A Verifiable Secret Shuffle of Homomorphic Encryptions

Jens Groth*
Department of Computer Science, UCLA
jg@cs.ucla.edu

July 27, 2005

Abstract

We suggest an honest verifier zero-knowledge argument for the correctness of a shuffle of homomorphic encryptions. A shuffle consists of a rearrangement of the input ciphertexts and a re-encryption of them. One application of shuffles is to build mix-nets.

Our scheme is more efficient than previous schemes in terms of both communication and computational complexity. Indeed, the HVZK argument has a size that is independent of the actual cryptosystem being used and will typically be smaller than the size of the shuffle itself. Moreover, our scheme is well suited for the use of multi-exponentiation techniques and batch-verification.

Additionally, we suggest a more efficient honest verifier zero-knowledge argument for a commitment containing a permutation of a set of publicly known messages. We also suggest an honest verifier zero-knowledge argument for the correctness of a combined shuffle-and-decrypt operation that can be used in connection with decrypting mix-nets based on ElGamal encryption.

All our honest verifier zero-knowledge arguments can be turned into honest verifier zero-knowledge proofs. We use homomorphic commitments as an essential part of our schemes. When the commitment scheme is statistically hiding we obtain statistical honest verifier zero-knowledge arguments, when the commitment scheme is statistically binding we obtain computational honest verifier zero-knowledge proofs.

Keywords: Shuffle, honest verifier zero-knowledge argument, homomorphic encryption, mix-net.

1 Introduction

Shuffle. A shuffle of ciphertexts $e_1, \dots, e_n$ is a new set of ciphertexts $E_1, \dots, E_n$ so that both sets of ciphertexts have the same plaintexts. If the cryptosystem is homomorphic we may shuffle $e_1, \dots, e_n$ by selecting a permutation $\pi \in \Sigma_n$ and setting $E_1 \leftarrow e_{\pi(1)}E(1), \dots, E_n \leftarrow e_{\pi(n)}E(1)$. If the cryptosystem is semantically secure, publishing $E_1, \dots, E_n$ reveals nothing about the permutation. On the other hand, this also means that nobody else can verify directly whether we shuffled correctly, substituted some ciphertexts, or performed some other malicious action. Our goal is to construct efficient honest verifier zero-knowledge (HVZK) arguments for the correctness of a shuffle.

Applications of HVZK shuffle arguments. Shuffling is the key building block in most mix-nets. A mix-net is a multi-party protocol run by a group of mix-servers to shuffle elements so that nobody knows the permutation linking the input and output. To mix ciphertexts we may let the mix-servers one after another make a shuffle with a random permutation and prove correctness of their shuffle. The arguments of correctness allow us to catch any cheater, and if at least one party is honest, it is impossible to link the input and

*Part of the work done while at BRICS, University of Aarhus and Cryptomathic.

output. In this role, shuffling constitutes an important building block in anonymization protocols and voting schemes.

Shuffle arguments have also found use as sub-protocols in more complex protocols or zero-knowledge arguments [KY04, Gro05b, Bra04].

Related work. Chaum invented mix-nets in [Cha81]. While being based on shuffling he did not suggest any method to guarantee correctness of the shuffles. Subsequent papers on mix-nets [BG02, PBDV04, JJR02, GJ04, JJ99, DK00, Jak98, OA00, Jak99, PIK93] have tried in many ways to guarantee correctness of a shuffle, most of which have been partially or fully broken [AI03, NSN03, Wik03, PP89]. Remaining are suggestions by [DK00, PBDV04, JJR02, Wik02], but they have various drawbacks. [DK00] require that only a small fraction of the mix-servers is corrupt. [PBDV04] require that a fraction of the senders producing the input to the mix-net is honest and restrict the class of possible permutations. [JJR02] allow mix-servers to compromise the privacy of a few senders and/or modify a few messages although they do run the risk of being caught. The mix-net in [Wik02] is less efficient than what one can build using the shuffle arguments in the present paper. Mix-nets based on shuffling and zero-knowledge arguments of correctness of a shuffle do not have these drawbacks.

Several papers have suggested zero-knowledge arguments for correctness of a shuffle, usually shuffling ElGamal ciphertexts. Sako and Kilian [SK95] use cut-and-choose methods and is thus not very efficient. Abe [Abe98](corrected in [AH01]) uses permutation networks and obtains reasonable efficiency. Currently there are two main paradigms that both yield practical HVZK arguments for correctness of a shuffle. Furukawa and Sako [FS01] suggest a paradigm based on permutation matrices. In this type of construction, you make a commitment to a permutation matrix, argue that you have committed to a permutation matrix and argue that the ciphertexts have been shuffled according to this permutation. It turns out that their protocol is not honest verifier zero-knowledge [FMM$^+$02], but it does hide the permutation [NSNK04]. Furukawa [Fur04a] develops the permutation matrix idea further and obtains a practical shuffle. [NSNK04, OT04] also use the permutation matrix idea of [FS01] to obtain HVZK arguments for correctness of a shuffle of Paillier ciphertexts [Pai99]. Following this paradigm we also have [FMM$^+$02, Fur04b] suggesting arguments for correctness of a combined shuffle-and-decrypt operation, an operation that is used in some decrypting mix-nets. The other paradigm is due to Neff [Nef01] and is based on polynomials being identical under permutation of their roots. A subsequent version [Nef03] corrects some flaws in [Nef01] and at the same time obtains higher efficiency. Unlike the Furukawa-Sako paradigm based arguments, Neff obtain an HVZK proof, i.e., soundness is unconditional but the zero-knowledge property is computational.

Our contributions. We suggest a 7-move HVZK argument for the correctness of a shuffle of homomorphic encryptions. We follow the Neff paradigm, basing the shuffle on invariance of polynomials under permutation of their roots. We use homomorphic commitments as a building block in our construction. If instantiated with a statistically hiding commitment we obtain a statistical HVZK *argument* for correctness of a shuffle. On the other hand, if instantiated with a statistically binding commitment scheme we obtain an HVZK *proof* of correctness of a shuffle.

The resulting HVZK argument is the most efficient HVZK argument for correctness of a shuffle that we know of both in terms of computation and communication. The scheme is well suited for multi-exponentiation techniques as well as randomized batch-verification giving us even higher efficiency. Unlike the permutation-matrix based approach it is also possible to work with a short public key, whereas key generation can be a significant cost in the permutation matrix paradigm. The only disadvantage of our scheme is the round-complexity. We use 7 rounds and the Furukawa-Sako paradigm can be used to obtain 3 round HVZK arguments for correctness of a shuffle.

Improving on the early version of the paper [Gro03] we enable shuffling of most known homomorphic cryptosystems. The size of the argument is almost independent of the cryptosystem that is being shuffled.

Furthermore, the commitment scheme we use does not have to be based on a group of the same order as the cryptosystem.

In Section 7, we give a more detailed comparison of our scheme and the other efficient HVZK arguments for correctness of a shuffle suggested in the literature.

As a building block, we use a shuffle of known contents and a corresponding argument of correctness of a shuffle of known contents. That is, given public messages $m_1, \ldots, m_n$, we can form a commitment to a permutation of these messages $c \leftarrow \mathrm{com}(m_{\pi(1)}, \ldots, m_{\pi(n)})$. We present an argument of knowledge for c containing a permutation of these messages. This has independent interest, for instance [Gro05b] uses an argument of correctness of a shuffle of known contents, it is not necessary to use a full-blown argument of correctness of a shuffle.

We also show how to modify our scheme into an HVZK argument of correctness of a shuffle-and-decrypt operation. This operation can be useful in decrypting mix-nets, it can save computational effort to combine the shuffle and decryption operations instead of performing each one of them by itself. [FMM$^+$02, Fur04b] already suggest arguments for the correctness of a shuffle-and-decrypt operation, however, while their arguments hide the permutation they are not HVZK. We obtain a more efficient argument that at the same time is HVZK.

2 Preliminaries

In this section, we define the three key concepts of this paper. We define homomorphic cryptosystems, since we will be shuffling homomorphic ciphertexts. We define homomorphic commitments, since they constitute an important building block in our schemes. Finally, we define honest verifier zero-knowledge (HVZK) arguments, since this paper is about HVZK arguments for the correctness of a shuffle. The reader already familiar with these concepts can go lightly over this section and return when needed.

2.1 Notation

All algorithms in protocols in this paper are envisioned as interactive probabilistic polynomial time uniform Turing machines. Adversaries are modeled as interactive probabilistic polynomial time non-uniform Turing machines. The different parties and algorithms get a security parameter as input, usually we omit writing this security parameter explicitly. For an algorithm A, we write $output \leftarrow A(input)$ for the process of selecting randomness r and making the assignment $output = A(input; r)$.

Recall that a function $\nu : \mathbb{N} \to [0;1]$ is negligible if for all monic polynomials $poly$ we have for all sufficiently large k that $\nu(k) < \frac{1}{poly(k)}$. For two functions f_1, f_2 we write $f_1 \approx f_2$ if $|f_1 - f_2|$ is negligible. We define security in terms of probabilities that become negligible as functions of the security parameter.

When referring to abelian groups, we will in this paper be thinking on "nice" groups, where membership can be decided efficiently, we can sample random elements from the groups, we can compute group operations, etc. In particular, we will make use of the group $\mathbb{Z}_q$, where we represent group elements as numbers in the interval $[0; q)$.

2.2 Special Honest Verifier Zero-Knowledge Arguments of Knowledge

Consider a pair of interactive algorithms (P, V) called the prover and the verifier. They may have access to a common reference string σ generated by a key generation algorithm K. We consider a polynomial time relation R, which may depend on σ. For an element x we call w a witness if $(\sigma, x, w) \in R$. We define a corresponding language L_σ consisting of elements that have a witness. We write view $\leftarrow < P(x), V(y) >$ for the public view produced by P and V when interacting on inputs x and y. This view ends with V either accepting or rejecting. We sometimes shorten the notation by saying $< P(x), V(y) >= b$ if V ends by accepting, $b = 1$, or rejecting, $b = 0$.

Definition 1 (Argument) *The triple (K, P, V) is called an argument for relation R if for all adversaries $\mathcal{A}$ we have*

Completeness:

$$\Pr\Big[\sigma \leftarrow K(); (x, w) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ or } < P(\sigma, x, w), V(\sigma, x) >= 1\Big] \approx 1.$$

Soundness:

$$\Pr\Big[\sigma \leftarrow K(); x \leftarrow \mathcal{A}(\sigma) : x \notin L_\sigma \text{ and } < \mathcal{A}, V(\sigma, x) >= 1\Big] \approx 0.$$

Consider a verifier that generates the challenges obliviously of the messages sent by P. We define special honest verifier zero-knowledge (SHVZK) [CDS94] as the ability to simulate the view with any set of challenges produced by V, but without access to the witness. We write $< P(\sigma, x, w), \text{challenges} >$ for the interactive protocol where the verifier simply forwards the specified challenges.

Definition 2 (Special honest verifier zero-knowledge) *The quadruple (K, P, V, S) is called a special honest verifier zero-knowledge argument for R if for all adversaries $\mathcal{A}$ we have*

$$\begin{aligned}
&\Pr\Big[\sigma \leftarrow K(); (x, w, \text{challenges}) \leftarrow \mathcal{A}(\sigma);\\
&\qquad \text{view} \leftarrow < P(\sigma, x, w), \text{challenges} >: \mathcal{A}(\text{view}) = 1\Big]\\
\approx\ &\Pr\Big[\sigma \leftarrow K(); (x, w, \text{challenges}) \leftarrow \mathcal{A}(\sigma);\\
&\qquad \text{view} \leftarrow S(\sigma, x, \text{challenges}) : \mathcal{A}(\text{view}) = 1\Big],
\end{aligned}$$

where we require that $\mathcal{A}$ does indeed produce (x, w) so $(\sigma, x, w) \in R$.

We call a scheme statistical SHVZK if the SHVZK property holds for unbounded adversaries.

Witness-extended emulation. The standard definition of a system for proof of knowledge does not work in our setting since we set up some public keys before making the argument of knowledge. A cheating prover may have non-zero probability of computing some trapdoor from the public keys and use that information in the argument. In this case, it may be impossible to extract a witness, but the standard definition calls for 100% probability of extracting the witness.

