J. Cryptol. (2010) 23: 546–579
DOI: 10.1007/s00145-010-9067-9

Journal of
CRYPTOLOGY

A Verifiable Secret Shuffle of Homomorphic Encryptions

Jens Groth[*]
Dept. of Computer Science, University College London, London, UK
j.groth@ucl.ac.uk

Communicated by Kazue Sako

Received 21 August 2005
Online publication 14 May 2010

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

Our scheme is more efficient than previous schemes both in terms of communication and computation. The honest verifier zero-knowledge 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 and batch-verification techniques.

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.

Key words. 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 $E_1, \dots, E_n$ with the same plaintexts in permuted order. We will consider homomorphic public-key cryptosystems in this paper. Informally,[1] we have for public key pk, messages m_1, m_2 and randomizers r_1, r_2 that the encryption function satisfies

$$\mathcal{E}_{pk}(m_1 m_2; r_1 + r_2) = \mathcal{E}_{pk}(m_1; r_1)\mathcal{E}_{pk}(m_2; r_2).$$

[1] See Sect. 2.2 for a formal definition of homomorphic encryption and a description of a few more required properties.

[*] Part of the work done while at University of Aarhus, Cryptomathic and UCLA.

If the cryptosystem is homomorphic, we may shuffle $e_1, \dots, e_n$ by selecting a permutation $\pi \in \Sigma_n$ and randomizers $R_1, \dots, R_n$ and setting

$$E_1 = e_{\pi(1)}\mathcal{E}_{pk}(1; R_1), \dots, E_n = e_{\pi(n)}\mathcal{E}_{pk}(1; R_n).$$

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 the shuffle is correct or incorrect. It could for instance be the case that some ciphertexts had been substituted for other ciphertexts. Our goal is to construct efficient honest verifier zero-knowledge (HVZK) arguments for the correctness of a shuffle. These arguments will make it possible to verify that a shuffle is correct (soundness) but will not reveal the permutation or the randomizers used in the re-encryption step (honest verifier zero-knowledge).

Applications Shuffling is the key building block in most mix-nets. A mix-net [8] 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 randomly chosen permutation. If at least one mix-server is honest and chooses a random permutation, it is impossible to link the input and output. In this role, shuffling constitutes an important building block in anonymization protocols and voting schemes.

In a mix-net it is problematic if a mix-server does not shuffle correctly. In a voting scheme it would for instance be disastrous if a mix-server could substitute some input votes for other votes of its own choosing. HVZK arguments for correctness of a shuffle are therefore useful to ensure that mix-servers follow the protocol. Each mix-server can, after making a shuffle, prove to the other mix-servers or any independent verifiers that the shuffle is correct. The soundness of the HVZK argument guarantees that the shuffle is correct. The honest verifier zero-knowledge property ensures that the HVZK argument does not leak the permutation, the randomizers or any other information pertaining to the shuffle.

Shuffle arguments have also found use as sub-protocols in more complex protocols or zero-knowledge arguments [7,26,32].

Related Work Chaum invented mix-nets in [8]. While his mix-net was based on shuffling, he did not suggest any method to guarantee correctness of the shuffles. Subsequent papers on mix-nets [6,15,22,28–31,42,46,48,52] have tried in many ways to guarantee correctness of a shuffle, most of which have been partially or fully broken [3,39,49,53]. Remaining are suggestions [15,31,48,52], of which the first three have various drawbacks. Desmedt and Kurosawa [15] require that at most a small fraction of the mix-servers is corrupt. Peng et al. [48] require that a fraction of the senders producing the input to the mix-net is honest and restrict the class of possible permutations. Jakobsson, Juels and Rivest [31] allow mix-servers to compromise the privacy of a few senders and/or modify a few messages although they risk being caught. Mix-nets based on shuffling and zero-knowledge arguments of correctness of a shuffle do not have these drawbacks and achieve better efficiency than [52].

Several papers have suggested zero-knowledge arguments for correctness of a shuffle, usually shuffling ElGamal ciphertexts [16]. Sako and Kilian [50] use cut-and-choose methods that are not very efficient. Abe [1] (corrected by Abe and Hoshino [2]) uses permutation networks and obtains reasonable efficiency.

Currently there are two main paradigms that yield practical HVZK arguments for correctness of a shuffle. Furukawa and Sako [20] suggest a paradigm based on permutation matrices in the common reference string model. In this type of construction, we make a commitment to a permutation matrix, argue that we 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 [19], but it does hide the permutation [40]. Furukawa [17] develops the permutation matrix idea further and obtains a practical HVZK argument for correctness of a shuffle. A couple of other works [40,44] also use the permutation matrix idea to obtain HVZK arguments for correctness of a shuffle of Paillier ciphertexts [45]. Following this paradigm, we also have Furukawa et al. [17,19] suggesting arguments for correctness of a combined shuffle-and-decrypt operation, an operation that is used in some decrypting mix-nets.

The other paradigm for verifying correctness of shuffles is due to Neff [36] and is based on polynomials being identical under permutation of their roots. Subsequent versions of that work [37,38] correct some flaws and at the same time obtain higher efficiency. Unlike the Furukawa–Sako paradigm based arguments, Neff obtains an HVZK proof, i.e., soundness is unconditional, but the zero-knowledge property is computational. Neff's proof does not require a common reference string, which gives an efficiency advantage in the setup. For the zero-knowledge property to hold, Neff's proof relies on the public key of the cryptosystem specifying a group where the decision Diffie–Hellman (DDH) assumption holds.

Our Contribution We suggest a 7-move public coin 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. Our HVZK argument has a common reference string, which contains a public key for a homomorphic commitment scheme. If instantiated with a statistically hiding commitment, we obtain a statistical HVZK *argument* for correctness of a shuffle, where soundness holds computationally. On the other hand, if instantiated with a statistically binding commitment scheme, we obtain an HVZK *proof* of correctness of a shuffle with unconditional soundness but computational honest verifier zero-knowledge.

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 and for randomized batch-verification giving us even higher efficiency. Unlike the permutation-matrix based approach, it is possible to work with a short public key for the commitment scheme, 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 seven rounds, and the Furukawa–Sako paradigm can be used to obtain three-round HVZK arguments for correctness of a shuffle.

Improving on the early version of the paper [23], 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 Sect. 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 $m_1, \dots, m_n$, we can form a commitment to a permutation of these messages $c \leftarrow \mathrm{com}_{ck}(m_{\pi(1)}, \dots, m_{\pi(n)})$. We present an argument of knowledge for c containing a permutation of these messages. This has independent interest, for instance [26] 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. Furukawa et al. [17,19] 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.

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 nonuniform polynomial-time or unbounded Turing machines. The different parties and algorithms get a security parameter κ as input; sometimes we omit writing this security parameter explicitly. For an algorithm A, we write $y \leftarrow A(x)$ for the process of selecting randomness r and making the assignment $y = A(x; r)$.

A function $\nu : \mathbb{N} \to [0; 1]$ is negligible if for all constants $\delta > 0$, we have for all sufficiently large κ that $\nu(\kappa) < \kappa^{-\delta}$. For two functions f_1, f_2, we write $f_1(\kappa) \approx f_2(\kappa)$ if $|f_1(\kappa) - f_2(\kappa)|$ is negligible. We define security in terms of probabilities that become negligible as functions of a security parameter κ.

2.2. *Homomorphic Encryption*

We use a probabilistic polynomial-time key generation algorithm to generate a public key and a secret key. The public key belongs to a key space $\mathcal{K}_{\mathrm{enc}}$ and specifies a message space $\mathcal{M}_{pk}$, a randomizer space $\mathcal{R}_{pk}$ and a ciphertext space $\mathcal{C}_{pk}$. It also specifies an efficiently computable encryption algorithm $\mathcal{E} : \mathcal{M}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$. The secret key specifies an efficiently computable decryption algorithm $\mathcal{D} : \mathcal{C}_{pk} \to \mathcal{M}_{pk} \cup \{\texttt{invalid}\}$.

We require that the cryptosystem has perfect decryption such that for all (pk, sk) output by the key generation algorithm and for all $m \in \mathcal{M}_{pk}$ and all $r \in \mathcal{R}_{pk}$, we have

$$\mathcal{D}_{sk}\big(\mathcal{E}_{pk}(m; r)\big) = m.$$

We require the message, randomizer and ciphertext spaces to be finite abelian groups $(\mathcal{M}_{pk}, \cdot, 1)$, $(\mathcal{R}_{pk}, +, 0)$ and $(\mathcal{C}_{pk}, \cdot, 1)$, where it is easy to compute group operations and decide membership. The encryption function must be homomorphic:

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

In this paper, we also demand that the order of the message space is divisible only by large prime-factors. More precisely, it must be the case that $|\mathcal{M}_{pk}|$ has no prime factors smaller than 2^{ℓ_e}, where ℓ_e is a security parameter specified in Sect. 2.6.

We need a root extraction property, which says that if a ciphertext raised to a nontrivial exponent encrypts 1, then the ciphertext itself encrypts 1. More precisely, we assume that there is a root extraction algorithm RootExt that given input $pk \in \mathcal{K}_{\text{enc}}$, $R \in \mathcal{R}_{pk}$, $E \in \mathcal{C}_{pk}$, $e \in \mathbb{Z}$ such that $\gcd(e, |\mathcal{M}_{pk}|) = 1$ and $E^e = \mathcal{E}_{pk}(1; R)$ outputs $r \in \mathcal{R}_{pk}$ such that $E = \mathcal{E}_{pk}(1; r)$. This property suffices for proving soundness, however, for proving witness-extended emulation, we further require that the root extraction algorithm runs in polynomial time.

Various cryptosystems [9,13,14,16,41,43,45] have the properties mentioned in this section or can be tweaked into cryptosystems with these properties. In particular, Paillier encryption [45] and ElGamal encryption [16] have the properties mentioned above and have polynomial-time root extraction.

2.3. *Homomorphic Commitment*

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

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 nonuniform polynomial-time adversaries $\mathcal{A}$, we have

$$\Pr\big[ck \leftarrow K_{\text{com}}\big(1^\kappa\big); (m_0, m_1) \leftarrow \mathcal{A}(ck); c \leftarrow \text{com}_{ck}(m_0) : \mathcal{A}(c) = 1\big]$$
$$\approx \Pr\big[ck \leftarrow K_{\text{com}}\big(1^\kappa\big); (m_0, m_1) \leftarrow \mathcal{A}(ck); c \leftarrow \text{com}_{ck}(m_1) : \mathcal{A}(c) = 1\big],$$

where $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}_{ck}$. If this also holds for unbounded $\mathcal{A}$, we call the commitment statistically hiding.

