Deterring Voluntary Trace Disclosure in Re-encryption Mix Networks

Philippe Golle
PARC
pgolle@parc.com

XiaoFeng Wang
Indiana University
xw7@indiana.edu

Markus Jakobsson
Indiana University
markus@indiana.edu

Alex Tsow
Indiana University
atsow@cs.indiana.edu

Abstract

An all too real threat to the privacy offered by a mix network is that individual mix administrators may volunteer partial tracing information to a coercer. While this threat can never be eliminated *– coerced mix servers could simply be forced to reveal all their secret data – we can* deter *administrators from succumbing to coercive attacks by raising the stakes. We introduce the notion of a* trace-deterring *mix permutation to guarantee privacy, and show how it ensures that a collateral key (used for an arbitrary purpose) be* automatically *revealed given* any *end-to-end trace from input to output elements. However,* no keying material *is revealed to a party who simply knows what input element corresponds to what output element. Our techniques are sufficiently efficient to be deployed in large-scale elections, thereby providing a sort of publicly verifiable privacy guarantee. Their impact on the size of the anonymity set – while quantifiable – are not of practical concern.*

1 Introduction

Mix servers transform a set of input elements to a permuted set of output elements. The use of probabilistic encryption methods for the generation of the input elements makes correlation of inputs and outputs infeasible as long as at least one of the mix servers involved is honest, and refuses to reveal its input-to-output correspondences.

Voluntary selective disclosure of mix traffic has recently been recognized as an emerging threat [3, 31]. Here, the attacker secures cooperation by means of social coercion (e.g., bribery) of the administrator of a mix server, thereby obtaining information about selected input-output correspondences for this server. Such information is referred to as a *trace*. A trace is only meaningful if it is performed for the same elements through each and every step of the mix network; this is referred to as an *end-to-end* trace. One can distinguish between a situation in which the trace information consists simply of what output element a given input element corresponds to, and a situation in which the mix server also outputs a proof that the two elements in question indeed do correspond to each other. This distinction is not of importance to us, and we will – for simplicity – assume that a trace simply *reports* a relationship, without any evidence.

Traces vary in the amount of information they leak about the correspondence between inputs and outputs. The strongest possible trace discloses the individual correspondence between one input and one output, or several such correspondences. Weaker traces may expose only the global correspondence between a subset of the inputs (e.g., a pair of inputs) and a subset of the outputs, without revealing the individual correspondences within the subset (e.g., which element of the input pair corresponds to which element of the output pair). Other than the trivial global correspondence between the set of *all* inputs and *all* outputs, all traces are undesirable, since they all compromise to some degree the privacy of the mix server.

In this paper, we address the problem of voluntary disclosure of traces. Our approach is to discourage coercion by ensuring the immediate disclosure of some collateral information of each server that collaborates in providing a trace. This collateral information may be the secret mix key of the server in question, thereby making it impossible to perform *partial* traces since all correspondences will be revealed as soon as one is. However, we recognize that this may sometimes be undesirable, and note that the collateral key may be of some other form as well, including simply a key whose disclosure provides publicly verifiable evidence of the server's breach of privacy. This approach may be particularly suitable for applications of mix networks such as electronic elections (see [7] for example).

We emphasize the following important properties of our technique for deterring voluntary trace disclosures in re-encryption mix networks:

First, exposure of a trace does not (in the context of electronic elections) link the *other* voters to their ballots as in previous coercion deterrent schemes (e.g., [31]).

While it is *possible* to embed the mix network's key into a trace, the collateral keys of mix servers can correspond to *any* agreed-upon public key. The holder of a trace learns the collateral keys of the mix servers. Knowledge of a collateral key provides irrefutable evidence of a breach of privacy.

Second, it is meaningful to consider an adversary that either infiltrates or silently coerces all mix servers. We model the stealthiness of this attack with an adversary who forces servers to write *any* selected information on a tape, but can *not* provide interaction with other servers. These practical constraints preclude an attack where the servers use a general multi-party protocol that (very inefficiently, and using lots of interaction) computes and proves the validity of a given end-to-end trace without revealing its intermediate steps or any secret information.

Third, the privacy of our construction is *stronger* than the secrecy of the collateral key. As will be clear from our protocol, it is not possible to compute traces from knowledge of the collateral information; however, the converse is necessarily true.

Fourth, careful selection of collateral information also has the ancillary benefit of increased diligence. An administrator may refuse cooperation with an adversary, but nevertheless fall short of best-practice technique. Rather than stonewall the existence of problems, the administrator benefits from proactive discovery of security threats.

We achieve our goals using a novel approach that we name *trace-deterring permutations*. The key idea behind our scheme is easily explained. We force mix servers to choose their permutations only from one of two sets of permutations. These sets are disjoint, and designed such that any non-trivial trace between inputs and outputs automatically reveals whether the permutation that produced the outputs from the inputs belongs to one set, or the other.

When presented with inputs, a mix server applies a permutation chosen either from one set of permutations or from the other. While the permutation is chosen uniformly at random within a set, the set itself is determined based on one bit of the server's secret collateral key. Any trace causes this bit to be leaked, since it automatically allows a verifier to learn which set the permutation was chosen from. Since there is only one bit of collateral key associated with each round of mixing, each server needs to perform a sequence of trace-deterring permutations in order to represent meaningful collateral keys.

The resulting Trace-Deterring (or *TD*) mixing protocol thus forces a server's permutation selection to correspond to a collateral secret key. This is done by means of appropriate commitments and proofs, which are both surprisingly simple. A single complete trace can be seen to reveal the secret collateral key for every mix server along the trace's path.

Organization. The rest of the paper is organized as follows. Section 2 reviews related work and its relation to our new technique. Section 3 discusses our attack model and introduces some necessary background on re-encryption mix networks. Section 4 gives a high-level overview of our trace-deterring (TD) techniques. Section 5 describes TD permutations. Section 6 describes the commitment protocol used by mix servers to commit to their collateral secret key. Section 7 describes a single round of TD mixing. Finally, section 8 presents the design of a complete TD mix network, and analyzes the security property of our approach. We conclude in section 9.

2 Related Work

Chaum first formalized mixing [6], a cryptographic laundering technique for preventing traffic analysis of electronic mail, providing unlinkability between sender and receiver. In Chaum's method, known as *decryption mixing*, the sender submits a serially encrypted message which is subsequently decrypted by the intermediate mix servers, and forwarded in a different order than received. However, decryption mixing cannot prevent the sender of a message from observing the trace of her own message. This allows an active attacker to insert a probe message to discover the collateral secret attached to a trace. Therefore, our technique is not designed for decryption mix networks.