We shall define an argument of knowledge through witness-extended emulation, the name taken from [Lin01]. This definition says, given an adversary that produces an acceptable argument with probability ϵ, there exists an emulator that produces a similar argument with probability ϵ, but at the same time provides a witness.

Definition 3 (Witness-extended emulation) *We say the argument has witness-extended emulation if for all deterministic polynomial time P^* there exists an expected polynomial time emulator E such that for all adversaries $\mathcal{A}$ we have*

$$\begin{aligned}
&\Pr\Big[\sigma \leftarrow K(); (x, s) \leftarrow \mathcal{A}(\sigma); \text{view} \leftarrow < P^*(\sigma, x, s), V(\sigma, x) >:\\
&\qquad \mathcal{A}(\text{view}) = 1\Big]\\
\approx\ &\Pr\Big[\sigma \leftarrow K(); (x, s) \leftarrow \mathcal{A}(\sigma); (\text{view}, w) \leftarrow E(\sigma, x, s) :\\
&\qquad \mathcal{A}(\text{view}) = 1 \text{ and if view is accepting then } (\sigma, x, w) \in R\Big].
\end{aligned}$$

We think of s as being the state of P^*, including the randomness. Then we have an argument of knowledge in the sense that from this state s and the public data σ, x the emulator should be able to extract a witness whenever P^* is able to make a convincing argument. This shows that the definition implies soundness.

Damgård and Fujisaki [DF02] have also suggested a definition of argument of knowledge in the presence of a public key. Their definition is a black-box definition. [Gro04] shows that black-box witness-extended emulation implies knowledge soundness as defined by [DF02]. The security proofs in this paper obtain black-box witness-extended emulation so our protocols have knowledge soundness as defined in [DF02].

SHVZK proofs. Sometimes unconditional soundness may be needed, i.e., soundness should hold even if the adversary is allowed to be unbounded. We call such a scheme a proof instead of an argument. We will construct both SHVZK arguments and SHVZK proofs in the paper.

2.3 Homomorphic commitment

We use a key generation algorithm to generate a public key pk. The public key specifies a message space $\mathcal{M}$, a randomizer space $\mathcal{R}$ and a commitment space $\mathcal{C}$ as well as an efficiently computable commitment function $\mathrm{com} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{C}$. There is also a probability distribution on $\mathcal{R}$ and we write $c \leftarrow \mathrm{com}(m)$ for the operation $r \leftarrow \mathcal{R}; c = \mathrm{com}(m; r)$.

We say the commitment scheme is hiding if a commitment does not reveal which message is inside. We define this by demanding that for all adversaries $\mathcal{A}$ we have

$$\begin{aligned} & \Pr\Big[pk \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}(pk); c \leftarrow \mathrm{com}(m_0) : m_0, m_1 \in \mathcal{M} \text{ and } \mathcal{A}(c) = 1\Big] \\ \approx\ & \Pr\Big[pk \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}(pk); c \leftarrow \mathrm{com}(m_1) : m_0, m_1 \in \mathcal{M} \text{ and } \mathcal{A}(c) = 1\Big]. \end{aligned}$$

If this also holds for unbounded $\mathcal{A}$, we call the commitment statistically hiding.

We say the commitment scheme is binding if it is impossible to change your mind about the content of a commitment once it is made. We specify this as the infeasibility to open a commitment to two different messages. For all adversaries $\mathcal{A}$ we have

$$\begin{aligned} \Pr\Big[&pk \leftarrow K(); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(pk) : \\ &(m_0, r_0), (m_1, r_1) \in \mathcal{M} \times \mathcal{R}, m_0 \neq m_1 \text{ and } \mathrm{com}(m_0, r_0) = \mathrm{com}(m_1; r_1)\Big] \approx 0. \end{aligned}$$

If this also holds for unbounded $\mathcal{A}$, we call the commitment statistically binding.

We are interested in commitment schemes where the message, randomizer and commitment spaces are abelian groups $(\mathcal{M}, +, 0), (\mathcal{R}, +, 0), (\mathcal{C}, \cdot, 1)$. With overwhelming probability over the choice of the public key, the commitment function must be homomorphic

$$\forall (m_0, r_0), (m_1, r_1) \in \mathcal{M} \times \mathcal{R} : \mathrm{com}(m_0 + m_1; r_0 + r_1) = \mathrm{com}(m_0; r_0)\mathrm{com}(m_1; r_1).$$

For our purposes, we use a homomorphic commitment scheme with message space $\mathbb{Z}_q^n$, where q is a prime. Other choices are possible, for instance letting q be a composite or using message space $\mathbb{Z}^n$. The reason we choose q to be prime is that it simplifies the presentation slightly and is the most realistic choice in practice. In particular, with q being prime we know that any non-trivial n-degree polynomial $P(T) \in \mathbb{Z}_q[T]$ has at most n roots, which will be useful later on.

We need a root extraction property, which says it is infeasible to create an opening of a commitment raised to a non-trivial exponent without being able to open the commitment itself. I.e., there is a root extraction algorithm $\mathrm{RootExt}$ so for all adversaries $\mathcal{A}$ we have

$$\Pr\Big[pk \leftarrow K(); (M,R,c,e) \leftarrow \mathcal{A}(pk); (m,r) \leftarrow \mathrm{RootExt}(M,R,c,e) : \\ \text{if}(M,R) \in \mathcal{M} \times \mathcal{R}, c \in \mathcal{C}, \gcd(e,q)=1 \\ \text{and } c^e = \mathrm{com}(M;R) \text{ then } (m,r) \in \mathcal{M} \times \mathcal{R}, c = \mathrm{com}(m;r)\Big] \approx 1,$$

As an example of an unconditionally hiding commitment scheme with these properties, we offer the following variation of the Pedersen commitment. We select primes q,p so $p = kq+1$ and k,q are coprime. The public key is $(q,p,g_1,\ldots,g_n,h)$, where $g_1,\ldots,g_n,h$ are randomly chosen elements of order q. Let $\mathbb{G}_k$ be the multiplicative group of elements with order dividing k. We have $\mathcal{M} = \mathbb{Z}_q^n, \mathcal{R} = \mathbb{G}_k \times \mathbb{Z}_q, \mathcal{C} = \mathbb{Z}_p^*$. To commit to $m_1,\ldots,m_n$ using randomness $(u,r) \in \mathbb{G}_k \times \mathbb{Z}_q$ we compute $c = ug_1^{m_1}\cdots g_n^{m_n}h^r \bmod p$. For the hiding property to hold we can always choose $u=1$ and simply pick $r \leftarrow \mathbb{Z}_q$ at random. This commitment scheme is homomorphic and has the root extraction property. Our little twist, adding the u-factor, of the Pedersen commitment scheme makes it extremely efficient to test membership of $\mathcal{C}$, we just have to verify $0 < c < p$.

As an example of an unconditionally binding commitment scheme consider selecting the public key $(q,p,g_1,\ldots,g_n,h)$ as above. The message space is $\mathcal{M} = \mathbb{Z}_q^n$, the randomizer space is $\mathbb{G}_k^{n+1} \times \mathbb{Z}$, and the commitment space is $\mathcal{C} = (\mathbb{Z}_p^*)^n$. We commit to $m_1,\ldots,m_n$ using randomizer $(u_1,\ldots,u_n,u,r)$ as $c = (u_1g_1^{r+m_1},\ldots,u_ng_n^{r+m_n},uh^r)$. We can simply use $u_1 = \cdots = u_n = u = 1$ when making the commitments, the hiding property follows from the DDH assumption.

2.4 Homomorphic cryptosystem

We use a key generation algorithm to generate a public key and a secret key. The public key specifies a message space $\mathcal{M}$, a randomizer space $\mathcal{R}$ and a ciphertext space $\mathcal{C}$. It also specifies an encryption algorithm $E : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$. The secret key specifies a decryption algorithm $D : \mathcal{C} \to \mathcal{M} \cup \{\texttt{invalid}\}$.

We require that with overwhelming probability the key generation algorithm specify keys such that decryption of an encrypted message always yields the message, i.e.,

$$\Pr\Big[(pk,sk) \leftarrow K() : \forall (m,r) \in \mathcal{M} \times \mathcal{R} : D(E(m;r)) = m\Big] \approx 1.$$

We require the message, randomizer and ciphertext spaces to be finite abelian groups $(\mathcal{M},\cdot,1), (\mathcal{R},+,0)$ and $(\mathcal{C},\cdot,1)$. The encryption function must be homomorphic with overwhelming probability over the public key

$$\forall (m_0,r_0),(m_1,r_2) \in \mathcal{M} \times \mathcal{R} : E(m_0m_1;r_0+r_1) = E(m_0;r_0)E(m_1;r_1).$$

In this paper, we demand that the order of the message space is divisible only by large prime-factors.

We also require any non-trivial root of an encryption of 1 has the same plaintext.

$$\forall R \in \mathcal{R} \forall E \in \mathcal{C} \forall e \text{ so } \gcd(e,|\mathcal{M}|) = 1 \text{ and } E^e = E(1;R) \exists r \in \mathcal{R} \text{ so } E = E(1;r).$$

This can be seen as a relaxed version of the root extraction property for homomorphic commitments, we know the message is 1, however, we may not be able to extract the randomness r so $E = E(1;r)$. Nonetheless, several cryptosystems in the literature do allow us to extract the randomness, in particular, Paillier encryption and ElGamal encryption allow full randomness extraction.

Various variants of ElGamal and Paillier encryption as well as other cryptosystems [Pai99, DJ01, DJ03, OU98, Gro05a, ElG84, CS98, NBD01] have the properties mentioned in this section or can be tweaked into cryptosystems with these properties.

2.5 Setup and parameters.

The common reference string will consist of public keys for the homomorphic cryptosystem and the homomorphic commitment scheme. We assume all parties, prover, verifier and adversary, know this common reference string.

The verifier will select public coin challenges from $\{0,1\}^{\ell_e}$. ℓ_e will be a sufficiently large length to make the risk of breaking soundness become negligible. In practice a choice of $\ell_e = 80$ suffices for interactive protocols. If we make the HVZK argument non-interactive using a hash-function, $\ell_e = 160$ may be sufficient. Another security parameter is ℓ_s. Here we require that for any a of length ℓ_a, we have that d and $a + d$ are indistinguishable, when d is chosen at random from $\{0,1\}^{\ell_a+\ell_s}$. In practice $\ell_s = 80$ will be fine.

We set up the commitment scheme with message space $\mathbb{Z}_q^n$. We demand that $|q| > \ell_e + \ell_s$. The reason for this choice is to make q large enough to avoid overflows that require a modular reduction in Section 4 and 5. When the cryptosystem has a message space where $m^q = 1$ for all messages, this requirement can be waived, see Section 6 for details. For notational convenience, we assume that the randomizer space of the commitment scheme is $\mathbb{Z}_q$, but other choices are possible.

3 HVZK Argument for Shuffle of Known Contents

Before looking into the question of shuffling ciphertexts, we investigate a simpler problem that will be used as a building block. We have messages $m_1, \ldots, m_n$. It is easy enough to pick a permutation π and a randomizer r and set $c = \mathrm{com}(m_{\pi(1)}, \ldots, m_{\pi(n)}; r)$. Can we prove knowledge of the permutation π and the randomizer r such that indeed c has been computed this way?

In this section, we present an SHVZK argument for a commitment containing a permutation of a set of known messages. The main idea is from Neff [Nef01], namely that a polynomial $p(X) = \prod_{i=1}^n (m_i - X)$ is stable under permutation of the roots, i.e., for any permutation π we have $p(X) = \prod_{i=1}^n (m_{\pi(i)} - X)$. A way to test whether two polynomials $p(X), q(X)$ are identical is to choose a random point x and evaluate whether $p(x) = q(x)$. Vice versa, if two polynomials are identical over a field $\mathbb{Z}_q$ then they have the same roots.

Using this idea, we formulate the following plan for arguing knowledge of c containing messages $m_1, \ldots, m_n$.

1. Use a standard HVZK argument with randomly chosen challenge e to argue knowledge of an opening $\mu_1, \ldots, \mu_n, r$ of c. As a byproduct of the argument of knowledge we get values $f_i = e\mu_i + d_i$, where d_i is fixed before receiving the random e.