We say the commitment scheme is binding if a commitment can be opened in one way only. For all nonuniform polynomial-time adversaries $\mathcal{A}$, we have

$$\Pr\big[ck \leftarrow K_{\text{com}}(1^\kappa); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck) : (m_0, r_0), (m_1, r_1) \in \mathcal{M}_{ck} \times \mathcal{R}_{ck} \\ \text{and } m_0 \neq m_1 \text{ and } \mathrm{com}_{ck}(m_0, r_0) = \mathrm{com}_{ck}(m_1; r_1)\big] \approx 0.$$

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

We will use commitment schemes where the message, randomizer and commitment spaces are abelian groups $(\mathcal{M}_{ck}, +, 0)$, $(\mathcal{R}_{ck}, +, 0)$, $(\mathcal{C}_{ck}, \cdot, 1)$. We require that we can efficiently compute group operations and decide membership. The choice of additive or multiplicative notation is not important, what matters is just that they are abelian groups. The commitment function must be homomorphic, i.e.,

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

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 homomorphic integer commitments [12,18,25] with 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 nontrivial degree n polynomial $P(X) \in \mathbb{Z}_q[X]$ has at most n roots, which will be useful later on.

We need a root extraction property, which says that it is infeasible to create an opening of a commitment raised to a nontrivial exponent without being able to open the commitment itself. More precisely, we assume that there is a polynomial-time root extraction algorithm RootExt that given $ck \in \mathcal{K}_{\text{com}}$, $M \in \mathcal{M}_{ck}$, $R \in \mathcal{R}_{ck}$, $c \in \mathcal{C}_{ck}$, $e \in \mathbb{Z}_q^*$ so $c^e = \mathrm{com}_{ck}(M; R)$ outputs a valid opening (m, r) of c.

Examples. As an example of a statistically hiding commitment scheme with these properties, we offer the following variation of Pedersen's commitment scheme [47]. The key generator selects primes q, p so $p = kq + 1$ and k, q are coprime. The commitment 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 u such that $u^k \equiv 1 \bmod p$. We have $\mathcal{M}_{ck} = \mathbb{Z}_q^n$, $\mathcal{R}_{ck} = \mathbb{G}_k \times \mathbb{Z}_q$, $\mathcal{C}_{ck} = \mathbb{Z}_p^*$. To commit to $(m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ using randomness $(u, r) \in \mathbb{G}_k \times \mathbb{Z}_q$ the committer computes $c = u g_1^{m_1} \cdots g_n^{m_n} h^r \bmod p$. For the statistical hiding property to hold, the committer may choose to always use $u = 1$ and simply pick $r \leftarrow \mathbb{Z}_q$ at random. The binding property holds computationally assuming that the discrete logarithm problem is hard in the order q subgroup of $\mathbb{Z}_p^*$. The commitment scheme is homomorphic and has the root extraction property. Our little twist of the Pedersen commitment scheme, adding the u-factor from $\mathbb{G}_k$, ensures we do not have to worry about what happens in the order k subgroup of $\mathbb{Z}_p^*$ and makes it extremely efficient to test membership of $\mathcal{C}_{ck}$; we just have to verify $0 < c < p$.

As an example of a statistically binding commitment scheme, consider selecting the commitment key $(q, p, g_1, \ldots, g_n, h)$ as described above. The message space is $\mathcal{M}_{ck} = \mathbb{Z}_q^n$, the randomizer space is $\mathbb{G}_k^{n+1} \times \mathbb{Z}_q$, and the commitment space is $\mathcal{C}_{ck} = (\mathbb{Z}_p^*)^{n+1}$. The committer commits to $(m_1, \dots, m_n) \in \mathbb{Z}_q^n$ using randomizer $(u_1, \dots, u_n, u, r) \in \mathbb{G}_k^{n+1} \times \mathbb{Z}_q$ by computing the commitment $c = (u_1 g_1^{r+m_1}, \dots, u_n g_n^{r+m_n}, uh^r)$. The committer may choose to always use $u_1 = \cdots = u_n = u = 1$ and picking $r \leftarrow \mathbb{Z}_q$ at random when making the commitments; the hiding property then holds computationally if the DDH problem is hard in the order q subgroup of $\mathbb{Z}_p^*$.

2.4. *Special Honest Verifier Zero-Knowledge Arguments of Knowledge*

Consider a pair of probabilistic polynomial-time interactive algorithms (P, V) called the prover and the verifier. They may have access to a common reference string σ generated by a probabilistic polynomial-time key generation algorithm K. We consider a polynomial-time decidable relation R, which may depend on the common reference string σ. For a statement x, we call w a witness if $(\sigma, x, w) \in R$. We define a corresponding language L_σ consisting of statements that have a witness. We write $\text{tr} \leftarrow \langle P(x), V(y) \rangle$ for the public transcript produced by P and V when interacting on inputs x and y. This transcript ends with V either accepting or rejecting. We sometimes shorten the notation by saying $\langle P(x), V(y) \rangle = b$ 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 nonuniform polynomial-time interactive adversaries $\mathcal{A}$, we have

Completeness:

$$\Pr\left[\sigma \leftarrow K(1^\kappa); (x, w) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ or } \langle P(\sigma, x, w), V(\sigma, x) \rangle = 1\right] \approx 1.$$

Soundness:

$$\Pr\left[\sigma \leftarrow K(1^\kappa); x \leftarrow \mathcal{A}(\sigma) : x \notin L_\sigma \text{ and } \langle \mathcal{A}, V(\sigma, x) \rangle = 1\right] \approx 0.$$

We call (K, P, V) a *proof* if soundness holds for unbounded adversaries.

It will sometimes be convenient to restrict the class of adversaries for which we have soundness. In that case, we say that we have soundness for a class of adversaries $\mathcal{ADV}$ if the definition above holds for all $\mathcal{A} \in \mathcal{ADV}$.

Definition 2 (Public coin). An argument (K, P, V) is said to be public coin if the verifier's messages are chosen uniformly at random independently of the messages sent by the prover.

We define special honest verifier zero-knowledge (SHVZK) [10] for a public coin argument as the ability to simulate the transcript for any set of challenges without access to the witness.

Definition 3 (Special honest verifier zero-knowledge). The public coin argument (K, P, V) is called a special honest verifier zero-knowledge argument for R if there exists a simulator S such that for all nonuniform polynomial time adversaries $\mathcal{A}$, we have

$$\begin{aligned}&\Pr\big[\sigma \leftarrow K(1^\kappa); (x, w, \rho) \leftarrow \mathcal{A}(\sigma);\\ &\quad \mathrm{tr} \leftarrow \big\langle P(\sigma, x, w), V(\sigma, x; \rho)\big\rangle : (\sigma, x, w) \in R \text{ and } \mathcal{A}(\mathrm{tr}) = 1\big]\\ &\approx \Pr\big[\sigma \leftarrow K(1^\kappa); (x, w, \rho) \leftarrow \mathcal{A}(\sigma);\\ &\qquad \mathrm{tr} \leftarrow S(\sigma, x, \rho) : (\sigma, x, w) \in R \text{ and } \mathcal{A}(\mathrm{tr}) = 1\big].\end{aligned}$$

We say that (K, P, V) has statistical SHVZK if the SHVZK property holds for unbounded adversaries.

We remark that a weaker definition of SHVZK arguments, where ρ is chosen uniformly at random instead of chosen by the adversary is common in the literature. We also remark that there are efficient techniques to convert SHVZK arguments into zero-knowledge arguments for arbitrary verifiers in the common reference string model [11,21,24].

Witness-Extended Emulation The standard definition of a system for proof of knowledge by Bellare and Goldreich [4] does not work in our setting since the adversary may have nonzero probability of computing some trapdoor pertaining to the common reference string and use that information in the argument [12]. In this case, it is possible that there exists a prover with 100% probability of making a convincing argument, where we nonetheless cannot extract a witness.

We shall define an argument of knowledge through witness-extended emulation, the name taken from Lindell [35]. Lindell's definition pertains to proofs of knowledge in the plain model, we will adapt his definition to the setting of public coin arguments in the common reference string model. Informally, our 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 4 (Witness-extended emulation). We say that the public coin argument (K, P, V) has witness-extended emulation if for all deterministic polynomial-time P^*, there exists an expected polynomial-time emulator E such that for all nonuniform polynomial-time adversaries $\mathcal{A}$, we have

$$\begin{aligned}&\Pr\big[\sigma \leftarrow K(1^\kappa); (x, s) \leftarrow \mathcal{A}(\sigma); \mathrm{tr} \leftarrow \big\langle P^*(\sigma, x, s), V(\sigma, x)\big\rangle : \mathcal{A}(\mathrm{tr}) = 1\big]\\ &\approx \Pr\big[\sigma \leftarrow K(1^\kappa); (x, s) \leftarrow \mathcal{A}(\sigma); (\mathrm{tr}, w) \leftarrow E^{\langle P^*(\sigma,x,s), V(\sigma,x)\rangle}(\sigma, x) :\\ &\qquad \mathcal{A}(\mathrm{tr}) = 1 \text{ and if tr is accepting then } (\sigma, x, w) \in R\big],\end{aligned}$$

where E has access to a transcript oracle $\langle P^*(\sigma, x, s), V(\sigma, x)\rangle$ that can be rewound to a particular round and run again with the verifier choosing fresh random coins.

We think of s as being the state of P^*, including the randomness. Then we have an argument of knowledge in the sense that the emulator can extract a witness whenever P^* is able to make a convincing argument. This shows that the definition implies soundness. We remark that the verifier's coins are part of the transcript and the prover is deterministic. So combining the emulated transcript with σ, x, s gives us the view of both prover and verifier and at the same time gives us the witness.

Our definition of witness-extended emulation treats both prover and verifier in a black-box manner. The emulator therefore only has access to an oracle that gives it transcripts with a deterministic prover and an honest probabilistic verifier. Treating not only the prover but also the verifier in a black-box manner makes the Fiat–Shamir heuristic described in the end of the section more convincing; we avoid the emulator querying the prover on eschewed challenges or challenges with implanted trapdoors.

In the paper it will sometimes be necessary to restrict the class of adversaries for which we have witness-extended emulation. In that case, we will say that we have witness-extended emulation for a class of adversaries $\mathcal{ADV}$ if the definition above holds for all $\mathcal{A} \in \mathcal{ADV}$.

