Efficiently Shuffling in Public

Udaya Parampalli[1], Kim Ramchen[1], and Vanessa Teague[1]

Department of Computer Science and Software Engineering
University of Melbourne
udaya,ramchen,vteague@csse.unimelb.edu.au

Abstract. We revisit *shuffling in public* [AW07a], a scheme which allows a shuffle to be precomputed. We show how to obfuscate a Paillier shuffle with $O(N \log^{3.5} N)$ exponentiations, leading to a very robust and efficient mixnet: when distributed over $O(N)$ nodes the mixnet achieves mixing in polylogarithmic time, independent of the level of privacy or verifiability required. Our construction involves the use of layered Paillier applied to permutation networks. With an appropriate network the shuffle may be confined to a particular subset of permutations, for example to rotations. While it is possible that the mixnet may produce biased output, we show that certain networks lead to an acceptable bias-efficiency tradeoff.

Keywords: Public key obfuscation, homomorphic encryption, electronic voting.

1 Introduction

A re-encryption mix permutes and re-encrypts its input [PIK94]. A series of mixes is called a *mixnet* and guarantees that input ciphertexts cannot be linked to the decrypted output unless all mixes collude. A *proof of shuffle* allows a mix to prove they have correctly processed their input. A fundamental challenge in electronic voting is the design of mixnets that can accomodate a large number of encrypted ballots in a relatively short space of time. An additional goal is *robustness*, a mixnet must be able to recover from the failure of faulty or dishonest mixes. In this paper we present efficient constructions for *shuffling in public* [AW07a], a scheme which allows a shuffle to be *precomputed* before any input is received. Evaluating the precomputed shuffle upon input is public, that is requires no secret information and can be performed even by *untrusted* parties. Evaluation is also highly *parallelisable*, thus the work required to mix votes can be distributed over an arbitrary number of workstations at election time.

1.1 Improving the Efficiency and Robustness of Mixnets

Most mixnets achieve robustness through the detection and replacement of corrupt mixes. Although a few schemes [Cha81,PIK94,GZB$^+$02] verify that the mixnet as a whole functioned properly, these schemes provide no way to recover from errors or identify dishonest mixes. The most common method to audit a

mix is to require it to output a zero knowledge proof. Cut and choose techniques [SK95] are generally applicable but inefficient, hence much work has been devoted to optimising the proof of a shuffle.

Abe [Abe99] and later Jakobsson and Juels [JJ99] presented the first practical proofs of a shuffle based upon re-encryption permutation networks. Furukawa and Sako [FS01] used a commitment to a permutation matrix and Neff [Nef01] used unique factorisation of polynomials to prove a shuffle of ElGamal ciphertexts with improved efficiency. Efficient arguments and proofs have also been devised in the case that the shuffle is restricted to a subset of permutations [RW04,dHSŠV09]. Generic techniques may be used to further optimise the above proofs, including pre-computation of re-encryption factors, fixed base and multi-exponentiation and batch proof techniques [BGR98] and PRGs for challenge generation. Wikström [Wik09] has also observed that a proof of shuffle may be split into offline and online phases. A mix provably commits to its permutation offline allowing a highly efficient commitment-consistent proof in the online phase.

Despite these enhancements there remain inherent limitations on the robustness and efficiency of mixnets which makes use of *private* techniques for online mixing. Firstly, if a mix is detected cheating then the mixnet must either be restarted or delayed until a replacement is found. Secondly, the opportunities for parallelisation are quite limited. As each mix must keep its permutation secret, it must perform its round of mixing and output a correct proof without assistance. Therefore the runtime is at least *linear* in the number of votes and mixes. Thirdly, it is commonly assumed that each mix server in a mixnet should belong to a different organisation (e.g. political party). Online private mixing depends upon a quorum of these co-operating in the short space of time before tallying begins.

In contrast, two schemes *shuffling in public* [AW07a] and *offline/online mixing* [AW07b] allow a shuffle to be precomputed. These schemes imply that no mix servers need be present at election time for mixing to take place. A major downside of offline/online mixing is that each voter requires a separate key to encrypt their vote. Additionally the scheme significantly restricts the number and size of votes. The main disadvantage of shuffling in public is its inefficiency, with generation and evaluation of the precomputed shuffle requiring $O(N^2)$ exponentiations. In this work we reduce both phases to $O(N \log^{3.5} N)$ exponentiations. Experiments indicate that our scheme is faster when $N > 1200$.

1.2 Shuffling in Public

The goal of shuffling in public is the *public-key obfuscation* of the shuffle phase of a mix-net comprising either a *decryption shuffle* or *re-encryption shuffle* functionality (program) [AW07a]. Informally, a public-key obfuscator $\mathcal{O}$ takes a program F and outputs a new program $\mathcal{O}(F)$ which outputs encryptions of F's outputs. That is $\exists \diamond \forall x \; \mathcal{O}(F) \diamond x = O(F(x))$ for some encryption function O and we say the operator $\diamond$ *evaluates* the obfuscated program on input x. A formal model is proposed in Definition 3 [AW07a] which builds upon an earlier definition by

Ostrovsky and Skeith [OS07]. Adida and Wikström present obfuscators for decryption and re-encryption shuffles in the BGN [BGN05] and Paillier [Pai99] cryptosystems respectively. They also prove that their obfuscators are semantically secure (Definition 4 [AW07a]). Given a set of parties who sample and obfuscate a shuffle before any input is received, one can construct a mixnet provided that joint decryption is verifiable.

1.3 Our Contributions

We public key obfuscate a Paillier shuffle using permutation networks. Our obfuscated shuffle comprises $O(N \log N)$ ciphertexts and requires $O(N \log^{3.5} N)$ exponentiations to generate and evaluate, rather than $O(N^2)$ ciphertexts and exponentiations in [AW07a]. Utilising a suitable network, we can restrict the space of permutations, for example we can obfuscate homomorphic rotation. We propose a distributed protocol for sampling and obfuscating a shuffle allowing the construction of a verifiable mixnet. A side effect of the use of permutation networks is that the resulting distribution over permutations may be biased. However it is possible to reduce the bias at the expense of increasing the complexity to $O(N \log^c N)$ for a constant $c > 3.5$. Moreover for some applications weaker anonymity may be acceptable.

1.4 Outline

The paper is organised as follows. In Section 2 we discuss cryptographic preliminaries. In Section 3 we review permutation networks. In Section 4 we show how to obfuscate shuffles of Damgård-Jurik ciphertexts as well as an operation to compose obfuscations. These ideas when applied to permutation re-encryption networks lead to an improved obfuscator for a Paillier shuffle. In Section 5 we provide a distributed protocol for sampling and obfuscating a shuffle via an arbitrary permutation network. In Section 6 we analyse the properties of the resulting mixnet and prove that it is secure under standard assumptions. In Section 7 we conclude and suggest future directions.