Re-encryption mixing [29, 1] achieves the property that the intermediate messages are unrecognizable to all, including their originators. For the first stage of this scheme, senders encrypt their messages once using a common public key. Servers forward randomly re-encrypted messages. In the second stage the mix servers collaboratively decrypt the messages with their share of the secret key. Our deterrent technique is based upon re-encryption mixing, and therefore does not make the collateral secret key vulnerable to an active attacker.

Since they were first proposed, mixes have been building blocks in strong electronic election schemes [6, 16, 29, 32, 23]. In this context, robustness has parity with unlinkability. Robustness primarily refers to systems in which each mix is asked to provide a proof or strong evidence for its honest behavior. For example, Ogata et al. [28] use cut and choose techniques to achieve robust mix-nets. Subsequent schemes improve both the efficiency of zero knowledge proofs [21] and attain *universal verifiability* [1, 2], i.e. verifiability by third party observers. Other protocols employ layer redundancy [12] and random partial checking [23] to achieve robustness.

No mixing protocol prevents an administrator from logging and later divulging input to output correspondences performed by his machine. This form of *voluntary disclosure* is an undetectable attack. So far, the only mitigation is deterrence: a secret that is valuable to the owner or administrator of a mix server is held as collateral. One such approach, *fragile mixing* [31], constrains the choice of permutations to those where knowledge of one input to output correspondence reveals all remaining correspondences. Assuming that administrators value the privacy of some messages in each batch, this method encourages them to uphold the secrecy of all linkages.

Our trace-deterring technique has significant advantages over fragile mixing. We do not need the assumption that every message batch contains some messages that are valuable to mix administrators. Any secret key can be used as collateral, which avoids the aforementioned secrecy-upholding problem. Disclosure of a trace could be made publicly verifiable, through the revelation of the collateral secret key. Finally, our trace-deterring technique does not constrain the permutation selection nearly to the same extent fragile mixing does. As we shall see, our technique allows a mix server to mix n inputs with a permutation chosen from a set of size $(n-1)!$, versus a set of size n for fragile mixing. For a given number of rounds of mixing, our technique thus offers better privacy than fragile mixing.

Other research is also related to the voluntary disclosure problem. For example, *proprietary certificates* [22, 4] address the problem of certificate-lending to achieve unauthorized access. This scheme binds collateral information to the private key of the proprietary certificate so that its divulgence punitively leaks the collateral information. Dwork, Lotspiech, and Naor [13] introduced the concept of "self-policing via sensitive information" with *signets*, a proposal for preventing illegal redistribution of digital content. These approaches are close to ours in spirit. They all hold some collateral secret to deter a party from acting dishonestly.

Traces are not the only method for linking sender and receiver in a mix network. Statistical disclosure [11], intersection [24], and timing attacks [14, 25] correlate the senders and receivers without determining traces. However, the collateral secret key in our scheme will not be revealed if any linkages are deduced in this manner.

Our work rests upon the correctness of several proofs of knowledge and commitment schemes: equal discrete logarithms [9], knowledge of discrete logarithm, verifiable shuffling [17, 26, 20], and splitting techniques from Pedersen's non-interactive secret sharing [30].

3 Preliminaries

3.1 Attack model

As is standard in the context of mix networks, we model all players as polynomial-time Turing Machines with read access to a public bulletin board. In addition, mix servers also have appendive write access to the bulletin board. The mix servers have a certified public key of some suitable format. In the case of decryption mixes (and many re-encryption mixes), the servers also have access to the corresponding secret key.

Corruption. A large number of different models have been developed to describe adversarial behavior. In the context of mix networks, it is commonly assumed that an adversary may control and fully coordinate the actions of some set of mix servers. This is referred to as *corrupting* the servers in question. Corruption may take place at any time during the lifetime of the mix network. It is always assumed that the adversary cannot corrupt a quorum of mix servers. For simplicity, the corruption is typically assumed to be *static*; however, one could divide time into intervals and consider a *mobile* adversary that may corrupt a different set of servers (below a quorum) in each time interval. In this paper, we make the standard assumption that the corrupting adversary is static.

Coercion. In addition to being able to corrupt any non-quorum of servers, we also allow the adversary to *coerce* any number of mix servers (possibly all servers). The coercion may take place at any time during the lifetime of the mix network. An adversary coerces a server by sending it a *coerce message*, containing a description of what secret information it wants the coerced party to divulge. (This may simply be *all* the secret information stored by the server in question.) The victim of the coercion responds to the attacker with the requested information; after that, no further interaction arising from the coercion is allowed. Thus, this restriction disallows interaction between servers as part of the coercion (apart from allowing coerce messages to be functions of responses to previous coerce messages.) This models an insider attack in which information can be stolen, but protocols cannot be replaced. This is a realistic assumption in all protocols where there may be some public audit of communication between legal protocol participants (as is afforded by the use of a bulletin board) and in which the attacker has temporary read access to some secret storage area, whether the corresponding coerced server is aware of this or not. To the best of our knowledge, this model of coercive behavior is novel.

3.2 ElGamal Encryption

Let g be a generator of $\mathcal{G}_q$, a multiplicative subgroup of order q where the Decisional Diffie-Hellman problem is hard. The secret key, x, is chosen at random from $\mathbb{Z}_q^*$, denoted $x \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$. The public key, y, is the value $g^x \in \mathcal{G}_q$. To encrypt a message $m \in \mathcal{G}_q$, one chooses $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$ and evaluates the ordered pair (g^γ, my^γ). Decryption of an ElGamal ciphertext (G, M) is computed by the expression $M \cdot G^{-x}$. One can re-encrypt a ciphertext (G, M) by choosing $\delta \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$ and evaluating (Gg^δ, My^δ). The decryption method remains the same. Decryption is a homomorphism from the pairwise multiplicative group of ciphertexts to the multiplicative group of plaintexts: Let (G, M) and (F, N) be ciphertexts for m and n respectively. Then $(G \times F, M \times N)$ is a ciphertext for $m \times n$. We will make use of the following protocols:

Proof of knowledge of discrete logarithm (KDL) [8]
A prover $\mathcal{P}$ proves to an honest verifier $\mathcal{V}$ the knowledge of the discrete logarithm base g for $a \in \mathbb{Z}_q^*$ without leaking out any information about $\log_g a$. We let $KDL_g\{a\}$ denote an instance of this protocol. The computational cost of the protocol $KDL_g\{a\}$ given in [8] is one modular exponentiation for $\mathcal{P}$ and two modular exponentiations for $\mathcal{V}$.

