Offline/Online Mixing

Ben Adida[1]* and Douglas Wikström[2]**

[1] Harvard, Center for Research on Computation and Society, `ben@eecs.harvard.edu`
[2] `douglas@wikstrom.net`

Abstract. We introduce an offline precomputation technique for mix-nets that drastically reduces the amount of online computation needed. Our method can be based on any additively homomorphic cryptosystem and is applicable when the number of senders and the maximal bit-size of messages are relatively small.

1 Introduction

Suppose some senders $S_1, \ldots, S_N$, each with input m_i, want to compute the sorted list $(m_{\pi(1)}, \ldots, m_{\pi(N)})$ while keeping the permutation π secret. A trusted party can provide this service. First, it collects all messages. Then, it shuffles the inputs according to π and outputs the result. A protocol, i.e. a list of machines $M_1, \ldots, M_k$, that emulates this service is called a *mix-net*, and the parties $M_1, \ldots, M_k$ are referred to as *mix-servers*. The assumption is that each sender S_i trusts that a certain fraction of the mix-servers $M_1, \ldots, M_k$ is honest. The notion of a mix-net was introduced by Chaum [8].

There are numerous proposals in the literature for how to construct a secure mix-net, but there are also several attacks. A rigorous definition of security of a mix-net was first given by Abe and Imai [1], though they did not construct a scheme satisfying their construction. Wikström [19] gives the first definition of a universally composable (UC) mix-net, and also the first UC-secure construction. In recent work, Wikström [20] gives a more efficient UC-secure scheme and Wikström and Groth [22] describes an adaptively secure construction.

In this paper we assume that a statically UC-secure mix-net can be constructed, and consider to what extent offline precomputation can be used to reduce the amount of online computation needed during execution.

1.1 Previous Work

General techniques, e.g., precomputation of re-encryption factors, fixed base exponentiation, and simultaneous exponentiation [15], can be used to lower the online computational complexity of most mix-nets in the literature. However, for the known constructions, it seems difficult to use these methods to completely

* Work done while at MIT, funded by the Caltech/MIT Voting Technology Project.
** Work done while at ETH Zürich, Department of Computer Science.

remove the large number of exponentiations needed in the proofs of shuffles used to provide security against active attacks.

We are not aware of any previous work on mix-nets using our approach, but it is inspired by the ground-breaking work on homomorphic election schemes introduced by Cohen[3] and Fischer [9] and further developed in a long line of papers [4, 10, 14].

In recent work [3], we consider a related precomputation technique with connections to public key obfuscation. By comparison, the solution we present here requires an individual key for each sender but is much more efficient. Thus, the two solutions are complementary.

1.2 Our Contributions

We describe a novel precomputation technique for mix-nets based on additively homomorphic cryptosystems such as the Paillier [18] cryptosystem. Although our technique is universally applicable, it only reduces the online complexity in terms of computation and communication when the number of senders and the maximal bit-size of their messages are reasonably small. We also introduce the notion of concatenation-friendly cryptosystems as a separate tool and prove that such schemes can be constructed from any additively homomorphic cryptosystem. Our technique may be of great value in some practical applications where online computational power is a scarce resource and the result is needed quickly.

1.3 Notation

We denote the natural numbers by $\mathbb{N}$, the integers by $\mathbb{Z}$, the integers modulo n by $\mathbb{Z}_n$, the multiplicative group modulo n by $\mathbb{Z}_n^*$, and the subgroup of squares modulo n by SQ_n. We interpret strings as integers in base two when convenient. We write $a\|b$ to denote the concatenation of the two strings a and b. We use PT and PT* to denote the set of polynomial time and non-uniform polynomial time Turing machines respectively, and let κ be the main security parameter. We say that a function $\epsilon(\kappa)$ is negligible if for every constant c and sufficiently large κ it holds that $\epsilon(\kappa) < \kappa^{-c}$. We denote by Sort the algorithm that, on input a list of strings, outputs the same strings in lexicographical order. If pk is the public key of a cryptosystem, we denote by M_{pk}, C_{pk}, and R_{pk} the plaintext space, the ciphertext space, and the randomness space respectively. We state our results using the Universal Composability (UC) framework [7]. We use slightly non-standard notation in that we use an explicit communication model, denoted $\mathcal{C}_{\mathcal{I}}$, that acts as a router between the parties. We refer the reader to [7, 21] for details on this variant of the UC model.

2 Additively Homomorphic Cryptosystems

There are several homomorphic cryptosystems in the literature, but not all are *additively* homomorphic. For our new scheme, we do not require the cryptosys-

[3] In his later work, Cohen published under the name Benaloh.

tem to have efficient decryption for all encrypted messages. More precisely, we use the following definitions.

Definition 1. *A weak cryptosystem* $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ *is a cryptosystem except we do not require that* D *run in polynomial time. If there exists polynomial* $T(\cdot)$ *and* $\kappa_s(\kappa) > 0$ *such that* $\{0,1\}^{\kappa_s} \subset M_{pk}$ *and such that* $\mathsf{D}_{sk}(\mathsf{E}_{pk}(m))$ *outputs* m *in time* $T(\kappa)$ *for every* $(pk, sk) = \mathsf{Kg}(1^\kappa)$ *and* $m \in \{0,1\}^{\kappa_s}$*, we call* $\mathcal{CS}$ *a* κ_s*-cryptosystem.*

Definition 2. *A weak cryptosystem* $\mathcal{CS}$ *is homomorphic if for every* $(pk, sk) = \mathsf{Kg}(1^\kappa)$*:*

1. *The message space* M_{pk} *and the randomizer space* R_{pk} *are additive abelian groups, and the ciphertext space* C_{pk} *is a multiplicative abelian group, and the group operations can be computed in polynomial time given* pk*.*
2. *For every* $m, m' \in M_{pk}$ *and* $r, r' \in R_{pk}$*:* $\mathsf{E}_{pk}(m, r)\mathsf{E}_{pk}(m', r') = \mathsf{E}_{pk}(m + m', r + r')$*.*

Definition 3. *A weak homomorphic cryptosystem* $\mathcal{CS}$ *is said to be additive if, for every* $(pk, sk) = \mathsf{Kg}(1^\kappa)$ *the message space* M_{pk} *is the additive modular group* $\mathbb{Z}_n$ *for some integer* $n > 1$*. In this case we identify the elements of* $\mathbb{Z}_n$ *with their bit-string representations as integers in base two.*

Efficient Examples. The Paillier cryptosystem [18, 11] is additively homomorphic, since $M_{pk} = \mathbb{Z}_n$, $R_{pk} = \mathbb{Z}_n^*$, and $C_{pk} = \mathbb{Z}_{n^2}^*$, where n is the κ-bit modulus contained in the public key pk. Similarly, the Okamoto-Uchiyama cryptosystem [17], a precursor of the Paillier cryptosystem, is additively homomorphic, since $M_{pk} = \mathbb{Z}_p$, $R_{pk} = \mathbb{Z}_n$, and $C_{pk} = \mathbb{Z}_n^*$, where n is the κ-bit modulus contained in the public key pk.