2 Preliminaries

2.1 Notation

We denote by κ the security parameter (i.e the bitlength of the RSA modulus), and say that a function $\epsilon(\kappa)$ is negligible if for each $c \in \mathbb{N}$ there exists $\kappa_0 \in \mathbb{N}$ such that for all $\kappa > \kappa_0$, $\epsilon(\kappa) < \kappa^{-c}$. We denote probabilistic polynomial time by PPT and assume all adversaries are PPT Turing machines. Let Σ_N be the symmetric group on N elements. By a "random encryption" of a message m, we will implicity mean an encryption of m where the randomisation factor is chosen uniformly and independently from the randomisation space. Suppose a

PPT Turing Machine A distinguishes distributions D_1 and D_0. We denote by $Adv(A)$ the advantage of A in distinguishing D_1 and D_0, where

$$Adv(A) = |\Pr_{t \leftarrow D_1}[A(t) = 1] - \Pr_{t \leftarrow D_0}[A(t) = 1]|$$

is a function of κ.

2.2 Homomorphic Encryption

Definition 1 (Homomorphic). *The public key of a homomorphic cryptosystem $\mathcal{CS} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ specifies a message space $(M_{pk}, +)$, a randomiser space $(R_{pk}, \cdot)$ and a ciphertext space $(C_{pk}, \times)$ all of which are abelian groups. Encryption is homomorphic*

$$\mathcal{E}_{pk}(m, r) \times \mathcal{E}_{pk}(m', r') = \mathcal{E}_{pk}(m + m', r \cdot r').$$

For any homomorphic cryptosystem we can define a scalar homomorphism generically

$$c \otimes \mathcal{E}_{pk}(m, r) = \underbrace{\mathcal{E}_{pk}(m, r) \times \ldots \times \mathcal{E}_{pk}(m, r)}_{c} = \mathcal{E}_{pk}(cm, r^c).$$

Definition 2 (Indistinguishability under Chosen Plaintext Attacks). *Let $\mathcal{CS} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a cryptosystem and $A = (A_1, A_2)$ be an adversary. Define*

Experiment $Exp_{A,\mathcal{CS}}^{IND-CPA-b}(\kappa)$:
$(pk, sk) \leftarrow \mathcal{G}(1^\kappa); (m_0, m_1, \delta) \leftarrow A_1(pk) : |m_0| = |m_1|; c \leftarrow \mathcal{E}_{pk}(m_b);$
$v \leftarrow A_2(m_0, m_1, \delta, c)$
return v

and let

$$Adv(A) = |\Pr[Exp_{A,\mathcal{CS}}^{IND-CPA-1}(\kappa) = 1] - \Pr[Exp_{A,\mathcal{CS}}^{IND-CPA-0}(\kappa) = 1]|$$

Then $\mathcal{CS}$ satisfies indistinguishability under chosen plaintext attacks (IND-CPA) if for any PPT A, $Adv(A)$ is negligible.

2.3 The Damgård-Jurik Cryptosystem

The Damgård-Jurik Cryptosystem [DJ01] is a generalisation of the Paillier Cryptosystem [Pai99] based on the isomorphism $\mathbb{Z}_{n^i} \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_{n^{i+1}}^*$. Let $\mathcal{E}_{i,n}$ be the i^{th} generalised Paillier encryption, where $i \leq s$ for some integer s.

Key Generation Let $n = pq$ be an RSA modulus. Let $\lambda = lcm(p-1, q-1)$. Compute via the Chinese Remainder Theorem d such that $d = 1 \mod n^s$ and $d = 0 \mod \lambda$.

Encryption Given a plaintext $m \in \mathbb{Z}_{n^i}$, choose a random $r \in \mathbb{Z}_n^*$. Let $\mathcal{E}_{i,n}(m, r) = (1+n)^m r^{n^i} \mod n^{i+1}$.

Decryption $\mathcal{D}_{i,n}(c) = \log_{(1+n)} c^d \bmod n^{i+1}$. We have:
$c^d = (1+n)^{md} r^{n^i d} = (1+n)^{md \bmod n^i} r^{n^i d} = (1+n)^m \bmod n^{i+1}$.
One can extract m given $(1+n)^m \bmod n^{i+1}$ using the binomial-expansion based algorithm presented in Section 3 of [DJ01].

Semantic Security Semantic security of the scheme is based upon the *Decision Composite Residuosity Assumption* (DCRA) [Pai99], which states that no PPT algorithm can distinguish the uniform distribution on $\mathbb{Z}^*_{n^2}$ from the uniform distribution on the subgroup of n^{th} residues in $\mathbb{Z}^*_{n^2}$. In fact an adversary with advantage $\epsilon_i(\kappa)$ against $\mathcal{E}_{i,n}$ implies an adversary with advantage at least $\epsilon_i(\kappa)/i$ against $\mathcal{E}_{1,n}$ as shown in [Gjø05].

2.4 Privacy of a Shuffle

Nguyen et al. in [NSNK04] formally define shuffle privacy by observing that a shuffle of ciphertexts is an "encryption" that hides the permutation. The corresponding security notion is "indistinguishability under chosen permutation attacks" (IND-CPA$_S$). A discussion is included in Appendix A.

3 Permutation Networks

A permutation network is a circuit composed of configurable switches that permutes a set of inputs. For convenience we assume that every switch accepts the same number of inputs. This includes the important special cases of rotation and shuffling - the networks we present are also *optimal* in the sense that the size and depth are minimal. We assume that the number of inputs, N, is a power of two.

Definition 3. *Suppose Ψ is a permutation network of dimension $\Delta \times W$. Then each layer consists of W independent switches where each switch imposes a fixed mapping on $N' = N/W$ inputs when its control bit is true. Let the switches in the i^{th} layer partition the set of inputs $\{1, \ldots, N\}$ into subsets $V_{i,1}, \ldots, V_{i,W}$ with corresponding mappings $\sigma_{i,1}, \ldots, \sigma_{i,W}$. Let $A^{(i)}$ be the adjacency matrix of the i^{th} layer. Then*

$$A^{(i)}_{lm} = \begin{cases} 1 \text{ if } l = m \vee \sigma_{i,j}(l) = m \text{ for some } j \in [W] \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

Let the control bits of the i^{th} layer be $b_{i,1}, \ldots, b_{i,W}$. Then the permutation imposed by that layer is $\pi_i \triangleq \sigma_{i,1}^{b_{i,1}} \ldots \sigma_{i,W}^{b_{i,W}}$. Moreover the state of the network is $\pi \stackrel{\Psi}{=} \pi_\Delta \ldots \pi_1$.

3.1 Rotation

We describe the *barrel shifter* network which is capable of implementing every possible rotation. Let the set of rotations be $\phi^0, \ldots, \phi^{N-1}$. Observe that a switch

on N inputs can output ϕ^0 or ϕ^{2^i} for each i in $[\log_2 N]$. Cascading these switches together, we obtain a network that has $\Delta = \log_2 N$ and $W = 1$. Clearly $\phi^j : 0 \leq j < N$ is output iff $b_{\log_2 N-1} \dots b_0 = j_2$, therefore each rotation is possible. Figure 1(a) shows an example for $N = 8$.

3.2 Shuffling

We require a *rearrangeable* permutation network, i.e. one that is capable of implementing all possible permutations of its input. Since there are $N!$ possible outputs, at least $\log_2 N! = \Omega(N \log N)$ switches are required. A number of networks meet this bound for example the Waksman network [Wak68] consists of $N \log_2 N - N + 1$ switches. For convenience we will use the slightly simpler Beneš network [Ben64] which consists of a butterfly network composed with a reflected butterfly network, where the middle layer is shared. Note that the network has $\Delta = 2 \log_2 N - 1, W = N/2$. Figure 1(b) shows an example for $N = 8$.