Proof of correct re-encryption (PCR) [9]
A prover $\mathcal{P}$ proves to an honest verifier $\mathcal{V}$ that an ElGamal ciphertext (G', M') is a re-encryption of a ciphertext (G, M) without leaking any other information. We let $PCR\{(G, M) \rightsquigarrow (G', M')\}$ denote an instance of this protocol. The proof consists of showing that $\log_g(G'/G) = \log_y(M'/M) = r$, without leaking any information about the value r. The computational cost of this protocol is 2 modular exponentiations for $\mathcal{P}$ and 4 modular exponentiations for $\mathcal{V}$.

Discrete logarithm proof systems [5, 10, 33]
An efficient zero-knowledge proof can be constructed for any monotone boolean formula whose atoms consist of the protocols KDL or PCR. This paper uses a single boolean formula, whose computational cost for $\mathcal{P}$ and $\mathcal{V}$ will be analyzed in the section where it is presented (Section 7).

3.3 Re-encryption Mix Networks

An ElGamal *mix* is a list of ciphertexts followed by a permuted list of the re-encrypted ciphertexts. Let $L = [(G_j, M_j)]$ and $L' = [(G'_j, M'_j)]$ be two lists of ElGamal ciphertexts. To indicate that L' consists of the elements of L re-encrypted and permuted according to a permutation π, we use the following notation:

$$L' = \mathsf{MIX}_{\pi}(L).$$

Verifiable mixing. Verifiable mixing protocols [17, 20, 27] allow a mix server to prove to a verifier that $L' = \mathsf{MIX}_{\pi}(L)$. More precisely, let $L = [(G_j, M_j)]$ and $L' = [(G'_j, M'_j)]$ be two lists of ElGamal encrypted messages. A verifiable mixing protocol allows the mix server to prove the existence of a permutation π and a sequence of exponents γ_j such that $(G'_j, M'_j) = (G_{\pi(j)}g^{\gamma_j}, M_{\pi(j)}y^{\gamma_j})$, without leaking any information about π or the values γ_j. Given n input ciphertexts, the computational cost of the most efficient verifiable mixing protocol [20] is $6n$ modular exponentiations for the prover (the mix server) and $6n$ modular exponentiations for the verifier. We denote a proof of verifiable mixing

$$PVM\{L \rightsquigarrow L'\}.$$

Equivalent mixing. We say that two mixes are *equivalent* when they share the same permutation. More precisely, let $L'_0 = \mathsf{MIX}_{\pi_0}(L_0)$ and $L'_1 = \mathsf{MIX}_{\pi_1}(L_1)$. These two mixes are equivalent if $\pi_0 = \pi_1$. To prove that two ElGamal ciphertext mixes are equivalent, we run a verifiable mix protocol on the pairwise product of ciphertexts of both mixes (see [19] for detail). The computational cost of a proof of equivalent mixing is thus equal to the cost of a verifiable mixing protocol. We denote a proof of equivalent mixing

$$PEM\{(L_0 \rightsquigarrow L'_0) = (L_1 \rightsquigarrow L'_1)\}.$$

4 Overview

In this section, we give a broad overview of our approach to deterring voluntary trace disclosure in mix networks. A regular mix network applies to a set of n inputs a permutation chosen uniformly at random from the set $\mathcal{T}$ of permutations on n elements. Our trace-deterring mixnet, in contrast, defines two disjoint sets of permutations $\mathcal{T}_0$ and $\mathcal{T}_1$. A mix server applies a permutation chosen either from $\mathcal{T}_0$ or from $\mathcal{T}_1$. The choice is dictated by the bits of a secret key held by the server (we call this secret key the collateral secret key of the server).

More precisely, let $n > 0$ denote the number of inputs in a mixing batch and let $\mathcal{T}$ denote the set of permutations on n elements. A trace-deterring partition, or TD-partition, is a partition of $\mathcal{T}$ into three disjoint subsets: $\mathcal{T}_0$, $\mathcal{T}_1$ and $\mathcal{T}_*$. Let b_i denote a bit of the server's collateral secret key. If $b_i = 0$, the server applies to the batch a permutation chosen (uniformly at random) from the set $\mathcal{T}_0$. If $b_i = 1$, the server applies to the batch a permutation chosen (uniformly at random) from the set $\mathcal{T}_1$. The set $\mathcal{T}_*$ consists of left-over permutations that are never used by the mix server.

Definition 1. (Trace-deterring partition) Let $\mathcal{T}$ denote the set of permutations on n elements, and let $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ be a partition of $\mathcal{T}$ into three disjoint subsets. We say that $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ is a *trace deterring* partition, or TD-partition, if

for all $\pi_0 \in \mathcal{T}_0$ and all $\pi_1 \in \mathcal{T}_1$ and all subsets $M \subset \mathbb{Z}_n$ such that $0 < |M| < n$, we have $\pi_0(M) \neq \pi_1(M)$.

This definition states that the knowledge of *any* (strict, non-empty) subset of the inputs, and the image of this subset by a permutation π chosen from $\mathcal{T}_0 \cup \mathcal{T}_1$ reveals whether $\pi \in \mathcal{T}_0$ or $\pi \in \mathcal{T}_1$. This property of TD-partitions deters a dishonest mix server from revealing to a third party any information that would help decrease the size of the anonymity set of a message. Indeed, if the server reveals any correspondence between a subset of its inputs and a subset of its outputs, the correspondence also reveals one bit of the server's collateral secret key. This bit is incriminating evidence of the server's breach of privacy (a single bit is very weak evidence, but as we shall see, a complete trace exposes the complete collateral key of a server).

Definition 1 is the strongest possible, in the sense that it prevents the mix server from revealing the image of *any* non-empty strict subset of the inputs. The server can not reveal the image of single input. Nor can it reveal what pair of outputs (as a set) corresponds to a pair of inputs, nor for that matter the image (as a set) of any strict subset of the inputs. We feel this strong definition is justified. Since it is difficult to anticipate the privacy requirements of specific applications, privacy primitives should be designed with the most conservative assumptions possible.