Damgård and Fujisaki [12] have suggested an alternative definition of an argument of knowledge in the presence of a common reference string. Witness-extended emulation as defined above implies knowledge soundness as defined by them [24].

The Fiat–Shamir Heuristic The Fiat–Shamir heuristic can be used to make public coin SHVZK arguments noninteractive. In the Fiat–Shamir heuristic the verifier's challenges are computed by applying a cryptographic hash-function to the transcript of the protocol.

Security can be argued heuristically in the random oracle model by Bellare and Rogaway [5]. In the random oracle model, the hash-function is modeled as a random oracle that returns a random string on each input it has not been queried before.

2.5. *Setup*

We will construct a 7-round public coin SHVZK argument for the relation

$$\begin{aligned} R = \big\{ &\sigma, (pk, e_1, \dots, e_n, E_1, \dots, E_n), (\pi, R_1, \dots, R_n) \big| \\ &\pi \in \Sigma_n \wedge R_1, \dots, R_n \in \mathcal{R}_{pk} \wedge \forall i : E_i = e_{\pi(i)} \mathcal{E}_{pk}(1; R_i) \big\}. \end{aligned}$$

The relation ignores σ, so this is a standard NP-relation. For soundness and witness-extended emulation, we restrict ourselves to the class of adversaries that produce valid $pk \in \mathcal{K}_{\text{enc}}$. For some cryptosystems, it is straightforward to check whether $pk \in \mathcal{K}_{\text{enc}}$. For ElGamal encryption, validity of a key can be decided in polynomial time. For Paillier encryption, all we need to verify is that there are no small prime factors in the modulus, which can be checked in heuristic polynomial time using Lenstra's [33] elliptic curve factorization method. For other homomorphic cryptosystems, it may not be easy to decide whether the key is correct, however, we may be working in a scenario, where it is correctly setup. For instance, in a mix-net it may be the case that the mix-servers use a multi-party computation protocol to generate the encryption key, and if a majority is honest, then we are guaranteed that the key is correct.

In the SHVZK argument we will suggest, the common reference string will be generated as a public key for a homomorphic commitment scheme for n elements as described in Sect. 2.3. Depending on the applications, there are many possible choices for who generates the commitment key and how they do it. For use in a mix-net, we could for instance imagine that there is a setup phase, where the mix-servers run a multi-party computation protocol to generate the commitment key.

It is possible to let the generation of the common reference string happen in the protocol itself. An unconditionally binding commitment scheme will give us statistical soundness. If we use a commitment scheme, where it is possible to verify that it is unconditionally binding, we can let the prover generate the commitment key and obtain an SHVZK proof. A statistically hiding commitment scheme will give us statistical SHVZK. If it is possible to verify whether a commitment key is statistically hiding, we can let the verifier pick the common reference string. This will give us a statistical SHVZK argument. The statistical SHVZK argument will be public coin if a random string can be used to specify a statistically hiding commitment key.

2.6. *Parameters*

The verifier will select public coin challenges from $\{0,1\}^{\ell_e}$. ℓ_e will be a sufficiently large security parameter, so the risk of breaking soundness is negligible. In practice a choice of $\ell_e = 80$ suffices for interactive protocols. If we make the SHVZK argument noninteractive using the Fiat–Shamir heuristic, $\ell_e = 160$ is low but may be sufficient for some applications. 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 statistically indistinguishable when d is chosen at random from $\{0,1\}^{\ell_a+\ell_s}$. This only leaks information about a in the unlikely situation that $a+d < 2^{\ell_a}$ or $2^{\ell_a+\ell_d} \le a+d$. In practice $\ell_s = 80$ will be sufficient.

We set up the commitment scheme with message space $\mathbb{Z}_q^n$. We demand that $2^{\ell_e+\ell_s} < q$. The reason for this choice is to make q large enough to avoid overflows that require a modular reduction in Sects. 4 and 5. When the cryptosystem has a message space where $m^q = 1$ for all messages, this requirement can be waived, see Sect. 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. SHVZK 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 $m_1, \dots, m_n$ and a commitment c. The problem is to prove knowledge of a permutation π and a randomizer r such that $c = \mathrm{com}_{ck}(m_{\pi(1)}, \dots, m_{\pi(n)}; r)$.

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 [36], 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 $p(X) = \prod_{i=1}^n (m_{\pi(i)} - X)$. We will prove knowledge of $\mu_1, \dots, \mu_n, r$, so $c = \mathrm{com}_{ck}(\mu_1, \dots, \mu_n; r)$, and prove that

$$\prod_{i=1}^n (m_i - X) = \prod_{i=1}^n (\mu_i - X).$$

Since we are working over a field $\mathbb{Z}_q$, this equality implies the existence of a permutation π, so $\mu_i = m_{\pi(i)}$.

To prove that the two polynomials are identical, we will let the verifier choose $x \in \mathbb{Z}_q$ at random and demonstrate that $\prod_{i=1}^n (m_i - x) = \prod_{i=1}^n (\mu_i - x)$. A degree n polynomial in $\mathbb{Z}_q[X]$ can have at most n roots, so there is overwhelming probability of failing the test unless indeed $\prod_{i=1}^n (m_i - X) = \prod_{i=1}^n (\mu_i - X)$.

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

1. Use a standard SHVZK argument with randomly chosen challenge e to argue knowledge of an opening $\mu_1, \dots, \mu_n, r$ of c. In this SHVZK argument of knowledge we get values $f_i = e\mu_i + d_i$, where d_i is committed to by the prover before receiving the random e from the verifier.
2. In the first round of the argument, the verifier will choose an evaluation point $x \in \mathbb{Z}_q$ at random. Once the prover sends out the values $f_1, \dots, f_n$, 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 will argue that $\prod_{i=1}^n (f_i - ex) = e^n \prod_{i=1}^n (m_i - x) + p_{n-1}(e)$. Since e is chosen at random, this means $\prod_{i=1}^n (\mu_i - x) = \prod_{i=1}^n (m_i - x)$ as we wanted.
4. To argue that $\prod_{i=1}^n (f_i - ex) = e^n \prod_{i=1}^n (m_i - x) + p_{n-1}(e)$ the prover will send $F_1, \dots, F_n$ of the form $F_j = e\prod_{i=1}^j (\mu_i - x) + \Delta_j$ to the verifier, where $\Delta_2, \dots, \Delta_{n-1}$ are chosen by the prover before receiving the random challenge e. We use $\Delta_1 = d_1$, so $F_1 = f_1 - ex$. We also use $\Delta_n = 0$, so $F_n = e\prod_{i=1}^n (m_i - x)$, which can be tested directly by the verifier. We will have the equalities $eF_{i+1} = F_i(f_{i+1} - ex) + f_{\Delta_i}$, where the f_{Δ_i}'s are linear in e. From the verifier's point of view these equalities imply that

$$e^n \prod_{i=1}^n (m_i - x) = e^{n-1} F_n = \prod_{i=1}^n (f_i - ex) - p_{n-1}(e),$$

where p_{n-1} is a degree $n-1$ polynomial in e. With overwhelming probability over e this implies $\prod_{i=1}^n (m_i - x) = \prod_{i=1}^n (\mu_i - x)$.

Theorem 1. *The protocol in Fig. 1 is a 4-move public coin special honest verifier zero-knowledge argument with witness-extended emulation for c being a commitment to a permutation of the messages $m_1, \dots, m_n$. If the commitment 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 an SHVZK proof.*

Proof. It is obvious that we are dealing with a 4-move public coin protocol. Perfect 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 $f_1, \dots, f_n, z, F_2, \dots, F_{n-1}, z_\Delta$ and $c_a \leftarrow \mathrm{com}_{ck}(0, \dots, 0)$ at random and then adjusts all other parts of the argument to fit these values. In the same figure, we describe a hybrid

Shuffle of Known Content Argument

Prover | Common input ck, $c, m_1, \ldots, m_n$ | Verifier

Prover's input
π, r so $c = \text{com}_{ck}(m_{\pi(1)}, \ldots, m_{\pi(n)}; r)$

$$\xleftarrow{\quad x \quad} \qquad x \leftarrow \{0,1\}^{\ell_e}$$

$d_1, \ldots, d_n \leftarrow \mathbb{Z}_q, r_d, r_\Delta \leftarrow \mathbb{Z}_q$
$\Delta_1 = d_1, \Delta_2, \ldots, \Delta_{n-1} \leftarrow \mathbb{Z}_q, \Delta_n = 0$
$a_i = \prod_{j=1}^{i}(m_{\pi(j)} - x), r_a \leftarrow \mathbb{Z}_q$
$c_d = \text{com}_{ck}(d_1, \ldots, d_n; r_d)$
$c_\Delta = \text{com}_{ck}(-\Delta_1 d_2, \ldots, -\Delta_{n-1} d_n; r_\Delta)$
$c_a = \text{com}_{ck}(\Delta_2 - (m_{\pi(2)} - x)\Delta_1 - a_1 d_2, \ldots,$
$\quad \Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - a_{n-1} d_n; r_a)$ $\xrightarrow{\quad c_d, c_\Delta, c_a \quad}$

$$\xleftarrow{\quad e \quad} \qquad e \leftarrow \{0,1\}^{\ell_e}$$

$f_i = e m_{\pi(i)} + d_i, \; z = er + r_d$
$f_{\Delta_i} = e(\Delta_{i+1} - (m_{\pi(i+1)} - x)\Delta_i - a_i d_{i+1})$
$\quad - \Delta_i d_{i+1}, \; z_\Delta = e r_a + r_\Delta$ $\xrightarrow{\quad f_1, \ldots, f_n, z \;\; f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta \quad}$

Check $c_d, c_a, c_\Delta \in \mathcal{C}_{ck}$
Check $f_1, \ldots, f_n, z, f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}, z_\Delta \in \mathbb{Z}_q$
Check $c^e c_d = \text{com}_{ck}(f_1, \ldots, f_n; z)$
Check $c_a^e c_\Delta = \text{com}_{ck}(f_{\Delta_1}, \ldots, f_{\Delta_{n-1}}; z_\Delta)$
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 $F_n = e\prod_{i=1}^{n}(m_i - x)$

Fig. 1. Argument of knowledge of shuffle of known content.

simulator that acts just as the simulator except when generating c_a. In the generation of c_a, the hybrid simulator 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.

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 this gives us $f_i, f_{\Delta_i}, z, z_\Delta$ randomly distributed over $\mathbb{Z}_q$.