3.3 Biased Networks

Abe and Hoshino observed that setting each switch uniformly and independently in most permutation networks leads to a biased distribution over Σ_N [AH01]. For example, in the Beneš network, there are $2^{(N/2)(\log_2 N-1)}$ switch settings that produce the identity permutation, while other permutations result from only one switch setting. This issue cannot be avoided in Protocol 1 leading to biased output of the mixnet. However we provide some results that suggest that for certain applications the protocol may be acceptable.

Definition 4. *Suppose Ψ is a permutation network. Let $k \leq N$ be a positive integer. Let C_k and P_k be the set of ordered (resp. unordered) k-tuples whose elements are drawn without replacement from $1, \dots, N$. For $\boldsymbol{t} \in C_k$, let $C_{\boldsymbol{t}}$ and $P_{\boldsymbol{t}}$ be the distributions of $\{\Psi(t_1), \dots, \Psi(t_k)\}$ and $(\Psi(t_1), \dots, \Psi(t_k))$ respectively, when all switches are set uniformly at random. The bias over ordered (resp. unordered) k-tuples of Ψ is*

$$\epsilon_{C_k}(N) = \max_{\boldsymbol{t} \in C_k} \|C_{\boldsymbol{t}} - U(C_k)\|, \quad \epsilon_{P_k}(N) = \max_{\boldsymbol{t} \in C_k} \|P_{\boldsymbol{t}} - U(P_k)\|$$

where U is the uniform distribution and $\| \cdot - \cdot \|$ denotes the statistical distance.

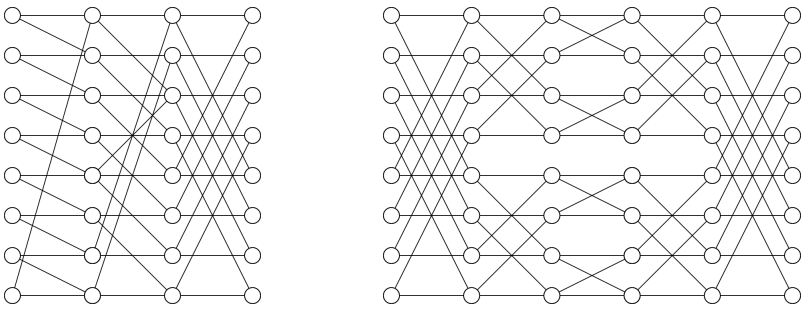
Proposition 1. *The bias over 1-tuples of the Beneš network is* 0.

Proof. It is well-known that the Butterfly network sends any input to each output with probability $1/N$ when set uniformly. This property is maintained when the network is composed with its reflection.

Theorem 1 (Lemma 4.2 [CKKK01]). *One can construct a permutation network of depth $O(\log^4 N)$ with bias over ordered $N/\log^2 N$-tuples in $O(1/N^2)$.*

Theorem 2 (Corollary 1.10 [CKKK01]). *There are permutation networks of depth $O(\log^2 N)$ with bias over unordered N-tuples in $O(1/N)$.*

Proposition 1 guarantees the privacy of any input in the absence of information about the images of other inputs. On the other hand, the network in Theorem 1 guarantees that the images of any $k \leq N/\log^2 N$ inputs cannot be determined except with low probability. Unfortunately privacy is only guaranteed *between* ordered k-tuples, for example it is possible that the order of inputs is maintained. For applications where there is a lot of redundancy in the inputs, such as first-past-the post voting, this level of bias may be acceptable. Theorem 2 is unfortunately non-constructive, but states that efficient networks with small bias exist.



(a) Barrel Shifter. (b) Beneš.

Fig 1. Permutation Networks.

4 Obfuscation of a Paillier Shuffle

In this section we show how to obfuscate Paillier shuffles via permutation networks. A key property we use is that the Damgård-Jurik cryptosystem supports *nested* homomorphic encryption. Let $\mathcal{E}_{i,n} : \mathcal{M}_{i,n} \times \mathcal{R}_n \to \mathsf{C}_{i,n}$ denote the i^{th} generalised Paillier encryption, where $\mathcal{M}_{i,n}$ and $\mathsf{C}_{i,n}$ are the message and ciphertext spaces.[1] Similiarly define re-encryption $\mathcal{RE}_{i,n} : \mathsf{C}_{i,n} \times \mathcal{R}_n \to \mathsf{C}_{i,n}$. Then $\mathsf{C}_{i,n} \subseteq \mathcal{M}_{i+1,n}$ for all $i \geq 1$ and additionally $\mathcal{E}_{i,n}(m,r) \otimes \mathcal{E}_{j,n}(m',r') = \mathcal{E}_{j,n}(\mathcal{E}_{i,n}(m,r) \times m', \mathcal{E}_{i,n}(m,r) \otimes r')$ for all $m \in \mathsf{C}_{i,n}, m' \in \mathcal{M}_{j,n}, r, r' \in \mathcal{R}_n$. The latter property appears to have been first observed by Lipmaa [Lip05] who used it in a recursive fashion to develop an efficient 1-out-of-n computationally private information retrieval (CPIR) protocol. Adida and Wikström in [AW07a] independently noticed this fact and used it define a form of homomorphic matrix multiplication.

4.1 Matrix Notation

Let $\bar{\mathcal{E}}_{i,n}$ and $\bar{\mathcal{RE}}_{i,n}$ be encryption and re-encryption defined for matrices of inputs. Let $\circ$ denote point-wise matrix multiplication and $\otimes$ denote point-wise

[1] Note that $\mathcal{R}_n$ is actually equivalent to $\mathsf{C}_{0,n}$.

matrix exponentiation, note that like ciphertext exponentiation the exponent matrix is written on the left. Variables in lower case will generally denote vectors, while variables in upper case will denote matrices.

Definition 5 (Homomorphic Matrix Multiplication [AW07a]). *Suppose* $d = (d_l) \in \mathsf{C}_{i,n}^{1\times N}$ *and* $C = (c_{lm}) \in \mathsf{C}_{j,n}^{N\times N}$. *Then* $d \star C \triangleq \left(\prod_{l=1}^{N}(c_{lm})^{d_l}\right)_{m=1}^{N}$.

Proposition 2. *Suppose* $d \in \mathsf{C}_{i,n}^{1\times N}$ *and* $C = \bar{\mathcal{E}}_{j,n}(M, R) \in \mathsf{C}_{j,n}^{N\times N}$. *Then* $d\star C = \bar{\mathcal{E}}_{j,n}(d \times M, d \star R)$.

Proof. The proof follows easily from the homomorphic properties of $\mathcal{E}_{j,n}$.

4.2 Obfuscation of Damgård-Jurik Shuffles

The main idea behind standard Paillier obfuscation is to represent the shuffle as a *modified* permutation matrix where the ones are replaced by re-encryption factors and then encrypt it (see Definition 9 [AW07a]). A straightforward generalisation allows one to obfuscate Damgård-Jurik shuffles of arbitrary degree (Proposition 3). Moreover it turns out that homomorphic matrix multiplication actually *composes* obfuscated shuffles, although this fact was not noted in [AW07a]. The composition of two obfuscated shuffles multiplies both underlying permutations, but re-encrypts using the re-encryption factors of the first shuffle. This is formalised in Lemma 1.