Example. One example of a TD-partition for $n = 3$ is $\mathcal{T}_0 = \{[1,2,3]\}$, $\mathcal{T}_1 = \{[2,3,1],[3,1,2]\}$ and $\mathcal{T}_* = \{[1,3,2],[2,1,3],[3,2,1]\}$. It is easy to verify that knowledge of any (strict, non-empty) subset of the inputs, and the image of that subset by a permutation chosen from $\mathcal{T}_0 \cup \mathcal{T}_1$ reveals whether the permutation belongs to $\mathcal{T}_0$ or $\mathcal{T}_1$. In this toy example, the sets $\mathcal{T}_0$ and $\mathcal{T}_1$ contain only "shift" permutations (like fragile mixing [31]), but in the next section we will define TD-partitions for which $|\mathcal{T}_0 \cup \mathcal{T}_1| = (n-1)!$.

In a nutshell, our approach is to let a mix perform multiple rounds of mixings on an input batch. In each round of mixing, a random permutation is chosen from either $\mathcal{T}_0$ or $\mathcal{T}_1$ according to one bit of the server's collateral secret key. Any trace between inputs and outputs discloses whether the permutation is in $\mathcal{T}_0$ or $\mathcal{T}_1$ and thus also discloses the corresponding bit of the secret collateral key.

Figure 1 illustrates the idea. In the figure, a collateral secret key string $10\ldots1$ is committed through a Key commitment scheme (Section 6). The TD-partition in this example defines $\mathcal{T}_0$ as the identity permutation singleton and $\mathcal{T}_1$ as the set of circular permutations (Section 5). The correspondence between the committed bits of the collateral key and the permutations applied in the mixing rounds is proved in zero-knowledge. If the mix traces any subset of the input messages, the input-to-output correspondence reveals whether the permutation applied is the identity or a circular permutation — thus also revealing whether the corresponding bit of the collateral key is 0 or 1.

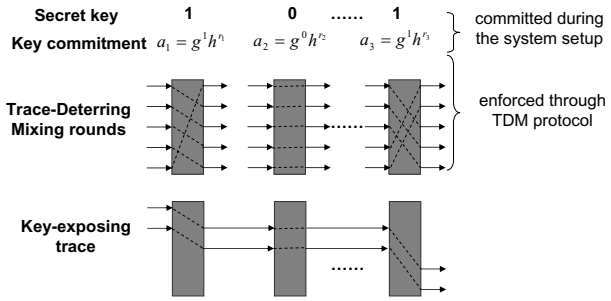

Figure 1. Trace-deterring protocol.

TD partition (Section 5). This paper defines a specific TD-partition that satisfies two additional properties required for our mix network application:

1. *Privacy.* At least one of the sets $\mathcal{T}_0$ or $\mathcal{T}_1$ must be sufficiently large to ensure that a mix network that selects permutations only from $\mathcal{T}_0 \cup \mathcal{T}_1$ offers good privacy.
2. *Correctness.* Let b_i denote one bit of the collateral secret key of a mix server. We need an efficient protocol that allows a mix server to prove that it applied a permutation selected from $\mathcal{T}_0$ if $b_i = 0$, or from $\mathcal{T}_1$ if $b_i = 1$.

Collateral key commitment (Section 6). The collateral key commitment (KC) protocol is run only once in a setup stage to commit every mix server to its collateral secret key. A mix server executes a zero-knowledge protocol with an honest verifier to prove that it has correctly committed to its collateral key.

Trace-deterring mixing round (Section 7). One round of trace deterring mixing (TDM) binds one bit of a server's collateral key to one permutation applied to a batch of messages. The TDM mixing protocol does not consume any keying information and can be run a polynomial number of times for (KC). In TDM, a mix server takes as input a commitment to a bit b of its collateral key and a list of ElGamal ciphertexts. The server re-encrypts and mixes this list according to a permutation chosen from the set $\mathcal{T}_0$ if $b = 0$ or from the set $\mathcal{T}_1$ if $b = 1$. Finally, the server outputs the permuted list and proves to an honest verifier that it executed the TDM protocol correctly.

Trace-deterring mix network (Section 8). Since there is only one bit of collateral key associated with each round of

mixing, each server needs to perform a sequence of trace-deterring permutations in order to represent meaningful collateral keys. If the server performs these transformations on the same batch consecutively, it can reveal only an end-to-end correspondence of a message to a third party. This can be done without any communication with other mix servers. We remove this option by interleaving the sequential mixing of independent servers. The definition of a TD-partition ensures that any complete trace of the input to output reveals the collateral keys.

5 Trace-Deterring Partition

Throughout this section and the rest of this paper, we let n denote the number of inputs submitted to the mix server. We let $\mathcal{T}$ denote the set of permutations on n elements. It is well-known that $|\mathcal{T}| = n!$. We let $\mathsf{Id} \in \mathcal{T}$ denote the identity permutation on n elements.

In what follows, we define a specific TD-partition that we will serve as the building block of our TD-mixing protocol. Our TD-partition is based on circular permutations, which are defined as follows:

Definition 2. (Circular permutation) Let $\pi \in \mathcal{T}$ be a permutation on n elements. We say that π is a circular permutation if its cyclic decomposition contains a single cycle of length n. In other words, a circular permutation is a permutation for which the successive images of any element form a cycle of length exactly n.

Throughout the rest of this paper, we let $\mathcal{C} \subset \mathcal{T}$ denote the set of circular permutations on n elements. The number of circular permutation is $|\mathcal{C}| = (n-1)!$. Circular permutations should not be confused with "shift" permutations (there are only n shift permutations on n elements). For example, with $n = 4$, the permutation π defined by $\pi(1) = 3$, $\pi(2) = 4$, $\pi(3) = 2$ and $\pi(4) = 1$ is a circular permutation (its cyclic decomposition contains a single cycle $(3, 4, 2, 1)$ of length 4), but it is not a shift permutation.

Proposition 1. *Let $\mathcal{T}_* = \mathcal{T} - (\mathcal{C} \cup \mathsf{Id})$ denote permutations that are neither circular nor the identity. The partition $(\{\mathsf{Id}\}, \mathcal{C}, \mathcal{T}_*)$ is trace-deterring.*

Proof. Let $\pi \in \mathcal{T}_1$ and let M be a subset of inputs elements such that $0 < |M| < n$. We must show that $\mathsf{Id}(M) \neq \pi(M)$. We have $\mathsf{Id}(M) = M$. We also have $\pi(M) \neq M$, for otherwise π would have a cycle of length strictly less than n, contradicting the assumption that π is a circular permutation. It follows that $\mathsf{Id}(M) \neq \pi(M)$. □

Proposition 2. *We define the* size *of a TD-partition $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ as $\max(|\mathcal{T}_0|, |\mathcal{T}_1|)$. The partition $(\{\mathsf{Id}\}, \mathcal{C}, \mathcal{T}_*)$ is of size $(n-1)!$*

The proof of this proposition is immediate: it is a well-known fact that $|\mathcal{C}| = (n-1)!$. The partition $(\{\mathsf{Id}\}, \mathcal{C}, \mathcal{T}_*)$ is of maximal size in the following sense: any TD-partition $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ such that $\mathcal{T}_0 = \{\mathsf{Id}\}$ must satisfy $\mathcal{T}_1 \subseteq \mathcal{C}$ (the proof is in the appendix).