Inefficient Examples. The Goldwasser-Micali cryptosystem [13], when based on quadratic residues, is additively homomorphic, since $M_{pk} = \mathbb{Z}_2$, $R_{pk} = SQ_n$, and C_{pk} is the subset of $\mathbb{Z}_n^*$ with Jacobi symbol 1. This example may be interesting despite its inefficiency, since the quadratic residuosity assumption is considered a very weak assumption. The Boneh-Goh-Nissim cryptosystem [5] can be viewed as an additively homomorphic $O(\kappa)$-cryptosystem. This is both inefficient and based on a very strong assumption, but it may still be interesting in connection with our ideas due to its special algebraic properties.

3 The Basic Idea

Our construction is simple provided that we use an additive homomorphic κ_s-cryptosystem such that $N\kappa_m < \kappa_s$, where N is the maximal number of senders and κ_m is the maximal bit-size of submitted messages.

The idea can be described as follows. Define $B_i = 2^{(i-1)\kappa_m}$ for $i = 1, \ldots, N$. The offline phase produces ciphertexts for the sequence of indexed positions where the inputs will end up, namely $B_1, \ldots, B_N$. Then, still in the offline phase, these ciphertexts are re-randomized and shuffled. Each sender is assigned one such encrypted index to use as his effective public key. The sender uses the additive homomorphic property of the cryptosystem to exponentiate his encrypted index to his plaintext value m_i, thereby creating a ciphertext of the value m_i offset to that sender's bit position (which remains hidden from the sender). The resulting ciphertext is then sent to the bulletin board. When all inputs are submitted, the offline phase ends. Then, they are aggregated using homomorphic addition. The plaintext of the resulting single ciphertext is the concatenation of all submitted messages, with each message at its appropriate offset. The idea is illustrated in Figure 1.

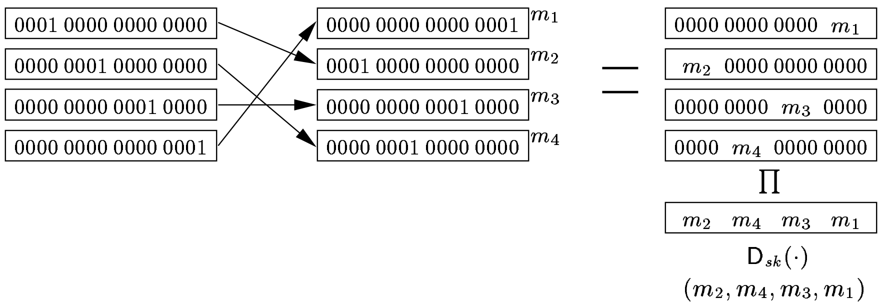

Fig. 1. The trivial ciphertexts are shuffled to produce a new list of re-encrypted and permuted ciphertexts. Then each sender uses its assigned ciphertext as a public key and the result is a new list of ciphertexts, where the messages of the senders are embedded. Finally, the mix-servers take the product of the ciphertexts and decrypt a single ciphertext to find the input messages, but in random order.

In the remainder of the paper we relax the restriction $N\kappa_m \leq \kappa_s$, give a more detailed description, and prove the security of the scheme, but, before we do so we give a more detailed description of the simple case. In the offline-phase, the mix-servers first form the list of trivial encryptions

$$(C_1, \ldots, C_N) = (\mathsf{E}_{pk}(B_1, 0), \ldots, \mathsf{E}_{pk}(B_N, 0)) \enspace .$$

Then, they mix the above list to produce a randomly re-encrypted and permuted list of ciphertext on the form

$$(C'_1, \ldots, C'_N) = (\mathsf{E}_{pk}(B_{\pi(1)}, s_1), \ldots, \mathsf{E}_{pk}(B_{\pi(N)}, s_N)) \enspace .$$

The sender S_i is then assigned the public key $pk_i = C'_i$. To send a message $m_i \in \{0,1\}^{\kappa_m}$, the sender S_i chooses $r_i \in R_{pk}$ randomly, computes the ciphertext

$$c_i = pk_i^{m_i} \mathsf{E}_{pk}(0, r_i)$$

and writes it on the bulletin board. It also proves knowledge of $r_i \in R_{pk}$ and $m_i \in \{0,1\}^{\kappa_m}$ such that the above holds. When the submission phase is over, the mix-servers compute the product $c = \prod_{i=1}^{N} c_i$. Note that we have

$$c = \prod_{i=1}^{N} pk_i^{m_i} \mathsf{E}_{pk}(0, r_i) = \mathsf{E}_{pk}(0, r) \prod_{i=1}^{N} \mathsf{E}_{pk}(B_{\pi(i)}, s_i)^{m_i} = \mathsf{E}_{pk}\left(\sum_{i=1}^{N} B_{\pi(i)} m_i, r' \right)$$

$$= \mathsf{E}_{pk}\left(\sum_{i=1}^{N} B^{\pi(i)-1} m_i, r' \right) = \mathsf{E}_{pk}\left(m_{\pi^{-1}(1)} \| \cdots \| m_{\pi^{-1}(N)}, r' \right) \ ,$$

for $r = \sum_{i=1}^{N} r_i$ and $r' = r + \sum_{i=1}^{N} s_i m_i$, since $m_i \in \{0,1\}^{\kappa_m}$. The mix-servers jointly compute $m'_1 \| \cdots \| m'_N = \mathsf{D}_{sk}(c)$, and output $\mathsf{Sort}(m'_1, \ldots, m'_N)$.

The Relation With Homomorphic Election Schemes. Recall that the idea behind the homomorphic election schemes [9] mentioned in the introduction is to use an additive homomorphic κ_s-cryptosystem and let a sender S_i encode a vote for party j by a ciphertext $c_i = \mathsf{E}_{pk}(M^j)$, where M is an integer larger than the number of senders N. The point is that the plaintext of the ciphertext product $\prod_{i=1}^{N} c_i$ is of the form $\sum_{j=0}^{C-1} a_j M^j$, where a_j is the number of senders that voted for candidate number j. If C is the number of candidates, this approach requires that $C \log N \leq \kappa_s$, but one can increase the number of candidates by using several ciphertexts. In some sense, our approach follows by switching the roles played by candidates and senders.

4 Model and Definitions

We define some ideal functionalities and the notion of concatenation-friendly cryptosystems to allow us to state our results more easily.

4.1 The Ideal Bulletin Board

We assume the existence of an ideal authenticated bulletin board. Each party can write to the bulletin board, nobody can erase anything from the bulletin board, and the messages that appear on the bulletin board are indexed in the order they appear We give a definition in Appendix B.