Definition 6. *Suppose* $\Lambda^\pi = (\lambda_{lm}^\pi)$ *is a permutation matrix. Let* $a = (a_l) \in \mathcal{M}_{i,n}^{1\times N}$ *and* $r = (r_l) \in \mathcal{R}_n^{1\times N}$ *be vectors. Then* $P_i^\pi[a, r] \triangleq (\lambda_{lm}^\pi \mathcal{E}_{i,n}(a_l, r_l))$.

Definition 7. *The* i^{th} *generalised Paillier shuffle is a functionality* $\mathcal{PS}_i = \{\mathcal{PS}_{i,N(\kappa),\kappa}\}_{\kappa\in\mathbb{N}}$, *where* $N(\cdot)$ *is a polynomially bounded and polynomially computable function, such that for every* $\kappa \in \mathbb{N}, \mathcal{PS}_{i,N(\kappa),\kappa} = \{PS_i[\pi, r]\}_{\pi\in\Sigma_{N(\kappa)}, r\in(\{0,1\}^*)^{N(\kappa)}}$ *and for every* $(n, sk) \in \mathcal{G}(1^\kappa)$ *and* $(c_1, \ldots, c_N) \in \mathsf{C}_{i,n}^{1\times N}$ *the circuit* $PS_i[\pi, r]$ *is defined by*

$$PS_i[\pi, r](n, c_1, \ldots, c_N) = (c_1, \ldots, c_N) \times P_i^\pi[0, r].$$

Proposition 3. *Let* $PS_i[\pi, r] \in \mathcal{PS}_{i,N(\kappa),\kappa}$ *be a shuffle and let* $i < j$. *Then* $C = \bar{\mathcal{E}}_{j,n}(P_i^\pi[0, r], S) : S \in_R {\mathcal{R}_n}^{N\times N}$ *is an obfuscation of* $PS_i[\pi, r]$.

Proof. Let $d \in \mathsf{C}_{i,n}^{1\times N}$. Define $d' = d \star C$. By Proposition 2

$$d' = d \star \bar{\mathcal{E}}_{j,n}(P_i^\pi[0, r], S) = \bar{\mathcal{E}}_{j,n}(d \times P_i^\pi[0, r], d \star S) = \bar{\mathcal{E}}_{j,n}(PS_i[\pi, r](d), d \star S).$$

Lemma 1 (Composition). *Suppose* $PS_i[\mu, r] \in \mathcal{PS}_{i,N(\kappa),\kappa}$, $PS_j[\nu, r'] \in \mathcal{PS}_{j,N,\kappa}$ *are shuffles with corresponding obfuscations* $C_i^\mu \triangleq \bar{\mathcal{E}}_{i+1,n}(P_i^\mu[0, r], S)$, $C_j^\nu \triangleq \bar{\mathcal{E}}_{j+1,n}(P_j^\nu[0, r'], S')$, *where* $i < j$. *Then* (C_i^μ, C_j^ν) *is an obfuscation of* $PS_i[\nu\mu, r]$.

Proof. Let $d \in \mathsf{C}_{i,n}^{1\times N}$. Define $d' = d \star C_i^{\mu}$ and $d'' = d' \star C_j^{\nu}$. By Proposition 2

$$\begin{aligned} d'' &= \bar{\mathcal{E}}_{j+1,n}(PS_j[\nu, r'](d'), d' \star S') \\ &= \bar{\mathcal{E}}_{j+1,n}(PS_j[\nu, r'](\bar{\mathcal{E}}_{i+1,n}(PS_i[\mu, r](d), d \star S)), d' \star S') \\ &= \bar{\mathcal{E}}_{j+1,n}(\bar{\mathcal{E}}_{i+1,n}(PS_i[\mu, r](d), d \star S + r'^{(j-i)}) \times \Lambda^{\nu}, d' \star S') \\ &= \bar{\mathcal{E}}_{j+1,n}(\bar{\mathcal{E}}_{i+1,n}(PS_i[\nu\mu, r](d), \nu(d \star S + r'^{(j-i)})), d' \star S'). \end{aligned}$$

4.3 Obfuscation of Shuffle Networks

The composition lemma implies that a sequence of obfuscated shuffles of arbitrary length may be composed by multiplication, the result is an obfuscated shuffle that composes *all* permutations but inherits re-encryption factors from only the *first* shuffle in the sequence. Therefore it is possible to obfuscate the set of shuffles induced by the layers of a re-encryption permutation network [Abe99,JJ99] and compose them, provided that the i^{th} layer is lifted to accept i^{th} generalised Paillier ciphertexts (Proposition 4). We further observe that each layer can be obfuscated using only $O(N)$ ciphertexts, by decomposing the corresponding permutation into switches (Proposition 5). Combining these observations yields an efficient obfuscator of an arbitrary shuffle (Definition 8). We prove that our obfuscator is semantically secure if the network has polylogarithmic depth and the DCRA holds (Theorem 3).

Proposition 4. *Let Ψ be a re-encryption permutation network with state $\pi \stackrel{\Psi}{=} \pi_\Delta \dots \pi_1$. Suppose that $PS_1[\pi_1, r^{(1)}] \in \mathcal{PS}_{1,N(\kappa),\kappa}, \dots, PS_\Delta[\pi_\Delta, r^{(\Delta)}] \in \mathcal{PS}_{\Delta,N(\kappa),\kappa}$ is the sequence of shuffles corresponding to the layers of Ψ. Let $\{C_i^{\pi_i} = \bar{\mathcal{E}}_{i+1,n}(P_i^{\pi_i}[0, r^{(i)}], S_i) : S_i \in \mathcal{R}_n^{N\times N}\}$ be obfuscations. Then $(C_1^{\pi_1}, \dots, C_\Delta^{\pi_\Delta})$ is an obfuscation of $PS_1[\pi, r^{(1)}]$.*

Proof. The result follows from recursive application of Lemma 1.

Proposition 5. *Let $PS_i[\pi_i, r^{(i)}]$ and $C_i^{\pi_i}$ be defined as in Proposition 4 and let $A^{(i)}$ be the adjacency matrix of the i^{th} layer of Ψ. Then $C_i'^{\pi_i} = A^{(i)} \otimes C_i^{\pi_i}$ is also an obfuscation of $PS_i[\pi_i, r^{(i)}]$.*

Proof. Observe that by Equation (1), Definition 3, $A^{(i)} \circ \Lambda^{\pi_i} = \Lambda^{\pi_i}$. By the homomorphic properties of ciphertext exponentiation

$$\begin{aligned} A^{(i)} \otimes C_i^{\pi_i} &= A^{(i)} \otimes \bar{\mathcal{E}}_{i+1,n}(P_i^{\pi_i}[0, r^{(i)}], S_i) \\ &= \bar{\mathcal{E}}_{i+1,n}(A^{(i)} \circ P_i^{\pi_i}[0, r^{(i)}], A^{(i)} \otimes S_i) \\ &= \bar{\mathcal{E}}_{i+1,n}(P_i^{\pi_i}[0, r^{(i)}], A^{(i)} \otimes S_i). \end{aligned}$$

Note that matrix $A^{(i)}$ is zero except for the co-ordinates which correspond to input and output nodes in the i^{th} layer which are linked by switch. It follows that $C_i'^{\pi_i}$ is a matrix which consists of only $2N$ non-trivial ciphertexts.

