

Deterring Voluntary Trace Disclosure in Re-encryption Mix-Networks

XIAOFENG WANG and PHILIPPE GOLLE
PARC
and
MARKUS JAKOBSSON and ALEX TSOW
Indiana University

Mix-networks, a family of anonymous messaging protocols, have been engineered to withstand a wide range of theoretical internal and external adversaries. An undetectable insider threat—voluntary partial trace disclosures by server administrators—remains a troubling source of vulnerability. An administrator's cooperation could be the resulting coercion, bribery, or a simple change of interests. While eliminating this insider threat is impossible, it is feasible to deter such unauthorized disclosures by bundling them with additional penalties. We abstract these costs with collateral keys, which grant access to customizable resources. This article introduces the notion of trace-detering mix-networks, which encode collateral keys for every server-node into every end-to-end message trace. The network reveals no keying material when the input-to-output transitions of individual servers remain secret. Two permutation strategies for encoding key information into traces, mix-and-flip and all-or-nothing, are presented. We analyze their trade-offs with respect to computational efficiency, anonymity sets, and colluding message senders. Our techniques have sufficiently low overhead for deployment in large-scale elections, thereby providing a sort of publicly verifiable privacy guarantee.

Categories and Subject Descriptors: C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks—*General: Security and protection (firewall)*; K.4.1 [**Computer Milieux**]: Computers and Society—*Public Policy Issues: Privacy*

General Terms: Security

Additional Key Words and Phrases: Anonymous messaging, electronic voting, insider threat, re-encryption mix-network, zero-knowledge protocol

ACM Reference Format:

Wang, X., Golle, P., Jakobsson, M., and Tsow, A. 2010. Deterring voluntary trace disclosure in re-encryption mix-networks. *ACM Trans. Info. Syst. Sec.* 13, 2, Article 18 (February 2010), 24 pages. DOI = 10.1145/1698750.1698758 <http://doi.acm.org/10.1145/1698750.1698758>

Authors' addresses: xw7@indiana.edu, pgolle@parc.com, markus@indiana.edu, atsow@cs.indiana.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org. © 2010 ACM 1094-9224/2010/02-ART18 \$10.00

DOI 10.1145/1698750.1698758 <http://doi.acm.org/10.1145/1698750.1698758>

ACM Transactions on Information and System Security, Vol. 13, No. 2, Article 18, Publication date: February 2010.

1. INTRODUCTION

Mix-servers transform sets of input messages by probabilistic encryption methods and subsequently by permutation of their output order. In a mix-network, messages are sent through a sequence of mix-servers. So long as the permutation of at least one server remains secret, the system prevents linking between message senders and receivers.

Voluntary selective disclosure of mix-traffic has recently been recognized as an emerging threat [Acquisti et al. 2003; Reiter and Wang 2004]. Here, the attacker secures a mix-server administrator’s cooperation by social coercion (e.g., bribery) in order to obtain selected input-to-output correspondences, or traces. End-to-end traces link message senders and receivers by following message paths through each and every node in the mix-network. One can distinguish between trace assertions—where input and output messages are simply correlated—and trace proofs, which augment assertions with mathematically verifiable reasoning. This distinction is unimportant for our protocol, and we will—for simplicity—assume that traces simply assert relationships without substantiating evidence.

The notion of trace can be generalized from the previously described message-to-message relationships to subset-to-subset relationships. In particular, a mix-server can assert correspondences between input and output subsets without revealing individual correspondences among messages. Moreover, subset trace assertions have the side effect of simultaneously asserting correspondences between their set complements as well. Thus, every nontrivial subset trace assertion reduces the anonymity set of itself or its complement by at least half. For this reason, all traces are undesirable and trace deterrence must apply to both individual and subset disclosures.

