]. Here the pub-

[7] By significant we mean not negligible.

lic key is an RSA-modulus N together with a small integer s. The encryption function takes a message $m \in \mathbf{Z}_{N^s}$ and a randomizer $r \in \mathbf{Z}_N^*$ as input and sets $E_{(N,s)}(m;r) = (1+n)^m r^{N^s} \bmod N^{s+1}$.

2.4 Compatibility of Commitments, Multicommitments and Encryptions

From now on, we will simply assume that we have some commitment scheme with key K, some multicommitment scheme with key MK and a cryptosystem with key pk. The keys may or may not overlap so that elements from one key are also included in another key. For instance, we could imagine the cryptosystem were an ElGamal scheme with primes $p, q | p-1$ and generators g, h and that the commitment scheme used the exact same elements.

We require that the keys be selected so that the message spaces are compatible in the following way. The message spaces of the commitment scheme and the multicommitment scheme are identical, i.e., $\mathcal{M}_K = \mathcal{M}_{MK}$. Furthermore, the message space for the cryptosystem, $\mathcal{M}_{pk}$ is a module over $\mathcal{M}_K$.

3 Proof of a Shuffle of Known Contents

Before going into the general protocol for proving a correct shuffle let us build some intuition by presenting, without proof, a proof system for a shuffle where the contents are known.

Say we have commitments $c_1, \ldots, c_n$ and want to prove that they contain $m_1, \ldots, m_n$ without revealing which commitment contains what. If the message space is an integral domain, we can, following [6], use the fact that an n'th degree non-zero polynomial has at most n roots. Let x be any element in the message space and set $c_x = com_K(x;0)$. In case $c_1, \ldots, c_n$ do not contain a shuffle of $m_1, \ldots, m_n$ there can at most be n challenges x where the product of the contents of $c_1 c_x^{-1}, \ldots, c_n c_x^{-1}$ is the same as $\prod_{i=1}^n (m_i - x)$. By choosing x at random, the verifier can therefore give the cheating prover a challenge he only has negligible chance to answer.

Proof of commitments containing a shuffle of known contents
Common input: $c_1, \ldots, c_n \in \mathcal{C}_K$ and $m_1, \ldots, m_n \in \mathcal{M}_K$.
Prover's input: A permutation $\pi \in \Sigma_n$ and $r_1, \ldots, r_n \in \mathcal{R}_K$ so that $c_1 = com_K(m_{\pi(1)}; r_1), \ldots, c_n = com_K(m_{\pi(n)}; r_n)$.

Initial challenge: Prover receives x chosen at random from $\mathcal{M}_K$.
Multiplication Proof: Make a 3-move proof using a multiplication proof of knowledge[8] that the product of the contents of $c_1 c_x^{-1}, \ldots, c_n c_x^{-1}$ equals the content in $com_K(\prod_{i=1}^n (m_i - x); 0)$.

[8] Such 3-move proofs are standard tools in cryptography. See for instance [3] for an example of such proofs.

Theorem 1. *The scheme is a public coin 4-move proof system for proving that the prover knows* π *and* $r_1, \ldots, r_n$ *so that* $c_1 = com_K(m_{\pi(1)}; r_1), \ldots, c_n = com_K(m_{\pi(n)}; r_n)$*. The proof system is special honest verifier zero-knowledge.*

The condition that the message space is an integral domain can be relaxed. In the assumptions on the commitment scheme, we only required that given some polynomial it should be unlikely that a randomly chosen element in $\mathcal{M}_K$ is a root, unless the polynomial is the zero-polynomial.

4 Proof of a Shuffle

Let us go back to the idea in the introduction for proving a shuffle this time having firmer grasp of the concepts. There are ciphertexts $E_1, \ldots, E_n, E'_1, \ldots, E'_n$, and the prover knows a permutation π and randomizers $r'_1, \ldots, r'_n$ so that $E'_1 = E_{\pi(1)}E_{pk}(0; r'_1), \ldots, E'_n = E_{\pi(n)}E_{pk}(0; r'_n)$. His aim is to convince the verifier that a permutation π exists so that for each $i = 1, \ldots, n$ the ciphertexts E'_i and $E_{\pi(i)}$ decrypt to the same.

The prover selects different elements $s_1, \ldots, s_n$ and creates commitments $c_{s,1} = com_K(s_{\pi(1)}), \ldots, c_{s,n} = com_K(s_{\pi(n)})$. He sends all of this to the verifier.

The verifier responds with $t_1, \ldots, t_n$ chosen at random in the message space. The prover sends $c_{t,1} = com_K(t_{\pi(1)}), \ldots, c_{t,n} = com_K(t_{\pi(n)})$ back.