Definition 8. *Let Ψ be a rearrangeable permutation network of depth Δ. The obfuscator $\mathcal{O}_\Psi$ for the Paillier shuffle $\mathcal{PS}_1$ takes as input the tuple $(1^\kappa, n, d, PS_1[\pi, r])$, where $(n, sk) \in \mathcal{G}(1^\kappa)$ and $PS_1[\pi, r] \in \mathcal{PS}_{1,N(\kappa),\kappa}$. It computes $\pi \overset{\Psi}{=} \pi_\Delta \dots \pi_1$ and generates shuffles $PS_1[\pi_1, r^{(1)}] \in \mathcal{PS}_{1,N(\kappa),\kappa}, \dots, PS_\Delta[\pi_\Delta, r^{(\Delta)}] \in \mathcal{PS}_{\Delta,N(\kappa),\kappa}$ such that $r^{(1)} = r$ and $r^{(2)}, \dots, r^{(N)} \in_R \mathcal{R}_n^{1\times N}$. It produces obfuscations $\{{C_i}^{\pi_i} = A^{(i)} \otimes \bar{\mathcal{E}}_{i+1,n}(P^{\pi_i}[0, r^{(i)}], S_i) : S_i \in \mathcal{R}_n^{N\times N}\}_{i=1}^{\Delta}$. It outputs a circuit with hard-coded ${C_1}^{\pi_1}, \dots {C_\Delta}^{\pi_\Delta}$ that, on input $d \in \mathsf{C}_{1,n}^{1\times N}$ outputs $d' = d \star {C_1}^{\pi_1} \star \dots \star {C_\Delta}^{\pi_\Delta} \in \mathsf{C}_{\Delta+1,n}$.*

Theorem 3. *The obfuscator $\mathcal{O}_\Psi$ is polynomially indistinguishable (Definition 4 [AW07a]) if Ψ has polylogarithmic depth and the DCRA holds.*

5 Distributed Sampling and Obfuscation of a Shuffle

We construct a distributed protocol for sampling and obfuscating a shuffle via an arbitrary permutation network. Suppose that mix servers $\mathcal{M}_1 - \mathcal{M}_k$ sample and obfuscate the shuffle. Denote the switch at position (i, j) in Ψ by $\chi_{i,j}$. Recall that the permutation in the i^{th} layer of a permutation network may be written as a product of the switches which are set to true, i.e. $\pi_i = \sigma_{i,1}^{b_{i,1}} \dots \sigma_{i,W}^{b_{i,W}}$. To ensure that the state of the permutation network is set uniformly at random, every mix flips the state of $\chi_{i,j}$ at random hence $b_{i,j} = 1$ with probability $1/2$. In practice $\chi_{i,j}$ is simply a permutation matrix of encrypted control bits, hiding $\sigma_{i,j}^{b_{i,j}}$. When the matrices $\chi_{i,1}, \dots, \chi_{i,W}$ are superimposed, they form the permutation matrix C_i of the shuffle π_i. Thus the obfuscated shuffle is the tuple $(C_1, \dots, C_\Delta)$.

Protocol 1 (Sampling and Obfuscation of a Shuffle).
COMMON INPUT: A Paillier public key n, integer N and permutation network Ψ of dimension $\Delta \times W$.
Mix server $\mathcal{M}_I$ proceeds as follows.

1. For $i = 1, \dots, \Delta$ do:
   (a) Generate $N \times N$ matrix C_i whose entries are all initially $\mathcal{E}_{i+1,n}(0, 0^*)$.
   (b) For $j = 1, \dots, W$ do:
      i. Generate N' double encrypted zeroes of the form $\mathcal{E}_{i+1,n}(\mathcal{E}_{i,n}(0))$ in a distributed way using Protocol 3 [AW07a]. Denote these $(c_{i,j}^{(1)}, \dots, c_{i,j}^{(N')})$.
      ii. Form the matrix:

$$\chi_{i,j}^{(0)} = \begin{pmatrix} c_{i,j}^{(1)} & , \dots , & c_{i,j}^{(N')} \\ \mathcal{E}_{i+1,n}(0, 0^*) & , \dots , & \mathcal{E}_{i+1,n}(0, 0^*) \end{pmatrix}.$$

      iii. For $l = 1, \dots, k$ do:

– If $l = I$, permute the rows of $\chi_{i,j}^{(l-1)}$ with $\mu_{i,j}^{(l)} \in_R \Sigma_2$ respectively and re-encrypt them with randomness $r_i^{(l)} \in_R \mathcal{R}_n^{2\times N'}$ publishing matrix

$$\chi_{i,j}^{(l)} = \bar{\mathcal{RE}}_{i+1,n}(\mu_{i,j}^{(l)}(\chi_{i,j}^{(l-1)}), r_i^{(l)}).$$

– If $l \neq I$, verify that the above equation holds.

iv. Suppose $V_{i,j} = \{l_1, \ldots, l_{N'}\}$. Update matrix C_i:

$$C_i \begin{pmatrix} l_1, l_1 & \ldots, & l_{N'}, l_{N'} \\ l_1, \sigma_{i,j}(l_1) & \ldots, & l_{N'}, \sigma_{i,j}(l_{N'}) \end{pmatrix} \leftarrow \chi_{i,j}^{(k)}.$$

2. Output $(C_1, \ldots, C_\Delta)$.

6 Mixnet Properties

We analyse the mixnet which result from mix servers generating an obfuscated shuffle according to Protocol 1, evaluating it upon input and requesting that a threshold number of decryption servers decrypt the output. Note that $N(\Delta{+}1) = O(N \log N)$ threshold decryptions are required to recover the input messages.

6.1 Privacy

We assume the existence of at least one honest mix in Protocol 1, hence the obfuscated shuffle should be identically distributed to the output of a trusted party running obfuscator $\mathcal{O}_\Psi$ (Definition 8), albeit according to a biased permutation distribution, namely that formed by setting the network uniformly at random. Therefore the security of the mixnet follows from the following theorem.

Theorem 4. *Suppose the DCRA holds. Let $(\mathcal{CS}^{pai}, S, (\mathcal{P}, \mathcal{V}))$ be the verifiable shuffle which results from a trusted party obfuscating a random Paillier shuffle according to Definition 8, evaluating it upon input and then revealing (and proving correct) each layer of intermediate decryptions. Then $(\mathcal{CS}^{pai}, S, (\mathcal{P}, \mathcal{V}))$ is IND-CPA$_S$ secure.*

We note that a weakness of the IND-CPA$_S$ model is that it does not guarantee that all information usable by an attacker remains hidden when the mixed ciphertexts are finally opened. In particular an attacker will at least know their own output and may combine this with knowledge of the bias to infer other outputs. Therefore analysis of a realisable ideal (biased) mixnet in the universally composable framework [Can01] is desirable but unfortunately beyond the scope of this paper.