An equivalent definition. Let ν be the permutation on $\mathbb{Z}_n$ defined by $\nu(i) = i + 1 \mod n$. The permutation ν is called the "shift" permutation. Let $\circ$ denote the composition of functions. We define the set of permutations that are conjugates of ν by elements of $\mathcal{T}$ as:

$$\{\pi^{-1} \circ \nu \circ \pi \mid \pi \in \mathcal{T}\}.$$

Proposition 3. *We have $\mathcal{C} = \{\pi^{-1} \circ \nu \circ \pi \mid \pi \in \mathcal{T}\}$.*

This proposition states that the set of circular permutations is exactly the same as the set of conjugates of the shift permutation ν. The proof is given in the appendix. This equivalent definition will prove useful in Section 7 to let a mix server prove that a batch of inputs was mixed correctly according to our TD-partition.

6 Collateral Key Commitment Protocol

The collateral key-commitment (KC) protocol lets a mix server e generate a *collateral* public key and commit to the bits of the corresponding secret key. The protocol KC takes as input a generator g of a group $\mathcal{G}_q$ of order q, and the public key $y \in \mathcal{G}_q$ of the mix network. The protocol outputs a collateral key $y_e \in \mathcal{G}_q$ for mix server e, together with a list of commitments $[a_i]$ to the bits of the corresponding secret key s_e (such that $y_e = g^{s_e}$). The protocol also allows the mix server to prove the correctness of the commitments, without leaking any information about the secret key. We denote the protocol by $(y_e, [a_i]) \leftarrow KC(g, q, y)$. We note that the KC protocol only needs to be executed once during the system bootstrap stage.

Protocol 1. Key commitment protocol $(y_e, [a_i]) \leftarrow KC(g, q, y)$

- $\mathcal{P}$ generates a secret key $s_e \overset{R}{\leftarrow} \mathbb{Z}_q^*$, and outputs the corresponding public key $y_e = g^{s_e}$. We denote the bits of the secret key s_e by b_i for $0 \leq i < k$.
- $\mathcal{V}$ sends $h \overset{R}{\leftarrow} \mathcal{G}_q/\{1\}$ to $\mathcal{P}$, where h is used to blind $\mathcal{P}$'s commitments.
- **Commitment.** For every bit b_i of the secret key ($0 \leq i < k$), $\mathcal{P}$ chooses a private exponent $r_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and outputs the commitment $a_i = g^{b_i} h^{r_i}$.
- **Proof.** $\mathcal{P}$ proves to $\mathcal{V}$ the correctness of the commitments a_i to the bits b_i of the collateral key as follows:

1. $\mathcal{P}$ proves $\left(KDL_h\{a_i\} \vee KDL_h\{a_i/g\}\right)$. As noted in Section 3.2, an efficient proof can be constructed for this boolean formula.
2. $\mathcal{P}$ computes $A = (a_0)^{2^0} \cdot (a_1)^{2^1} \cdot \ldots \cdot (a_{k-1})^{2^{k-1}}$. Note that $A = g^{s_e} h^R$, where $R = \sum 2^i r_i$.
3. $\mathcal{P}$ proves $KDL_h\{A/y_e\}$ to $\mathcal{V}$ using knowledge of R.

The commitment scheme used in Protocol 1 is well-known to be complete, sound and zero-knowledge (see [30]).

7 One Round of Trace-Deterring Mixing

This section introduces a protocol to perform one round of trace-deterring mixing (TDM). In TDM, a mix server takes as input a commitment to a bit b of its collateral key and a list of ElGamal ciphertexts. The server re-encrypts and mixes this list according to a permutation chosen from the set $\mathcal{T}_0$ if $b = 0$ or from the set $\mathcal{T}_1$ if $b = 1$. The sets $\mathcal{T}_0$ and $\mathcal{T}_1$ are defined according the the TD-partition $(\{\mathsf{Id}\}, \mathcal{C}, T_*)$ of Section 5. Finally, the server outputs the permuted list and proves to an honest verifier that it executed the TDM protocol correctly. Since any input-to-output trace discloses the secret bit, the administrator of the mix server is deterred from leaking any information about the permutation to a third party.

Let $0 \leq j < n$ and let $L_0 = [(G_j, M_j)]$ be an input batch to a mix server. We denote an instance of TDM as $TDM(y, b_i, L_0)$, where y is the public key mix server e uses to encrypt and re-encrypt all the incoming messages. The protocol is as follows:

Protocol 2. Trace-Deterring-Mixing $TDM(y, b_i, L_0)$

1. The mix server e chooses a permutation π_i uniformly at random from $\mathcal{T}$. The mix server computes $L_1 = \mathsf{MIX}_{\pi_i}(L_0)$ and outputs L_1.
2. The server outputs a list L_2 defined as follows:
   - If $b_i = 0$, the server defines $L_2 = \mathsf{MIX}_{\mathsf{Id}}(L_1)$.
   - If $b_i = 1$, the server defines $L_2 = \mathsf{MIX}_{\nu}(L_1)$, where ν is the shift permutation defined in Section 5.
3. The server computes $L_3 = \mathsf{MIX}_{\pi_i^{-1}}(L_2)$ and outputs L_3.

Soundness. We prove first that the protocol TDM is sound. More precisely, we prove that TDM guarantees that the list L_0 is permuted according to Id when $b_i = 0$ and is permuted according to a permutation chosen randomly from the set $\mathcal{C}$ of circular permutations when $b_i = 1$.