2. Choose an evaluation point x at random. It is straightforward to compute $f_i - ex = e(\mu_i - x) + d_i$.

3. We have $\prod_{i=1}^n (f_i - ex) = e^n \prod_{i=1}^n (\mu_i - x) + p_{n-1}(e)$, where $p_{n-1}(\cdot)$ is a polynomial of degree $n-1$. We therefore wish to argue $\prod_{i=1}^n (f_i - ex) = e^n \prod_{i=1}^n (m_i - x) + p_{n-1}(e)$, which would mean $\prod_{i=1}^n (\mu_i - x) = \prod_{i=1}^n (m_i - x)$.

4. To argue the latter we roll up the partial product in $F_j = e \prod_{i=1}^j (\mu_i - x) + \Delta_j$. We start with $F_1 = f_1 - ex$. We then compute $eF_2 = F_1(f_2 - ex) + f_{\Delta_1}$, where f_{Δ_1} is used to remove superfluous factors. We compute $F_3, \ldots, F_n$ in the same manner. We use $\Delta_n = 0$, so in the end it is sufficient to test whether $F_n = e \prod_{i=1}^n (m_i - x)$.

Theorem 4 *The protocol in Figure 1 is a 4-move public coin special honest verifier zero-knowledge argument of knowledge for c being a commitment to a permutation of the messages $m_1, \ldots, m_n$. If the commitment*

Shuffle of Known Content Argument

Prover	Common input	Verifier
	$c, m_1, \ldots, m_n$ and public keys	
Prover's input π, r so $c = \mathrm{com}(m_{\pi(1)}, \ldots, m_{\pi(n)}; r)$		
	$\xleftarrow{\quad x \quad}$	$x \leftarrow \{0,1\}^{\ell_e}$
$d_1, \ldots, d_n \leftarrow \mathbb{Z}_q, r_d, r_\Delta \leftarrow \mathbb{Z}_q$ $\Delta_1 = d_1, \Delta_2, \ldots, \Delta_{n-1} \leftarrow \mathbb{Z}_q, \Delta_n = 0$ $a_i = \prod_{j=1}^{i}(m_{\pi(j)} - x), r_a \leftarrow \mathbb{Z}_q$ $c_d = \mathrm{com}(d_1, \ldots, d_n; r_d)$ $c_\Delta = \mathrm{com}(-\Delta_1 d_2, \ldots, -\Delta_{n-1} d_n; r_\Delta)$ $c_a = \mathrm{com}(\Delta_2 - (m_{\pi(2)} - x)\Delta_1 - a_1 d_2, \ldots,$ $\Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} d_n - a_{n-1} d_n; r_a)$	$\xrightarrow{\quad c_d, c_\Delta, c_a \quad}$	
	$\xleftarrow{\quad e \quad}$	$e \leftarrow \{0,1\}^{\ell_e}$
$f_i = e m_{\pi(i)} + d_i, z = er + r_d$ $f_{\Delta_i} = e(\Delta_{i+1} - (m_{\pi(i+1)} - x)\Delta_i - a_i d_{i+1})$ $-\Delta_i d_{i+1}, z_\Delta = e r_a + r_\Delta$	$f_1, \ldots, f_n, z$ $f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta$ $\longrightarrow$	
	Check $c_d, c_a, c_\Delta \in \mathcal{C}$ Check $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta \in \mathbb{Z}_q$ Check $c^e c_d = \mathrm{com}(f_1, \ldots, f_n; z)$ Check $c_a^e c_\Delta = \mathrm{com}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)$ Define $F_1, \ldots, F_n$ so $F_1 = f_1 - ex, eF_2 = F_1(f_2 - ex) + f_{\Delta_1}, \ldots,$ $eF_n = F_{n-1}(f_n - ex) + f_{\Delta_{n-1}}$ Check $F_n = e\prod_{i=1}^{n}(m_i - x)$	

Figure 1: Argument of Knowledge of Shuffle of Known Content.

scheme is statistically hiding then the argument is statistical honest verifier zero-knowledge. If the commitment scheme is statistically binding, then we have unconditional soundness, i.e., the protocol is a SHVZK proof.

Proof. It is obvious that we are dealing with a 4-move public coin protocol. Completeness is straightforward to verify. Remaining is to prove special honest verifier zero-knowledge and witness-extended emulation.

Special honest verifier zero-knowledge. Figure 2 describes how the simulator acts given challenges x, e. The simulator does not use any knowledge of π, r. It first selects $f_1, \ldots, f_n, z, F_2, \ldots, F_{n-1}, z_\Delta$ and $c_a \leftarrow \mathrm{com}(0, \ldots, 0)$ and then adjusts all other parts of the argument to fit these values. In the same figure, we describe a hybrid simulator that acts just as the simulator except when generating c_a. In the generation of c_a, it does use knowledge of π to compute d_i, a_i, Δ_i values. It then produces c_a in the same manner as a real prover would do it using those values. Finally, for comparison we have the real prover's protocol in an unordered fashion.

Simulator	Hybrid	Prover
$f_i \leftarrow \mathbb{Z}_q$, $z \leftarrow \mathbb{Z}_q$		$f_i = em_{\pi(i)} + d_i$, $z = er + r_d$
$F_i \leftarrow \mathbb{Z}_q$, $z_\Delta \leftarrow \mathbb{Z}_q$		$F_i = ea_i + \Delta_i$, $z_\Delta = er_a + r_\Delta$
$F_1 = f_1 - ex$, $F_n = e\prod_{i=1}^n (m_i - x)$		
$f_{\Delta_i} = eF_{i+1} - F_i(f_{i+1} - ex)$		
	$d_i = f_i - em_{\pi(i)}$	$d_i \leftarrow \mathbb{Z}_q$, $r_d \leftarrow \mathbb{Z}_q$
	$a_i = \prod_{j=1}^i (m_{\pi(j)} - x)$,$r_a \leftarrow \mathbb{Z}_q$	
	$\Delta_i = F_i - ea_i$	$\Delta_i \leftarrow \mathbb{Z}_q$, $r_\Delta \leftarrow \mathbb{Z}_q$
$c_a \leftarrow \mathrm{com}(0, \ldots, 0)$	$c_a \leftarrow \mathrm{com}(\Delta_2 - (m_{\pi(2)} - x)\Delta_1 - a_1 d_2,$	
	$\ldots, \Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - a_{n-1}d_n; r_a)$	
$c_d = \mathrm{com}(f_1, \ldots, f_n; z)c^{-e}$		
$c_\Delta = \mathrm{com}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)c_a^{-e}$		

Figure 2: Simulation of Known Shuffle Argument.

The simulated argument and the hybrid argument differ only in the content of c_a. The hiding property of the commitment scheme therefore gives us indistinguishability between hybrid arguments and simulated arguments. If the commitment scheme is statistically hiding then the arguments are statistically indistinguishable.

A hybrid argument is statistically indistinguishable from a real argument. The only difference is that a real prover starts out by picking $d_i, \Delta_i, r_d, r_\Delta$ at random, however, in both protocols we get $f_i, f_{\Delta_i}, z, z_\Delta$ randomly distributed over $\mathbb{Z}_q$. One can now check that indeed we compute c_d, c_Δ so they are on the form $c_d = \mathrm{com}(d_1, \ldots, d_n; r_d)$ and $c_\Delta = \mathrm{com}(-\Delta_1 d_2, \ldots, -\Delta_{n-1}d_n; r_\Delta)$ when running the prover protocol described in Figure 2.

Witness-extended emulation. The emulator E first runs P^* with the algorithm of the real verifier. This is the view that E outputs and by construction it is perfectly indistinguishable from a real SHVZK argument. If the view is rejecting, then E halts with $w = \texttt{no witness}$. However, if the view is accepting then E must try to find a witness $w = (\pi, r)$.

To extract a witness E rewinds and runs P^* on the same challenge x until it gets another acceptable argument. Call the two arguments $(x, c_d, c_\Delta, c_a, e, f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta)$ and $(x, c_d, c_\Delta, c_a, e', f'_1, \ldots, f'_n, z', f'_{\Delta_1}, \ldots, f'_{\Delta_{n-1}}, z'_\Delta)$. We have $c^e c_d = \mathrm{com}(f_1, \ldots, f_n; z)$ and $c^{e'} c_d = \mathrm{com}(f'_1, \ldots, f'_n; z')$. This gives us $c^{e-e'} = \mathrm{com}(f_1 - f'_1, \ldots, f_n - f'_n; z - z')$. If $e \neq e'$, E can run the root extraction algorithm in an attempt to learn an opening $\mu_1, \ldots, \mu_n, r$ of c. From such an opening we can also find an opening $d_1, \ldots, d_n, r_d$ of c_d with $d_i = f_i - e\mu_i, r_d = z - er$.

Let us at this point argue that E runs in expected polynomial time. If P^* is in a situation where it has probability $\epsilon > 0$ of making the verifier accept on challenge x, then the expected number of runs to get an acceptable view is $\frac{1}{\epsilon}$. Of course if P^* fails, then we do not need to sample a second run. We therefore get a total expectation of 2 runs of P^*. A consequence of E using an expected polynomial number of queries to P^* is that it only has negligible probability of ending in a run where $e' = e$ or any other event with negligible probability occurs. Therefore, with overwhelming probability, we do not need a witness or we have found an opening $\mu_1, \ldots, \mu_n, r$ of c.

We still need to argue that the probability for extracting an opening of c, yet $\mu_1, \ldots, \mu_n$ not being a permutation of $m_1, \ldots, m_n$, is negligible. Assume there is a polynomial $poly$ such that P^* has more than $\frac{1}{poly()}$ chance of producing a convincing argument. In that case we can pick a challenge x at random, and thereafter pick three random challenges e, e', e''.

With probability at least $\frac{1}{poly()^3}$ P^* manages to create accepting arguments on all three of these challenges. Call the first two arguments $(x, c_d, c_\Delta, c_a, e, f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta)$ and $(x, c_d, c_\Delta, c_a, e', f'_1, \ldots, f'_n, z', f'_{\Delta_1}, \ldots, f'_{\Delta_{n-1}}, z'_\Delta)$. We have $c_a^e c_\Delta = \mathrm{com}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)$ and $c_a^{e'} c_\Delta = \mathrm{com}(f'_{\Delta_1}, \ldots, f'_{\Delta_{n-1}}; z'_\Delta)$ so $c_a^{e-e'} = \mathrm{com}(f_{\Delta_1} - f'_{\Delta_1}, \ldots, f_{\Delta_{n-1}} - f'_{\Delta_{n-1}}; z_\Delta - z'_\Delta)$. From this, we can extract an opening $\alpha_1, \ldots, \alpha_{n-1}, r_a$ of c_a. This also gives us an opening $\delta_1, \ldots, \delta_{n-1}, r_\Delta$ of c_Δ, where $\delta_i = f_{\Delta_i} - e\alpha_i, r_\Delta = z_\Delta - er_a$.

Consider now the third challenge e''. Since we know openings of c, c_d we have $f''_i = e''\mu_i + d_i$, and since we know openings of c_a, c_Δ we have $f''_{\Delta_i} = e''\alpha_i + \delta_i$. From the way we build up F''_n and from $F''_n = e'' \prod_{i=1}^n (m_i - x)$ we deduce

$$(e'')^n \prod_{i=1}^n (m_i - x) = (e'')^{n-1} F''_n = (e'')^n \prod_{i=1}^n (\mu_i - x) + p_{n-1}(e''),$$

where $p_{n-1}(\cdot)$ is a polynomial of degree $n-1$. Since e'' is chosen at random this implies with overwhelming probability that $\prod_{i=1}^n (\mu_i - x) = \prod_{i=1}^n (m_i - x)$.

We now have two polynomials evaluating to the same value in a random point x. With overwhelming probability, they must be identical. This in turn implies that $\mu_1, \ldots, \mu_n$ is a permutation of $m_1, \ldots, m_n$ as we wanted to show.

If the commitment scheme is statistically binding, then even an unbounded adversary is stuck with the values that have been committed to, without any ability to change them. With x, e chosen at random by the verifier, even an unbounded adversary has negligible chance of cheating. □

4 HVZK Argument for Shuffle of Homomorphic Encryptions

A set of ciphertexts $e_1, \ldots, e_n$ can be shuffled by selecting a permutation π, selecting randomizers $R_1, \ldots, R_n$, and setting $E_1 = e_{\pi(1)} E(1; R_1), \ldots, E_n = e_{\pi(n)} E(1; R_n)$. The task for the prover is now to argue 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.