Remark 1. Alternatively it is possible to construct an unbiased mixnet as follows. The I^{th} mix samples a shuffle from $\mathcal{PS}_{(I-1)\Delta+1}$ at random and obfuscates it by setting the state of the Beneš network accordingly and applying the obfuscator in Definition 8. The obfuscated shuffles are then homomorphically multiplied with the input. Note, however, this approach incurs an overhead of $k^{3.5}$ (see Section 6.2) thus is only practicable for small k.

6.2 Complexity

The expansion factor of the mix-net is $1/\Delta$. We count the effective number of multiplications modulo n for each stage of the mixing process, and compare them to [AW07a]. We assume that multiplication modulo n^s is $s^{1.5}$ times as costly as multiplication modulo n, and that exponentiation is performed by repeated squaring. This implies complexity proportional to $\Delta^{3.5}$. Note that κ_c is a parameter of Protocol 2 [AW07a] and satisfies $\kappa_c \ll \kappa$.

	Sample & Obfuscate	Prove	Evaluate	Decrypt
[AW07a] (Shuffle)	$O(N^2\kappa)$	$O((N^2 + N\kappa_c)\kappa\ )$	$O(N^2\kappa)$	$O(N \log N\kappa)$
Proposed (Shuffle & Rotate)	$O(N \log^{3.5} N\kappa)$	$O(N \log^{3.5} N\kappa_c\kappa)$	$O(N \log^{3.5} N\kappa)$	$O(N \log^{3.5} N (\log N + \kappa))$

An implementation using GMP [Gra12] suggests that our scheme is faster when $N > 1200$.

6.3 Parallelisation

Generating an obfuscated shuffle makes use of a private mixnet, therefore parallelisation is limited to that within individual mixes, of course the verification of each mix's shuffle proofs can be distributed over other mixes or the public. The evaluation of the shuffle is public, though, hence can be safely parallelised over arbitrarily many parties. The most obvious parallelisation has k processors evaluate $O(N/k)$ switches at each layer of the network, resulting in $O(\log N)$ parallel steps. Thus when $k \approx N$ it is possible to mix votes in polylogarithmic time.

7 Conclusion

We have presented a more efficient method of obfuscating a Paillier shuffle based upon re-encryption permutation networks. An interesting further direction is to investigate to what extent it is possible to distribute the sampling and obfuscation of a shuffle over a variable number of parties, without incurring a prohibitive loss in efficiency. Such a protocol could conceivably allow voters to directly contribute to the anonymisation of their votes without any assistance from third parties.

References

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '99, pages 258–273. Springer-Verlag, 1999.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 317–324. Springer-Verlag, 2001.

[AW07a] Ben Adida and Douglas Wikström. How to shuffle in public. In *Proceedings of the 4th conference on Theory of cryptography*, TCC'07, pages 555–574, 2007.

[AW07b] Ben Adida and Douglas Wikström. Offline/online mixing. In Christian Cachin et al., editor, *34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *Lecture Notes in Computer Science*, pages 484–495. Springer/ Heidelberg, 2007.

[Ben64] V. E. Beneš. Permutation groups, complexes, and rearrangeable connecting networks. *The Bell System Technical Journal*, 43(4):1619–1640, 1964.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer-Verlag, 2005.

[BGR98] Mihir Bellare, Juan Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *Advances in Cryptology EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer Berlin / Heidelberg, 1998.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, FOCS '01, pages 136–, Washington, DC, USA, 2001. IEEE Computer Society.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24:84–90, February 1981.

[CKKK01] A. Czumaj, P. Kanarek, M. Kutyłowski, and Loryś K. Switching networks for generating random permutations. In *Switching Networks: Recent Advances*, volume 5 of *Network Theory and Applications*, pages 25–61. Kluwer Academic Publishers, 2001.

[dHSŠV09] Sebastiaan de Hoogh, Berry Schoenmakers, Boris Škorić, and José Villegas. Verifiable rotation of homomorphic encryptions. In *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, pages 393–410, Berlin, Heidelberg, 2009. Springer-Verlag.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 119–136. Springer-Verlag, 2001.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 368–387. Springer-Verlag, 2001.

[Gjø05] Kristian Gjøsteen. Homomorphic cryptosystems based on subgroup membership problems. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 314–327. Springer Berlin / Heidelberg, 2005.

[Gra12] T. Granlund. Gnu multiple precision arithmetic library. `http://gmplib.org/`, 2012.

[GZB+02] Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '02, pages 451–465. Springer-Verlag, 2002.

[JJ99] Markus Jakobsson and Ari Juels. Millimix: Mixing in small batches. Technical report, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), 1999.

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *ISC '05*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer-Verlag, 2005.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pages 116–125. ACM, 2001.

[NSNK04] Lan Nguyen, Rei Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 61–75. Springer Berlin / Heidelberg, 2004.

[OS07] Rafail Ostrovsky and William E. Skeith. Private searching on streaming data. *Journal of Cryptology*, 20:397–430, 2007.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin / Heidelberg, 1999.

[PIK94] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 248–259. Springer-Verlag, 1994.

[RW04] Michael K. Reiter and Xiaofeng Wang. Fragile mixing. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 227–235, New York, NY, USA, 2004. ACM.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth. In *Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'95, pages 393–403. Springer-Verlag, 1995.

[Wak68] Abraham Waksman. A permutation network. *J. ACM*, 15:159–163, 1968.

[Wik09] Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Gonzlez Nieto, editors, *Information Security and Privacy*, volume 5594 of *Lecture Notes in Computer Science*, pages 407–421. Springer Berlin / Heidelberg, 2009.

A Shuffle Privacy

A *verifiable shuffle* is a tuple $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ where $\mathcal{RP}$ is a public-key cryptosystem with a re-encryption algorithm, S is a PPT algorithm that shuffles input ciphertexts and $(\mathcal{P}, \mathcal{V})$ is a proof system that proves the existence of re-encryption factors linking input and output ciphertexts [NSNK04]. One definition for security of a verifiable shuffle is *indistinguishability under chosen permutation attacks* (IND-CPA$_S$) and is an extension of classical IND-CPA security for cryptosystems. A related definition is *semantic privacy under chosen permutation attacks* (SP-CPA$_S$) which specifies that whatever can be computed after the shuffle execution could be computed using only prior information. Nguyen et al. [NSNK04] prove that the two notions are equivalent.