In this article, we address the problem of voluntary trace disclosure by binding the release of sensitive collateral information to trace exposure. This strategy increases the cost of revealing traces across individual mix-servers. A natural candidate for collateral is the server’s secret mix-key. Its exposure would spoil the secrecy of all mixing performed under the key, including the remaining correspondences from the partial trace that led to its revelation. In many cases, this incentive is an inappropriate deterrent. The information guarded by our collateral key scheme is customizable. The information cache may even be empty: Collateral key disclosure in this case simply provides publicly verifiable evidence of a server’s privacy breach. This approach seems well suited for privacy-critical applications, such as electronic elections (e.g., see Chaum [2004]).

We emphasize the following properties of our trace-detering protocol for re-encryption mix-networks.

—In the context of electronic elections, exposing traces does not link other voters to their ballots as in previous anticoercion schemes (e.g., Reiter and Wang [2004]). While it is possible to embed mix-server keys into a trace, their collateral keys may correspond to any agreed on public key. The holder of an end-to-end trace learns the collateral keys for all mix-servers. Their demonstration (via zero-knowledge protocols) constitutes irrefutable evidence of a

network-wide privacy breach.

- Our model considers an adversary that either infiltrates or silently coerces individual mix-servers to write any selected information to a tape, but is unable to force them to participate in a general multiparty computation (MPC) that computes and proves the validity of a given end-to-end trace without revealing their intermediate steps or any secret information. This assumption is realistic because a general MPC is typically very inefficient, involving lots of interactions and a large amount of traffic, which significantly raises the colluders' risks of being caught by auditing agencies. In Section 6, we also demonstrate that our technique can either force colluders to perform a large-scale MPC when the adversary is active (capable of injecting messages into our mix-net) or discourage any uncorrupted server from participating in such a collusion when the adversary is passive.
- 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. Since collateral key exposure constitutes proof of compromise, server administrators have a direct incentive to proactively eliminate security threats rather than stonewall the existence of problems.

Our trace-detering framework embeds collateral keys into all end-to-end traces through the mix-network. This property arises from the fusion of individual node behavior and network organization. Two zero-knowledge protocols—a key-commitment and a permutation-selection proof—ensure adherence to the mixing discipline.

At the node level, servers execute trace-detering (TD) transformations. The protocol forces mix-servers to choose permutations from one of two disjoint sets, encoding one bit of the collateral key; for example, a permutation from the first set encodes a 0 while a permutation from the other encodes a 1. While the collateral key determines the permutation set, selection within the set is uniformly random. Trace-detering permutations have the property that every nontrivial subset (i.e., nonempty and nonuniverse) correspondence between inputs and outputs determine their membership in one of the two bit-encoding sets; thus, every trace exposes a key-bit.

At the network level, mixing operates in rounds (see Figure 4). Each server encodes one bit of its collateral key into the trace per round. Thus, fully exposing an n -bit collateral key requires n rounds. A network with k servers produces end-to-end traces that are $k \cdot n$ steps long, encoding collateral keys for all servers. In practice, full key exposure is unnecessary; fewer rounds will compromise enough of each collateral key to deter trace disclosure.

This article presents two transformation techniques. The first approach, mix-and-flip, transforms inputs in two stages. The mix stage divides input messages into two equal-sized sets, mixing each independently in the conventional manner. The flip stage encodes a key bit by either transposing or freezing the input halves. Permutations in the mix stage are fully random and arbitrary within the halves. Enforcing mix-and-flip transformations has the

same overhead as verifying conventional re-encryption mixing. An extension of mix-and-flip encodes multiple bits per round, at the cost of reducing the anonymity set per node; this insight can reduce the number of mixing rounds necessary for effective deterrence. Though efficient, our analysis exposes some weaknesses with respect to subset traces. We explore ways to minimize its impact.