Simulator	Hybrid	Prover
$f_i \leftarrow \mathbb{Z}_q, z \leftarrow \mathbb{Z}_q$		$f_i = em_{\pi(i)} + d_i, z = er + r_d$
$F_i \leftarrow \mathbb{Z}_q, z_\Delta \leftarrow \mathbb{Z}_q$		$F_i = ea_i + \Delta_i, z_\Delta = er_a + r_\Delta$
$F_1 = f_1 - ex, F_n = e\prod_{i=1}^{n}(m_i - x)$		
$f_{\Delta_i} = eF_{i+1} - F_i(f_{i+1} - ex)$		
	$d_i = f_i - em_{\pi(i)}$	$d_i \leftarrow \mathbb{Z}_q, r_d \leftarrow \mathbb{Z}_q$
	$a_i = \prod_{j=1}^{i}(m_{\pi(j)} - x), r_a \leftarrow \mathbb{Z}_q$	
	$\Delta_i = F_i - ea_i$	$\Delta_i \leftarrow \mathbb{Z}_q, r_\Delta \leftarrow \mathbb{Z}_q$
$c_a \leftarrow \mathrm{com}_{ck}(0, \dots, 0)$	$c_a \leftarrow \mathrm{com}_{ck}(\Delta_2 - (m_{\pi(2)} - x)\Delta_1 - a_1 d_2, \dots, \Delta_n - (m_{\pi(n)} - x)\Delta_{n-1} - a_{n-1}d_n; r_a)$	
$c_d = \mathrm{com}_{ck}(f_1, \dots, f_n; z)c^{-e}$		$c_d = \mathrm{com}_{ck}(d_1, \dots, d_n; r_d)$
$c_\Delta = \mathrm{com}_{ck}(f_{\Delta_1}, \dots, f_{\Delta_{n-1}}; z_\Delta)c_a^{-e}$		$c_\Delta = \mathrm{com}_{ck}(-\Delta_1 d_2, \dots; r_\Delta)$

Fig. 2. Simulation of known shuffle argument.

Given these values, the commitment c_a is computed in the same way by both protocols. Moreover, in both protocols we get $c_d = \mathrm{com}_{ck}(d_1, \dots, d_n; r_d)$ and $c_\Delta = \mathrm{com}_{ck}(-\Delta_1 d_2, \dots, -\Delta_{n-1}d_n; r_\Delta)$.

Witness-Extended Emulation. The emulator E first runs $\langle P^*, V\rangle$ to get a transcript tr. This is the transcript E will output, and by construction it is perfectly indistinguishable from a real SHVZK argument. If the transcript is rejecting, then E halts with (tr, $\bot$). However, if the transcript is accepting, then E must try to find a witness $w = (\pi, r)$.

To extract a witness E rewinds and runs $\langle P^*, V\rangle$ again on the same challenge x until it gets another acceptable argument. Call the two arguments $(x, c_d, c_\Delta, c_a, e, f_1, \dots, f_n, z, f_{\Delta_1}, \dots, f_{\Delta_{n-1}}, z_\Delta)$ and $(x, c_d, c_\Delta, c_a, e', f'_1, \dots, f'_n, z', f'_{\Delta_1}, \dots, f'_{\Delta_{n-1}}, z'_\Delta)$. We have $c^e c_d = \mathrm{com}_{ck}(f_1, \dots, f_n; z)$ and $c^{e'} c_d = \mathrm{com}_{ck}(f'_1, \dots, f'_n; z')$. This gives us $c^{e-e'} = \mathrm{com}_{ck}(f_1 - f'_1, \dots, f_n - f'_n; z - z')$. If $e \neq e'$, E can run the root extraction algorithm to get an opening $\mu_1, \dots, \mu_n, r$ of c.

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 transcript 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 two queries to $\langle P^*, V\rangle$. A consequence of E using an expected polynomial number of queries to $\langle P^*, V\rangle$ is that there is only negligible probability of ending in a run where $e' = e$ or any other event with negligible probability occurs, e.g., breaking the binding property of the commitment scheme. Therefore, with overwhelming probability, either we do not need a witness or we have found an opening $\mu_1, \dots, \mu_n, r$ of c.

We need to argue that the probability for extracting an opening of c such that $\mu_1, \dots, \mu_n$ is not a permutation of $m_1, \dots, m_n$ is negligible. Assume that there is a constant $\delta > 0$ such that P^* has more than $\kappa^{-\delta}$ chance of producing a convincing argument. In that case we can run it with a random challenge x and rewind to get three random challenges e, e', e''. With probability at least $\kappa^{-3\delta}$, P^* manages to create accepting arguments on all three of these challenges. Call the first two arguments $(x, c_d, c_\Delta, c_a, e, f_1, \dots, f_n, z, f_{\Delta_1}, \dots, f_{\Delta_{n-1}}, z_\Delta)$ and $(x, c_d, c_\Delta, c_a, e', f'_1, \dots, f'_n, z', f'_{\Delta_1}, \dots, f'_{\Delta_{n-1}}, z'_\Delta)$. We have $c_a^e c_\Delta = \mathrm{com}_{ck}(f_{\Delta_1}, \dots, f_{\Delta_{n-1}}; z_\Delta)$ and $c_a^{e'} c_\Delta =$

$\mathrm{com}_{ck}(f'_{\Delta_1},\dots,f'_{\Delta_{n-1}};z'_\Delta)$, so $c_a^{e-e'}=\mathrm{com}_{ck}(f_{\Delta_1}-f'_{\Delta_1},\dots,f_{\Delta_{n-1}}-f'_{\Delta_{n-1}};z_\Delta-z'_\Delta)$. With overwhelming probability $e\neq e'$, so we can extract an opening $\alpha_1,\dots,\alpha_{n-1},r_a$ of c_a. This also gives us an opening $\delta_1,\dots,\delta_{n-1},r_\Delta$ of c_Δ, where $\delta_i=f_{\Delta_i}-e\alpha_i$, $r_\Delta=z_\Delta-er_a$. Since we know an opening of c, we also have an opening $d_1,\dots,d_n,r_d$ of c_d with $d_i=f_i-e\mu_i$, $r_d=z-er$.

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 $\mu_1,\dots,\mu_n$ is a permutation of $m_1,\dots,m_n$ 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. SHVZK Argument for Shuffle of Homomorphic Encryptions

A set of ciphertexts $e_1,\dots,e_n$ can be shuffled by selecting a permutation π, selecting randomizers $R_1,\dots,R_n$, and setting $E_1=e_{\pi(1)}\mathcal{E}_{pk}(1;R_1),\dots,E_n=e_{\pi(n)}\mathcal{E}_{pk}(1;R_n)$. The task for the prover is to argue that some permutation π exists so that the plaintexts of $E_1,\dots,E_n$ and $e_{\pi(1)},\dots,e_{\pi(n)}$ 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 $t_1,\dots,t_n$ and computes $\prod_{i=1}^n e_i^{t_i}$ and $\prod_{i=1}^n E_i^{t_{\pi(1)}}$. Finally, the prover proves that the two resulting ciphertexts have the same plaintext. 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 [20,36]:

1. The prover commits to the permutation π as $c\leftarrow\mathrm{com}_{ck}(\pi(1),\dots,\pi(n))$. He makes an SHVZK argument of knowledge of c containing a permutation of the numbers $1,\dots,n$. At this step, the prover is bound to some permutation he knows, but the permutation remains hidden.
2. The prover creates a commitment $c_d\leftarrow\mathrm{com}_{ck}(-d_1,\dots,-d_n)$ to random d_i's. The verifier selects at random $t_1,\dots,t_n$, 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 SHVZK arguments of knowledge of multiplicative relationship and equivalence to show that the products $\prod_{i=1}^n e_i^{t_i}$ and $\prod_{i=1}^n E_i^{f_i}$ differ only by a factor $E_d = \prod_{i=1}^n E_i^{d_i} \mathcal{E}_{pk}(1; R)$ for some randomizer R 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 SHVZK 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 $c^\lambda c_d \mathrm{com}_{ck}(f_1, \ldots, f_n; 0)$ contains a shuffle of $\lambda + t_1, \ldots, \lambda n + t_n$. 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.

Theorem 2. *The protocol in Fig. 3 is a 7-move public coin special honest verifier zero-knowledge argument for correctness of a shuffle of homomorphic ciphertexts. If the cryptosystem has polynomial-time root extraction, then the argument has witness-extended emulation. If the commitment scheme is statistically hiding, then the argument is statistical SHVZK. If the commitment scheme is statistically binding, then the scheme is an 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. With sufficiently large ℓ_s, we have with overwhelming probability that $2^{\ell_e} \leq t_{\pi(i)} + d_i < 2^{\ell_e + \ell_s} < q$ when added as integers. With this in mind, it is straightforward to verify completeness. It remains to prove that we have special honest verifier zero-knowledge and witness-extended emulation.

Special Honest Verifier Zero-Knowledge. Given challenges $t_1, \ldots, t_n, \lambda$ and challenges for the known shuffle, we wish to simulate a transcript that is indistinguishable from a real argument. We describe in Fig. 4 a simulator that simulates the argument without access to the permutation π or the randomizers $R_1, \ldots, R_n$. It picks $c, c_d, f_1, \ldots, f_n, Z$ at random and fits the other parts of the protocol to these values. In the same figure, we also include a hybrid argument that works like the simulator except for generating c, c_d correctly using knowledge of π. Finally, we include for comparison the real prover in a somewhat unordered description.