The verifier sends a random λ to the prover. The prover demonstrates by using a shuffle proof of known contents that he knows that $c_{s,1}c_{t,1}^{\lambda}, \ldots, c_{s,n}c_{t,n}^{\lambda}$ contain a shuffle of $s_1 + \lambda t_1, \ldots, s_n + \lambda t_n$.

Unless indeed $(c_{s,1}, c_{t,1}), \ldots, (c_{s,n}, c_{t,n})$ contain a shuffle of the pairs $(s_1, t_1), \ldots, (s_n, t_n)$ there is overwhelming probability over the choices of λ that this proof will fail. In other words, we get three pieces of information from this proof: The commitments $c_{s,1}, \ldots, c_{s,n}$ sent in the first round did contain $s_1, \ldots, s_n$. The commitments $c_{t,1}, \ldots, c_{t,n}$ sent in the third round did contain $t_1, \ldots, t_n$. The s_i's and t_i's were shuffled using the same permutation.

What we have so far is that we can get the prover to commit to a permutation through the choices of $c_{s,1}, \ldots, c_{s,n}$, and subsequently we can ensure that he commits to $t_1, \ldots, t_n$ permuted in the same way. Let us call this permutation π.

The prover can conclude his proof by computing $E''_1 = (E'_1)^{t_{\pi(1)}}E_{pk}(0), \ldots, E''_n = (E'_n)^{t_{\pi(n)}}E_{pk}(0)$. Using basic multiplication proofs, he can demonstrate to the verifier that this set of ciphertexts has been correctly formed. Finally, he can show through an equality proof that $E''_1 \cdots E''_n$ has the same content as $E_1^{t_1} \cdots E_n^{t_n}$.

From the verifier's point of view, this demonstrates that $(E'_1)^{t_{\pi(1)}} \cdots (E'_n)^{t_{\pi(n)}}$ has the same content as $E_1^{t_1} \cdots E_n^{t_n}$. With overwhelming probability over the choice of $t_1, \ldots, t_n$ this is only possible if each of the pairs $(E'_1, E_{\pi(1)}), \ldots, (E'_n, E_{\pi(n)})$ are pairs of ciphertexts with the same plaintext. We have thus proven the shuffle without revealing π.

To optimize the proof we note the following. First, we only use $E''_1, \ldots, E''_n$ temporarily. We may as well prove that $(E'_1)^{t_{\pi(1)}} \cdots (E'_n)^{t_{\pi(n)}}$ has the same content as $E_1^{t_1} \cdots E_n^{t_n}$ directly if possible, and indeed this is possible. This way we save n encryptions.

Another optimization comes from the use of multicommitments. When making a shuffle proof for $c_{s,1}c_{t,1}^{\lambda}, \ldots, c_{s,n}c_{t,n}^{\lambda}$ containing a shuffle of $s_1 + \lambda t_1, \ldots, s_n + \lambda t_n$ we need to make multiplications and prove them done correctly. However, up to this point we only use additions. Therefore we can instead make multicommitments to $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ and from this create multicommitments to $s_1 + \lambda t_1 - x, \ldots, s_n + \lambda t_n - x$, where x is the randomly chosen challenge to be used in the shuffle proof with known contents. First now do we begin to prove that the product of the contents in the multicommitments matches the product $\prod_{i=1}^{n}(s_i + \lambda t_i - x)$.

When starting to make this proof we use the fact that we can combine the multiplication proof used in the shuffle proof of known contents and the multiplication proofs used with the encryptions. Instead of raising the ciphertexts to $t_1, \ldots, t_n$ we can raise them to $s_1 + \lambda t_1 - x, \ldots, s_n + \lambda t_n - x$. Since $t_1, \ldots, t_n, \lambda$ and x are all chosen by the verifier these values have the same random distribution as the original $t_1, \ldots, t_n$. But in the basic multiplication proofs used on the ciphertexts we create the values $f_1 = e(s_{\pi(1)} + \lambda t_{\pi(1)} - x) + d_1, \ldots, f_n = e(s_{\pi(n)} + \lambda t_{\pi(n)} - x) + d_n$ and reveal them. Having those values helps us create an efficient proof of the shuffle of known contents. The idea here is that we can compute $e^n \prod_{i=1}^{n}(s_i + \lambda t_i - x)$ as $f_1 \cdots f_n - d_1 f_2 \cdots f_n - (f_1 - d_1)d_2 f_3 \cdots f_n - \cdots - (f_1 - d_1) \cdots (f_{n_1} - d_{n-1})d_n$.

Let us now write down the proof system for proving a shuffle that we have been aiming for throughout the paper. For simplicity and without loss of generality we assume $k|n$.

Proof of shuffle of ciphertexts
Common input: $E_1, \ldots, E_n, E'_1, \ldots, E'_n \in \mathcal{C}_{pk}$.
Prover's input: A permutation $\pi \in \Sigma_n$ and randomizers $r'_1, \ldots, r'_n \in \mathcal{R}_{pk}$ satisfying $E'_1 = E_{\pi(1)}E_{pk}(0; r'_1), \ldots, E'_n = E_{\pi(n)}E_{pk}(0; r'_n)$.