4.2 The Ideal Mix-Net

We use an ideal mix-net functionality similar to the one in [19]. The only essential difference is that we explicitly allow the adversary to prohibit senders from submitting an input. This makes the ideal functionality more realistic.

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

1. Initialize a list $L = \emptyset$, a database D, a counter $c = 0$, and set $J_S = \emptyset$ and $J_M = \emptyset$.
2. Repeatedly wait for inputs
   - Upon receipt of $(S_i, \texttt{Send}, m_i)$ with $m_i \in \{0,1\}^{\kappa_m}$ and $i \notin J_S$ from $\mathcal{C_I}$, store this tuple in D under the index c, set $c \leftarrow c + 1$, and hand $(\mathcal{S}, S_i, \texttt{Input}, c)$ to $\mathcal{C_I}$.
   - Upon receipt of $(M_j, \texttt{Run})$ from $\mathcal{C_I}$, store $(M_j, \texttt{Run})$ in D under the index c, set $c \leftarrow c + 1$, and hand $(\mathcal{S}, M_j, \texttt{Input}, c)$ to $\mathcal{C_I}$.
   - Upon receipt of $(\mathcal{S}, \texttt{AcceptInput}, c)$ such that something is stored under the index c in D do
     (a) If $(S_i, \texttt{Send}, m_i)$ with $i \notin J_S$ is stored under c, then append m_i to L, set $J_S \leftarrow J_S \cup \{i\}$, and hand $(\mathcal{S}, S_i, \texttt{Send})$ to $\mathcal{C_I}$.
     (b) If $(M_j, \texttt{Run})$ is stored under c, then set $J_M \leftarrow J_M \cup \{j\}$. If $|J_M| > k/2$, then sort the list L lexicographically to form a list L', hand $((\mathcal{S}, M_j, \texttt{Output}, L'), \{(M_l, \texttt{Output}, L')\}_{l=1}^{k})$ to $\mathcal{C_I}$ and ignore further messages. Otherwise, hand $\mathcal{C_I}$ the list $(\mathcal{S}, M_j, \texttt{Run})$.

4.3 The Ideal Mixer

Since our focus in this paper is to minimize the online work needed by the mix-servers and not how to construct a secure mix-net from scratch, we assume the existence of a powerful ideal functionality that allows us to invoke the different phases of a mix-net without going into details. We use this functionality during the offline phase only. Although it is essentially equivalent to an ideal mix-net, we call it a mixer to distinguish it from the ideal mix-net above, and we parameterize it by a cryptosystem. The functionality outputs a public key, waits for a list of ciphertexts to mix, and then finally waits for ciphertexts to decrypt.

Functionality 2 ($\mathcal{CS}$-Mixer). The ideal functionality for a $\mathcal{CS}$-mixer, $\mathcal{F}_{\text{mixer}}$, running with mix-servers $M_1, \ldots, M_k$, senders $S_1, \ldots, S_N$, and ideal adversary $\mathcal{S}$ proceeds as follows

1. Set $J_M = \emptyset$, compute $(pk, sk) = \mathsf{Kg}(1^\kappa)$, and hand $((\mathcal{S}, \texttt{PublicKey}, pk), \{(M_j, \texttt{PublicKey}, pk)\}_{j=1}^{k})$ to $\mathcal{C_I}$.
2. Wait for an input on the form $(M_j, \texttt{Mix}, L_j)$ with $j \notin J_M$ and set $J_M \leftarrow J_M \cup \{j\}$.
   (a) If there is an $L = (c_i)_{i=1}^{N}$ such that $L_j = L$ for more than $k/2$ distinct j, where $c_i \in C_{pk}$, choose $r_i \in R_{pk}$ randomly and compute $L' = (c_{\pi(1)}\mathsf{E}_{pk}(0, r_1), \ldots, c_{\pi(N)}\mathsf{E}_{pk}(0, r_N))$ for a random $\pi \in \Sigma_N$. Then hand $((\mathcal{S}, \texttt{Mixed}, L'), \{(M_j, \texttt{Mixed}, L'\}_{j=1}^{k})$ to $\mathcal{C_I}$, and go to the next step.
   (b) Otherwise hand $(\mathcal{S}, M_j, \texttt{Mix}, L_j)$ to $\mathcal{C_I}$ and wait for another input.
3. Repeatedly wait for messages. Upon receiving $(M_j, \texttt{Decrypt}, c)$ check if c has been received. If so set $J_c \leftarrow J_c \cup \{j\}$. Otherwise initialize $J_c = \emptyset$. If $|J_c| > k/2$, then hand $((\mathcal{S}, \texttt{Decrypted}, c, \mathsf{D}_{sk}(c)), \{(M_j, \texttt{Decrypted}, c, \mathsf{D}_{sk}(c))\}_{i=1}^{k})$ to $\mathcal{C_I}$, and otherwise hand $(\mathcal{S}, M_j, \texttt{Decrypt}, c)$ to $\mathcal{C_I}$.

Proving that this functionality can be realized in an efficient and UC-secure way is beyond the scope of this paper. It can be achieved following [20, 22].

4.4 Ideal Zero-Knowledge Proof of Knowledge of Plaintexts

We assume the existence of an ideal zero-knowledge proof of knowledge for correct encryption. The corresponding relation is defined below.

Definition 4 (Plaintext Knowledge). *Define the relation $\mathcal{R}_{\mathsf{enc}}$ as consisting of the pairs $((pk, pk', c), (m, r))$ such that $c = (pk')^m \mathsf{E}_{pk}(0, r)$ and $m \in \{0,1\}^{\kappa_m}$.*

Functionality 3 (Zero-Knowledge Proof of Knowledge). Let $\mathcal{L}$ be a language given by a binary relation R. The ideal *zero-knowledge proof of knowledge* functionality $\mathcal{F}_{\mathrm{ZK}}^{R}$ of a witness w to an element $x \in \mathcal{L}$, running with provers $S_1, \ldots, S_N$, and verifiers $M_1, \ldots, M_k$, proceeds as follows.

1. Upon receipt of $(S_i, \mathtt{Prover}, x, w)$ from $\mathcal{C_I}$, store w under the tag (S_i, x), and hand $(\mathcal{S}, S_i, \mathtt{Prover}, x, R(x, w))$ to $\mathcal{C_I}$. Ignore further messages from S_i.
2. Upon receipt of $(M_j, \mathtt{Question}, S_i, x)$ from $\mathcal{C_I}$, if $J_{S_i,x}$ is not initialized set $J_{S_i,x} = \emptyset$ and otherwise $J_{S_i,x} \leftarrow J_{S_i,x} \cup \{j\}$. Let w be the string stored under the tag (S_i, x) (the empty string if nothing is stored). If $|J_{S_i,x}| = k$, then hand $((\mathcal{S}, M_j, \mathtt{Verifier}, S_i, x, R(x, w)), \{(M_j, \mathtt{Verifier}, S_i, x, R(x, w))\}_{j=1}^{k})$ to $\mathcal{C_I}$ and otherwise $(\mathcal{S}, M_j, \mathtt{Question}, S_i, x)$.