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. 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 $\prod_{i=1}^n E_i^{-d_i} \mathcal{E}_{pk}(1; R_d) =$

Shuffle of Homomorphic Ciphertexts

Prover | Common input | Verifier

Common input: ck; $pk, e_1, \dots, e_n, E_1, \dots, E_n$

Prover's input
$\pi, R_1, \dots, R_n$ so $E_i = e_{\pi(i)}\mathcal{E}_{pk}(1; R_i)$

$r \leftarrow \mathbb{Z}_q,\ R_d \leftarrow \mathcal{R}_{pk}$
$d_1, \dots, d_n \leftarrow \{0,1\}^{\ell_e+\ell_s},\ r_d \leftarrow \mathbb{Z}_q$
$c = \mathrm{com}_{ck}(\pi(1), \dots, \pi(n); r)$
$c_d = \mathrm{com}_{ck}(-d_1, \dots, -d_n; r_d)$
$E_d = \prod_{i=1}^n E_i^{-d_i}\mathcal{E}_{pk}(1; R_d)$ $\xrightarrow{c,\ c_d,\ E_d}$

$\xleftarrow{t_1, \dots, t_n}$ $t_i \leftarrow \{0,1\}^{\ell_e}$

$f_i = t_{\pi(i)} + d_i$
$Z = \sum_{i=1}^n t_{\pi(i)}R_i + R_d$ $\xrightarrow{f_1, \dots, f_n,\ Z}$

$\xleftarrow{\lambda}$ $\lambda \leftarrow \{0,1\}^{\ell_e}$

$\mathrm{Arg}(\pi, \rho | c^{\lambda}c_d\mathrm{com}_{ck}(f_1, \dots, f_n; 0)$[a]
$= \mathrm{com}_{ck}(\lambda\pi(1) + t_{\pi(1)}, \dots,$
$\lambda\pi(n) + t_{\pi(n)}; \rho))$ $\longleftarrow$ $\longrightarrow$ $\longleftarrow$ $\longrightarrow$

Check $c, c_d \in \mathcal{C}_{ck},\ E_d \in \mathcal{C}_{pk}$
and $2^{\ell_e} \le f_1, \dots, f_n < 2^{\ell_e+\ell_s},\ Z \in \mathcal{R}_{pk}$
Verify $\mathrm{Arg}(\pi, \rho)$
Check $\prod_{i=1}^n e_i^{-t_i}\prod_{i=1}^n E_i^{f_i}E_d = \mathcal{E}_{pk}(1; Z)$

[a] Given $m_1, \dots, m_n, c$, we write $\mathrm{Arg}(\pi, \rho | c = \mathrm{com}_{ck}(m_{\pi(1)}, \dots, m_{\pi(n)}; \rho))$ as a shorthand for carrying out the SHVZK argument in Fig. 1 of knowledge of π, ρ such that $c = \mathrm{com}_{ck}(m_{\pi(1)}, \dots, m_{\pi(n)}; \rho)$.

Fig. 3. Argument of shuffle of homomorphic ciphertexts.

$\mathcal{E}_{pk}(1; Z)\prod_{i=1}^n e_i^{t_i}\prod_{i=1}^n E_i^{-f_i}$, so there is no difference in the way E_d is computed by respectively the hybrid simulator and the prover. 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.

Soundness and Witness-Extended Emulation. The proof of soundness will follow from the proof of witness-extended emulation, so let us start with describing the emulator. We first run $\langle P^*, V\rangle$ to give us a transcript $\mathrm{tr} = (c, c_d, E_d, t_1, \dots, t_n, f_1, \dots, f_n, Z,$

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

Fig. 4. Simulation of shuffle argument.

$\lambda, \mathrm{tr}_{\mathrm{known}})$, where $\mathrm{tr}_{\mathrm{known}}$ is the transcript of the 4-move argument for a shuffle of known contents. If P^* fails to produce an acceptable argument, then we output $(\mathrm{tr}, \perp)$. On the other hand, if the argument is acceptable, then we must extract a witness $\pi, R_1,\ldots,R_n$ 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 rewind $\langle P^*, V\rangle$ to get more transcripts with randomly chosen challenges $t_1,\ldots,t_n,\lambda$ and use the witness-extended emulator for the argument of shuffle of known contents to get openings of $c^\lambda c_d \mathrm{com}_{ck}(f_1,\ldots,f_n,0)$. We do this until we have obtained $n+3$ acceptable arguments.

If we have probability ϵ for getting an acceptable transcript on random challenges $t_1,\ldots,t_n,\lambda$, then we expect to use $\frac{n+2}{\epsilon}$ attempts to sample $n+2$ extra transcripts. Since we only need to extract a witness when the transcript is accepting, we have an expected number of $n+3$ runs. One has to be careful when combining expected polynomial-time algorithms, since the composed algorithm may not be expected polynomial time. In our case, however, we will run the witness-extended emulator on transcripts that have the same distribution as real arguments; in particular the inputs to the witness-extended emulator will always have a size that is polynomial in the security parameter, so we do really get expected polynomial time for the emulator.

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 binding property of the commitment scheme or have collisions among the randomly chosen challenges.

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, ρ and π', ρ' for $c^\lambda c_d \mathrm{com}_{ck}(f_1,\ldots,f_n;0)$ and $c^{\lambda'} c_d \mathrm{com}_{ck}(f_1',\ldots,f_n';0)$ containing shuffles of respectively $\lambda i + t_i$ and $\lambda' i + t_i'$. This gives us

$$c^{\lambda-\lambda'} = \mathrm{com}_{ck}\big(f_1' - f_1 + \lambda\pi(1) + t_{\pi(1)} - \lambda'\pi'(1) - t'_{\pi'(1)},\ldots,$$
$$f_n' - f_n + \lambda\pi(n) + t_{\pi(n)} - \lambda'\pi'(n) - t'_{\pi'(n)}; \rho - \rho'\big).$$

We run the root extractor to get an opening $s_1, \dots, s_n, r$ of c. Given this opening, we can compute an opening $-d_1, \dots, -d_n, r_d$ of c_d with $-d_i \equiv \lambda\pi(i) + t_{\pi(i)} - \lambda s_i - f_i \bmod q$ and $0 \le d_i < q$.

We will now argue that $s_1, \dots, s_n$ is a permutation of $1, \dots, n$. Suppose for some constant $\delta > 0$ that P^* has more than $\kappa^{-\delta}$ chance of producing a valid argument for an infinite number of $\kappa \in \mathbb{N}$ and that we are looking at such a security parameter k. In the third transcript, we have run P^* with randomly chosen challenges $t_1, \dots, t_n, \lambda$, 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 sent by the prover before receiving λ, this has negligible chance of happening unless $s_i = \pi(i)$. We conclude that indeed $s_1, \dots, s_n$ is a permutation of $1, \dots, n$. This in turn tells us that $f_i \equiv t_{\pi(i)} + d_i \bmod q$ for the argument to go through with more than negligible probability. Since $2^{\ell_e} \le f_i < 2^{\ell + \ell_s} < q$, the equality $f_i = t_{\pi(i)} + d_i$ holds over the integers as well.

The last $n + 1$ acceptable transcripts we enumerate $j = 1, \dots, n + 1$. Call the $t_1, \dots, t_n$ used in the jth argument for $t_1^{(j)}, \dots, t_n^{(j)}$. We have corresponding answers $f_i^{(j)} = t_{\pi(i)}^{(j)} + d_i, Z^{(j)}$. Consider the integer vectors $(t_1^{(j)}, \dots, t_n^{(j)}, 1)$ and the corresponding matrix T containing these as row vectors. For any prime p dividing $|\mathcal{M}_{pk}|$, there is overwhelming probability that the vectors are linearly independent modulo p since $|\mathcal{M}_{pk}|$ only has large prime divisors. This means that $\gcd(\det(T), p) = 1$ for all p dividing the order of $\mathcal{M}_{pk}$, and thus $\gcd(\det(T), |\mathcal{M}_{pk}|) = 1$. 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

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

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

$$\prod_{i=1}^{n} e_i^{-t_i^{(j)}} \prod_{i=1}^{n} E_i^{t_{\pi(i)}^{(j)}} \left(\prod_{i=1}^{n} E_i^{d_i} E_d\right)^1 = \prod_{i=1}^{n} e_i^{-t_i^{(j)}} \prod_{i=1}^{n} E_i^{f_i^{(j)}} E_d = \mathcal{E}_{pk}\bigl(1; Z^{(j)}\bigr).$$

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

$$\begin{aligned}
\bigl(e_k^{-1} E_{\pi^{-1}(k)}\bigr)^{\det(T)} &= \prod_{i=1}^{n} \bigl(e_i^{-1} E_{\pi^{-1}(i)}\bigr)^{\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} \left(\prod_{i=1}^{n} E_i^{d_i} E_d\right)^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{i=1}^{n} e_i^{-\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} \prod_{i=1}^{n} E_i^{\sum_{j=1}^{n+1} a_{kj} t_{\pi(i)}^{(j)}} \left(\prod_{i=1}^{n} E_i^{d_i} E_d\right)^{\sum_{j=1}^{n+1} a_{kj} 1} \\
&= \prod_{j=1}^{n+1} \left(\prod_{i=1}^{n} e_i^{-t_i^{(j)}} \prod_{i=1}^{n} E_i^{t_{\pi(i)}^{(j)}} \left(\prod_{i=1}^{n} E_i^{d_i} E_d\right)^1\right)^{a_{kj}}
\end{aligned}$$

$$= \prod_{j=1}^{n+1} \mathcal{E}_{pk}\big(1; Z^{(j)}\big)^{a_{kj}} = \mathcal{E}_{pk}\left(1; \sum_{j=1}^{n+1} a_{kj} Z^{(j)}\right).$$

We now know from the root extraction property that there exists an $R_{\pi^{-1}(k)}$, so $e_k^{-1} E_{\pi^{-1}(k)} = \mathcal{E}_{pk}(1; R_{\pi^{-1}(k)})$, which shows that the argument is sound. If the commitment scheme is statistically binding, we get statistical soundness; where we recall that the SHVZK argument for shuffle of known content has statistical soundness when the commitment is statistically binding. If the cryptosystem has polynomial-time root extraction, we can run the root extractor to find the randomizers $R_1, \dots, R_n$, so we have witness-extended emulation. □

We remark that the proof of soundness shows that the SHVZK argument for correctness of a shuffle is an argument of knowledge of π. However, we may not have full witness-extended emulation where we also learn the rerandomization factors $R_1, \dots, R_n$, unless the cryptosystem has polynomial-time root extraction.

5. Combining Shuffling and Decryption

For efficiency reasons, it may be desirable to combine 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 and then threshold decrypting, 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.

The public key is of the form $(g, y_1, \dots, y_N)$, 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 $(g, \prod_{j=1}^N y_j)$ of 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 of the form $(u_1, v_1), \dots, (u_n, v_n)$ under the key $(g, \prod_{j=s}^N y_j)$. It selects a permutation π at random, as well as randomizers $R_1, \dots, R_n$. The output is $(U_1, V_1), \dots, (U_n, V_n)$ under the key $(g, Y = \prod_{j=s+1}^N y_j)$, where

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

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

A couple of papers have already investigated this problem [17,19], but their arguments are not SHVZK. 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. Neff [38] has independently of this work also investigated the combination of shuffle and decryption operations.