An obvious problem with this idea is the lack of zero-knowledge. We remedy it in the following way:

1. The prover commits to the permutation π as $c \leftarrow \mathrm{com}(\pi(1), \ldots, \pi(n))$. He makes an HVZK argument of knowledge of c containing a permutation of the numbers $1, \ldots, n$. At this step, the prover is bound to some permutation he knows, but the permutation remains hidden.

2. The prover creates a commitment $c_d \leftarrow \mathrm{com}(d_1, \ldots, d_n)$ to random d_i's. The verifier selects at random $t_1, \ldots, t_n$ and the prover permutes them according to π. The prover will at some point reveal values $f_i = t_{\pi(i)} + d_i$, but since the d_i's are random this does not reveal the permutation π. As part of the argument, we will argue that the f_i's have been formed correctly, using the same permutation π that we used to form c.

3. Finally, the prover uses standard HVZK arguments of knowledge of multiplicative relationship and equivalence to show that the products $e_1^{t_1} \cdots e_n^{t_n}$ and $E_1^{t_{\pi(1)}} \cdots E_n^{t_{\pi(n)}}$ differ only by an encryption of 1 without revealing anything else. This last step corresponds to carrying out the naïve proof system in zero-knowledge using a secret permutation π that was fixed before receiving the t_i's.

To carry out this process we need to convince the verifier that c and $f_1, \ldots, f_n$ contain respectively $1, \ldots, n$ and $t_1, \ldots, t_n$ permuted in the same order. 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. The HVZK argument of knowledge for a shuffle of known content can therefore be used.

To see that the pairs (i, t_i) match we let the verifier pick λ at random, and let the prover demonstrate that $c^{\lambda} c_d \mathrm{com}(f_1, \ldots, f_n; 0)$ contains a shuffle of $1+\lambda t_1, \ldots, n+\lambda t_n$. If a pair (i, t_i) does not appear in the same spot in respectively c and $f_1, \ldots, f_n$, then with high likelihood over the choice of λ the shuffle argument will fail.

Shuffle of Homomorphic Ciphertexts

Prover	Common input $e_1, \ldots, e_n, E_1, \ldots, E_n$ and public keys	Verifier
Prover's input $\pi, R_1, \ldots, R_n$ so $E_i = e_{\pi(i)} E(1; R_i)$		
$r \leftarrow \mathbb{Z}_q$, $R_d \leftarrow \mathcal{R}$ $d_1, \ldots, d_n \leftarrow \mathbb{Z}_q, r_d \leftarrow \mathbb{Z}_q$ $c = \mathrm{com}(\pi(1), \ldots, \pi(n); r)$ $c_d = \mathrm{com}(-d_1, \ldots, -d_n; r_d)$ $E_d = \prod_{i=1}^n E_i^{-d_i} E(1; R_d)$	$\xrightarrow{c, c_d, E_d}$	
	$\xleftarrow{t_1, \ldots, t_n\,[a]}$	$t_i \leftarrow \{0,1\}^{\ell_e}$
$f_i = t_{\pi(i)} + d_i$ $Z = \sum_{i=1}^n t_{\pi(i)} R_i + R_d$	$\xrightarrow{f_1, \ldots, f_n, Z}$	
	$\xleftarrow{\lambda}$	$\lambda \leftarrow \{0,1\}^{\ell_e}$
$\mathrm{Arg}(\pi, \rho \mid c^{\lambda} c_d \mathrm{com}(f_1, \ldots, f_n; 0)$ $= \mathrm{com}(\lambda\pi(1) + t_{\pi(1)},$ $\ldots, \lambda\pi(n) + t_{\pi(n)}; \rho))$	$\longleftarrow$ $\longrightarrow$ $\longrightarrow$ $\longrightarrow$	
		Check $c, c_d \in \mathcal{C}_{com}, E_d \in \mathcal{C}$ and $2^{\ell_e} \leq f_1, \ldots, f_n < q, Z \in \mathcal{R}$ Verify $\mathrm{Arg}(\pi, \rho)$ Check $\prod_{i=1}^n e_i^{-t_i} \prod_{i=1}^n E_i^{f_i} E_d = E(1; Z)$

[a] See Section 6 for ways to choose $t_1, \ldots, t_n$ efficiently.

Figure 3: Argument of Shuffle of Homomorphic Ciphertexts.

Theorem 5 *The protocol in Figure 3 is a 7-move public coin special honest verifier zero-knowledge argument of knowledge for correctness of a shuffle of homomorphic ciphertexts. If the commitment scheme is statistically hiding and the argument of knowledge of a shuffle of known content is statistical SHVZK, then the entire argument is statistical SHVZK. If the commitment scheme is statistically binding and we use a SHVZK proof of shuffle of known contents, then the entire scheme is a SHVZK proof of a shuffle.*

Proof. Using the 4-move argument of knowledge for shuffle of known contents from this paper the protocol is a 7-move public coin protocol. Recall that $|q| > 2^{\ell_e} + \ell_s$, so with overwhelming probability $2^{\ell_e} \leq t_{\pi(i)} + d_i < q$ when added as integers. With this in mind, it is straightforward to verify completeness. It remains to prove zero-knowledge and witness-extended emulation.

Special honest verifier zero-knowledge. Given challenges $t_1, \ldots, t_n, \lambda$ as well as challenges for the known shuffle we wish to produce something that is indistinguishable from a real argument. We describe in Figure 4 a simulator that simulates the argument without access to the permutation π or the randomizers $R_1, \ldots, R_n$. In the same figure, we also include a hybrid argument that works like the simulator except for generating some commitments correctly. Finally, we include for comparison the real prover in a somewhat unordered description.

Simulator	Hybrid	Prover
$c \leftarrow \mathrm{com}(0, \ldots, 0)$	$c \leftarrow \mathrm{com}(\pi(1), \ldots, \pi(n))$	
	$d_i = f_i - t_{\pi(i)}$	$d_i \leftarrow \mathbb{Z}_q$
$c_d \leftarrow \mathrm{com}(0, \ldots, 0)$	$c_d \leftarrow \mathrm{com}(-d_1, \ldots, -d_n)$	
$f_i \leftarrow \mathbb{Z}_q$		$f_i = t_{\pi(i)} + d_i$
$Z \leftarrow \mathcal{R}$		$R_d \leftarrow \mathcal{R}, Z = \sum_{i=1}^n t_{\pi(i)} R_i + R_d$
$E_d = E(1; Z) \prod_{i=1}^n e_i^{t_i} \prod_{i=1}^n E_i^{-f_i}$		$E_d = \prod_{i=1}^n E_i^{-d_i} E(1; R_d)$
Simulate $\mathrm{Arg}(\pi, \rho \mid$		$\mathrm{Arg}(\pi, \rho \mid$
$c^\lambda c_d \mathrm{com}(f_1, \ldots, f_n; 0)$		$c^\lambda c_d \mathrm{com}(f_1, \ldots, f_n; 0)$
$= \mathrm{com}(\lambda\pi(1) + t_{\pi(1)},$		$= \mathrm{com}(\lambda\pi(1) + t_{\pi(1)},$
$\ldots, \lambda\pi(n) + t_{\pi(n)}; \rho)$		$\ldots, \lambda\pi(n) + t_{\pi(n)}; \rho)$

Figure 4: Simulation of Shuffle Argument.

Simulated arguments and hybrid arguments only differ in the content of c and c_d. The hiding property of the commitment scheme therefore implies indistinguishability between simulated arguments and hybrid arguments, and if the commitment scheme is statistically hiding, then the two types of arguments are statistically indistinguishable.

Since $|q| > \ell_e + \ell_s$ there is overwhelming probability that we do not need to make any modular reductions when computing the d_i's and f_i's and that the f_i's are at least 2^{ℓ_e}. Under this condition, we have for the prover that $\prod_{i=1}^n E_i^{-d_i} E(1; R_d) = E(1; Z) \prod_{i=1}^n e_i^{t_i} \prod_{i=1}^n E_i^{-f_i}$, so there is no difference in the way E_d is computed. The only remaining difference is that the hybrid argument contains a simulated argument of knowledge of shuffle of known content, whereas the prover makes a real proof. The SHVZK property of this argument gives us indistinguishability between hybrid arguments and real arguments, and statistical SHVZK gives us statistical indistinguishability.

Witness-extended emulation. We first run P^* on randomly chosen challenges $t_1, \ldots, t_n, \lambda$ and the HVZK argument of known shuffle. This gives us a correctly formed view. If P^* fails to produce an acceptable argument, then we output (view, `no witness`). On the other hand, if the argument is acceptable, then we must attempt to extract a witness for $E_1, \ldots, E_n$ being a shuffle of $e_1, \ldots, e_n$. In the following we let ϵ be the probability of P^* outputting an acceptable argument.

In order to extract a witness we run P^* on randomly chosen challenges $t_1, \ldots, t_n, \lambda$ and use the witness-extended emulator for the argument of shuffle of known contents. We do this until we have obtained $n+3$ acceptable arguments. Recall from the proof of Theorem 4 that the witness-extended emulator for the shuffle of known contents has exactly the same distribution on the public view as that produced in a real argument with P^*. Therefore, each attempted run has probability ϵ of leading to an acceptable argument and we use an expected $(n+3)/\epsilon$ runs. Since we only need to extract a witness when P^* produced a valid argument we get an expected number of $n+3$ runs. In each run, the witness-extended emulator for the argument of shuffle of known contents is fed with polynomial size input and all runs have the same size of input. This means we can simply sum the expected polynomial time run times. The witness-extended emulator of the argument of a shuffle of homomorphic ciphertexts therefore uses expected polynomial time.

Since the witness-extended emulator uses expected polynomial time there is overwhelming probability that either we do not get an acceptable argument, or alternatively we do get an acceptable argument but no event with negligible probability occurs. In particular, with overwhelming probability we do not break the commitment scheme, unsuccessfully run root extractors, etc.

From the sampling process we have two acceptable arguments $c, c_d, E_d, t_1, \ldots, t_n, f_1, \ldots, f_n, Z, \lambda$ and $c, c_d, E_d, t'_1, \ldots, t'_n, f'_1, \ldots, f'_n, Z', \lambda'$ as well as witnesses π, r and π', r' for $c^\lambda c_d \mathrm{com}(f_1, \ldots, f_n; 0)$ and $c^{\lambda'} c_d \mathrm{com}(f'_1, \ldots, f'_n; 0)$ containing shuffles of $\lambda i + t_i$ and $\lambda' i + t'_i$ respectively. This gives us

$$c^{\lambda-\lambda'} = \mathrm{com}(f'_1 - f_1 + \lambda\pi(1) + t_{\pi(1)} - \lambda'\pi'(1) - t'_{\pi'(1)}, \ldots, f'_n - f_n + \lambda\pi(n) + t_{\pi(n)} - \lambda'\pi'(n) - t'_{\pi'(n)}; r - r').$$

We run the root extractor and find an opening $s_1, \ldots, s_n, r$ of c. Given this opening we can then compute an opening $-d_1, \ldots, -d_n, r_d$ of c_d with $-d_i = \lambda\pi(i) + t_{\pi(i)} - \lambda s_i - f_i$ and $0 \leq d_i < q$.

Next, we wish to argue that $s_1, \ldots, s_n$ is a permutation of $1, \ldots, n$, i.e., they define a unique permutation π. Suppose for some polynomial $poly()$ in the security parameter that P^* has more than $\frac{1}{poly()}$ chance of producing a valid argument. We run P^* with randomly chosen challenges $t_1, \ldots, t_n, \lambda$ and from the witness-extended emulator we get a permutation π so $\lambda s_i - d_i + f_i = \lambda\pi(i) + t_{\pi(i)}$. Since f_i is chosen before λ this has negligible chance of happening unless $s_i = \pi(i)$. We conclude that indeed $s_1, \ldots, s_n$ is a permutation of $1, \ldots, n$. This in turn tells us that $f_i = t_{\pi(i)} + d_i$ for the argument to go through with not negligible probability. Since $2^{\ell_e} \leq f_i < q$ this equality holds over the integers as well.