Note that if $b_i = 0$, we have

$$L_3 = \mathsf{MIX}_{\pi_i^{-1}}\Big(\mathsf{MIX}_{\mathsf{Id}}(\mathsf{MIX}_{\pi_i}(L_0))\Big)$$

and thus $L_3 = \mathsf{MIX}_{\mathsf{Id}}(L_0)$. In other words, the list L_3 consists of re-encryptions of the elements of the list L_0 without any modification to their order. If $b_i = 1$, then

$$L_3 = \mathsf{MIX}_{\pi_i^{-1}}\Big(\mathsf{MIX}_{\nu}(\mathsf{MIX}_{\pi_i}(L_0))\Big).$$

By Proposition 3, we know that $\pi_i^{-1} \circ \nu \circ \pi_i$ is a circular permutation. This shows that if the protocol is executed correctly, the list L_0 is permuted according to Id when $b_i = 0$ and is permuted according to a permutation chosen randomly from the set $\mathcal{C}$ of circular permutations when $b_i = 1$.

The mix (the prover $\mathcal{P}$) must next prove to a verifier $\mathcal{V}$ that it executed the TDM protocol correctly. The proof proceeds as follows:

Protocol 3. Generating a proof of correct execution of TDM

1. To prove correct operation in steps 1 and 3 of the TDM protocol, $\mathcal{P}$ first proves to $\mathcal{V}$ the correctness of the mixing:

   $$PVM\{L_0 \rightsquigarrow L_1\} \quad \text{and} \quad PVM\{L_3 \rightsquigarrow L_2\}.$$

   $\mathcal{P}$ then proves to $\mathcal{V}$ that the mix that transforms L_0 into L_1 (step 1) is equivalent to the mix that transforms L_3 into L_2 (step 3). This proof is given by running

   $$PEM\{\ (L_0 \rightsquigarrow L_1) \ = \ (L_3 \rightsquigarrow L_2)\ \}$$

2. $\mathcal{P}$ proves to $\mathcal{V}$ correct operation in step 2 as follows. Recall from Section 6 that the server's commitment to the bit b_i is a value $a_i = g^{b_i} h^{r_i}$. let $L_1 = [(G_j, M_j)]$ and $L_2 = [(G'_j, M'_j)]$ denote the elements of the lists L_1 and L_2 (for $0 \leq j < n$). The server must prove that:
   - either $b_i = 0$ (i.e. $a_i = h^{r_i}$) and the ciphertext (G'_j, M'_j) is a re-encryption of (G_j, M_j) for $j = 0, \ldots, n-1$,
   - or $b_i = 1$ (i.e. $a_i = g h^{r_i}$) and the ciphertext (G'_{j+1}, M'_{j+1}) is a re-encryption of (G_j, M_j) for $j = 0, \ldots, n-1$ (the list was shifted). Note that in the notation (G'_{j+1}, M'_{j+1}), the subscript indices are taken modulo n. In other words, with a slight abuse of notation, we let $(G'_n, M'_n) = (G'_0, M'_0)$.

   Formally, let

   $$F_0 \ = \ \bigwedge_{0 \leq j < n} PCR\left\{(G_j, M_j) \rightsquigarrow (G'_j, M'_j)\right\}$$

   $$F_1 \ = \ \bigwedge_{0 \leq j < n} PCR\left\{(G_j, M_j) \rightsquigarrow (G'_{j+1}, M'_{j+1})\right\}$$

$\mathcal{P}$ proves to $\mathcal{V}$ the following formula:

$$\Big(KDL_h\,\{a_i\} \;\wedge\; F_0\Big) \vee \Big(KDL_h\,\{a_i/g\} \;\wedge\; F_1\Big).$$

As noted in Section 3.2, an efficient proof can be constructed for this boolean formula.

This protocol can be converted into a noninteractive version in the random oracle model by using the Fiat-Shamir heuristic [15].

The completeness, soundness and zero-knowledge of protocol 3 follow directly from the corresponding properties of the well-known building blocks that make up the protocol.

Computational complexity of the TDM protocol. Let n denote the number of ciphertext inputs (i.e., the number of elements in the list L_0):

- The cost of Steps 1 and 3 of the TDM protocol (with the accompanying proof of correctness), is the cost of two verifiable mixings and one proof of equivalent mixing: $18n$ modular exponentiations for both the prover and the verifier.
- The cost of Step 2 of the protocol is the cost of re-encrypting n elements for the prover (which is $2n$ modular exponentiations) plus the cost of the proof for the boolean formula. Using the technique of [5], the computational cost of proving the boolean formula comes to $4n + 3$ modular exponentiations for $\mathcal{P}$ and $4n + 4$ modular exponentiations for $\mathcal{V}$.

The total computational complexity of the TDM protocol is thus $24n$ modular exponentiations for the prover $\mathcal{P}$ and $22n$ modular exponentiations for the verifier $\mathcal{V}$.

8 A Trace-Deterring Mix Network

In this section, we discuss how to construct a complete trace-deterring mix network using as a building block the TDM protocol of Section 7. Our trace-deterring techniques are compatible with the standard construction of mix networks, but add a new property which discourages mix administrators from disclosing input-to-output message correspondences.

As discussed in Section 4, a mix server must perform a sequence of TD-Mixing operations over a batch of input messages, each corresponding to one bit of its collateral secret key. If all these TD-Mixings are executed consecutively, the mix server can disclose to a third party the input of a message to the first TD-Mixing and its output of the last TD-Mixing, without revealing any of its secret bits in-between. To prevent this attack, we propose a *loop* construction of a mix cascade. A cascade is composed of multiple mix servers, each belonging to a different organization. A batch of messages flows through the cascade, and then goes back to the head of the cascade to start another round. Each round commits to one bit of these mix servers' collateral keys.

Figure 2 illustrates this construction. In the figure, the mix cascade performs k loops on an input batch, where k is the number of bits in the collateral key of a mix server. The m servers permute the batch according to the bit string $11\ldots0$ in the first loop and $10\ldots1$ in the second loop, and so on, until all k strings are used. Note that the m mix servers are assumed to belong to different organizations. Their reluctance to reveal their secret bits to one another prevents them from colluding.

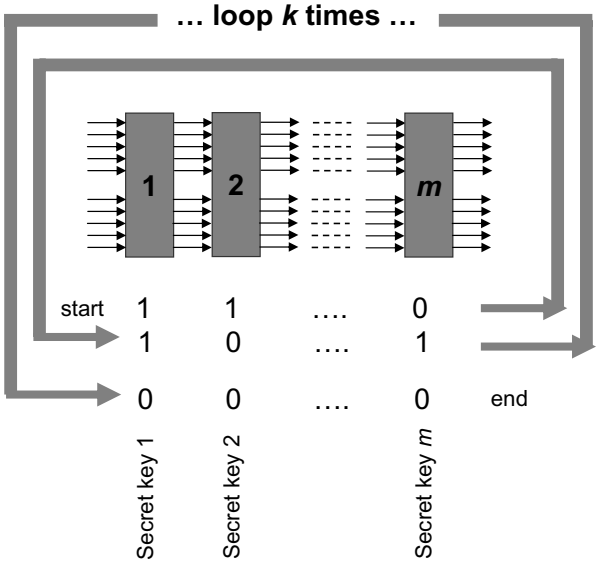

Figure 2. A Trace-Deterring mix cascade.