Initial message: Select $s_1, \ldots, s_n$ as different elements from $\mathcal{M}_K$.
Select $r_{s,1}, \ldots, r_{s,\frac{n}{k}}$ at random from $\mathcal{R}_{MK}$.
Let $c_{s,1} = mcom_{MK}(s_{\pi(1)}, \ldots, s_{\pi(k)}; r_{s,1}), \ldots,$
$c_{s,\frac{n}{k}} = mcom_{MK}(s_{\pi(n-k+1)}, \ldots, s_{\pi(n)}; r_{s,\frac{n}{k}})$.
Send $s_1, \ldots, s_n, c_{s,1}, \ldots, c_{s,\frac{n}{k}}$ to the verifier.
First challenge: $t_1, \ldots, t_n$ chosen at random from $\mathcal{M}_K$.
First answer: Select $r_{t,1}, \ldots, r_{t,\frac{n}{k}}$ at random from $\mathcal{R}_{MK}$.
Set $c_{t,1} = mcom_{MK}(t_{\pi(1)}, \ldots, t_{\pi(k)}; r_{t,1}), \ldots,$
$c_{t,\frac{n}{k}} = mcom_{MK}(t_{\pi(n-k+1)}, \ldots, t_{\pi(n)}; r_{t,\frac{n}{k}})$.
Send $c_{t,1}, \ldots, c_{t,\frac{n}{k}}$ to the verifier.
Second challenge: Choose λ, x at random from $\mathcal{M}_K$.
Second answer: For $j = 1, \ldots, n$ let $a_j = \prod_{i=1}^{j}(s_{\pi(i)} + \lambda t_{\pi(i)} - x)$.

Select $d_1, \ldots, d_n$ at random from $\mathcal{M}_{MK}$. Select $r_{d,1}, \ldots, r_{d,\frac{n}{k}}$ at random from $\mathcal{R}_{MK}$. Set $c_{d,1} = mcom_{MK}(d_1, \ldots, d_k; r_{d,1}), \ldots, c_{d,\frac{n}{k}} = mcom_{MK}(d_{n-k+1}, \ldots, d_n; r_{d,\frac{n}{k}})$.
Select $r_1, \ldots, r_n$ at random from $\mathcal{R}_K$. Select r at random from $\mathcal{R}_K$. Set $c_1 = com_K(d_1; r_1), c_2 = com_K(a_1 d_2; r_2), \ldots, c_n = com_K(a_{n-1} d_n; r_n)$. Set $c = com_K(0; r)$.[9]
Select r' at random from $\mathcal{R}_{pk}$. Set $E' = (E'_1)^{d_1} \cdots (E'_n)^{d_n} E_{pk}(0; r')$.
Send $c_{d,1}, \ldots, c_{d,\frac{n}{k}}, c_1, \ldots, c_n, c, E'$ to the verifier.

Third challenge: Select at random $e \in \mathcal{M}_K$.

Final answer: Set $f_1 = e(s_{\pi(1)} + \lambda t_{\pi(1)} - x) + d_1, \ldots, f_n = e(s_{\pi(n)} + \lambda t_{\pi(n)} - x) + d_n$. Set $z_1 = e(r_{s,1} + \lambda r_{t,1}) + r_{d,1}, \ldots, z_{\frac{n}{k}} = e(r_{s,\frac{n}{k}} + \lambda r_{t,\frac{n}{k}}) + r_{d,\frac{n}{k}}$.
Set $z = r - ef_2 \cdots f_n r_1 - \cdots - e^n r_n$.
Set $z' = r' - (e(s_{\pi(1)} + \lambda t_{\pi(1)} - x) + d_1) r'_1 - \cdots - (e(s_{\pi(n)} + \lambda t_{\pi(n)} - x) + d_n) r'_n$.
Send $f_1, \ldots, f_n, z_1, \ldots, z_{\frac{n}{k}}, z, z'$ to the verifier.