The remaining $n+1$ acceptable arguments we enumerate $j = 1, \ldots, n+1$. Call the $t_1, \ldots, t_n$ used in the j'th argument for $t_1^{(j)}, \ldots, t_n^{(j)}$. We have corresponding answers $f_i^{(j)} = t_{\pi(i)}^{(j)} + d_i, Z^{(j)}$. Consider the integer vectors $(t_1^{(j)}, \ldots, t_n^{(j)}, 1)$ and the corresponding matrix T containing these as row vectors. For any prime dividing $|\mathcal{M}|$, there is overwhelming probability that the vectors are linearly independent modulo p since $|\mathcal{M}|$ only has large prime divisors. This means $\gcd(\det(T), p) = 1$ for all p dividing the order of $\mathcal{M}$ and thus $\gcd(\det(T), |\mathcal{M}|) = 1$. Let A be the transposed cofactor matrix of T, then we have

$$AT = \det(T)I.$$

Calling the entries of A for a_{kj}, we have

$$\sum_{j=1}^{n+1} a_{kj}(t_1^{(j)}, \ldots, t_n^{(j)}, 1) = (0, \ldots, 0, \det(T), 0, \ldots, 0),$$

where $\det(T)$ is placed in position k. For all j the verification gives us

$$\prod_{i=1}^n e_i^{-t_i^{(j)}} \prod_{i=1}^n E_i^{t_{\pi(i)}^{(j)}} (\prod_{i=1}^n E_i^{d_i} E_d)^1 = \prod_{i=1}^n e_i^{-t_i^{(j)}} \prod_{i=1}^n E_i^{f_i^{(j)}} E_d = E(1; Z^{(j)}).$$

For all $k = 1, \ldots, n$ we have

$$(e_k^{-1} E_{\pi^{-1}(k)})^{\det(T)} \quad = \quad \prod_{i=1}^n (e_i^{-1} E_{\pi^{-1}(i)})^{\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} (\prod_{i=1}^n E_i^{d_i} E_d)^{\sum_{j=1}^{n+1} a_{kj} 1}$$

$$
\begin{aligned}
&= \prod_{i=1}^{n} e_i^{-\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} \prod_{i=1}^{n} E_i^{\sum_{j=1}^{n+1} a_{kj} t_{\pi(i)}^{(j)}} (\prod_{i=1}^{n} E_i^{d_i} E_d)^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{j=1}^{n+1} \Big(\prod_{i=1}^{n} e_i^{-t_i^{(j)}} \prod_{i=1}^{n} E_i^{t_{\pi(i)}^{(j)}} (\prod_{i=1}^{n} E_i^{d_i} E_d)^1 \Big)^{a_{kj}} \\
&= \prod_{j=1}^{n+1} E(1; Z^{(j)})^{a_{kj}} = E(1; \sum_{j=1}^{n+1} a_{kj} Z^{(j)}).
\end{aligned}
$$

We now know from the root extraction property that there exists an $R_{\pi^{-1}(k)}$ so $e_k^{-1} E_{\pi^{-1}(k)} = E(1; R_{\pi^{-1}(k)})$. This way we have witnesses for all k for e_k and $E_{\pi^{-1}(k)}$ having the same plaintext. If the cryptosystem has the root extraction property, we can run the root extractor to obtain the randomizers $R_1, \ldots, R_n$.

If the commitment scheme is statistically binding, then even an unbounded adversary cannot change its mind about the values it has committed to. Assume furthermore that the argument of knowledge for a shuffle of known content is a SHVZK proof. The proof shows that in this case we extract a witness even when we face an unbounded adversary, so we actually have a SHVZK proof for shuffle of ciphertexts. $\square$

5 Combining Shuffling and Decryption

To save time it is possible to combine the shuffling and decryption into one operation. Consider for instance the case where we are using ElGamal encryption and share the secret key additively between the mix-servers. Instead of first mixing, then performing threshold decryption, it makes sense to combine the shuffle operations and the decryption operations. This saves computation and each mix-server only has to be activated once instead of twice. While restricting the choice of parameters, namely we must use an ElGamal like cryptosystem and we must share the secret key additively between all the mix-servers, this is a realistic real-life scenario. In particular, it protects against any single honest-but-curious mix-server.

The public key is on the form $(g, y_1, \ldots, y_N)$, where $y_j = g^{x_j}$ and x_j is the secret key of server j. Inputs to the mix-net are ElGamal encryptions under the key $(g, \prod_{j=1}^{N} y_j)$ on the form $(g^r, (\prod_{j=1}^{N} y_j)^r m)$. The first server shuffles and decrypts with respect to its own key. This leaves us with encryptions under the key $(g, \prod_{j=2}^{N} y_j)$ that the second server can shuffle and decrypt, etc. Once the last server shuffles and decrypts we get the plaintexts out.

Server s gets input ciphertexts on the form $(u_1, v_1), \ldots, (u_n, v_n)$ under the key $(g, \prod_{j=s}^{N} y_j)$. It selects a permutation π at random, as well as randomizers $R_1, \ldots, R_n$. The output is $(U_1, V_1), \ldots, (U_n, V_n)$ under the key $(g, Y = \prod_{j=s+1}^{N} y_j)$, where

$$
U_i = g^{R_i} u_{\pi(i)} \text{ and } V_i = Y^{R_i} v_{\pi(i)} u_{\pi(i)}^{-x_s}.
$$

What we need is an HVZK argument of knowledge for correctness of such a shuffle-and-decrypt operation.

A couple of papers have already investigated this problem [FMM$^+$02, Fur04b], but their arguments are not HVZK. Instead, they use a weaker security notion saying that an adversary does not learn anything about the permutation. We will suggest an argument that is SHVZK and at the same time is more efficient in terms of computation and communication but has worse round-complexity.

The argument is essentially the same as the argument for correctness of a shuffle of ciphertexts, we have written out everything using the ElGamal notation in this section. The only difference from the shuffle argument is that we add some extras to argue also correctness of the partial decryption. We prove knowledge of the secret key x_s, and argue that it has been used to make partial decryptions. For this purpose, we add an initial message $D = g^{d_x}$. Later, the prover will receive a challenge e and respond with $f = ex_s + d_x$. We use the hidden x_s to ensure that $u_i^{x_s}$ is removed as intended from the output ciphertexts. Due to the e-factor, we raise an entire part of the verification to e. The d_x-part that is used to hide x_s forces us to add some extra elements into the protocol, but this part will not be raised to e.

The full protocol can be seen in Figure 5. The cryptosystem is ElGamal encryption over a group of prime order Q. We include in the public keys an additional homomorphic commitment scheme COM, which has $\mathbb{Z}_Q$ as message space. For notational convenience, we assume the randomizers for these commitments are chosen at random from $\mathbb{Z}_Q$. The public keys include a generator g for the group $\mathbb{G}_Q$ of order Q over which we do the ElGamal encryption, and two public keys y_s and Y.

Theorem 6 *The protocol in Figure 5 is a 7-move public coin special honest verifier zero-knowledge argument of knowledge for correctness of a shuffle and partial decryption of ElGamal ciphertexts. If the commitment schemes are statistically hiding and the argument of knowledge of a shuffle of known content is statistical SHVZK, then the entire argument is statistical SHVZK. If the commitment schemes are statistically binding and we are using a SHVZK proof of shuffle of known content, then the entire argument is a SHVZK proof.*

Sketch of proof. Obviously, we have a 7-move public coin protocol. Completeness is straightforward to verify.

Special honest verifier zero-knowledge. To argue special honest verifier zero-knowledge we describe a simulator that runs without knowledge of $\pi, R_1, \ldots, R_n, x_s$ and also a hybrid simulator that does use knowledge of these secret values.

The simulator gets the challenges $t_1, \ldots, t_n, \lambda, e$ as well as challenges for the argument of knowledge of a shuffle of known contents as input. It selects at random $f_1, \ldots, f_n \leftarrow \mathbb{Z}_q, Z, f, f_V, z_V \leftarrow \mathbb{Z}_Q, c, c_d \leftarrow \mathrm{com}(0, \ldots, 0), C_1 \leftarrow \mathrm{COM}(0)$ and $V_d \leftarrow \mathbb{G}_Q$. It computes $U_d = g^Z \prod_{i=1}^n u_i^{t_i} \prod_{i=1}^n U_i^{-f_i}, U = Y^{eZ} g^{f_V} (\prod_{i=1}^n u_i^{-t_i})^f (\prod_{i=1}^n v_i^{-t_i} \prod_{i=1}^n V_i^{f_i} V_d)^{-e}, D = g^f y_s^{-e}$ and $C_2 = \mathrm{COM}(f_V; z_V) C_1^{-e}$. It also simulates the argument of knowledge of shuffle of known contents.

The hybrid simulator also selects $f_1, \ldots, f_n \leftarrow \mathbb{Z}_q, Z, f, f_V, z_V \leftarrow \mathbb{Z}_Q$. It computes $c \leftarrow \mathrm{com}(\pi(1), \ldots, \pi(n)), d_i \leftarrow f_i - t_{\pi(i)}, c_d \leftarrow \mathrm{com}(-d_1, \ldots, -d_n)$. It selects $r_V \leftarrow \mathbb{Z}_Q$ and $C_1 \leftarrow \mathrm{COM}(r_V)$. It sets $V_d = Y^Z (\prod_{i=1}^n u_i^{-t_i})^{x_s} \prod_{i=1}^n v_i^{t_i} \prod_{i=1}^n V_i^{-f_i} g^{r_V}$. As the simulator it computes $U_d = g^Z \prod_{i=1}^n u_i^{t_i} \prod_{i=1}^n U_i^{-f_i}, U = Y^{eZ} g^{f_V} (\prod_{i=1}^n u_i^{-t_i})^f (\prod_{i=1}^n v_i^{-t_i} \prod_{i=1}^n V_i^{f_i} V_d)^{-e}, D = g^f y_s^{-e}$ and $C_2 = \mathrm{COM}(f_V; z_V) C_1^{-e}$ and simulates the argument of knowledge of shuffle of known contents.

Let us argue that simulated arguments and hybrid arguments are indistinguishable. In both distributions, V_d is random, in the simulation because V_d is selected at random, in the hybrid argument because of the g^{r_V} factor. The only difference between the two types of arguments is the way we compute the commitments. In the simulated argument we compute c, c_d, C_1 as commitments to 0, while in the hybrid argument we compute them as commitments to $\pi(1), \ldots, \pi(n), -d_1, \ldots, -d_n$ and r_V. The hiding properties of the two commitment schemes give us indistinguishability between simulated arguments and hybrid arguments. Furthermore, if both commitment schemes are statistically hiding, then we have statistical indistinguishability between simulated arguments and hybrid arguments.

Next, we argue that hybrid arguments and real arguments are indistinguishable. First, we note that $f_1, \ldots, f_n, Z, f, f_V, z_V$ have the same distribution in the two arguments. Let r_1 be the randomness used in forming C_1. In the hybrid argument we can compute $d_i = f_i - t_{\pi(i)}, d_V = f_V - er_V, r_2 = z_V - er_1, R_d = Z - \sum_{i=1}^n t_{\pi(i)} R_i, d_x = f - ex_s$. These values have the same distribution as they would have if chosen by a real prover. Furthermore, it is straightforward to verify that $c, c_d, U_d, V_d, D, U, C_1, C_2$ attain the same values as computed by a real prover. The only difference between hybrid arguments and real arguments is therefore in the simulation of the argument of knowledge of a shuffle of known contents. The SHVZK property of this argument of shuffle of known contents implies indistinguishability between hybrid arguments and real arguments. Moreover, if the argument of shuffle of known contents is statistical SHVZK then hybrid arguments and real arguments are statistically indistinguishable.