Note that the functionality synchronizes the response. For cryptosystems such as Paillier [18] and ElGamal [12] with messages encoded in the exponent, the above functionality can be efficiently realized using the Naor and Yung [16] double ciphertext trick and an efficient proof of membership in an interval [6].

4.5 Concatenation Friendly Cryptosystems

To simplify the exposition, we introduce the notion of concatenation-friendly cryptosystems. Informally, a concatenation-friendly cryptosystem allows concatenation of plaintexts under the cover of encryption. We show that this feature can be obtained from any additively homomorphic κ_s-cryptosystem for an arbitrary $\kappa_s > 0$.

Definition 5. *Let $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ be a $N\kappa_s$-cryptosystem. We say that $\mathcal{CS}$ is (N, κ_m)-*concatenation friendly *if there exists $\mathsf{Shift}, \mathsf{Exp} \in \mathrm{PT}$, such that for every $\kappa \in \mathbb{N}$ and every $(pk, sk) = \mathsf{Kg}(1^\kappa)$:*

1. *For every $m \in \{0,1\}^{\kappa_m}$ we have $\mathsf{D}_{sk}(\mathsf{Exp}_{pk}(\mathsf{E}_{pk}(0), m)) = 0$.*
2. *For every $1 \le t \le N$ and $m_c \in \{0,1\}^{\kappa_m}$:*

$$\mathsf{D}_{sk}(\mathsf{Exp}_{pk}(\mathsf{E}_{pk}(\mathsf{Shift}_{pk}(t)), m_c)) = 0^{(t-1)\kappa_m} \| m_c \| 0^{(N-t)\kappa_m} \ .$$

3. *For every $m_l \in \{0,1\}^{(t-1)\kappa_m}$, $m_c \in \{0,1\}^{\kappa_m}$, $m_r \in \{0,1\}^{(N-t)\kappa_m}$:*

$$\mathsf{D}_{sk}\left(\mathsf{E}_{pk}(m_l \| 0^{\kappa_m} \| m_r) \mathsf{E}_{pk}(0^{(t-1)\kappa_m} \| m_c \| 0^{(N-t)\kappa_m})\right) = m_l \| m_c \| m_r \ .$$

We abuse notation and write c^m instead of $\mathsf{Exp}_{pk}(c, m)$, and also drop the subscript pk from $\mathsf{Shift}_{pk}(\cdot)$. We stress that, in general, the operation computed by Exp is *not* the standard exponentiation operator.

Proposition 1. *Let N, κ_m, and $\kappa_s > 0$ be polynomially bounded. If there exists a polynomially indistinguishable additively homomorphic κ_s-cryptosystem, then there exists a (N, κ_m)-concatenation friendly and polynomially indistinguishable $N\kappa_m$-cryptosystem.*

Proof. Let $\mathsf{CS}^{\mathsf{ah}} = (\mathsf{Kg}^{\mathsf{ah}}, \mathsf{E}^{\mathsf{ah}}, \mathsf{D}^{\mathsf{ah}})$ be a polynomially indistinguishable additively homomorphic κ_s-cryptosystem for some polynomial $\kappa_s(\kappa) > 0$. Define Kg equal to $\mathsf{Kg}^{\mathsf{ah}}$.

The idea is to "pack" the bits of a message into a list of ciphertexts in such a way that we can "concatenate" messages from $\{0,1\}^{\kappa_m}$ under encryption as required by the definition. We assume that an integer $0 < t_m \leq \kappa_s$ has been fixed and define $t_r = \lceil \kappa_m / t_m \rceil$. The integer t_r decides into how many pieces we divide a message $m \in \{0,1\}^{\kappa_m}$, and t_m decides how many bits we have in each such piece. Note that we may choose a value of t_m lower than strictly necessary, so that, later, we can optimize the number of bits encrypted under $\mathsf{CS}^{\mathsf{ah}}$ depending on the specific values of N, κ_m, and κ_s, without breaking the symmetry required for concatenation under the cover of encryption.

On input pk and $m \in \{0,1\}^{N\kappa_m}$, the encryption algorithm E first writes $m = m_1 \| \ldots \| m_N$ with $m_j \in \{0,1\}^{\kappa_m}$. Then it writes $m_j = m_{1,j} \| \ldots \| m_{t_r,j}$ with $m_{i,j} \in \{0,1\}^{t_m}$. This gives a $t_r \times N$-matrix

$$m = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,N} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{t_r,1} & m_{t_r,2} & \cdots & m_{t_r,N} \end{pmatrix}$$

where the jth column corresponds to m_j. Then it defines

$$M_{i,j} = m_{i,jt_M+1} \| \ldots \| m_{i,jt_M+t_M}$$

for $j = 0, \ldots, t'_M$ where t_M is chosen maximal under the restriction $t_m t_M \leq \kappa_s$, and $t'_M = N/t_M - 1$. Finally, the algorithm chooses $r_{i,j} \in R^{\mathsf{ah}}_{pk}$ randomly and defines

$$c = \left(\mathsf{E}^{\mathsf{ah}}_{pk}(M_{i,j}, r_{i,j})\right)_{i=1,j=0}^{t_r,t'_M} .$$

The decryption algorithm D takes as input a secret key sk and a ciphertext $c = (c_{i,j})$ and proceeds as follows. It first computes

$$(M_{i,j}) = (\mathsf{D}^{\mathsf{ah}}_{sk}(c_{i,j}))$$

for $i = 1, \ldots, t_r$, $j = 0, \ldots, t'_M$ and interprets $M_{i,j}$ as $m_{i,jt_M+1} \| \cdots \| m_{i,jt_M+t_M}$ by truncating the string in the natural way. Then, it outputs the concatenation m of the columns in the matrix $m = (m_{i,l})$, where i ranges over $\{1, \ldots, t_r\}$ and l ranges over $\{1, \ldots, N\}$.

The encryption and decryption algorithms obviously run in polynomial time, since each individual operation does, and it is easy to see that an encrypted message is always recovered. Thus, $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ is a $N\kappa_m$-cryptosystem.

The polynomial indistinguishability of the scheme follows by a standard hybrid argument, since a ciphertext essentially consists of a polynomial length list of ciphertexts of a polynomially indistinguishable cryptosystem [13].