Verification: Check that $f_1 \ldots, f_n \in \mathcal{M}_K$. Check that $c_{s,1}, \ldots, c_{s,\frac{n}{k}}, c_{t,1}, \ldots, c_{t,\frac{n}{k}}, c_{d,1}, \ldots, c_{d,\frac{n}{k}} \in \mathcal{C}_{MK}$. Check that $z_1, \ldots, z_{\frac{n}{k}} \in \mathcal{R}_{MK}$. Check that $c_1, \ldots, c_n, c \in \mathcal{C}_K$. Check that $z \in \mathcal{R}_K$. Check that $E' \in \mathcal{C}_{pk}$. Check that $z' \in \mathcal{R}_{pk}$.
Let $c_x = mcom_{MK}(x, \ldots, x; 0)$.
Check that $mcom_{MK}(f_1, \ldots, f_k; z_1) = (c_{s,1} c_{t,1}^{\lambda} c_x^{-1})^e c_{d,1}, \ldots,$
$mcom_{MK}(f_{n-k+1}, \ldots, f_n; z_{\frac{n}{k}}) = (c_{s,\frac{n}{k}} c_{t,\frac{n}{k}}^{\lambda} c_x^{-1})^e c_{d,\frac{n}{k}}$.
Check that $com_K(e^{n+1} a_n - ef_1 \cdots f_n; z) = c_1^{-ef_2 \cdots f_n} \cdots c_n^{-e^n} c$.
Check that $(E'_1)^{f_1} \cdots (E'_n)^{f_n} E_{pk}(0; z') = (E_1^{s_1 + \lambda t_1 - x} \cdots E_n^{s_n + \lambda t_n - x})^e E'$.

Theorem 2. *The scheme is a public coin 7-move proof system for a correct shuffle. It is complete, sound and special honest verifier zero-knowledge. If the commitments are statistically hiding, then the entire proof is statistical special honest verifier zero-knowledge.*

Proof. It is easy to see that we are dealing with a 7-move public coin protocol. Completeness follows by straightforward verification. Soundness follows from Lemma 1. Special honest verifier zero-knowledge follows from Lemma 2. □

Lemma 1. *The proof system is sound.*

Proof. We will assume that the proof system is not sound and derive a contradiction with the assumptions made on the commitment schemes and the cryptosystem that we use. Therefore, let us assume there is some adversary $\mathcal{A}$ that has a significant probability of succeeding in the following game. Public keys for the schemes are generated and given to the adversary. It then produces two sets of encryptions $E_1, \ldots, E_n$ and $E'_1, \ldots, E'_n$ and engages in a proof with the verifier. It is succesful if the verifier accepts the proof. By the definition of significant this means that there is some inverse polynomial, $\epsilon(l)$, in the security parameter l where $\mathcal{A}$ is succesful with probability $\epsilon(l)$ for an infinite number of possible security parameters.

[9] The a_j's can be computed recursively using the formula $a_j = a_{j-1}(s_{\pi(j)} + \lambda t_{\pi(j)} - x)$.

From the adversary $\mathcal{A}$ we will construct an algorithm $\mathcal{B}$ that is capable of violating the assumptions made on the schemes we use. Basically $\mathcal{B}$ will run $\mathcal{A}$ to get two sets of ciphertexts $E_1, \ldots, E_n$ and $E'_1, \ldots, E'_n$ and then use rewinding techniques to find witnesses for these decrypting to the same plaintexts. If they do not decrypt to the same plaintexts then the root decryption assumption has been violated.

Sampling of proofs

$\mathcal{B}$ works in the following way. It runs $\mathcal{A}$ to get ciphertexts $E_1, \ldots, E_n$ and $E'_1, \ldots, E'_n$. By the assumption we have $\epsilon(l)$ probability for these ciphertexts not decrypting to the same families of plaintexts, yet $\mathcal{A}$ being succesful in the proof. Let $p(l)$ be an easily computable polynomial in the security parameter that is sufficiently larger[10] than $\text{time}_{\mathcal{A}}(l)/\epsilon(l)$. $\mathcal{B}$ lets $\mathcal{A}$ send the initial message of the proof, and then selects at random $p(l)$ first challenges $t_1, \ldots, t_n$. On each such challenge $t_1, \ldots, t_n$ it runs the adversary $p(l)$ times to receive $p(l)$ answers to the first challenge. For each such answer it selects at random $p(l)$ second challenges λ, x. It splits $\mathcal{A}$ further into $p(l)$ copies and lets each copy produce an answer to one of the second challenges λ, x. For each of these answers it selects $p(l)$ final challenges e. It runs $\mathcal{A}$ to the end to see if it gets acceptable final answers to some of these final challenges.

All in all $\mathcal{B}$ runs $p(l)^3$ copies of $\mathcal{A}$. We hope to find n sets of challenges $t_1, \ldots, t_n$ so that for each of those sets we have 2 sets of challenges λ, x, where we in turn have $n + 2$ sets of challenges e with acceptable answers. All in all, we hope to end up with $2n(n + 2)$ related proofs. With overwhelming probability this happens if we have set $p(l)$ to be a large enough polynomial and the probability of $\mathcal{A}$ for succeeding after having sent the initial message is larger than $\epsilon(l)$.

With overwhelming probability these acceptable proofs include n linearly independent vectors $(s_1 + \lambda t_1 - x, \ldots, s_n + \lambda t_n - x)$. We also have with overwhelming probability that the two second challenges λ, x have different λ's. Finally, we have with overwhelming probability that the vectors $\boldsymbol{e} = (1, e, \ldots, e^{n+1})$ are linearly independent. To see the latter note that vectors of such a form can be seen as rows in a Vandermonde matrix and with high probability the Vandermonde matrix has determinant different from 0.