Definition 9 (Indistinguishability under Chosen Permutation Attacks). *Let*
$(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ *be a verifiable shuffle and* $A = (A_1, A_2)$ *be a pair of PPT algorithms. Let* $t \in \{0,1\}^{poly(\kappa)}$. *Define*

Experiment $Exp^{IND-CPA_S-b}_{A,(\mathcal{RP},S,(\mathcal{P},\mathcal{V}))}(\kappa, t)$:

$$(pk, sk) \leftarrow \mathcal{G}(1^\kappa); ((\pi_{(0)}, \pi_{(1)}, L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}}), \delta) \leftarrow A_1(pk, t);$$
$$L_{out} \leftarrow S(pk, L_{in}, \pi_{(b)});$$
$$o_{(b)} \leftarrow (L_{out}, VIEW_{\mathcal{V}}^{\mathcal{P}}(pk, L_{in}, L_{out}), L_{in}, L_{in}^{(p)}, C_{E_{pk}}^{L_{in}^{(p)}});$$
$$v \leftarrow A_2(\delta, o_{(b)})$$
return v

and let

$$Adv(A) = \max_{t \in \{0,1\}^{poly(\kappa)}} | \Pr[Exp^{IND-CPA_S-1}_{A,(\mathcal{RP},S,(\mathcal{P},\mathcal{V}))}(\kappa, t) = 1] - \Pr[Exp^{IND-CPA_S-0}_{A,(\mathcal{RP},S,(\mathcal{P},\mathcal{V}))}(\kappa, t) = 1] |$$

Then $(\mathcal{RP}, S, (\mathcal{P}, \mathcal{V}))$ *satisfies indistinguishability under chosen plaintext attacks (IND-CPA*$_S$*) if for any A with polynomially bounded auxiliary input,* $Adv(A)$ *is negligible.*

B Proofs

B.1 Proof of Theorem 3

Proof. Suppose there is an adversary A against the the obfuscator $\mathcal{O}_\Psi$ with advantage $\epsilon(\kappa)$. Let A' be an adversary in the IND-CPA experiment for $\mathcal{E}_{\Delta+1,n}$. When A outputs challenge circuits PS^0, PS^1, adversary A' generates sequences $\{P_i^0\}_{i=1}^{\Delta}$, $\{P_i^1\}_{i=1}^{\Delta}$ as would be generated by $\mathcal{O}_\Psi$. When the IND-CPA experiment returns $S^b = \{\bar{\mathcal{E}}_{\Delta+1,n}(P_i^b)\}_{i=1}^{\Delta}$, A' produces $S_{red}^b = \{A^{(i)} \otimes \bar{\mathcal{E}}_{\Delta+1,n}(P_i^b)$

$(\text{mod } n^{i+1})\}_{i=1}^{\Delta}$ and passes it to A, outputting 1 iff A does. Since S^b_{red} is identically distributed to $\mathcal{O}_{\Psi}(PS^b)$, the advantage of A' in the IND-CPA experiment is identical to that of A in distinguishing the obfuscated shuffles. Then by the remarks in Section 2.3, A' has advantage at least $\epsilon(\kappa)/(\Delta+1) = \epsilon(\kappa)/O(\log^c N)$ in breaking the DCRA. Since $N < 2^\kappa$, $\log^c N < \kappa^c$ thus $\epsilon(\kappa)/\kappa^c$ must be negligible if the DCRA holds. Then $\epsilon(\kappa)$ is also negligible and polynomial indistinguishability of $\mathcal{O}_{\Psi}$ follows.

B.2 Proof of Theorem 4

Proof. The proof is via a hybrid argument. First define $\bar{\mathcal{D}}_{i,n}$ to be the vector form of $\mathcal{D}_{i,n}$, and define $\bar{\mathcal{D}}_{j:i,n} = \bar{\mathcal{D}}_{j,n} \circ \ldots \circ \bar{\mathcal{D}}_{i,n}$ for $j > i$. Suppose $\mathit{\Pi}_{(b)}$ is the distribution of the challenge $o_{(b)}$ in $Exp^{IND-CPA_S-b}_{A,\ (\mathcal{CS}^{pai},S,(\mathcal{P},\mathcal{V}))}$, where $A = (A_1, A_2)$ is an adversary. Define the following hybrid distributions:

$$
\begin{aligned}
\mathit{\Pi}_{(b)} = \Bigg(& L^{(p)}_{in}, C^{L^{(p)}_{in}}_{\mathcal{E}_{1,n}}, L_{in}, L_{eval}, \bar{\mathcal{D}}_{\Delta+1,n}(L_{eval}), \ldots, \bar{\mathcal{D}}_{\Delta+1:2,n}(L_{eval}) \Bigg) : \\
& (L_{in}, L^{(p)}_{in}, C^{L^{(p)}_{in}}_{\mathcal{E}_{1,n}}, \pi_{(0)}, \pi_{(1)}) \leftarrow A_1(n,t), \pi_{(b)} \overset{\Psi}{=} \pi_{\Delta} \ldots \pi_1, \\
& C_i = A^{(i)} \otimes \bar{\mathcal{E}}_{i+1,n}(P^{\pi_i}[0, r^{(i)}], S_i) : r^{(i)} \in \mathcal{R}_n^{1\times N}, S_i \in \mathcal{R}_n^{N\times N}, \\
& L_{eval} \leftarrow L_{in} \star \prod_{i=1}^{\Delta} C_i.
\end{aligned}
$$

$$
\begin{aligned}
\hat{\mathit{\Pi}}_{(b)} = \Bigg(& L^{(p)}_{in}, C^{L^{(p)}_{in}}_{\mathcal{E}_{1,n}}, L_{in}, L_{eval}, \bar{\mathcal{D}}_{\Delta+1,n}(L_{eval}), \ldots, \bar{\mathcal{D}}_{\Delta+1:2,n}(L_{eval}) \Bigg) : \\
& (L_{in}, L^{(p)}_{in}, C^{L^{(p)}_{in}}_{\mathcal{E}_{1,n}}, \pi_{(0)}, \pi_{(1)}) \leftarrow A_1(n,t), \pi_{(b)} \overset{\Psi}{=} \pi_{\Delta} \ldots \pi_1, \\
& C_i = A^{(i)} \otimes \bar{\mathcal{E}}_{i+1,n}(P^{\pi_i}[x^{(i)}, r^{(i)}], S_i) : x^{(i)} \in \mathcal{M}_{i,n}^{1\times N}, r^{(i)} \in \mathcal{R}_n^{1\times N}, S_i \in \mathcal{R}_n^{N\times N}, \\
& L_{eval} \leftarrow L_{in} \star \prod_{i=1}^{\Delta} C_i.
\end{aligned}
$$

$$
\begin{aligned}
\tilde{\mathit{\Pi}}_{\Psi} = \Bigg(& L^{(p)}_{in}, C^{L^{(p)}_{in}}_{\mathcal{E}_{1,n}}, L_{in}, L_{eval}, y_{\Delta}, \ldots, y_1 \Bigg) : \\
& (L_{in}, L^{(p)}_{in}, C^{L^{(p)}_{in}}_{\mathcal{E}_{1,n}}, \pi_{(0)}, \pi_{(1)}) \leftarrow A_1(n,t), \\
& C_i = A^{(i)} \otimes \bar{\mathcal{E}}_{i+1,n}(\mathbf{0}, S_i) : S_i \in \mathcal{R}_n^{N\times N}, \\
& L_{eval} \leftarrow L_{in} \star \prod_{i=1}^{\Delta} C_i, y_{\Delta} \in_R \mathsf{C}_{\Delta,n}^{1\times N}, \ldots, y_1 \in_R \mathsf{C}_{1,n}^{1\times N}.
\end{aligned}
$$

We are required to show that the distributions $\mathit{\Pi}_{(0)}$ and $\mathit{\Pi}_{(1)}$ are computationally indistinguishable. By transitivity it suffices to prove that $\mathit{\Pi}_{(b)}$ and $\tilde{\mathit{\Pi}}_{(b)}$ are computationally indistinguishable for each b. However this in turn follows from combining Lemmas 2 and 3.