It remains to show that the scheme is (N, κ_m)-concatenation friendly. We define multiplication component-wise, i.e., $cc' = (c_{i,j})(c'_{i,j}) = (c_{i,j}c'_{i,j})$. The output of $\mathsf{Shift}(t)$ is defined as the concatenation of the columns in the $t_r \times N$-matrix $(z_{i,l})$ where $z_{i,l} = 0$ for all elements except that $z_{1,t} = z_{2,t} = \ldots = z_{t_r,t} = 1$. In other words the tth column consists of ones and all other elements are zero. Finally, we define the Exp algorithm as follows. We write $m = (m_1, \ldots, m_{t_r})$ with $m_i \in \{0,1\}^{t_m}$ as above. Then we define

$$c^m = (c_{i,j}^{m_i}) \ .$$

Consider now t and m_c as in Definition 5, and denote by $z = (z_{i,l}) = \mathsf{Shift}(t)$, and define $Z_{i,j} = z_{i,jt_M+1}\|\ldots\|z_{i,jt_M+t_M}$ for $j = 0, \ldots, t'_M$. We have

$$\begin{aligned}\mathsf{E}_{pk}(\mathsf{Shift}(t))^m &= \left(\left(\mathsf{E}^{\mathsf{ah}}_{pk}(Z_{i,j})\right)_{i=1,j=0}^{t_r,t'_M}\right)^m = \left(\mathsf{E}^{\mathsf{ah}}_{pk}(Z_{i,j})^{m_i}\right)_{i=1,j=0}^{t_r,t'_M} \\ &= \left(\mathsf{E}^{\mathsf{ah}}_{pk}(z_{i,jt_M+1}\|\ldots\|z_{i,jt_M+t_M})^{m_i}\right)_{i=1,j=0}^{t_r,t'_M} \ .\end{aligned}$$

If we write $\mathsf{E}^{\mathsf{ah}}_{pk}(z_{i,jt_M+1}\|\ldots\|z_{i,jt_M+t_M})^{m_i} = \mathsf{E}^{\mathsf{ah}}_{pk}(z'_{i,jt_M+1}\|\ldots\|z'_{i,jt_M+t_M})$, we may conclude that $z'_{i,l} = 0$ for all i and l except that $z'_{i,t} = m_i$ for $i = 1, \ldots, t_r$. In other words, the second requirement is satisfied. Note that if $\mathsf{Shift}(t)$ is replaced by 0 above, we see in a similar way that the first requirement is satisfied.

Consider now m_l, m_c, and m_r as in Definition 5 and write $m = m_l\|0^{\kappa_m}\|m_r$ and $m' = 0^{(t-1)\kappa_m}\|m_c\|0^{(N-t)\kappa_m}$. We have

$$\begin{aligned}\mathsf{E}_{pk}(m)\mathsf{E}_{pk}(m') &= \left(\mathsf{E}^{\mathsf{ah}}_{pk}(M_{i,j})\right)_{i=1,j=0}^{t_r,t'_M}\left(\mathsf{E}^{\mathsf{ah}}_{pk}(M'_{i,j})\right)_{i=1,j=0}^{t_r,t'_M} \\ &= \left(\mathsf{E}_{pk}(m_{i,jt_M+1}\|\ldots\|m_{i,jt_M+t_M})\mathsf{E}_{pk}(m'_{i,jt_M+1}\|\ldots\|m'_{i,jt_M+t_M})\right)_{i=1,j=0}^{t_r,t'_M} \ .\end{aligned}$$

From the additive homomorphism of $\mathsf{CS}^{\mathsf{ah}}$ we conclude that

$$\begin{aligned}&\mathsf{E}_{pk}(m_{i,jt_M+1}\|\ldots\|m_{i,jt_M+t_M})\mathsf{E}_{pk}(m'_{i,jt_M+1}\|\ldots\|m'_{i,jt_M+t_M}) \\ &= \mathsf{E}_{pk}(\bar{m}_{i,jt_M+1}\|\ldots\|\bar{m}_{i,jt_M+t_M})\end{aligned}$$

with $\bar{m}_{i,l} = m_{i,l}$ for $l \neq t$ and $\bar{m}_{i,l} = m'_{i,l}$ otherwise. Thus, the third requirement is satisfied.

Finally, note that it is an easy task to optimize the value of t_m with regards to minimizing the number of individual ciphertexts.

5 Detailed Protocol and Security Analysis

We are now ready to describe the details of our scheme and prove its security.

Protocol 1 (Online/Offline Mix-Net). The online/offline mix-net $\pi_{\mathrm{MN}}^{\mathrm{o/o}}$ executing with senders $S_1, \ldots, S_N$, mix-servers $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows.

Sender S_i

1. Wait until $(M_j, \texttt{SenderPublicKeys}, (pk_i)_{i=1}^N)$ appears on $\mathcal{F}_{\mathrm{BB}}$ for more than $k/2$ distinct j.
2. Wait for an input $(\texttt{Send}, m_i)$ with $m_i \in \{0,1\}^{\kappa_m}$. Then choose $r_i \in R_{pk}$ randomly and compute $c_i = pk_i^{m_i} \mathsf{E}_{pk}(0, r_i)$.
3. Hand $(\texttt{Prover}, (pk, pk_i, c_i), (m_i, r_i))$ to $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$.
4. Hand $(\texttt{Send}, c_i)$ to $\mathcal{F}_{\mathrm{BB}}$.

Mix-Server M_j
Offline Phase
1. Wait for a message $(\texttt{PublicKey}, pk)$ from $\mathcal{F}_{\mathrm{mixer}}$.
2. Form the list $L = (\mathsf{E}_{pk}(\mathsf{Shift}(1), 0), \ldots, \mathsf{E}_{pk}(\mathsf{Shift}(N), 0)$. Hand $(\texttt{Mix}, L)$ to $\mathcal{F}_{\mathrm{mixer}}$, and wait until it returns $(\texttt{Mixed}, (pk_i)_{i=1}^N)$.
3. Hand $(\texttt{Write}, \texttt{SenderPublicKeys}, (pk_i)_{i=1}^N)$ to $\mathcal{F}_{\mathrm{BB}}$.
4. Initialize $J_M = \emptyset$ and repeatedly wait for new inputs or the next new message on $\mathcal{F}_{\mathrm{BB}}$.
   – On input $(\texttt{Run})$, hand $(\texttt{Write}, \texttt{Run})$ to $\mathcal{F}_{\mathrm{BB}}$.
   – If $(M_j, \texttt{Run})$ appears on $\mathcal{F}_{\mathrm{BB}}$, then set $J_M \leftarrow J_M \cup \{j\}$. If $|J_M| > k/2$, go to Step 5.
   – If $(S_\gamma, \texttt{Send}, c_\gamma)$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $\gamma \notin J_S$ then do:
     (a) Set $J_S \leftarrow J_S \cup \{\gamma\}$.
     (b) Hand $(\texttt{Question}, S_\gamma, (pk, pk_\gamma, c_\gamma))$ to $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ and wait for a reply $(\texttt{Verifier}, S_\gamma, (pk, pk_\gamma, c_\gamma), b_\gamma)$ from $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$.