We now go backwards through the conversations that constitute the acceptable proofs we have found and see what we can conclude. First we look at a conversation that has been completed up to the point where the verifier is about to pick the final challenge e and give it to the prover. We see what can be deduced from the $n + 2$ answers to this challenge. Next we take a step back and look at a conversation where the verifier is about to pick the second challenge λ, x. Again we see what we can conclude from the $2(n + 2)$ acceptable continuations of a proof from this point. Finally, we step back to the point where the verifier is picking the first challenge, consisting of $t_1, \ldots, t_n$, and see what we can conclude

[10] We leave it to the reader to use Chernoff bounds to estimate exactly what sufficiently larger means.

from the n sets of initial challenges that are succesfully answered $2(n+2)$ times each.

We start out by looking at the situation where a conversation has been conducted up to the point where the verifier is selecting the final challenge.

The final challenge

Given acceptable answers $f_1, \ldots, f_n, z_1, \ldots, z_{\frac{n}{k}}, z, z'$ and $\widetilde{f}_1, \ldots, \widetilde{f_n}$, $\widetilde{z}_1, \ldots, \widetilde{z_{\frac{n}{k}}}, \widetilde{z}, \widetilde{z'}$ to two different challenges $e, \widetilde{e}$, we have for $i = 1, \ldots, \frac{n}{k}$:

$$mcom_{MK}(f_{ik-k+1}, \ldots, f_{ik}; z_i) = (c_{s,i} c_{t,i}^{\lambda} c_x^{-1})^e c_{d,i}$$
$$\wedge mcom_{MK}(\widetilde{f_{ik-k+1}}, \ldots, \widetilde{f_{ik}}; \widetilde{z}_i) = (c_{s,i} c_{t,i}^{\lambda} c_x^{-1})^{\widetilde{e}} c_{d,i} \ . \quad (1)$$

This implies that

$$mcom_{MK}(f_{ik-k+1} - \widetilde{f_{ik-k+1}}, \ldots, f_{ik} - \widetilde{f_{ik}}; z_i - \widetilde{z}_i) = (c_{s,i} c_{t,i}^{\lambda} c_x^{-1})^{e-\widetilde{e}}.$$

By the root opening assumption, this means that we can find an opening of $c_{s,i} c_{t,i}^{\lambda} c_x^{-1}$. The binding property of the multicommitment scheme implies that this is the only way we can open the multicommitments. We call the contents of $c_{s,i} c_{t,i}^{\lambda} c_x^{-1}$ for $m_{ki-k+1}, \ldots, m_{ki}$. We can now go back and find an opening of $c_{d,i}$ too. We write $d_{ki-k+1}, \ldots, d_{ki}$ for the content of this commitment. We have $f_i = em_i + d_i$, for $i = 1, \ldots, n$.

Looking at the encryptions we get from the answers to two challenges $e, \widetilde{e}$ that

$$(E_1')^{f_1 - \widetilde{f}_1} \cdots (E_n')^{f_n - \widetilde{f}_n} E_{pk}(0; z' - \widetilde{z'}) = (E_1^{s_1 + \lambda t_1 - x} \cdots E_n^{s_n + \lambda t_n - x})^{e - \widetilde{e}}.$$

Plugging in the f_i's and the corresponding m_i's we conclude that

$$E_{pk}(0; z' - \widetilde{z'}) = (E_1^{s_1 + \lambda t_1 - x} \cdots E_n^{s_n + \lambda t_n - x} (E_1')^{-m_1} \cdots (E_n')^{-m_n})^{e - \widetilde{e}}.$$

The root decryption assumption then tells us that $E_1^{s_1 + \lambda t_1 - x} \cdots E_n^{s_n + \lambda t_n - x}$ and $(E_1')^{m_1} \cdots (E_n')^{m_n}$ decrypt to the same.

A prover having significant chance of answering the last challenge can also be used to to conclude that $a_n = m_1 \cdots m_n$, where $a_n = \prod_{i=1}^{n} (s_{\pi(i)} + \lambda t_{\pi(i)} - x)$.

By writing out the f_i's we see that an answer to a challenge e satisfies

$$com_K(e^{n+1}(m_1 \cdots m_n - a_n) + e^n(d_1 m_2 \cdots m_n + \cdots + m_1 \cdots m_{n-1} d_n) + \cdots$$
$$+ e(d_1 \cdots d_n); -z) c = (c_1^{m_2 \cdots m_n} \cdots c_n^1)^{e^n} \cdots (c_1^{d_2 \ldots d_n})^e.$$

From a linear combination of $n+2$ linearly independent vectors on the form $(1, \ldots, e^{n+1})$ we can get a vector on the form $(0, \ldots, 0, \gamma)$, where γ with overwhelming probability is prime to $|\mathcal{M}_K|$. From answers to $n+2$ different e's that satisfy (1) this linear combination gives us an element $z_\gamma \in \mathcal{R}_K$ so that

$$com_K(\gamma(m_1 \cdots m_n - a_n); z_\gamma) = 1 \ .$$

This in turn shows that $z_\gamma = 0$ and $a_n = m_1 \cdots m_n$.