Shuffle and Decryption of ElGamal Ciphertexts

Prover	Common input	Verifier
	$(u_1, v_1), \ldots, (u_n, v_n)$ $(U_1, V_1), \ldots, (U_n, V_n)$ and public keys	
Prover's input $\pi, x_s, R_1, \ldots, R_n$ so $y_s = g^{x_s}$ and $(U_i, V_i) = (g^{R_i} u_{\pi(i)}, Y^{R_i} v_{\pi(i)} u_{\pi(i)}^{-x_s})$		
$r \leftarrow \mathbb{Z}_q$, $R_d \leftarrow \mathcal{R}$ $d_1, \ldots, d_n \leftarrow \mathbb{Z}_q, r_d \leftarrow \mathbb{Z}_q$ $c = \mathrm{com}(\pi(1), \ldots, \pi(n); r)$ $c_d = \mathrm{com}(-d_1, \ldots, -d_n; r_d)$ $U_d = \prod_{i=1}^n U_i^{-d_i} g^{R_d}$ $V_d = \prod_{i=1}^n V_i^{-d_i} Y^{R_d} g^{r_V}$ $d_x, r_V, d_V, r_1, r_2 \leftarrow \mathbb{Z}_Q, D = g^{d_x}$ $C_1 = \mathrm{COM}(r_V; r_1), C_2 = \mathrm{COM}(d_V; r_2)$	$\xrightarrow{c, c_d, U_d, V_d, D, C_1, C_2}$	
	$\xleftarrow{t_1, \ldots, t_n}$	$t_i \leftarrow \{0,1\}^{\ell_e}$
$f_i = t_{\pi(i)} + d_i, Z = \sum_{i=1}^n t_{\pi(i)} R_i + R_d$ $U = g^{d_V} (\prod_{i=1}^n u_i^{-t_i})^{d_x}$	$\xrightarrow{f_1, \ldots, f_n, Z, U}$	
	$\xleftarrow{\lambda, e}$	$\lambda, e \leftarrow \{0,1\}^{\ell_e}$
$\mathrm{Arg}(\pi, \rho \vert c^\lambda c_d \mathrm{com}(f_1, \ldots, f_n; 0)$ $= \mathrm{com}(\lambda\pi(1) + t_{\pi(1)},$ $\ldots, \lambda\pi(n) + t_{\pi(n)}; \rho))$	$\leftarrow$ $\rightarrow$ $\leftarrow$ $\rightarrow$	
$f = ex_s + d_x, f_V = er_V + d_V, z_V = er_1 + r_2$	$\xrightarrow{f, f_V, z_V}$	

Check $c, c_d \in \mathcal{C}_{com}, U_d, V_d, D, U \in \mathbb{G}_Q$
and $C_1, C_2 \in \mathcal{C}_{COM}$
and $2^{\ell_e} \leq f_1, \ldots, f_n < q, Z, f, f_V, z_V \in \mathbb{Z}_Q$
Verify $\mathrm{Arg}(\pi, \rho)$
Check $\prod_{i=1}^n u_i^{-t_i} \prod_{i=1}^n U_i^{f_i} U_d = g^Z$
Check $(\prod_{i=1}^n u_i^{-t_i})^{-f} (\prod_{i=1}^n v_i^{-t_i} \prod_{i=1}^n V_i^{f_i} V_d)^e U = Y^{eZ} g^{f_V}$
Check $y_s^e D = g^f$ and $C_1^e C_2 = \mathrm{COM}(f_V; z_V)$

Figure 5: Argument of Shuffle and Decryption of ElGamal Ciphertexts.

Witness-extended emulation. As in the proof of Theorem 5 we use an emulator that runs the prover P^* on a real verifier and outputs this view. In case the argument is acceptable the emulator rewinds and runs P^* until it has $n + 3$ acceptable arguments. By a similar argument, this emulator runs in expected polynomial time.

As in the proof of Theorem 5, we can extract openings of c and c_d. As argued there we can find a permutation π so c contains $\pi(1), \ldots, \pi(n)$. We call the opening of c_d for $-d_1, \ldots, -d_n$. This gives us $f_1, \ldots, f_n$ on the form $f_i = t_{\pi(i)} + d_i$.

From the equations $y_s^e D = g^f$ and $y_s^{e'} D = g^{f'}$ we get $y_s^{e-e'} = g^{f-f'}$. If $e \neq e'$ we then have $y_s = g^{x_s}$, where $x_s = (f - f')(e - e')^{-1}$. This also means $D = g^f y_s^{-e} = g^{f-ex_s}$, so $D = g^{d_x}$, where $d_x = f - ex_s$. We now have π and x_s, but still need to extract the randomizers $R_1, \ldots, R_n$.

We also have $C_1^e C_2 = \mathrm{COM}(f_V; z_V)$ and $C_1^{e'} C_2 = \mathrm{COM}(f'_V; z'_V)$ so $C_1^{e-e'} = \mathrm{COM}(f_V - f'_V; z_V - z'_V)$. The root extraction property gives us an opening r_V, r_1 of C_1, and from this we can compute an opening d_V, r_2 of C_2. With overwhelming probability the prover must use $f_V = er_V + d_V$ when forming acceptable arguments.

As in the proof of Theorem 5 we form the matrix T containing challenge rows on the form $(t_1^{(j)}, \ldots, t_n^{(j)}, 1)$ for $j = 1, \ldots, n+1$. Calling the entries of the transposed cofactor matrix a_{kj}, we have

$$\sum_{j=1}^{n+1} a_{kj}(t_1^{(j)}, \ldots, t_n^{(j)}, 1) = (0, \ldots, 0, \det(T), 0, \ldots, 0),$$

where $\det(T)$ is placed in position k.

For all j, the verification gives us

$$\prod_{i=1}^n u_i^{-t_i^{(j)}} \prod_{i=1}^n U_i^{t_{\pi(i)}^{(j)}} (\prod_{i=1}^n U_i^{d_i} U_d)^1 = \prod_{i=1}^n u_i^{-t_i^{(j)}} \prod_{i=1}^n U_i^{f_i^{(j)}} U_d = g^{Z^{(j)}}.$$

For all $k = 1, \ldots, n$ we have

$$\begin{aligned}
(u_k^{-1} U_{\pi^{-1}(k)})^{\det(T)} &= \prod_{i=1}^n (u_i^{-1} U_{\pi^{-1}(i)})^{\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} (\prod_{i=1}^n U_i^{d_i} U_d)^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{i=1}^n u_i^{-\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} \prod_{i=1}^n U_i^{\sum_{j=1}^{n+1} a_{kj} t_{\pi(i)}^{(j)}} (\prod_{i=1}^n U_i^{d_i} U_d)^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{j=1}^{n+1} \Big(\prod_{i=1}^n u_i^{-t_i^{(j)}} \prod_{i=1}^n U_i^{t_{\pi(i)}^{(j)}} (\prod_{i=1}^n U_i^{d_i} U_d)^1 \Big)^{a_{kj}} \\
&= \prod_{j=1}^{n+1} g^{Z^{(j)} a_{kj}} = g^{\sum_{j=1}^{n+1} a_{kj} Z^{(j)}}.
\end{aligned}$$

Define $R_k = (\sum_{j=1}^{n+1} a_{kj} Z^{(j)}) \det(T)^{-1}$. Then we have $U_{\pi^{-1}(k)} = g^{R_{\pi^{-1}(k)}} u_k$.

The final part of the proof is to show that for all i we have $V_i = Y^{R_i} v_{\pi(i)} u_{\pi(i)}^{-x_s}$. From the equations

$$(\prod_{i=1}^n u_i^{-t_i^{(j)}})^{-f^{(j)}} (\prod_{i=1}^n v_i^{-t_i^{(j)}} \prod_{i=1}^n V_i^{f_i^{(j)}} V_d)^{e^{(j)}} U^{(j)} = Y^{e^{(j)} Z^{(j)}} g^{f_V^{(j)}},$$

we get

$$\Big(\prod_{i=1}^n (v_i u_i^{-x_s})^{-t_i^{(j)}} \prod_{i=1}^n V_i^{f_i^{(j)}} V_d g^{-r_V} \Big)^{e^{(j)}} \prod_{i=1}^n u_i^{d_x t_i^{(j)}} U^{(j)} g^{-d_V} = Y^{e^{(j)} Z^{(j)}}.$$

Given any challenge $t_1^{(j)}, \ldots, t_n^{(j)}$ there is negligible probability over $e^{(j)}$ of producing an acceptable argument unless

$$\prod_{i=1}^n (v_i u_i^{-x_s})^{-t_i^{(j)}} \prod_{i=1}^n V_i^{f_i^{(j)}} V_d g^{-r_V} = Y^{Z^{(j)}}.$$

Using the same matrix T as before we get for $k = 1, \ldots, n$

$$\begin{aligned}
(v_k^{-1} u_k^{x_s} V_{\pi^{-1}(k)})^{\det(T)} &= \prod_{i=1}^{n} (v_i^{-1} u_k^{x_s} V_{\pi^{-1}(i)})^{\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} (\prod_{i=1}^{n} V_i^{d_i} V_d g^{-r_V})^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{i=1}^{n} (v_i u_i^{-x_s})^{-\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} \prod_{i=1}^{n} V_i^{\sum_{j=1}^{n+1} a_{kj} t_{\pi(i)}^{(j)}} (\prod_{i=1}^{n} V_i^{d_i} V_d g^{-r_V})^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{j=1}^{n+1} \Big(\prod_{i=1}^{n} (v_i u_i^{-x_s})^{-t_i^{(j)}} \prod_{i=1}^{n} V_i^{t_{\pi(i)}^{(j)}} (\prod_{i=1}^{n} V_i^{d_i} V_d g^{-r_V})^{1} \Big)^{a_{kj}} \\
&= \prod_{j=1}^{n+1} Y^{Z^{(j)} a_{kj}} = Y^{\sum_{j=1}^{n+1} a_{kj} Z^{(j)}}.
\end{aligned}$$

We then have $V_{\pi^{-1}(k)} = Y^{R_{\pi^{-1}(k)}} v_k u_k^{-x_s}$.

Finally, if the commitment schemes are statistically binding and we use a proof of knowledge of shuffle of known content, then the proof shows that we have a SHVZK proof of a shuffle. □

6 Speed, Space and Tricks

Adjusting the key length of the commitment scheme. When carrying out the shuffle argument we use a homomorphic commitment scheme. If we use for instance the Pedersen commitment scheme, then the public key for the commitment scheme contains $n+1$ elements and the cost of making a commitment is a multi-exponentiation of those $n+1$ elements. Depending on the group sizes, it may be costly to compute and distribute such a long key.

It is possible to trade off key length and computational cost when making a commitment. Assume for simplicity in the following that $n = kl$. Assume furthermore that we have a homomorphic commitment scheme that allows us to commit to k elements at once. We can now commit to n elements $m_1, \ldots, m_n$ by setting

$$c = \Big(c_1, \ldots, c_l\Big) \leftarrow \Big(\mathrm{com}(m_1, \ldots, m_k), \ldots, \mathrm{com}(m_{k(l-1)+1}, \ldots, m_{kl})\Big).$$

Using the Pedersen commitment scheme, this forces us to make l multi-exponentiations of $k+1$ elements when making a commitment, but permits a shorter public key.

Batch verification. In the verification phase, the argument of shuffle of known contents has us checking

$$c^e c_d = \mathrm{com}(f_1, \ldots, f_n; z) \text{ and } c_a^e c_d = \mathrm{com}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, 0; z_\Delta).$$

Here we have implemented the latter commitment, which is a commitment to $n-1$ elements, by using the n-element commitment and adding a dummy zero. We note that the important thing here is not the fact that z is the randomizer, but rather that we know some randomizer such that the above equations hold.

If we use one of the commitment schemes suggested in Section 2.3 we can verify both commitments at once using randomization techniques. Namely, pick $\alpha \leftarrow \{0,1\}^{\ell_e}$ at random and verify

$$(c^e c_d)^\alpha c_a^e c_\Delta = \mathrm{com}(\alpha f_1 + f_{\Delta_1}, \ldots, \alpha f_n + 0; \alpha z + z_\Delta).$$

Suppose, this equality holds for two different α, α', then

$$((c^e c_d)^{-1} \mathrm{com}(f_1, \ldots, f_n; z))^{\alpha - \alpha'} = \mathrm{com}(0, \ldots, 0; 0).$$

We can now run the root extractor to find u so

$$(c_a^e c_d)^{-1}\mathrm{com}(f_1;\ldots,f_n;z) = \mathrm{com}(0,\ldots,0;u).$$

In other words, we have an opening $f_1,\ldots,f_n,z-u$ of $c_a^e c_d$. We also have an opening $f_{\Delta_1},\ldots,f_{\Delta_{n-1}},0,\alpha u+z_\Delta$ of $c_a^e c_\Delta$. This means that with overwhelming probability we can find openings of $c^e c_d$ and $c_d^e c_\Delta$ to respective messages $f_1,\ldots,f_n$ and $f_{\Delta_1},\ldots,f_{\Delta_{n-1}}$.