We describe a trace-deterring mix-cascade protocol built on Abe's scheme [1]. In an initial setup step, every mix server on the cascade commits to the bits of its collateral key. For every batch of input messages, the mix cascade operates in two stages: 1) re-encryption and mixing; and 2) decryption of the final outputs. During the first stage, the mix cascade re-encrypts the ciphertext elements of the input batch, permutes them according to the TDM protocol, and proves correct execution of the TDM protocol as discussed in Section 7. In the decryption stage, all mixes in the cascade collaborate to decrypt the output batch and forward these decryptions to the receivers.

In the formal description of the protocol, given below, we let m denote the number of servers in the cascade. Let y denote the public key of the mix network (used to encrypt and re-encrypt inputs). The corresponding decryption key

x is shared among all mix servers, such that a quorum of servers can decrypt. In addition, we assume that each mix server e has a secret collateral key s_e and we let y_e denote the corresponding public key. The collateral key s_e *can* be the same as the server's share of the key x, but it need not be the same (recall that our scheme allows a mix server to use any secret key as a collateral key).

Protocol 4. TD mix cascade

System initialization. The mix servers jointly generate an ElGamal private/public key pair $(x, y = g^x)$ using a threshold protocol [18]. The public key y is known to all servers. The servers also hold shares of the secret key x, in such a way that a quorum of servers can decrypt.

Every mix server e in the cascade then runs $KC(g, q, y)$ to generate a collateral public key y_e and commit to the corresponding secret key s_e of k bits with a sequence of commitments $[a_i = g^{b_i} h^{r_i}]$, where $1 \leq i \leq k$.

Creation of an input batch. A user randomly draws a value $\gamma_j \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and posts an encryption of her message M_j to the bulletin board:

$$(G_{0,j}, M_{0,j}) = (g^{\gamma_j}, M_j y^{\gamma_j})$$

After collecting n messages on the bulletin board, the mix cascade starts to process the batch.

Re-encryption and mixing.

1. In the ℓ^{th} round of the cascade (for $0 \leq l < k$), mix server e processes its inbound message batch $[(G_{\ell m+e-1,j}, M_{\ell m+e-1,j})]$ by running

   $$TDM(y, b_{\ell,e}, [(G_{\ell m+e-1,j}, M_{\ell m+e-1,j})])$$

   and proves correct execution of the protocol TDM as described in Section 7.
2. After the output batch $[(G_{\ell m,j}, M_{\ell m,j})]$ is produced, mix server m sends the batch back to the head of the cascade if $\ell < k$.

Decryption. A quorum of mix servers jointly decrypt the final output batch and output the corresponding plaintexts.

Performance improvements. To lower the computational cost, a TD-mix cascade need not necessarily mix its inputs exactly as many times as there are bits in the collateral keys of the mix servers. For every message batch, the cascade may instead use only λ randomly chosen bits of the collateral keys of mix servers. The value λ must be large enough to constitute a credible deterrent to individual mix servers' misbehavior.

All the usual techniques commonly used to speed-up the operation of re-encryption mix networks can also be used in our trace-deterring mixnet. For example, mix servers may pre-compute the values $(g^{\gamma_{i,j}}, y^{\gamma_{i,j}})$ used to re-encrypt ciphertexts for all bits $0 \leq i < k$.

Discussion of threats. We construct a mix cascade in a way which interleaves mix servers of different organizations. This prevents a dishonest mix server from unilaterally exposing the end-to-end correspondence of a message across all the permutations it performed without leaking out any correspondences in-between. Specifically, a mix server cannot link an input message to its first permutation to its output from the server's last permutation because the linkage of that message between any of two permutations it performs is interrupted by other mix servers' permutations. However, if all mix servers of a cascade collude, they can offer this end-to-end correspondence to a third party. Our TD mix network deters global trace collusion because a conspiring mix server has to reveal the output of the message under trace to its neighbor, amounting to disclosure of a secret bit. Therefore, the cost of such collusion is to reveal one's secret collateral key to another party.

9 Conclusion

We have presented a deterrent to the voluntary selective disclosure of mix correspondences. This method improves upon previous efforts in three significant ways: *trace disclosures become provable*, *the disclosure penalty is customizable*, and *the anonymity set is large*. We introduce the notions of trace-deterring permutations, formalize the trace-deterring mixing protocol and examine its deployment in a mix network.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. Dr. Wang is supported by the NSF grant IIS-0549313.

References

[1] M. Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Proceedings of EUROCRYPT 1998*, pages 437–447. Springer-Verlag, LNCS 1403, 1998.

[2] M. Abe. Mix-networks on permutation networks. In *Proceedings of ASIACRYPT 1999*, pages 258–273. Springer-Verlag, LNCS 1716, 1999.

[3] A. Acquisti, R. Dingledine, and P. Syverson. On the economics of anonymity. In *Proceedings of Financial Cryptography 2003*, pages 84–102. Springer-Verlag, LNCS 2742, 2003.

[4] A. Boldyreva and M. Jakobsson. Theft-protected proprietary certificates. In *Proceedings of the 2002 Digital Rights Management Workshop*, pages 208–220, 2002.

[5] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. In *Technical report TR 260*. Dept. of Computer Science, ETH Zurich, 1997.

[6] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[7] D. Chaum. E-voting: Secret-ballot receipts: True voter-verifiable elections. 2(1):38–47, Jan./Feb. 2004.

[8] D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In *Proceedings of CRYPTO '86*, pages 200–212. Springer-Verlag, LNCS 263, 1987.

[9] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Proceedings of CRYPTO '92*, pages 89–105. Springer-Verlag, LNCS 740, 1993.

[10] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of CRYPTO 1994*, pages 174–187. Springer-Verlag, LNCS 893, 1994.

[11] G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of the 6th Information Hiding Workshop*, pages 293–308. Springer-Verlag, LNCS 3200, 2004.

[12] Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. In *Proceedings of EUROCRYPT 2000*, pages 557–572. Springer-Verlag, LNCS 1803, 2000.