The argument is essentially the same as the SHVZK 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 also argue 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, the prover sends an initial message $D = g^{d_x}$ in the first round. Later, the prover will receive a challenge e and respond with $f = ex_s + d_x$. We use the hidden x_s in f to ensure that $u_i^{x_s}$ is removed as intended from the output ciphertexts. The e-factor in f and the d_x-part that is used to hide x_s forces us to add some extra elements to the protocol.

The full argument can be seen in Fig. 5. The cryptosystem is ElGamal encryption over a group of prime order Q. We include in the common reference string a public key CK for an additional homomorphic commitment scheme COM_{CK}, which has $\mathbb{Z}_Q$ as message space. For notational convenience, we assume that the randomizers for these commitments are chosen at random from $\mathbb{Z}_Q$. The commitment key CK includes a generator g for the group $\mathbb{G}_Q$ of order Q over which we do the ElGamal encryption. The ElGamal encryption key contains y_s and Y from $\mathbb{G}_Q$.

Theorem 3. *The protocol in Fig. 5 is a* 7*-move public coin special honest verifier zero-knowledge argument for correctness of a shuffle and partial decryption of ElGamal ciphertexts with witness-extended emulation. If the commitment schemes are statistically hiding, then the entire argument is statistical SHVZK. If the commitment schemes are statistically binding, then the entire argument is an 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 $\pi, R_1, \dots, R_n, x_s$ and also a hybrid simulator that does use knowledge of these secret values.

The simulator gets the challenges $t_1, \dots, t_n, \lambda, e$ and challenges for the argument of knowledge of a shuffle of known contents as input. It selects at random $f_1, \dots, f_n \leftarrow \{0,1\}^{\ell_e+\ell_s}, Z, f, f_V, z_V \leftarrow \mathbb{Z}_Q, c, c_d \leftarrow \mathrm{com}_{ck}(0, \dots, 0)$, $C_1 \leftarrow \mathrm{COM}_{CK}(0)$ 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 $C_2 = \mathrm{COM}_{CK}(f_V; z_V) C_1^{-e}$. It also simulates the argument of knowledge of shuffle of known contents.

The hybrid simulator chooses $f_1, \dots, f_n \leftarrow \{0,1\}^{\ell_e+\ell_s}, Z, f, f_V, z_V \leftarrow \mathbb{Z}_Q$. It sets $c \leftarrow \mathrm{com}_{ck}(\pi(1), \dots, \pi(n))$, $d_i \leftarrow f_i - t_{\pi(i)}$, $c_d \leftarrow \mathrm{com}_{ck}(-d_1, \dots, -d_n)$. It selects $r_V \leftarrow \mathbb{Z}_Q$ and $C_1 \leftarrow \mathrm{COM}_{CK}(r_V)$. It sets $V_d = Y^Z (\prod_{i=1}^n u_i^{-t_i})^{x_s} \times \prod_{i=1}^n v_i^{t_i} \prod_{i=1}^n V_i^{-f_i} g^{r_V}$. As the simulator, it computes $U = Y^{eZ} g^{f_V} (\prod_{i=1}^n u_i^{-t_i})^f \times (\prod_{i=1}^n v_i^{-t_i} \prod_{i=1}^n V_i^{f_i} V_d)^{-e}$, $U_d = g^Z \prod_{i=1}^n u_i^{t_i} \prod_{i=1}^n U_i^{-f_i}$, $D = g^f y_s^{-e}$ and $C_2 = \mathrm{COM}_{CK}(f_V; z_V) C_1^{-e}$ and simulates the argument of knowledge of shuffle of known contents.

Shuffle and Decryption of ElGamal Ciphertexts

Prover | Common input | Verifier

Common input: ck, CK

$pk = (Q, \mathbb{G}_Q, g, y_s, Y)$

$(u_1, v_1), \ldots, (u_n, v_n)$

$(U_1, V_1), \ldots, (U_n, V_n)$

Prover's input

$\pi, x_s, R_1, \ldots, R_n$ so $y_s = g^{x_s}$ and

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

$r \leftarrow \mathbb{Z}_q,\ R_d \leftarrow \mathcal{R}_{pk}$

$d_1, \ldots, d_n \leftarrow \mathbb{Z}_q,\ r_d \leftarrow \mathbb{Z}_q$

$c = \mathrm{com}_{ck}(\pi(1), \ldots, \pi(n); r)$

$c_d = \mathrm{com}_{ck}(-d_1, \ldots, -d_n; r_d)$

$U_d = \prod_{i=1}^n U_i^{-d_i} g^{R_d}$

$V_d = \prod_{i=1}^n V_i^{-d_i} Y^{R_d} g^{r_V}$

$d_x, r_V, d_V, r_1, r_2 \leftarrow \mathbb{Z}_Q,\ D = g^{d_x}$

$C_1 = \mathrm{COM}_{CK}(r_V; r_1),\ C_2 = \mathrm{COM}_{CK}(d_V; r_2)$ $\xrightarrow{c, c_d, U_d, V_d, D, C_1, C_2}$

$\xleftarrow{t_1, \ldots, t_n}$ $t_i \leftarrow \{0,1\}^{\ell_e}$

$f_i = t_{\pi(i)} + d_i,\ Z = \sum_{i=1}^n t_{\pi(i)} R_i + R_d$

$U = g^{d_V} (\prod_{i=1}^n u_i^{-t_i})^{d_x}$ $\xrightarrow{f_1, \ldots, f_n, Z, U}$

$\xleftarrow{\lambda, e}$ $\lambda, e \leftarrow \{0,1\}^{\ell_e}$

$\mathrm{Arg}(\pi, \rho | c^\lambda c_d \mathrm{com}_{ck}(f_1, \ldots, f_n; 0)$

$= \mathrm{com}_{ck}(\lambda\pi(1) + t_{\pi(1)}, \ldots,$

$\lambda\pi(n) + t_{\pi(n)}; \rho))$[a] $\xleftarrow{}$ $\xrightarrow{}$ $\xleftarrow{}$ $\xrightarrow{}$

$f = e x_s + d_x,\ f_V = e r_V + d_V,\ z_V = e r_1 + r_2$ $\xrightarrow{f, f_V, z_V}$

Check $c, c_d \in \mathcal{C}_{ck}, U_d, V_d, D, U \in \mathbb{G}_Q$ and $C_1, C_2 \in \mathcal{C}_{CK}$

and $2^{\ell_e} \leq f_1, \ldots, f_n < 2^{\ell_e + \ell_s}, Z, f, f_V, z_V \in \mathbb{Z}_Q$

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 $C_1^e C_2 = \mathrm{COM}_{CK}(f_V; z_V)$

[a] Given $m_1, \ldots, m_n, c$, we write $\mathrm{Arg}(\pi, \rho | c = \mathrm{com}_{ck}(m_{\pi(1)}, \ldots, m_{\pi(n)}; \rho))$ as a shorthand for carrying out the SHVZK argument in Fig. 1 of knowledge of π, ρ such that $c = \mathrm{com}_{ck}(m_{\pi(1)}, \ldots, m_{\pi(n)}; \rho)$.

Fig. 5. Argument of shuffle and decryption of ElGamal ciphertexts.

Let us argue that simulated arguments and hybrid arguments are indistinguishable. In both distributions, V_d is random. In the simulation it is random because V_d is selected at random; in the hybrid argument it is random because of the g^{r_V} factor. The only difference between the two types of arguments is the way we compute the commitments c, c_d, C_1. 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 respectively $\pi(1), \dots, \pi(n)$, $-d_1, \dots, -d_n$ 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 $f_1, \dots, f_n, Z, f, f_V, z_V$ 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.

Witness-Extended Emulation. As in the proof of Theorem 2, we use an emulator that runs $\langle P^*, V\rangle$ and outputs the transcript. In case the argument is acceptable the emulator rewinds and runs $\langle P^*, V\rangle$ until it has $n+3$ acceptable arguments. As in the proof of Theorem 2, we can prove that this emulator runs in expected polynomial time.

As in the proof of Theorem 2, we can extract openings of c and c_d. As argued there, we can find a permutation π, so c contains $\pi(1), \dots, \pi(n)$. We call the opening of c_d for $-d_1, \dots, -d_n$. This gives us $f_1, \dots, f_n$ of 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 that $D = g^f y_s^{-e} = g^{f-ex_s}$, 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 $R_1, \dots, R_n$.

We also have $C_1^e C_2 = \mathrm{COM}_{CK}(f_V; z_V)$ and $C_1^{e'} C_2 = \mathrm{COM}_{CK}(f'_V; z'_V)$, so $C_1^{e-e'} = \mathrm{COM}_{CK}(f_V - f'_V; z_V - z'_V)$. By raising both sides to $(e-e')^{-1} \bmod Q$ we perform a root extraction and get an opening r_V, r_1 of C_1. From the opening of C_1, 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 2, we form the matrix T containing challenge rows of the form $(t_1^{(j)}, \dots, t_n^{(j)}, 1)$ for $j = 1, \dots, n+1$. Calling the entries of the transposed cofactor matrix a_{kj}, we have

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

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

For all j, the verification gives us

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

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

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

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

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

we get

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

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

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

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

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

$$= \prod_{i=1}^{n} \left(v_i u_i^{-x_s}\right)^{-\sum_{j=1}^{n+1} a_{kj} t_i^{(j)}} \prod_{i=1}^{n} V_i^{\sum_{j=1}^{n+1} a_{kj} t_{\pi(i)}^{(j)}} \left(\prod_{i=1}^{n} V_i^{d_i} V_d g^{-r_V}\right)^{\sum_{j=1}^{n+1} a_{kj} 1}$$

$$= \prod_{j=1}^{n+1} \left(\prod_{i=1}^{n} \left(v_i u_i^{-x_s}\right)^{-t_i^{(j)}} \prod_{i=1}^{n} V_i^{t_{\pi(i)}^{(j)}} \left(\prod_{i=1}^{n} V_i^{d_i} V_d g^{-r_V}\right)^1 \right)^{a_{kj}}$$

$$= \prod_{j=1}^{n+1} Y^{Z^{(j)} a_{kj}} = Y^{\sum_{j=1}^{n+1} a_{kj} Z^{(j)}}.$$

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, then the shuffle of known content is statistically sound with statistical witness-extended emulation, and we have an SHVZK proof of a shuffle with statistical witness-extended emulation. □

6. Speed, Space and Tricks

Adjusting the Key Length of the Commitment Scheme We use a homomorphic commitment scheme in the shuffle argument. If we use the Pedersen commitment scheme, then the public key 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 $m_1, \dots, m_n$ by setting

$$c = (c_1, \dots, c_l) \leftarrow \left(\mathrm{com}_{ck}(m_1, \dots, m_k), \dots, \mathrm{com}_{ck}(m_{k(l-1)+1}, \dots, m_{kl})\right).$$

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

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

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 Sect. 2.3, we can verify both commitments at once using randomization techniques. Namely, pick $\alpha \leftarrow \{0,1\}^{\ell_e}$ at random and verify

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

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

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

We can now run the root extractor to find u, so

$$(c^e c_d)^{-1}\mathrm{com}_{ck}(f_1; \dots, f_n; z) = \mathrm{com}_{ck}(0, \dots, 0; u).$$

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

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

$$\left(\prod_{j=1}^{l} c_j^{\alpha_j} c_{a,j}^{\beta_j}\right)^e \prod_{j=1}^{l} c_{d,j}^{\alpha_j} c_{\Delta,j}^{\beta_j} = \mathrm{com}_{ck}\left(\sum_{j=1}^{l}(\alpha_j f_{k(j-1)+1} + \beta_j f_{\Delta,k(j-1)+1}), \dots, \right.$$
$$\left. \sum_{j=1}^{l}(\alpha_j f_{kj} + \beta_j f_{\Delta,kj}); \sum_{j=1}^{l}(\alpha_j z_j + \beta_j z_{\Delta,j})\right).$$

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 nonrandomized approach, we would end up making $2n + 4l$ exponentiations.