The randomization method generalizes to the case where we have multiple commitment equations to verify. As the number of commitment equation increases the cost of each of them goes down. In addition, if we use a key of length $k+1$ elements for the commitments, then we have l commitments that we can verify with these techniques. In the latter case, we have $c=(c_1,\ldots,c_l), c_d=(c_{d,1},\ldots,c_{d,l}), c_a=(c_{a,1},\ldots,c_{a,l}), c_\Delta=(c_{\Delta,1},\ldots,c_{\Delta,l})$. We pick $\alpha_1,\ldots,\alpha_l,\beta_1,\ldots,\beta_l \leftarrow \{0,1\}^\ell$ and verify

$$\begin{aligned}&(\prod_{j=1}^{l} c_j^{\alpha_j} c_{a,j}^{\beta_j})^e \prod_{j=1}^{l} c_{d,j}^{\alpha_j} c_{\Delta,j}^{\beta_j}\\ =\ & \mathrm{com}\Big(\sum_{j=1}^{l}(\alpha_j f_{k(j-1)+1}+\beta_j f_{\Delta,k(j-1)+1}),\ldots,\sum_{j=1}^{l}(\alpha_j f_{kj}+\beta_j f_{\Delta,kj});\sum_{j=1}^{l}(\alpha_j z_j+\beta_j z_{\Delta,j})\Big).\end{aligned}$$

This costs $4l+k+2$ exponentiations, mostly to ℓ_e-bit exponents. If for instance $k\approx\sqrt{n}$, then the price is approximately $5\sqrt{n}$ exponentiations. Using the straightforward non-randomized approach, we would end up making $2n+4l$ exponentiations.

Randomization can also bring down the cost of ciphertext exponentiation in the verification process. Suppose we are using the shuffle in a mix-net for instance, then the output ciphertexts from one shuffle will be the input ciphertexts of another shuffle. Calling the output ciphertexts of shuffle j for $E_{1,j},\ldots,E_{n,j}$, we have to check for all j that

$$\prod_{i=1}^{n} E_{i,j-1}^{-t_{i,j}} \prod_{i=1}^{n} E_{i,j}^{f_{i,j}} E_{d,j} = E(1;Z_j).$$

Assume the order of the ciphertext space has no prime divisors smaller than 2^ℓ. Suppose we perform a total of N shuffles. Picking $\alpha_0=0,\alpha_{N+1}=0$ and $\alpha_1,\ldots,\alpha_N\leftarrow\{0,1\}^\ell$ at random we can check

$$\prod_{j=1}^{N}(\prod_{i=1}^{n} E_{i,j-1}^{-\alpha_j t_{i,j}} \prod_{i=1}^{n} E_{i,j}^{\alpha_j f_{i,j}} E_{d,j}^{\alpha_j}) = \prod_{j=0}^{N}(\prod_{i=1}^{n} E_{i,j}^{-\alpha_{j+1}t_{i,j+1}+\alpha_j f_{i,j}} E_{d,j}^{\alpha_j}) = E(1;\sum_{j=1}^{N}\alpha_j Z_j).$$

This test has at most probability $2^{-\ell}$ of passing if either of the N equations is false. The straightforward approach calls for N multi-exponentiations of $2n$ ciphertexts. With the randomized method, we only make one multi-exponentiation of $N(n+1)$ ciphertexts. Even though the exponents are ℓ bits longer, this is a significant gain.

Online/offline. Many of the prover's computations can be pre-computed. To carry out the shuffle itself it is straightforward to select $R_1,\ldots,R_n$ in advance and correspondingly compute the rerandomization factors $E(1;R_1),\ldots,E(1;R_n)$. This way the shuffle itself can be done very quickly.

In the argument of shuffle of known contents we can compute c_d, c_Δ in advance and in the argument of shuffle of homomorphic ciphertexts we can compute c and c_d in advance. This leaves us with the task of computing c_a in the argument of correctness of known contents, and in the shuffle of homomorphic ciphertexts we need to compute E_d.

Multi-exponentiation techniques. While pre-computation and randomization lessens the burden for respectively the prover and the verifier, there is still something that remains. The prover has to compute $E_d = \prod_{i=1}^n E_i^{-d_i} E(1; R_d)$, containing a multi-exponentiation of n ciphertexts. Likewise, the verifier will also have to compute a multi-exponentiation of many ciphertexts. These are the most expensive operations the prover, respectively the verifier, will run into.

While most multi-exponentiation techniques focus on relatively few elements, our situation is different. First, all the ciphertexts are different and cannot be guessed beforehand so pre-computation is not that useful. Second, we have a huge number of ciphertexts. Lim [Lim00] has suggested a method for precisely this situation that uses relatively few multiplications. Using his methods, the cost of the multi-exponentiation corresponds to $\mathcal{O}(n/\log n)$ single exponentiations of ciphertexts.

Multi-exponentiation techniques may of course also be applied when computing the commitments and in any pre-computation phase.

Reducing the length of the exponents. The easiest case is when both the commitment scheme and the cryptosystem have a message space of the same order. Suppose for instance that we are shuffling ElGamal ciphertexts where the message space has prime order q. As a commitment scheme, we can then pick the Pedersen commitment scheme with message space $\mathbb{Z}_q$. This allows us to reduce all exponents modulo q.

In some cases, voting for instance, it may be important that the messages be protected for a long time into the future. For this reason, we may for instance select ElGamal encryption with a large modulus as the cryptosystem. However, the verification of the argument may be something that takes place right away, soundness only has to hold a short time into the future. Since the Pedersen commitment scheme is statistically hiding, we get a statistically hiding argument for the correctness of a shuffle and do not need to worry about the argument itself revealing the messages or the permutation. We can therefore use a Pedersen commitment scheme with a relatively short modulus. The only important thing here is that the orders of the message spaces match.

Of course, there may be situations where we have a huge message space for the cryptosystem. In this case, the cost of a correspondingly large message space for the commitment scheme may be prohibitive. If we are using the Fiat-Shamir heuristic to compute the challenges, another trick may be worth considering to bring down the length of the exponents. Recall, we choose ℓ_s to be large enough so d and $a+d$ are statistically indistinguishable when d is chosen as a random $|a| + \ell_s$-bit number. A reasonable choice would be $\ell_s = 80$. However, in the Fiat-Shamir heuristic we may get by with a much smaller ℓ_s, for instance $\ell_s = 20$. The idea is to check that we do not create an underflow or overflow that reveals the number we are trying to hide. Therefore, if we are trying to hide message $a \in \{0,1\}^{\ell_a}$, then we choose d as a random $\ell_a + \ell_s$-bit number and compute $a + d$. However, if $a + d \notin [2^{\ell_a}; 2^{\ell_a+\ell_s})$ then we reject the argument and start over again. This distribution hides a perfectly, but does of course increase the risk of having to start over again if at some point we do not end up within the interval. However, with a suitable choice of ℓ_s the gain we get from having shorter exponents outweigh the small risk of having to start over again.

Picking the challenges. The important part when we pick $t_1, \ldots, t_n$ is that $n + 1$ random vectors on the form $(t_1^{(j)}, \ldots, t_n^{(j)}, 1)$ should have overwhelming chance of being linearly independent. This is the property that makes the proof of witness-extended emulation go through.

Instead of the verifier picking all of $t_1, \ldots, t_n$ at random, he may instead pick a seed t for a pseudo-random number generator at random. Then $t_1, \ldots, t_n$ are generated from this number generator. There is overwhelming probability that $n+1$ vectors $(t_1^{(j)}, \ldots, t_n^{(j)}, 1)$ generated from seeds $t^{(j)}$ are linearly independent. Furthermore, now we only have to pick a random seed and transmit this instead of picking n elements $t_1, \ldots, t_n$ as the challenge. In cases where the verifier is implemented as a multi-party computation, this may be a significant simplification of the protocol.

In case the cryptosystem has message space of order q and the commitment scheme uses message space $\mathbb{Z}_q$ we just need linear independence over $\mathbb{Z}_q$. One way to obtain this is by picking t at random and setting $t_1 = t^1, \ldots, t_n = t^n$. Vectors on the form $(1, (t^{(j)})^1, \ldots, (t^{(j)})^n)$ correspond to rows in a Vandermonde matrix. The vectors are independent, since the determinant is non-zero, as long as the seeds $t^{(0)}, \ldots, t^{(n)}$ are distinct. If we are using multiparty computation, then we can let each server pick a random input to a collision-free hash function. As long as one of them is honest, the collision-freeness of the hash function ensures that many such runs would give different seeds $t^{(0)}, \ldots, t^{(n)}$, and thus we would obtain the needed linear independence.

We can also use a hash-function to pick x, λ and e, all we need is collision-freeness. This way we get witness-extended, as long as at least one of the parties is honest. However, we may not have a uniform distribution on the outputs of the hash-function, so we may need to apply standard techniques, for instance from [GMY03], to retain the zero-knowledge property.

Parallel shuffling. If we have many sets of ciphertext that we want to shuffle using the same permutation, we can recycle many parts of the protocol. We only need one set of challenges $t_1, \ldots, t_n, \lambda, x, e$, the argument for shuffle of known contents can be reused and so can $c, c_d, f_1, \ldots, f_n$. The only extra work the prover needs to do is to compute a separate E_d for each of the sets and correspondingly send a Z to the verifier for each of the sets. The verifier will then for each of the sets verify $\prod_{i=1}^n e_i^{-t_i} \prod_{i=1}^n E_i^{f_i} E_d = E(1; Z)$. The extra cost for the prover, for each additional set, is a multi-exponentiation of n ciphertexts when computing E_d. For the verifier, each additional set costs a multi-exponentiation of $2n$ ciphertexts.

Selecting the cryptosystem for a mix-net. Throughout the paper we have assumed that the input and output ciphertexts were valid ciphertexts. When designing a mix-net, for instance using the shuffle arguments presented here, it is of course relevant to verify that indeed the input and output ciphertexts are valid. Attacks exist [Wik03] that will compromise the privacy of the mix-net if this check is not performed. We will comment on how an ElGamal cryptosystem can be set up such that this check of the ciphertexts can be done efficiently and be integrated with the argument of correctness of a shuffle.

Let $p = 2qp_1 \ldots p_k + 1$, where $q, p_1, \ldots, p_k$ are distinct primes larger than some bound 2^ℓ. We let g be a randomly chosen generator of the unique subgroup G_q of order q. We choose the secret key $x \leftarrow \mathbb{Z}_q$ and let $y = g^x$. To encrypt a message $m \in G_q$ we choose $(b_1, b_2, r) \leftarrow \{-1, 1\} \times \{-1, 1\} \times \mathbb{Z}_q$ and return the ciphertext $(b_1 g^r, b_2 y^r m)$.

This cryptosystem allows for an efficient batch-verification of membership in $\mathcal{C} = \pm G_q \times \pm G_q$. Assume we have ElGamal ciphertexts $(u_1, v_1), \ldots, (u_n, v_n)$. We choose $\alpha_i \leftarrow [0; 2^\ell)$ and check whether $(\prod_{i=1}^n u_i^{\alpha_i})^q = \pm 1$ and $(\prod_{i=1}^n v_i^{\alpha_i})^q = \pm 1$. The tests have probability $2^{-\ell}$ of passing if any of the ciphertexts does not belong to $\mathcal{C}$.

If we use $\ell = \ell_e$ we may use $t_1, \ldots, t_n$ as our $\alpha_1, \ldots, \alpha_n$. We check in the shuffle argument that

$$\prod_{i=1}^n u_i^{-t_i} \prod_{i=1}^n U_i^{f_i} U_d = \pm g^Z \text{ and } \prod_{i=1}^n v_i^{-t_i} \prod_{i=1}^n V_i^{f_i} V_d = \pm y^Z.$$

As a side effect of these computations we may get out $\prod_{i=1}^n u_i^{t_i}$ and $\prod_{i=1}^n v_i^{t_i}$. It only costs a couple of exponentiations more to test $(\prod_{i=1}^n u_i^{t_i})^q = \pm 1$ and $(\prod_{i=1}^n v_i^{t_i})^q = \pm 1$. The test of validity of the ciphertexts therefore comes at a very low cost. Of course the output ciphertexts can be incorporated into the verification in a similar manner.