Online Phase
5. Let $J_S' \subset J_S$ be the set of γ such that $b_\gamma = 1$. Compute $c = \prod_{\gamma \in J_S'} c_\gamma$, hand $(\texttt{Decrypt}, c)$ to $\mathcal{F}_{\mathrm{mixer}}$, and wait until a message $(\texttt{Decrypted}, c, m)$ is returned by $\mathcal{F}_{\mathrm{mixer}}$.
6. Write $m = m_1 \| \ldots \| m_N$, where $m_i \in \{0,1\}^{\kappa_m}$, set $m' = (m_1, \ldots, m_N)$, and return $(\texttt{Output}, \mathsf{Sort}(m'))$.

5.1 Online Complexity

The complexity of our scheme depends heavily on the application, the cryptosystem used, the number of parties N and the maximal bit-size κ_m of messages. The setting where our techniques reduce the online complexity the most is when the verification of the submissions can be considered part of the offline phase and $N\kappa_m \leq O(\kappa)$. For this case, the online complexity both in terms of computation and communication between the mix-servers is drastically reduced, as illustrated by the following example.

The most natural practical set-up is to use the Paillier cryptosystem [18] with $N\kappa_m \leq O(\kappa)$. In this case, the online complexity consists of performing $O(N)$ multiplications and $O(1)$ joint decryptions. This can be done using $O(k)$ exponentiations, with a small hidden constant. The fastest mix-net based on the Paillier cryptosystem requires at least $\Omega(kN)$ exponentiations with small constants with precomputation. Thus, we get a speed-up on the order of N.

We have chosen to consider the submission phase as part of the offline phase. If this is not reasonable, then our techniques are still applicable, but they do not reduce the complexity as much. In the Paillier example, this would give a speedup on the order of k. We expect most applications with $N\kappa \leq O(\kappa)$ to be somewhere between these to extremes.

5.2 Security Analysis

We denote by $\mathcal{M}_l$ the set of static adversaries that corrupt at most l mix-servers and arbitrarily many senders. The following proposition captures the security properties of the protocol.

Proposition 2. *Let $\mathcal{CS}$ be a concatenation-friendly and polynomially indistinguishable cryptosystem. Then $\pi_{\mathrm{MN}}^{\mathrm{o/o}}$ securely realizes $\mathcal{F}_{\mathrm{MN}}$ with respect to $\mathcal{M}_{k/2}$ adversaries in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}, \mathcal{F}_{\mathrm{mixer}})$-hybrid model.*

The proof, given in Appendix A, is a simplified version of the proof in [20].

6 Conclusion

A mix-net allows any polynomial number N of senders to send any of exponentially many possible messages, i.e, the only restriction is that $N\kappa_m$ is polynomial in κ, where κ_m is the maximal bit-size of submitted messages.

The homomorphic election schemes may be viewed as a mix-net with the restriction that $2^{\kappa_m} \log N \leq O(\kappa)$, i.e., each sender can send one out of very few messages, but there can be many senders. The advantage of this is that homomorphic election schemes are much more efficient than general mix-nets.

In this paper we have considered the dual restriction $\kappa_m N \leq O(\kappa)$, i.e., there can be few senders, but each sender can send one out of many messages. We have shown that, in this case also, there exists a solution that is much more efficient than a general mix-net in the online phase.

References

1. M. Abe and H. Imai. Flaws in some robust optimistic mix-nets. In *Australasian Conference on Information Security and Privacy – ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 39–50. Springer Verlag, 2003.
2. B. Adida and D. Wikström. How to shuffle in public. Cryptology ePrint Archive, Report 2005/394, 2005. http://eprint.iacr.org/.

3. B. Adida and D. Wikström. How to shuffle in public. In *4th Theory of Cryptography Conference (TCC)*, volume 4392 of *Lecture Notes in Computer Science*, pages 555–574. Springer Verlag, 2007. Accepted for publication at Theory of Cryptography Conference 2007 (full version [2]).
4. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *26th ACM Symposium on the Theory of Computing (STOC)*, pages 544–553. ACM Press, 1994.
5. D. Boneh, Eu-Jin Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 325–342. Springer Verlag, 2005.
6. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE Computer Society Press, 2001. (Full version at Cryptology ePrint Archive, Report 2000/067, http://eprint.iacr.org, October, 2001.).
8. D. Chaum. Untraceable electronic mail, return addresses and digital pseudo-nyms. *Communications of the ACM*, 24(2):84–88, 1981.
9. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *28th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 372–382. IEEE Computer Society Press, 1985.
10. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer Verlag, 1997.
11. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography – PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer Verlag, 2001.
12. T. El Gamal. A public key cryptosystem and a signiture scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
13. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
14. J. Katz, S. Myers, and R. Ostrovsky. Cryptographic counters and applications to electronic voting. In *Advances in Cryptology – Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 78–92. Springer Verlag, 2001.
15. A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
16. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attack. In *22th ACM Symposium on the Theory of Computing (STOC)*, pages 427–437. ACM Press, 1990.
17. T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318. Springer Verlag, 1998.
18. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Verlag, 1999.
19. D. Wikström. A universally composable mix-net. In *1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 315–335. Springer Verlag, 2004.

20. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *Advances in Cryptology – Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 273–292. Springer Verlag, 2005. (Full version [21]).
21. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. Cryptology ePrint Archive, Report 2004/137, 2005. http://eprint.iacr.org/.
22. D. Wikström and J. Groth. An adaptively secure mix-net without erasures. In *33rdInternational Colloquium on Automata, Languages and Programming (ICALP)*, volume 4052 of *Lecture Notes in Computer Science*, pages 276–287. Springer Verlag, 2006.

A Omitted Proofs

Proof (Proposition 2). The proof is a simplified version of the proof in [20] and proceeds by contradiction. We first describe an ideal adversary parameterized by a hybrid adversary and then show that the resulting ideal model is indistinguishable from the hybrid model in which the adversary is executing.

The Ideal Adversary. Denote by $\mathcal{A}$ the hybrid adversary. The ideal adversary $\mathcal{S} = \mathcal{S}(\mathcal{A})$ runs the hybrid adversary as a black-box. The simplest way to describe it is to assume that it simulates all parties in an execution of the hybrid model except those corrupted by $\mathcal{A}$. Denote by I_M and I_S the indices of the corrupted mix-servers and senders respectively. The honest mix-servers M_j for $j \notin I_M$, the honest senders S_i for $i \notin I_S$, and the ideal functionalities $\mathcal{F}_{\mathrm{BB}}$, $\mathcal{F}_{\mathrm{mixer}}$, and $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$, and the environment $\mathcal{Z}'$ are all simulated by $\mathcal{S}$. Precisely how they are simulated is described below. We stress that $\mathcal{Z}$ is the environment that tries to distinguish the two models, whereas $\mathcal{Z}'$ is the environment simulated by $\mathcal{S}$.