Going a step back in the proof we now look at a situation where the prover is about to receive the two challenges λ and x.

The second challenge

We first look at the challenge x. For such challenges we have seen that the contents $m_1, \ldots, m_n$ of $c_{s,1}c_{t,1}^{\lambda}c_x^{-1}, \ldots, c_{s,\frac{n}{k}}c_{t,\frac{n}{k}}^{\lambda}c_x^{-1}$ satisfy

$$a_n = \prod_{i=1}^{n}(s_i + \lambda t_i - x) = m_1 \ldots m_n = \prod_{i=1}^{n}(m'_i - x) \ ,$$

where $m'_1, \ldots, m'_n$ are the contents of $c_{s,1}c_{t,1}^{\lambda}, \ldots, c_{s,\frac{n}{k}}c_{t,\frac{n}{k}}^{\lambda}$. We have an evaluation of an n'th degree polynomial on both sides and by the few polynomial roots assumption the two evaluations must be equal if the prover has to have a significant chance of answering the challenges x and e. This in turn means that the two polynomials we evaluate have identical roots. Therefore the elements $m'_1, \ldots, m'_n$ are some permutation of $s_1 + \lambda t_1, \ldots, s_n + \lambda t_n$.

We proceed to look at the challenge λ. Being able to answer two different challenges λ and $\widetilde{\lambda}$ gives us openings of $c_{s,i}c_{t,i}^{\lambda}c_x^{-1}$ and $c_{s,i}c_{t,i}^{\widetilde{\lambda}}c_x^{-1}$ for $i = 1, \ldots, \frac{n}{k}$. By the homomorphic property of the multicommitment scheme this gives us openings of $c_{t,1}^{\lambda-\widetilde{\lambda}}, \ldots, c_{t,\frac{n}{k}}^{\lambda-\widetilde{\lambda}}$. By the root opening assumption we can therefore open $c_{t,1}, \ldots, c_{t,\frac{n}{k}}$. We call the contents for $\hat{t_1}, \ldots, \hat{t_n}$. We are now also able to open $c_{s,1}, \ldots, c_{s,\frac{n}{k}}$. We call the contents for $\hat{s_1}, \ldots, \hat{s_n}$.

Let us look at $s_1 + \lambda t_1$. With overwhelming probability over the choices of $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ there are at most n different values of λ where it would correspond to one of $\hat{s_1} + \lambda\hat{t_1}, \ldots, \hat{s_n} + \lambda\hat{t_n}$, unless there exists $i \in \{1, \ldots, n\}$ where $\hat{s_i} = s_1$ and $\hat{t_i} = t_1$. Arguing similarly for the pairs $(s_2, t_2), \ldots, (s_n, t_n)$ we deduce that there is a unique permutation π so that $\hat{s_i} = s_{\pi(i)}$ and $\hat{t_i} = t_{\pi(i)}$ for $i = 1, \ldots, n$. Since the prover committed to $s_1, \ldots, s_n$ in the initial message the permutation was fixed already then.

We take another step back in the conversation and see what we can conclude from the answers following the first challenge.

The first challenge

Since $s_1 + \lambda t_1 - x, \ldots, s_n + \lambda t_n - x$ are linearly indendent we can for $i = 1, \ldots, n$ find linear combinations of the vectors $(s_1 + \lambda t_1 - x, \ldots, s_n + \lambda t_n - x)$ giving us the vector $(0, \ldots, 0, \gamma_i, 0, \ldots, 0)$, where γ_i is prime to $|\mathcal{M}_{pk}|$. Returning to the encryptions we can therefore for $i = 1, \ldots, n$ find z'_{γ_i} so that $E_{pk}(0; z'_{\gamma_i}) = (E'_i)^{\gamma_i}E_{\pi(i)}^{-\gamma_i}$. This by the root decryption assumption implies that $E_{\pi(i)}$ and E'_i decrypt to the same.

This means that we have in polynomial time found witnesses on the form $E_{pk}(0; z'_{\gamma_i}) = (E'_iE_{\pi(i)}^{-1})^{\gamma_i}$ for the plaintexts of $E_1, \ldots, E_n$ and $E'_1, \ldots, E'_n$ decrypting to the same. But this contradicts the root decryption assumption if the ciphertexts are to decrypt to different plaintexts. We must therefore conclude that the adversary has negligible chance of cheating with the proof. □

Lemma 2. *The proof system is special honest verifier zero-knowledge. If the commitments and multicommitments are statistically or perfectly hiding then the proof system is statistical or perfect special honest verifier zero-knowledge.*

Proof. We first describe the simulator. Afterwards, we argue that its output is indistinguishable from a real proof.