7 Comparison of Shuffle Arguments

The literature contains several arguments and proofs for correctness of a shuffle. The most efficient arguments and proofs generally follow one of two paradigms. In the paradigm of Furukawa and Sako [FS01] we commit to a permutation matrix and subsequently argue that indeed we committed to a permutation matrix and furthermore that we have shuffled the ciphertext using the same permutation. This idea has been improved in [Fur04a]. The second paradigm, used in this paper, was suggested by Neff [Nef01]. In this paradigm one uses the fact that polynomials are stable under permutation of the roots. Both paradigms have their merits, here we will compare them and give a rough guide to which one to use.

7.1 HVZK Proof

The first question we must ask ourselves is whether we need computational or unconditional soundness. The schemes based on permutation matrices are arguments, and we see no way to turn them into HVZK proofs. If the situation calls for an HVZK proof we therefore recommend following the Neff paradigm. An unfortunate consequence is that this paradigm leads to 7-move HVZK proofs, so if both unconditional soundness and low round complexity is desirable then we are in trouble. It is an interesting open problem to come up with a highly efficient 3-move HVZK proof for correctness of a shuffle.

We remark that for HVZK proofs it is reasonable to use groups of the same size both for the cryptosystem and for the commitments. Therefore, we do not need to distinguish between exponentiations for the cryptosystem and exponentiations for the commitments, their cost is comparable. Neff [Nef03] suggests an HVZK proof where the prover uses 8 exponentiations and the verifier uses 12 exponentiations. In comparison, in our scheme using the statistically binding commitment scheme from Section 2.3 the prover uses 7 exponentiations and the verifier 9 exponentiations. However, our scheme does require a longer public key than Neff's scheme to get this kind of efficiency.

7.2 HVZK Argument

For ease of comparison with the other schemes we use the standard setting of using ElGamal encryption and Pedersen commitments with primes q, p where $q|p-1, |q| = 160, |p| = 1024$. Whether this choice is reasonable is of course something that depends on the application of the shuffle. As argued earlier when we use statistically hiding commitments and the verification takes place shortly after the shuffle, we only need from the argument that the soundness holds a short time into the future. In this case the binding property of the commitment scheme only needs to be temporarily so it is reasonable to choose a small security parameter. For the commitment scheme $|p| = 1024$ may therefore be reasonable enough. For higher efficiency we might also decide to use elliptic curve groups for the commitment scheme. On the other hand, in some cases we need strong guarantees that the cryptosystem does not reveal anything about the messages many years into the future. In such a case it would be reasonable to choose a longer security parameter for the cryptosystem.

The permutation matrix based approach was suggested by Furukawa and Sako [FS01]. Their scheme is not HVZK [FMM$^+$02], but it does hide the permutation, a property called indistinguishability under chosen permutation attack IND-CPA in [NSNK04]. In their argument the prover uses $8n$ exponentiations and the verifier $10n$ exponentiations. Furukawa [Fur04a] suggests a 3-move HVZK argument where both the prover and the verifier uses $9n$ exponentiations. He observes that letting $q = 2 \bmod 3$ allows a simplification of the protocol so the prover and verifier only need to make $8n$ exponentiations. Making some further changes to the protocol (unpublished) we have been able to push that further down to $6n$ exponentiations for the prover and $7n$ exponentiations for the verifier. In comparison, our scheme uses $6n$ exponentiations for both the prover and verifier. In the earlier version [Gro03] the communication complexity was higher and the scheme was less fit for multi-exponentiations so we list both results separately. Table 1[1] summarizes the complexities of the various homomorphic shuffles without using randomization or batching in the verification.

	Furukawa-Sako [FS01]	Groth [Gro03]	Furukawa (improved) [Fur04a]	(unpublished)	proposed
Prover (expo.)	8n	6n	9n	(6n)	6n
Verifier (expo.)	10n	6n	9n	(7n)	6n
Communication (bits)	5120n	1184n	1344n		480n
Rounds	3	7	3		7
Key length (bits)	1024n	adjustable	1024n		adjustable
Privacy	IND-CPA	SHVZK	SHVZK		SHVZK

Table 1: Comparison of shuffle arguments

Table 1 should of course be read with care. More important than the number of exponentiations is what happens when we throw in randomization, batching and multi-exponentiation techniques. As described in Section 6 the scheme we propose allow using such techniques. We therefore obtain better efficiency than the other schemes and larger flexibility in terms of trading off key length and computational efficiency.

For situations where round complexity matters the permutation matrix based approach gives us 3-move schemes and seems like the best choice. In cases where round complexity is of less importance the scheme we have suggested here is the best choice. It offers a relatively short public key so the cost of key generation is not too large. It offers the better computational and communicational complexities. In particular, if we are using the Fiat-Shamir heuristic to make the shuffle argument non-interactive, then round complexity does not matter much and the present scheme is the superior choice.

7.3 HVZK Argument for Shuffle of Known Contents

We have suggested a 4-move SHVZK argument for shuffle of known contents. When implemented with Pedersen commitments this argument requires the prover to make 3n exponentiations and the verifier to make 2n exponentiations. The communication complexity is 320n bits sent from the prover.

If we implement the argument with the statistically binding commitment from Section 2.3 the prover makes 3n exponentiations and the verifier makes 4n exponentiations.

We do not know of other HVZK arguments for shuffle of known contents in the literature. In some cases we only need an HVZK argument for shuffle of known contents [Gro05b], and in such cases our scheme offers a significant saving in comparison with a full shuffle argument.

7.4 Combined HVZK Argument for Shuffle and Decryption

The 7-move SHVZK argument for a shuffle-and-decrypt operation costs 6n exponentiations for the prover and 7n exponentiations for the verifier. The prover sends 480n bits to the verifier when making the argument.

In comparison [Fur04b] suggests where a 5-move argument, which is not SHVZK but instead has a witness hiding property. In that argument the prover uses 6n exponentiations and 1344n bits of communication, while the verifier uses 8n exponentiations.

If we implement our scheme as a SHVZK proof, then the prover uses 8n exponentiations and the verifier uses 10n exponentiations.

[1] It is possible to reduce the communication complexity further to 320n bits [Gro04] by combining parts of the argument of shuffle of known contents and the full shuffle argument.

References

[Abe98] Masayuki Abe. Universally verifiable mix-net with verification work indendent of the number of mix-servers. In *proceedings of EUROCRYPT '98, LNCS series, volume 1403*, pages 437–447, 1998.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *proceedings of PKC '01, LNCS series, volume 1992*, pages 317–324, 2001.

[AI03] Masayuki Abe and Hideki Imai. Flaws in some robust optimistic mix-nets. In *proceedings of ACISP '03, LNCS series, volume 2727*, pages 39–50, 2003.

[BG02] Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *ACM CCS '02*, pages 68–77, 2002.

[Bra04] Felix Brandt. Efficient cryptographic protocol design based on el gamal encryption, 2004. Manuscript.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *proceedings of CRYPTO '94, LNCS series, volume 893*, pages 174–187, 1994.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[CS98] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *proceedings of CRYPTO '98, LNCS series, volume 1462*, pages 13–25, 1998. Full paper available at `http://eprint.iacr.org/2001/108`.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *proceedings of ASIACRYPT '02, LNCS series, volume 2501*, pages 125–142, 2002.

[DJ01] Ivan Damgård and Mads J. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *proceedings of PKC '01, LNCS series, volume 1992*, 2001.

[DJ03] Ivan Damgård and Mads J. Jurik. A length-flexible threshold cryptosystem with applications. In *proceedings of ACISP '03, LNCS series, volume 2727*, pages 350–364, 2003.

[DK00] Yvo Desmedt and Kaoru Kurosawa. How to break a practical mix and design a new one. In *in proceedings of EUROCRYPT '00, LNCS series, volume 1807*, pages 557–572, 2000.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *proceedings of CRYPTO '84, LNCS series, volume 196*, pages 10–18, 1984.

[FMM+02] Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *proceedings of Financial Cryptography '02, LNCS series, volume 2357*, pages 16–30, 2002.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 368–387, 2001.

[Fur04a] Jun Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. Manuscript, 2004. Full version of [Fur04b].

[Fur04b] Jun Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. In *proceedings of PKC '04, LNCS series, volume 2947*, pages 319–332, 2004.

[GJ04] Philippe Golle and Ari Juels. Parallel mixing. In *proceedings of ACM CCS '04*, pages 220–226, 2004.

[GMY03] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 177–194, 2003. Full paper available at `http://eprint.iacr.org/2003/037`.

[Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *proceedings of PKC '03, LNCS series, volume 2567*, pages 145–160, 2003.

[Gro04] Jens Groth. Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, 2004. PhD thesis. xii+119 pp.

[Gro05a] Jens Groth. Cryptography in subgroups of $\mathbb{Z}_n^*$. In *proceedings of TCC '05, LNCS series, volume 3378*, pages 50–65, 2005.

[Gro05b] Jens Groth. Non-interactive zero-knowledge arguments for voting. In *proceedings of ACNS '05, LNCS series, volume 3531*, 2005.

[Jak98] Markus Jakobsson. A practical mix. In *proceedings of EUROCRYPT '98, LNCS series, volume 1403*, pages 448–461, 1998.

[Jak99] Markus Jakobsson. Flash mixing. In *proceedings of PODC '99*, pages 83–89, 1999.

[JJ99] Markus Jakobsson and Ari Juels. Millimix: Mixing in small batches, 1999. DIMACS technical report 99-33, `http://www.informatics.indiana.edu/markus/papers/millimix.pdf`.

[JJR02] Markus Jakobson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security '02*, pages 339–353, 2002.

[KY04] Aggelos Kiayias and Moti Yung. The vector-ballot e-voting approach. In *proceedings of Financial Cryptography '04, LNCS series, volume 3110*, pages 74–89, 2004.

[Lim00] Chae Hoon Lim. Efficient multi-exponentiation and application to batch verification of digital signatures, 2000. `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps`.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant round secure two-party computation. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 408–432, 2001. Full paper available at `http://eprint.iacr.org/2001/107`.

[NBD01] Juan Manuel Gonzàlez Nieto, Colin Boyd, and Ed Dawson. A public key cryptosystem based on the subgroup membership problem. In *proceedings of ICICS '01, LNCS series, volume 2229*, pages 352–363, 2001.

[Nef01] Andrew C. Neff. A verifiable secret shuffle and its application to e-voting. In *CCS '01*, pages 116–125, 2001. Full paper available at `http://www.votehere.net/vhti/documentation/egshuf.pdf`.

[Nef03] Andrew C. Neff. Verifiable mixing (shuffling) of elgamal pairs, 2003. `http://www.votehere.net/vhti/documentation/egshuf.pdf`.

[NSN03] Lan Nguyen and Reihaneh Safavi-Naini. Breaking and mending resilient mix-nets. In *proceedings of PET '03, LNCS series, volume 2760*, pages 66–80, 2003.

[NSNK04] Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In *proceedings of ACNS '04, LNCS series, volume 3089*, pages 61–75, 2004.

[OA00] Miyako Ohkubo and Masayuki Abe. A length-invariant hybrid mix. In *proceedings of ASIACRYPT '00, LNCS series, volume 1976*, pages 178–191, 2000.

[OT04] Takao Onodera and Keisuke Tanaka. A verifiable secret shuffle of paillier's encryption scheme, 2004. Tokyo Institute of Technology, research report C-193.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *proceedings of EUROCRYPT '98, LNCS series, volume 1403*, pages 308–318, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite residuosity classes. In *proceedings of EUROCRYPT '99, LNCS series, volume 1592*, pages 223–239, 1999.

[PBDV04] Kun Peng, Colin Boyd, Ed Dawson, and Kapalee Viswanathan. A correct, private, and efficient mix network. In *proceedings of PKC '04, LNCS series, volume 2947*, pages 439–454, 2004.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *proceedings of EUROCRYPT '93, LNCS series, volume 765*, pages 248–259, 1993.

[PP89] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct rsa-implementation of mixes. In *proceedings of EUROCRYPT '89, LNCS series, volume 434*, pages 373–381, 1989.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *proceedings of EUROCRYPT '95, LNCS series, volume 921*, pages 393–403, 1995.

[Wik02] Douglas Wikström. The security of a mix-center based on a semantically secure cryptosystem. In *proceedings of INDOCRYPT '02, LNCS series, volume 2551*, pages 368–381, 2002.

[Wik03] Douglas Wikström. Five practical attacks for optimistic mixing for exit-polls. In *proceedings of SAC '03, LNCS series, volume 3006*, pages 160–175, 2003.