In the ideal model the ideal adversary $\mathcal{S}$ corrupts the dummy mix-servers $\tilde{M}_j$ for $j \in I_M$ and the dummy senders $\tilde{S}_i$ for $i \in I_S$, but it has no control over the honest dummy mix-servers $\tilde{M}_j$ for $j \notin I_M$ or the honest dummy senders $\tilde{S}_i$ for $i \notin I_S$.

We now consider how $\mathcal{S}$ performs its simulation.

- **Forwarding messages.** Let $j \in I_M$. Any message sent to $\mathcal{Z}'$ by M_j is forwarded to $\mathcal{Z}$ by $\tilde{M}_j$. Conversely, any message received by $\tilde{M}_j$ is forwarded to M_j by $\mathcal{Z}'$. Similarly, any message sent to $\mathcal{Z}'$ by $\mathcal{A}$ is forwarded to $\mathcal{Z}$ by $\mathcal{S}$ and any message received by $\mathcal{S}$ from $\mathcal{Z}$ is forwarded to $\mathcal{A}$ by $\mathcal{Z}'$.
- **Honest mix-servers.** M_j for $j \notin I_M$ is simulated honestly. However, when $\mathcal{S}$ receives $(\tilde{M}_j, \mathtt{Input}, e)$ from $\mathcal{F}_{\mathrm{MN}}$ it inputs $(\mathtt{Run})$ to the simulated M_j. When $\mathcal{F}_{\mathrm{BB}}$ receives $(\mathtt{Write}, M_j, \mathtt{Run})$ it interrupts the simulation of $\mathcal{F}_{\mathrm{BB}}$, stores the value e_2 of the second index in $\mathcal{F}_{\mathrm{BB}}$ together with e, i.e., it stores (M_j, e, e_2), and continues the simulation of $\mathcal{F}_{\mathrm{BB}}$.
  Recall that $\mathcal{A}$ and $\mathcal{S}$ decides when the output of ideal functionalities are delivered to the honest mix-servers. Without going into details we assume that $\mathcal{S}$ instructs $\mathcal{F}_{\mathrm{MN}}$ to deliver an output to M_j when $\mathcal{A}$ instructs $\mathcal{F}_{\mathrm{mixer}}$ to deliver the plaintext of the cryptotext submitted for decryption by the mix-servers.

- **Corrupt mix-servers.** M_j for $j \in I_M$ is controlled by the adversary, but the corresponding corrupted dummy mix-server $\tilde{M}_j$ must be behave in a way that is consistent with M_j. When $\mathcal{F}_{\mathrm{BB}}$ receives $(\mathtt{Write}, M_j, \mathtt{Run})$ the simulation of $\mathcal{F}_{\mathrm{BB}}$ is interrupted and $\tilde{M}_j$ sends $(\mathtt{Run})$ to $\mathcal{F}_{\mathrm{MN}}$. When $\mathcal{S}$ receives $(\tilde{M}_j, \mathtt{Input}, e)$ from $\mathcal{F}_{\mathrm{MN}}$ it stores (M_j, e, e_2), where e_2 is the value of the second index in the simulation of $\mathcal{F}_{\mathrm{BB}}$. Then the simulation of $\mathcal{F}_{\mathrm{BB}}$ is continued.
- **Both honest and corrupt mix-servers.** When $\mathcal{F}_{\mathrm{BB}}$ is about to hand $(\mathcal{A}, \mathtt{AcceptInput}, e')$ to $\mathcal{C}_{\mathcal{I}}$ the simulation is interrupted. If (M_j, e, e') is stored, then it is deleted and $\mathcal{S}$ hands $(\mathtt{AcceptInput}, e)$ to $\mathcal{F}_{\mathrm{MN}}$ and waits until it receives either $(\tilde{M}_j, \mathtt{Run})$ or $(\tilde{M}_j, \mathtt{Output}, L')$ before it continues the simulation of $\mathcal{F}_{\mathrm{BB}}$.
- **Honest senders.** S_i for $i \notin I_S$ is not simulated honestly. When $\mathcal{S}$ receives $(\tilde{S}_i, \mathtt{Input}, e)$ from $\mathcal{F}_{\mathrm{MN}}$ it instructs S_i to use 0 as its message and it is instructed to replace the witness (m_i, r_i) in its message to $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ by $\perp$. On the other hand $\mathcal{S}$ instructs $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ to behave as if this was a valid witness. When $\mathcal{F}_{\mathrm{BB}}$ receives $(\mathtt{Write}, S_i, \mathtt{Send}, c_i)$ it interrupts the simulation of $\mathcal{F}_{\mathrm{BB}}$ and stores (S_i, c_i, e, e_2), where e_2 is the current value of the second index in $\mathcal{F}_{\mathrm{BB}}$. Then the simulation of $\mathcal{F}_{\mathrm{BB}}$ is continued. Before the functionality $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ hands $((\mathcal{S}, M_j, \mathtt{Verifier}, S_i, (pk, pk_i, c_i), 1), \ldots)$ to $\mathcal{C}_{\mathcal{I}}$ the simulation is interrupted. If (S_i, c_i, e, e_2) is stored, then this tuple is deleted and $\mathcal{S}$ hands $(\mathtt{AcceptInput}, e)$ to $\mathcal{F}_{\mathrm{MN}}$ and waits until it receives $(S_i, \mathtt{Send})$. Then the simulation of M_j is continued. Note that if some honest mix-server accepts c_i as a valid input, then all honest mix-servers will eventually do so as well.
- **Corrupt senders.** S_i for $i \in I_S$ is controlled by the adversary $\mathcal{A}$, but we must make sure that whenever a cryptotext is accepted in the simulated hybrid model, the corresponding plaintext is input to $\mathcal{F}_{\mathrm{MN}}$ by $\tilde{S}_i$ in the ideal model. Before $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ hands $((\mathcal{S}, M_j, \mathtt{Verifier}, S_i, (pk, pk_i, c_i), 1), \ldots)$ to $\mathcal{C}_{\mathcal{I}}$ the simulation is interrupted and $\tilde{S}_i$ is instructed to hand $(\mathtt{Send}, m_i)$ to $\mathcal{F}_{\mathrm{MN}}$. When it returns $(\tilde{S}_i, \mathtt{Input}, e)$, then $\mathcal{S}$ immediately hands the message $(\mathtt{AcceptInput}, e)$ to $\mathcal{F}_{\mathrm{MN}}$. When it returns $(\tilde{S}_i, \mathtt{Send})$ the simulation of $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ is continued. Note that $\mathcal{F}_{\mathrm{MN}}$ receives and accepts as input the plaintext m_i of a corrupt sender if and only if m_i will be part of the output of the honest mix-servers in the simulation (if they give an output at all).
- **Simulation of $\mathcal{F}_{\mathrm{mixer}}$.** Note that now the ciphertexts of honest parties do not contain the correct messages m_i, but instead 0. This flaw in the simulation is corrected by modifying $\mathcal{F}_{\mathrm{mixer}}$. Recall that each sender is assigned a public key pk_i, where $pk_i = \mathsf{E}_{pk}(\mathsf{Shift}(\pi^{-1}(i)))$ for some random permutation $\pi \in \Sigma_N$. The re-encryption and permutation needed to form $(pk_i)_{i=1}^N$ is performed honestly except that if $i \notin I_S$, then pk_i is replaced by $\mathsf{E}_{pk}(0)$, i.e., an encryption of zero. Note that the secret key is not needed during this phase.