Lemma 2. *Suppose the DCRA holds. Then the distributions $\Pi_{(b)}$ and $\hat{\Pi}_{(b)}$ are computationally indistinguishable.*

Proof. Suppose there is an adversary $A = (A_1, A_2)$ against $\Pi_{(b)}$ and $\hat{\Pi}_{(b)}$ with advantage $\epsilon(\kappa)$. Let A' be the adversary that distinguishes a ciphertext c_Δ that is a random encryption of 0 or a uniform message under $\mathcal{E}_{\Delta,n}$ as follows.

Adversary $A'(c_\Delta, n, t)$

1. Set $(L_{in}, L_{in}^{(p)}, C_{\mathcal{E}_{1,n}}^{L_{in}^{(p)}}, \pi_{(0)}, \pi_{(1)}) \leftarrow A_1(n, t)$
2. Compute $\pi_{(b)} \stackrel{\Psi}{=} \pi_\Delta \dots \pi_1$.
3. For $i = 1, \dots, \Delta$ do:
   (a) Compute $c_i \equiv c_\Delta \bmod n^{i+1}$.
   (b) Apply Protocol 2 on (c_i, π_i) to generate modified permutation matrix $P_i^{\pi_i}[a^{(i)}, r^{(i)}]$.
   (c) Generate $C_i = A^{(i)} \otimes \bar{\mathcal{E}}_{i+1,n}(P_i^{\pi_i}[a^{(i)}, r^{(i)}], S_i) : S_i \in_R \mathcal{R}_n^{N \times N}$.
4. Set $L_{eval} \leftarrow L_{in} \star \prod_{i=1}^{\Delta} C_i$.
5. For $i = 1, \dots, \Delta$ do:
   (a) Compute $\bar{\mathcal{D}}_{\Delta+1:i+1,n}(L_{eval}) = \pi_\Delta \dots \pi_{i+1}\ (l_{i-1} \times P_i^{\pi_i}[a^{(i)}, r^{(i)}])$.
6. Run A_2 on

$$o_{(b)} = (L_{in}^{(p)}, C_{\mathcal{E}_{1,n}}^{L_{in}^{(p)}}, L_{in}, L_{eval}, \bar{\mathcal{D}}_{\Delta+1,n}(L_{eval}), \dots, \bar{\mathcal{D}}_{\Delta+1:2,n}(L_{eval}))$$

   and output 1 iff A_2 does.

Clearly A' has advantage $\epsilon(\kappa)$ in breaking the semantic security of $\mathcal{E}_{\Delta,n}$. Since $\Delta = O(\log^c N)$ the DCRA implies $\epsilon(\kappa)$ is negligible, hence the lemma follows.

Lemma 3. *Suppose the DCRA holds. Then the distributions $\hat{\Pi}_{(b)}$ and $\tilde{\Pi}$ are computationally indistinguishable.*

Proof. Suppose there is an adversary $A = (A_1, A_2)$ against $\hat{\Pi}_{(b)}$ and $\tilde{\Pi}$ with advantage $\epsilon(\kappa)$. Let A' be the adversary that distinguishes a ciphertext $c_{\Delta+1}$ that is a random encryption of 0 or 1 under $\mathcal{E}_{\Delta+1}$ as follows.

Adversary $A'(c_{\Delta+1}, n, t)$

1. Set $(L_{in}, L_{in}^{(p)}, C_{\mathcal{E}_{1,n}}^{L_{in}^{(p)}}, \pi_{(0)}, \pi_{(1)}) \leftarrow A_1(n, t)$.
2. Compute $\pi_{(b)} \stackrel{\Psi}{=} \pi_\Delta \dots \pi_1$.
3. For $i = 1, \dots, \Delta$ do:
   (a) Compute $c_i \equiv c_{\Delta+1} \bmod n^{i+2}$.
   (b) Generate modified permutation matrix $P_i^{\pi_i}[x^{(i)}, r^{(i)}]$ where $x^{(i)} \in_R {\mathcal{M}_{i,n}}^{1 \times N}, r^{(i)} \in_R \mathcal{R}_n^{1 \times N}$.
   (c) Apply Protocol 3 on (c_i, π_i) to generate matrix of ciphertexts M_i.
   (d) Generate $C_i = A^{(i)} \otimes (P_i^{\pi_i}[x^{(i)}, r^{(i)}] \otimes M_i \oplus \mathcal{E}_{i,n}(\mathbf{0}, S_i)) : S_i \in_R \mathcal{R}_n^{N \times N}$.
4. Set $L_{eval} \leftarrow L_{in} \star \prod_{i=1}^{\Delta} C_i$.
5. For $i = 1, \dots, \Delta$ do:

(a) Compute $\bar{\mathcal{D}}_{\Delta+1:i+1,n}(L_{eval}) = \pi_\Delta \dots \pi_{i+1}\ (l_{i-1} \times P_i^{\pi_i}[x^{(i)}, r^{(i)}])$.

6. Run A_2 on

$$o_{(b)} = (L_{in}^{(p)}, C_{\mathcal{E}_{1,n}}^{L_{in}^{(p)}}, L_{in}, L_{eval}, \bar{\mathcal{D}}_{\Delta+1,n}(L_{eval}), \dots, \bar{\mathcal{D}}_{\Delta+1:2,n}(L_{eval}))$$

and output 1 iff A_2 does.

Clearly A' has advantage $\epsilon(\kappa)$ in breaking the semantic security of $\mathcal{E}_{\Delta+1,n}$. Since $\Delta = O(\log^c N)$ the DCRA implies $\epsilon(\kappa)$ is negligible, hence the lemma follows.

Protocol 2.

Input Ciphertext c which is a random encryption of 0 or a uniformly chosen message under $\mathcal{E}_i$ and permutation $\pi_i \in \Sigma_N$.

Output Modified permutation matrix $P_i^{\pi_i}[a, r]$ with distribution

$$P_i^{\pi_i}[0, r] : r \in_R \mathcal{R}_n^{1\times N} \text{ if } c \text{ is a random encryption of 0,}$$
$$P_i^{\pi_i}[x, r] : x \in_R {\mathcal{M}_{i,n}}^{1\times N}, r \in_R {\mathcal{R}_n^{1\times N}} \text{ otherwise.}$$

Procedure Use standard amplification to generate N independent copies $c_1, \dots, c_N$ which have the same distribution as c. Replacing the ones in Λ^{π_i} with $\{c_i\}_{i=1}^N$ yields the required distribution.

Protocol 3.

Input Ciphertext c which is a random encryption of 0 or 1 under $\mathcal{E}_{i+1,n}$ and a permutation $\pi_i \in \Sigma_N$.

Output Matrix M_i with distribution

$$\bar{\mathcal{E}}_{i+1,n}(\mathbf{0}, S) : S \in_R \mathcal{R}_n^{N\times N} \text{ if } c \text{ is a random encryption of 0,}$$
$$\bar{\mathcal{E}}_{i+1,n}(\Lambda^{\pi_i}, S) : S \in_R \mathcal{R}_n^{N\times N} \text{ otherwise.}$$

Procedure Use standard amplification to generate N independent copies $c_1, \dots, c_N$ which have the same distribution as c. Replacing the ones in Λ^{π_i} with $\{c_i\}_{i=1}^N$ and the zeroes with random encryptions of zero yields the required distribution.