[13] C. Dwork, J. Lotspiech, and M. Naor. Digital signets: self-enforcing protection of digital information. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 489–498. ACM Press, 1996.

[14] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 25–32. ACM Press, 2000.

[15] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *In Proceedings of CRYPTO '86*, pages 186–194. Springer-Verlag, LNCS 263, 1987.

[16] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of AUSCRYPT 1992*, pages 244–251. Springer-Verlag, LNCS 718, 1992.

[17] J. Furukawa and K. Sako. An efficient scheme for proving a shuffling. In *Proceedings of CRYPTO 2001*, pages 368–387. Springer-Verlag, LNCS 2139, 2001.

[18] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proceedings of EUROCRYPT 1999*, pages 295–310. Springer-Verlag, LNCS 1592, 1999.

[19] P. Golle and M. Jakobsson. Reusable anonymous return channels. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 94–100. ACM Press, 2003.

[20] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 145–160. Springer-Verlag, LNCS 2567, 2002.

[21] M. Jakobsson and A. Juels. An optimally robust hybrid mix network. In *Proceedings of Principles of Distributed Computing 2001*, pages 284–292. ACM Press, 2001.

[22] M. Jakobsson, A. Juels, and P. Q. Nguyen. Proprietary certificates. In *Proceedings of the The Cryptographer's Track at the 2002 RSA Conference on Topics in Cryptology*, pages 164–181. Springer-Verlag, LNCS 2271, 2002.

[23] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of USENIX 2002*, pages 339–353, 2002.

[24] D. Kesdogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In *Proceedings of the 5th Information Hiding Workshop*, pages 53–69. Springer-Verlag, LNCS 2578, 2002.

[25] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing attacks in low-latency mix-based systems. In *Proceedings of Financial Cryptography 2004*, pages 251–265. Springer-Verlag, LNCS 3110, 2004.

[26] A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 2002 ACM Conference on Computer and Communication Security*, pages 116–125. ACM Press, 2001.

[27] A. Neff. Verifiable Mixing (Shuffling) of ElGamal Pairs. Technical report, VOTEHERE, 2003.

[28] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *Proceedings of Information and Communications Security 1997*, pages 440–444. Springer-Verlag, LNCS 1334, 1997.

[29] C. Park, K. Itoh, and K. Kurosawa. All/nothing election scheme and anonymous channel. In *Proceedings of EUROCRYPT 1993*, pages 248–259. Springer-Verlag, LNCS 765, 1993.

[30] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of CRYPTO '91*, pages 129–140. Springer-Verlag, LNCS 576, 1992.

[31] M. Reiter and X. Wang. Fragile mixing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 227–235. ACM Press, 2004.

[32] K. Sako and J. Kilian. Receipt-free MIX-type voting scheme - a practical solution to the implementation of a voting booth. In *Proceedings of EUROCRYPT 1995*, pages 393–403. Springer-Verlag, LNCS 921, 1995.

[33] A. D. Santis, G. D. Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *Proceedings of the IEEE FOCS 1994*, pages 454–465, 1994.

A Circular Permutations

Proposition 4. *Let $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ be a trace-deterring (TD) partition as defined in Section 5. If $\mathcal{T}_0 = \{\mathsf{Id}\}$, then $\mathcal{T}_1 \subseteq \mathcal{C}$, where $\mathcal{C}$ denotes the subsets of permutations on n elements that are circular.*

Proof. Let $\pi \in \mathcal{T}_1$. We must show that π is a circular permutation. The proof is by contradiction. Assume that π has a cycle C of length $\alpha < n$. Then $\mathsf{Id}(C) = \pi(C) = C$ and therefore $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ cannot be a TD-partition. □

Proposition 5. *Let $\mathcal{T}$ denote the set of all permutations on n elements and let $\mathcal{C} \subset \mathcal{T}$ denote the subset of circular permutations on n elements (see Definition 2). Let ν denote the shift permutation on n elements. We have $\mathcal{C} = \{\pi^{-1} \circ \nu \circ \pi \mid \pi \in \mathcal{T}\}$.*

The proof follows from the two following lemmas:

Lemma 6. *For all $\pi \in \mathcal{T}$, the permutation $\pi \circ \nu \circ \pi^{-1}$ is a circular permutation.*

Proof. The proof is by contradiction. Let us assume that the successive images of the input 1 by the permutation $\pi \circ \nu \circ \pi^{-1}$ are not all different. Then there exist $i, j \in \{1, \ldots, n\}$ such that $i \neq j$ and

$$(\pi \circ \nu \circ \pi^{-1})^{(i)}(1) = (\pi \circ \nu \circ \pi^{-1})^{(j)}(1)$$

But $(\pi \circ \nu \circ \pi^{-1})^{(i)} = \pi \circ (\nu^{(i)}) \circ \pi^{-1}$ and so the equation above can be rewritten as

$$\pi \circ (\nu^{(i)}) \circ \pi^{-1}(1) = \pi \circ (\nu^{(j)}) \circ \pi^{-1}(1).$$

It follows that $(\nu^{(i)})(\pi^{-1}(1)) = (\nu^{(j)})(\pi^{-1}(1))$, which is a contradiction since ν is a circular permutation. □

Lemma 7. *Let $\sigma \in \mathcal{C}$ be a circular permutation. There exists $\pi \in \mathcal{T}$ such that $\sigma = \pi \circ \nu \circ \pi^{-1}$.*

Proof. The proof is constructive. Let σ be a circular permutation on n elements. For $i \in \{1, \ldots, n\}$, let us define $\pi(i) = \sigma^{(i)}(1)$. We must prove that π thus defined is a permutation and that $\sigma = \pi \circ \nu \circ \pi^{-1}$.

We show first that π is a permutation. Let $i, j \in \{1, \ldots, n\}$ such that $i \neq j$. Since σ is a circular permutation, we have $\sigma^{(i)}(1) \neq \sigma^{(j)}(1)$, and therefore $\pi(i) \neq \pi(j)$. This shows that π is a bijection, and therefore a permutation of the set $\{1, \ldots, n\}$.

Next, we show that $\sigma = \pi \circ \nu \circ \pi^{-1}$. Observe that for $i \in \{1, \ldots, n\}$, we have

$$\sigma \circ \pi(i) = \sigma \circ \sigma^{(i)}(1) = \sigma^{(i+1)}(1) = \pi(i+1) = \pi \circ \nu(i)$$

and therefore $\sigma \circ \pi = \pi \circ \nu$. It follows that $\sigma = \pi \circ \nu \circ \pi^{-1}$. □