  Decryption is also simulated without using the secret key. When it is about to compute $\mathsf{D}_{sk}(c)$, where c is a cryptotext submitted for decryption by more than $k/2$ mix-servers, it behaves as follows. Suppose $\mathcal{F}_{\mathrm{MN}}$ outputs the list of messages $(m'_1, \ldots, m'_{N'})$. $\mathcal{S}$ appends zero-messages $m'_{N'+1} = m'_{N'+2} = \ldots = m'_N = 0$ to this list. Then it chooses a random permutation $\psi \in \Sigma_N$ under the restriction that $\psi(i) = \pi(i)$ for all $i \in I_S$. Finally, it replaces $\mathsf{D}_{sk}(c)$ in its simulation by $(m'_{\psi(1)}, \ldots, m'_{\psi(N)})$. To see that the above simulation always can be performed, note that if $\mathcal{F}_{\mathrm{mixer}}$ has received c from more than $k/2$ mix-servers, then $\mathcal{F}_{\mathrm{MN}}$ has received $(\mathtt{Run})$ from more than $k/2$ dummy mix-servers, and already output the list of messages submitted by dummy senders.

– **Simulation of $\mathcal{F}_{\mathrm{BB}}$ and $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$.** These are simulated as described above.

Analysis of the Ideal Adversary. Our goal is to show that the distribution of the environment when interacting with the ideal model is negligibly close to its distribution when interacting with the hybrid model.

We first make two simple observations.

1. The message used by an honest sender S_i to form its ciphertext c_i using its assigned public key pk_i is always zero. It is easy to see that changing this such that the correct value m_i is used does not change the distribution of the output of the environment, since the distribution of c_i does not change by this modification. This follows, since also pk_i is an encryption of zero. Given this modification we may also assume that $\mathcal{F}_{\mathrm{ZK}}^{\mathcal{R}_{\mathsf{enc}}}$ is simulated completely honestly, since it no longer needs to lie on the behalf of honest senders, i.e., we may assume that S_i uses a valid witness.
2. We may assume that $\psi = \pi$. This follows, since the distribution of $(pk_i)_{i=1}^N$ is independent of the choice of π provided that $\pi(i) = \psi(i)$ for $i \in I_S$.

The proposition now follows from the indistinguishability of the underlying cryptosystem by a standard hybrid argument. Denote by T_l a simulation of the ideal model with the above two modifications, except that in the simulation of $\mathcal{F}_{\mathrm{mixer}}$ the public key pk_i is computed honestly also for $i \in I_S$ provided that $i \leq l$. By inspection we see that the distribution of the output of T_N is identical to the distribution of the output of the environment $\mathcal{Z}$ in the hybrid model.

Thus, if the proposition is false then there exists an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ such that $|\Pr[T_0 = 1] - \Pr[T_N = 1]|$ is non-negligible. A hybrid argument implies that there exists some fixed l such that $|\Pr[T_l = 1] - \Pr[T_{l+1} = 1]|$ is non-negligible.

Denote by A' the adversary in an indistinguishability experiment executed with the cryptosystem $\mathcal{CS}$. It accepts a public key pk as input. Then it hands $(0, \mathsf{Shift}(l+1))$ to the experiment and is given a cryptotext pk_{l+1} such that $\mathsf{D}_{sk}(pk_{l+1})$ equal either 0 or $\mathsf{Shift}(l+1)$. Then it simulates T_l except that it uses the value of pk_{l+1} instead of generating it. By construction, $\Pr[A' = 1 | \mathsf{D}_{sk}(pk_{l+1}) = 0] = \Pr[T_l = 1]$ and $\Pr[A' = 1 | \mathsf{D}_{sk}(pk_{l+1}) = \mathsf{Shift}(l+1)] = \Pr[T_{l+1} = 1]$. It follows that the indistinguishability of the cryptosystem is broken, and the proposition is true.

B The Ideal Bulletin Board

Functionality 4 (Bulletin Board). The ideal bulletin board functionality, $\mathcal{F}_{\text{BB}}$, running with parties $P_1, \ldots, P_n$ and ideal adversary $\mathcal{S}$ proceeds as follows. $\mathcal{F}_{\text{BB}}$ holds two databases D_1 and D_2 indexed on integers. Initialize two counters $c_1 = 0$ and $c_2 = 0$.

– Upon receiving $(P_i, \texttt{Write}, m_i)$, $m_i \in \{0,1\}^*$, from $\mathcal{C}_\mathcal{I}$, store (P_i, m_i) in D_2 by the index c_2 in the database, set $c_2 \leftarrow c_2 + 1$, and hand $(\mathcal{S}, \texttt{Input}, c_2, P_i, m_i)$ to $\mathcal{C}_\mathcal{I}$.
– Upon receiving $(\mathcal{S}, \texttt{AcceptInput}, c)$ from $\mathcal{C}_\mathcal{I}$ check if a tuple (P_i, m_i) is stored in the database D_2 under c. If so, then store (P_i, m_i) in D_1 under the index c_1, set $c_1 \leftarrow c_1 + 1$, and hand $(\mathcal{S}, \texttt{AcceptInput}, c)$ to $\mathcal{C}_\mathcal{I}$.
– Upon receiving $(P_j, \texttt{Read}, c)$ from $\mathcal{C}_\mathcal{I}$ check if a tuple (P_i, m_i) is stored in the database D_1 under c. If so hand $((\mathcal{S}, P_j, \texttt{Read}, c, P_i, m), (P_j, \texttt{Read}, c, P_i, m_i))$ to $\mathcal{C}_\mathcal{I}$. If not, hand $((\mathcal{S}, P_j, \texttt{NoRead}, c), (P_j, \texttt{NoRead}, c))$ to $\mathcal{C}_\mathcal{I}$.