The simulation

We are given $t_1, \ldots, t_n, \lambda, x, e$ as input, and wish to produce something that is indistinguishable from a real proof.

We let $c_{s_1} = mcom_{MK}(0, \ldots, 0), \ldots, c_{s,\frac{n}{k}} = mcom_{MK}(0, \ldots, 0)$.

We let $c_{t,1} = mcom_{MK}(0, \ldots, 0), \ldots, c_{t,\frac{n}{k}} = mcom_{MK}(0, \ldots, 0)$.

We let $c_1 = com_K(0), \ldots, c_n = mcom_K(0)$.

We select $f_1, \ldots, f_n$ at random from $\mathcal{M}_{MK}$, $z_1, \ldots, z_{\frac{n}{k}}$ at random from $\mathcal{R}_{MK}$, z at random from $\mathcal{R}_K$ and z' at random from $\mathcal{R}_{pk}$.

We let $c_x = mcom_{MK}(x, \ldots, x; 0)$.

We set $c_{d,1} = mcom_{MK}(f_1, \ldots, f_k; z_1)(c_{s,1}c_{t,1}^{\lambda}c_x^{-1})^{-e}, \ldots,$
$c_{d,\frac{n}{k}} = mcom_{MK}(f_{n-k+1}, \ldots, f_n; z_{\frac{n}{k}})(c_{s,\frac{n}{k}}c_{t,\frac{n}{k}}^{\lambda}c_x^{-1})^{-e}$.

We set $c = com_K(e^{n+1}a_n - ef_1 \cdots f_n; z)c_1^{ef_2 \cdots f_n} \cdots c_n^{e^n}$.

Finally we let $E' = E_{pk}(0; z')(E'_1)^{f_1} \cdots (E'_n)^{f_n}(E_1^{s_1+\lambda t_1 - x} \cdots E_n^{s_n+\lambda t_n - x})^{-e}$.

The simulated proof is $(s_1, \ldots, s_n, c_{s,1}, \ldots, c_{s,\frac{n}{k}}, t_1, \ldots, t_n, c_{t,1}, \ldots, c_{t,\frac{n}{k}}, \lambda, x, c_{d,1}, \ldots, c_{d,\frac{n}{k}}, c_1, \ldots, c_n, c, E', e, f_1, \ldots, f_n, z_1, \ldots, z_{\frac{n}{k}}, z, z')$.

Proof that the simulation works

Let us argue that this simulated proof is indistinguishable from a real proof. We define the following sequence of experiments:

Exp_1**:** We carry out a real proof with challenges $t_1, \ldots, t_n, \lambda, x, e$.

Exp_2**:** First we pick $d_1, \ldots, d_n, r_{d,1}, \ldots, r_{d,\frac{n}{k}}, r, r'$. Then we carry out a real proof using these values.

Exp_3**:** We pick $f_1, \ldots, f_n, z_1, \ldots, z_{\frac{n}{k}}, z, z'$. We then carry out a real proof, except when making the second answer. In that step the values $d_1, \ldots, d_n, r_{d,1}, \ldots, r_{d,\frac{n}{k}}, r, r'$ are fitted to the uniquely determined values that will make the entire proof acceptable.

Exp_4**:** We pick as in Exp_3 the elements $f_1, \ldots, f_n, z_1, \ldots, z_{\frac{n}{k}}, z, z'$ first. Afterwards we fit $c_{d,1}, \ldots, c_{d,\frac{n}{k}}, c, E'$ to the other commitments and ciphertexts, as we do in the simulation.

Exp_5**:** We carry out Exp_4 this time committing to zeros when making $c_{s,1}, \ldots, c_{s,\frac{n}{k}}, c_{t,1}, \ldots, c_{t,\frac{n}{k}}, c_1, \ldots, c_n$.

Exp_6**:** We make a simulated proof.

Exp_1 and Exp_2 are the same experiments where we have only changed the order in which we pick some elements. They are therefore perfectly indistinguishable.

Exp_2 and Exp_3 are perfectly indistinguishable since the elements we pick get the same distribution either way.

Exp_3 and Exp_4 are perfectly indistinguishable since we get the same $c_{d,1}, \ldots, c_{d,\frac{n}{k}}, c, E'$ no matter which method we use.

Exp_4 and Exp_5 are indistinguishable due to the hiding property of the commitment scheme. If the commitment scheme is statistically or perfectly hiding, the two experiments are statistically respectively perfectly indistinguishable.

Exp_5 and Exp_6 are the same experiments except for the order in which we pick some of the elements.