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

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

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

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

This test has at most probability $2^{-\ell_e}$ 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_e bits longer, this is a significant gain.

Online/Offline Many of the prover's computations can be precomputed. We can select $R_1, \dots, R_n$ in advance and compute the rerandomization factors $\mathcal{E}_{pk}(1; R_1), \dots, \mathcal{E}_{pk}(1; R_n)$. 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 precomputation and randomization lessens the burden for respectively the prover and the verifier, there is still something that remains. The prover has to compute $E_d = \prod_{i=1}^{n} E_i^{-d_i} \mathcal{E}_{pk}(1; R_d)$, containing a multi-exponentiation of n ciphertexts. Likewise, the verifier will also have to compute a multi-exponentiation of many ciphertexts. These are typically 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 precomputation is not that useful. Second, we have a huge number of ciphertexts. Lim [34] 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 can of course also be applied when computing the commitments and in any precomputation 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, so 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 bring down the length of the exponents.

Recall that 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 $a+d \notin [2^{\ell_a}; 2^{\ell_a+\ell_s})$, we 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 of 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 pseudorandom 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 prime 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 of 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 nonzero, 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 emulation, as long as at least one of the verifiers 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 [24] to retain the zero-knowledge property.

Parallel Shuffling As observed by Neff [36], 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 $\prod_{i=1}^n e_i^{-t_i} \prod_{i=1}^n E_i^{f_i} E_d = \mathcal{E}_{pk}(1; Z)$. 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 [53] 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 $p = 2qp_1 \cdots p_k + 1$, 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 $\mathbb{G}_q$ of order q. We choose the secret key $x \leftarrow \mathbb{Z}_q$ and set $y = g^x$. To encrypt a message $m \in \mathbb{G}_q$ 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 $\mathcal{C}_{pk} = \pm\mathbb{G}_q \times \pm\mathbb{G}_q$. Assume that we have ElGamal ciphertexts $(u_1, v_1), \ldots, (u_n, v_n)$. We choose $\alpha_i \leftarrow \{0, 1\}^\ell$ 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 $\mathcal{C}_{pk}$.

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

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

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 [20] we commit to a permutation matrix and subsequently argue that indeed we committed to a permutation matrix and furthermore that we have shuffled the ciphertexts using the same permutation. This idea was improved by Furukawa [17]. The second paradigm, used in this paper, was suggested by Neff [36]. 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. *SHVZK Proof*

The schemes based on permutation matrices are *arguments*, and we see no way to turn them into SHVZK *proofs*. If the situation calls for an SHVZK proof, we therefore recommend following the Neff paradigm. An unfortunate consequence is that this paradigm leads to 7-move SHVZK 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 SHVZK proof for correctness of a shuffle.

Our shuffle argument can be used for many different cryptosystems. Neff [36,37] investigated the case of ElGamal encryption, which we will look a little closer at now. For SHVZK proofs, it is reasonable to use groups of the same size both for the cryptosystem and for the commitment scheme, since typically they will both be governed by the same security parameter that is chosen so both the cryptosystem and the SHVZK proof will keep the permutation secret. Therefore, we do not need to distinguish between exponentiations for the cryptosystem and exponentiations for the commitments; their cost is comparable. Neff [37] suggests an SHVZK proof where the prover uses $8n$ exponentiations and the verifier uses $12n$ exponentiations, where n is the number of ciphertexts in the shuffle. This has been improved to using $8n$ exponentiations for the prover and $10n$ exponentiations for the verifier [38]. If we assume a group size of 1024 bits and exponents of 160 bits, Neff's proof [38] uses $7488n$ bits of communication from the prover to the verifier. In comparison, in our scheme using the statistically binding commitment scheme from Sect. 2.3, the prover uses $7n$ exponentiations, and the verifier $9n$ exponentiations. Further, assuming exponents of size 160 bits and group elements of size 1024 bits, we use $5600n$ bits of communication from the prover to the verifier. However, whereas Neff's scheme only relies on a DDH group wherein his cryptosystem is set, our scheme needs a common reference string with a commitment key to get this kind of efficiency. To make the setting completely comparable, we could let the prover select the unconditionally binding commitment key and send it to the verifier in the first round. By adjusting the commitment key length to giving a commitment key for committing to $\sqrt{n}$ elements at a time, we still get slightly better performance.

7.2. *SHVZK Argument*

ElGamal Encryption For ease of comparison with other arguments for correctness of a shuffle in the literature, we will evaluate our scheme using ElGamal encryption and Pedersen commitments with primes q, p where $q|p-1$, $|q| = 160$, $|p| = 1024$. Whether this choice is reasonable 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 larger security parameter for the cryptosystem.

The permutation matrix based approach was suggested by Furukawa and Sako [20]. Their scheme is not SHVZK [19], but it does satisfy a weaker security notion called indistinguishability under chosen permutation attack, IND-CPA, as defined by Nguyen, Safavi-Naini and Kurosawa [40].[2] In Furukawa and Sako's argument the prover uses $8n$

[2] IND-CPA security considers an adversary that does not know the secret key for the cryptosystem. The adversary chooses two permutations and sees a shuffle under one of the permutations and an argument for correctness of the shuffle. The argument is IND-CPA secure if the adversary cannot distinguish which permutation was used.

Table 1. Comparison of shuffle arguments for ElGamal encryption.

	Furukawa–Sako [20]	Groth [23]	Furukawa [17,27]	Proposed
Prover (single expo.)	$8n$	$6n$	$7n$	$6n$
Verifier (single expo.)	$10n$	$6n$	$8n$	$6n$
Prover's communication (bits)	$5120n$	$1184n$	$1344n$	$480n$
Rounds	3	7	3	7
Common reference string (bits)	$1024n$	adjustable	$1024n$	adjustable
Privacy	IND-CPA	SHVZK	SHVZK	SHVZK

exponentiations, and the verifier $10n$ exponentiations. Furukawa [17] suggests a 3-move SHVZK 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. Groth and Lu [27] use this simplification to get an SHVZK argument that for ElGamal encryption is very similar to Furukawa's scheme. In that scheme the prover uses $7n$ exponentiations, and the verifier $8n$ exponentiations.

In comparison, our scheme uses $6n$ exponentiations for both the prover and verifier. In the earlier version [23] the communication complexity was higher, and the scheme was less fit for multi-exponentiations, so we list both results separately. Table 1[3,4] summarizes the complexities of the various arguments for correctness of shuffling ElGamal ciphertexts. Table 1 only counts the cost of proving the correctness of a shuffle, not the cost of doing the shuffle itself, and does not take into account the optimizations that can be achieved by using randomization and batching in the verification.

Table 1 should of course be read with care. More important than the number of single exponentiations is what happens when we use randomization, batching and multi-exponentiation techniques. As described in Sect. 6, our scheme is well suited to take advantage of such techniques, and after applying such techniques to all the schemes we still have better efficiency than the other schemes and more flexibility in terms of trading off key length and computational efficiency.

Paillier Encryption Several arguments for correctness of a shuffle of Paillier ciphertexts have also been suggested. Most of these arguments for correctness of a shuffle follow the Furukawa–Sako paradigm and yield 3-round arguments. Nguyen, Safavi-Naini and Kurosawa [40] were the first to suggest a 3-round argument for correctness of a shuffle for Paillier encryption. They were followed by Onodera and Tanaka [44] that achieved much better efficiency. Recently Groth and Lu [27] have suggested a shuffle argument based on homomorphic integer commitments as well as one that uses ideas from Furukawa [17]; we include the latter scheme in the table.

[3] At first glance it might look like the verifier in our scheme should use $7n$ exponentiations to verify the shuffle. However, the commitment to $f_1, \dots, f_n$ in the full SHVZK argument for a shuffle and the commitment to the $f_1', \dots, f_n'$ in the SHVZK shuffle of known content can be combined such that only one commitment needs to be computed by the verifier. This saves us from making n exponentiations and makes the verifier's computational complexity $6n$ exponentiations.

[4] It is possible to reduce the communication complexity of our scheme further to $320n$ bits [24] by combining parts of the argument of shuffle of known contents and the full shuffle argument.

Table 2. Comparison of shuffle arguments for Paillier encryption.

	Nguyen et al. [40]	Onodera–Tanako [44]	Groth–Lu [27]	Proposed
Prover (single expo.)	$9n$	$1.3n$	$0.7n$	$0.4n$
Verifier (single expo.)	$8n$	$0.7n$	$0.7n$	$0.5n$
Prover's communication (bits)	$9216n$	$2413n$	$1664n$	$720n$
Rounds	3	3	3	7
Common reference string (bits)	$2048n$	$1024n$	$1024n$	adjustable
Privacy	IND-CPA	SHVZK	SHVZK	SHVZK

In Table 2 we compare the arguments for correctness of a shuffle of Paillier ciphertexts. The parameters we have chosen are a 1024-bit Paillier modulus, which gives 2048-bit ciphertexts, 160-bit challenges, and for statistical hiding, we use $\ell_s = 80$. We base our scheme on Pedersen commitment with primes $|p| = 1024$, $|q| = 241$. To measure the prover's and the verifier's computational loads, we count the number of exponentiations with 1024-bit exponents using a 2048-bit modulus. We assume that the computational load grows linearly in the length of the exponent and quadratically in the length of the modulus. As for ElGamal encryption, the table should be read with care since multi-exponentiation and batch-verification techniques can improve the performance of the schemes.