From this, we deduce that Exp_1 and Exp_6 are indistinguishable, in other words real proofs and simulated proofs are indistinguishable. Furthermore, when the commitment schemes are statistically or perfectly hiding, then the real proof and the simulated proof are statistically, respectively perfectly, indistinguishable. □

While this proof system is intended for proving correctness of a shuffle of encryptions, we can use a virtually identical proof if we wish to prove a shuffle of commitments $c_1, \ldots, c_n, c'_1, \ldots, c'_n$. When making the proof for commitments the root opening property of commitment schemes allows us to extract in polynomial time with high probability a witness for a shuffle consisting of a permutation π and openers $r_1, \ldots, r_n$ so that the commitments satisfy $c'_i = c_{\pi(i)} com_K(0; r_i)$ for $i = 1, \ldots, n$.

5 Speed, Space and Tricks

We start by mentioning a speedup. As $s_1, \ldots, s_n$ we may as well use the values $0, \ldots, n-1$. With this convention, the prover does not need to send $s_1, \ldots, s_n$ to the verifier. Furthermore, by choosing $s_1, \ldots, s_n$ as small as possible we may save some computation when making the multicommitments to $s_1, \ldots, s_n$. We use this speedup in the following estimate of the computational and communicational effort involved in making a proof of a shuffle.

We look at the special case of an ElGamal cryptosystem built over a 1024-bit prime p and a 160-bit prime q as in the introduction, and using the commitment scheme we have looked at before with these parameters as well as the suggested generalization to a multicommitment scheme. The prover makes $6n + 3\frac{n}{k} + 3$ exponentiations, the verifier makes $6n + 3\frac{n}{k} + 6$ exponentiations. The prover sends $1184n + 3232\frac{n}{k} + 3392$ bits during the proof. For optimal efficiency of the shuffle we select $k = n$. However, in cases where n is not known in advance, or where the public keys may need to be computed using a complicated multi-party computation protocol, we may wish to use a smaller k.

When comparing our scheme with that of [4] we have achieved significant improvements, in particular in the communication complexity. However, in many settings 1024-bit ElGamal encryption is not sufficiently strong, in particular not when long-term protection of data is needed. Here we may take advantage of the fact that we can use any commitment and multicommitment schemes, as long as the message spaces are compatible with the message space of the cryptosystem, and get further improvements. Let us say for instance that we have set up an ElGamal cryptosystem with a 3000-bit prime p and a 300-bit prime $q|p-1$. We may then use another 1500-bit prime p' for the commitment schemes instead of p

if $q|p'-1$. This speeds up the group operations considerably when computing the commitments, and because the proof system is statistical zero-knowledge it does not compromise the systems ability to hide the underlying permutation used in the shuffle.

An approach along the same lines that can be used quite generally is to use an integer commitment scheme as a virtual $\mathbf{Z}_N$ commitment scheme, where N is some appropriate integer in connection with the message space of the cryptosystem. This works by simply computing with the contents of the commitment schemes modulo N. However, some care must be taken not to accidentally leak information.

In the proof above, we picked the challenges $t_1, \ldots, t_n, \lambda, x, e$ from the message space $\mathcal{M}_K$. When the message space is small, such as $\mathbf{Z}_q$, where q is a 160-bit prime this is reasonable enough. However, when the message space is large we can make the protocol more efficient by picking the challenges from some suitable small subset of $\mathcal{M}_K$, say for instance $\{0, \ldots, 2^t - 1\}$ where t is a secondary security parameter of, say, 160 bits.

Finally, we mention that if we have several lists of ciphertexts that we need to shuffle according to the same permutation π, then we do not have to pay much extra for each additional π-shuffle. Almost all of the protocol can be reused.

References

[1] R. Cramer, I. Damgård and B. Schoenmakers, "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols", CRYPTO '94, LNCS series, volume 893: p. 174-187, 1994 146

[2] I. Damgård and E. Fujisaki, "An Integer Commitment Scheme based on Groups with Hidden Order", Cryptology ePrint Archive, Report 2001/064, 2001 148

[3] I. Damgård and M. J. Jurik, "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System", PKC 2001, LNCS series, volume 1992: p. 119-136, 2001 150, 151

[4] J. Furukawa and K. Sako, "An Efficient Scheme for Proving a Shuffle", CRYPTO '01, LNCS series, volume 2139: p. 368-387, 2001 145, 146, 159

[5] M. Jakobson, A. Juels and R. L. Rivest, "Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking", USENIX Security '02, 2002 146

[6] A. Neff, "A Verifiable Secret Shuffle and its Application to E-Voting", ACM CCS '01: p. 116-125, 2001 145, 146, 147, 151

[7] P. Paillier, "Public-key cryptosystems based on composite residuosity classes", EUROCRYPT '99, LNCS series, volume 1592: p. 223-239, 1999 150