Conclusion 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. As described in Sect. 6, we can adjust the length of the common reference string, so the cost of commitment key generation is not too large. Moreover, our scheme offers the best computational and communicational complexities. In particular, if we are using the Fiat–Shamir heuristic to make the shuffle argument noninteractive, then round complexity does not matter much, and the present scheme is the superior choice.

7.3. *SHVZK 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 Sect. 2.3, the prover makes $3n$ exponentiations, and the verifier makes $4n$ exponentiations.

We do not know of other SHVZK arguments for shuffle of known contents in the literature. In cases where we only need an SHVZK argument for shuffle of known contents [26], our scheme offers a significant saving in comparison with a full shuffle argument.

7.4. *Combined SHVZK Argument for Shuffle and Decryption*

The 7-move SHVZK argument for a shuffle-and-decrypt operation for ElGamal encryption costs $6n$ exponentiations for the prover and $7n$ exponentiations for the verifier. The prover sends $480n$ bits to the verifier when making the argument if we use the parameters suggested earlier.

In comparison, Furukawa [17] suggests a 5-move argument which is not SHVZK but instead has a witness hiding property. In his argument the prover uses $6n$ exponentiations and $1344n$ bits of communication, and the verifier uses $8n$ exponentiations.

If we implement our scheme as an SHVZK proof, then the prover uses $8n$ exponentiations, the verifier uses $10n$ exponentiations, and the prover's communication is 5600 bits.

Acknowledgements

We greatly appreciate discussions we have had with Heiko Stamer and would like to thank him for sharing the insights gained from his implementation of the protocols in the paper [51]. We would also like to thank C. Andrew Neff for sharing drafts of related work [38] with us.

References

[1] M. Abe, Universally verifiable mix-net with verification work independent of the number of mix-servers, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 1403 (Springer, Berlin, 1998), pp. 437–447

[2] M. Abe, F. Hoshino, Remarks on mix-network based on permutation networks, in *PKC*. Lecture Notes in Computer Science, vol. 1992 (Springer, Berlin, 2001), pp. 317–324

[3] M. Abe, H. Imai, Flaws in some robust optimistic mix-nets, in *ACISP*. Lecture Notes in Computer Science, vol. 2727 (Springer, Berlin, 2003), pp. 39–50

[4] M. Bellare, O. Goldreich, On defining proofs of knowledge, in *CRYPTO*. Lecture Notes in Computer Science, vol. 740 (Springer, Berlin, 1992), pp. 390–420

[5] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *ACM CCS* (1993), pp. 62–73

[6] D. Boneh, P. Golle, Almost entirely correct mixing with applications to voting, in *ACM CCS* (2002), pp. 68–77

[7] F. Brandt, Efficient cryptographic protocol design based on distributed ElGamal encryption, in *ICISC*. Lecture Notes in Computer Science, vol. 3935 (Springer, Berlin, 2006), pp. 32–47

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

[9] R. Cramer, V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, in *CRYPTO*. Lecture Notes in Computer Science, vol. 1462 (Springer, Berlin, 1998), pp. 13–25

[10] R. Cramer, I. Damgård, B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, in *CRYPTO*. Lecture Notes in Computer Science, vol. 893 (Springer, Berlin, 1994), pp. 174–187

[11] I. Damgård, Efficient concurrent zero-knowledge in the auxiliary string model, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 418–430

[12] I. Damgård, E. Fujisaki, A statistically-hiding integer commitment scheme based on groups with hidden order, in *ASIACRYPT*. Lecture Notes in Computer Science, vol. 2501 (Springer, Berlin, 2002), pp. 125–142

[13] I. Damgård, M.J. Jurik, A generalisation, a simplification and some applications of Paillier's probabilistic public-key system, in *PKC*. Lecture Notes in Computer Science, vol. 1992 (Springer, Berlin, 2001)

[14] I. Damgård, M.J. Jurik, A length-flexible threshold cryptosystem with applications, in *ACISP*. Lecture Notes in Computer Science, vol. 2727 (Springer, Berlin, 2003), pp. 350–364

[15] Y. Desmedt, K. Kurosawa, How to break a practical MIX and design a new one, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 557–572

[16] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **31**(4), 469–472 (1985)

[17] J. Furukawa, Efficient and verifiable shuffling and shuffle-decryption. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **88-A**(1), 172–188 (2005)

[18] E. Fujisaki, T. Okamoto, Statistical zero knowledge protocols to prove modular polynomial relations, in *CRYPTO*. Lecture Notes in Computer Science, vol. 1294 (Springer, Berlin, 1997), pp. 16–30

[19] J. Furukawa, H. Miyauchi, K. Mori, S. Obana, K. Sako, An implementation of a universally verifiable electronic voting scheme based on shuffling, in *Financial Cryptography*. Lecture Notes in Computer Science, vol. 2357 (Springer, Berlin, 2002), pp. 16–30

[20] J. Furukawa, K. Sako, An efficient scheme for proving a shuffle, in *CRYPTO*. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 368–387

[21] J.A. Garay, P.D. MacKenzie, K. Yang, Strengthening zero-knowledge protocols using signatures. *J. Cryptol.* **19**(2), 169–209 (2006)

[22] P. Golle, A. Juels, Parallel mixing, in *ACM CCS* (2004), pp. 220–226,

[23] J. Groth, A verifiable secret shuffle of homomorphic encryptions, in *PKC*. Lecture Notes in Computer Science, vol. 2567 (Springer, Berlin, 2003), pp. 145–160

[24] J. Groth, Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS (2004). Ph.D. thesis, pp. xii+119

[25] J. Groth, Cryptography in subgroups of $\mathbb{Z}_n^*$, in *TCC*. Lecture Notes in Computer Science, vol. 3378 (Springer, Berlin, 2005), pp. 50–65

[26] J. Groth, Non-interactive zero-knowledge arguments for voting, in *ACNS*. Lecture Notes in Computer Science, vol. 3531 (Springer, Berlin, 2005)

[27] J. Groth, S. Lu, Verifiable shuffle of large size ciphertexts, in *PKC*. Lecture Notes in Computer Science, vol. 4450 (Springer, Berlin, 2007), pp. 377–392

[28] M. Jakobsson, A practical mix, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 1403 (Springer, Berlin, 1998), pp. 448–461

[29] M. Jakobsson, Flash mixing, in *PODC* (1999), pp. 83–89

[30] M. Jakobsson, A. Juels, Millimix: Mixing in small batches (1999)

[31] M. Jakobson, A. Juels, R.L. Rivest, Making mix nets robust for electronic voting by randomized partial checking, in *USENIX Security* (Springer, Berlin, 2002), pp. 339–353

[32] A. Kiayias, M. Yung, The vector-ballot e-voting approach, in *Financial Cryptography*. Lecture Notes in Computer Science, vol. 3110 (Springer, Berlin, 2004), pp. 74–89

[33] H.W. Lenstra, Factoring integers with elliptic curves. *Ann. Math.* **126**, 649–673 (1987)

[34] C.H. Lim, Efficient multi-exponentiation and application to batch verification of digital signatures. Manuscript (2000)

[35] Y. Lindell, Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptol.* **16**(3), 143–184 (2003)

[36] C.A. Neff, A verifiable secret shuffle and its application to e-voting, in *ACM CCS* (2001), pp. 116–125

[37] C.A. Neff, Verifiable mixing (shuffling) of ElGamal pairs (2003)

[38] C.A. Neff, Personal communication (2005)

[39] L. Nguyen, R. Safavi-Naini, Breaking and mending resilient mix-nets, in *PET*. Lecture Notes in Computer Science, vol. 2760 (Springer, Berlin, 2003), pp. 66–80

[40] L. Nguyen, R. Safavi-Naini, K. Kurosawa, Verifiable shuffles: a formal model and a Paillier-based three-round construction with provable security. *Int. J. Inf. Secur.* **5**(4), 241–255 (2006)

[41] J. Manuel González Nieto, C. Boyd, E. Dawson, A public key cryptosystem based on a subgroup membership problem. *Des. Codes and Cryptogr.* **36**(3), 301–316 (2005)

[42] M. Ohkubo, M. Abe, A length-invariant hybrid mix, in *ASIACRYPT*. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 178–191

[43] T. Okamoto, S. Uchiyama, A new public-key cryptosystem as secure as factoring, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 1403 (Springer, Berlin, 1998), pp. 308–318

[44] T. Onodera, K. Tanaka, Shufle for Paillier's encryption scheme. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E88-A**(5), 1241–1248 (2005)

[45] P. Paillier, Public-key cryptosystems based on composite residuosity classes, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 223–239

[46] C. Park, K. Itoh, K. Kurosawa, Efficient anonymous channel and all/nothing election scheme, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 765 (Springer, Berlin, 1993), pp. 248–259

[47] T.P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in *CRYPTO*. Lecture Notes in Computer Science, vol. 576 (Springer, Berlin, 1991), pp. 129–140

[48] K. Peng, C. Boyd, E. Dawson, K. Viswanathan, A correct, private, and efficient mix network, in *PKC*. Lecture Notes in Computer Science, vol. 2947 (Springer, Berlin, 2004), pp. 439–454

[49] B. Pfitzmann, A. Pfitzmann, How to break the direct RSA-implementation of mixes, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 434 (Springer, Berlin, 1989), pp. 373–381

[50] K. Sako, J. Kilian, Receipt-free mix-type voting scheme—a practical solution to the implementation of a voting booth, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 921 (Springer, Berlin, 1995), pp. 393–403

[51] H. Stamer, Efficient electronic gambling: an extended implementation of the toolbox for mental card games, in *WEWoRC 2005*, ed. by C. Wolf, S. Lucks, P.-W. Yau. Lecture Notes in Informatics, vol. P-74 (Gesellschaft für Informatik e.V., 2005), pp. 1–12

[52] D. Wikström, The security of a mix-center based on a semantically secure cryptosystem, in *INDOCRYPT*. Lecture Notes in Computer Science, vol. 2551 (Springer, Berlin, 2002), pp. 368–381

[53] D. Wikström, Five practical attacks for optimistic mixing for exit-polls, in *SAC*. Lecture Notes in Computer Science, vol. 3006 (Springer, Berlin, 2003), pp. 160–175