The next approach, all-or-nothing, eliminates mix-and-flip's vulnerability to subset-traces. All-or-nothing transformations come from either the set of permutations that change all message positions or the singleton set containing the identity (the nothing-set). Any nontrivial subset trace exposes membership to one of these two sets, thereby providing an effective coding medium for collateral key bits. Zero-knowledge enforcement of proper key-encoding leverages the group-theoretic observation that the all-set is characterized by the conjugates of any cyclic generator, while the nothing-set is characterized by the conjugates of the identity.

Organization. The rest of the article 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 describes the mix-and-flip approach, its weaknesses, and workarounds. Section 5 presents the notion of trace-detering partition and the all-or-nothing mixing technique. Section 6 shows how to construct a trace-detering mix-network and analyzes its security properties. Section 7 concludes the article.

2. RELATED WORK

Chaum [1981] first formalized mixing, 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 does not prevent message senders from observing the traces of their own messages. 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 [Park et al. 1993; Abe 1998] 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 builds on re-encryption mixing, which conceals traces—and consequently the collateral keys—from message senders.

Since they were first proposed, mix-networks have been building blocks in strong electronic election schemes [Chaum 1981; Fujioka et al. 1992; Park et al. 1993; Sako and Kilian 1995; Jakobsson et al. 2002]. In this context, robustness—where each server provides a proof or strong evidence for its honest

behavior—has parity with unlinkability. For example, Ogata et al. [1997] use cut and choose techniques to achieve robust mix-networks. Subsequent schemes improve both the efficiency of zero-knowledge proofs [Jakobsson and Juels 2001] and attain universal verifiability [Abe 1998, 1999], that is, verifiability by third-party observers. Other protocols employ layer redundancy [Desmedt and Kurosawa 2000] and random partial checking [Jakobsson et al. 2002] to achieve robustness.

No mixing protocol prevents administrators from logging and later divulging input-to-output correspondences performed by their machines. This form of voluntary disclosure is an undetectable attack. So far, the only defense 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 [Reiter and Wang 2004], 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-detering 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-detering technique does not constrain the permutation selection to the same extent fragile mixing does. 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.

Other research is also related to the voluntary disclosure problem. For example, proprietary certificates [Jakobsson et al. 2002; Boldyreva and Jakobsson 2002] 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 et al. [1996] 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 [Danezis and Serjantov 2004], intersection [Kesdogan et al. 2002], and timing attacks [Felten and Schneider 2000; Levine et al. 2004] 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 on the correctness of several proofs of knowledge and commitment schemes: equal discrete logarithms [Chaum and Pedersen 1993], knowledge of discrete logarithm, verifiable shuffling [Furukawa and Sako 2001; Neff 2001; Groth 2002], and splitting techniques from Pedersen’s noninteractive secret sharing [Pedersen 1992].

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, a device that publishes the messages from mix-servers. In addition, mix-servers also have append-only 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 article, we make the standard assumption that the corrupting adversary is static.

Coercion and Collusion. In addition to being able to corrupt any nonquorum of servers, we also allow the adversary to coerce individual mix-servers. However, we assume that she cannot coerce all the servers to participate in an MPC that reveals the end-to-end trace of a message. The rationale behind this assumption comes from the distrust among colluding parties who do not believe that other parties will honestly follow the MPC protocol. As a result, the attack models that enable relatively efficient MPC, such as honest majority [Damagard and Ishai 2005] and covert adversary [Canetti and Ostrovsky 1999], can be insufficient to attract colluders, who would rather opt for the very expensive general MPC [Katz et al. 2003]. This also increases the risks that such activities are caught by the parties like auditors. Actually, we show in Section 6 that when only the passive adversary is considered, our technique can completely deter a mix-server from joining such a collusion: This is because once an end-to-end trace of a message is revealed, the same MPC can also be utilized to disclose any participant's secret. When the adversary is active, our approach can control the scale of the MPC colluders need to perform.

Honest Verifier. Our techniques are based on several existing zero-knowledge proofs that assume the presence of honest verifiers who follow the protocol properly. The verifiers can be the parties that use the mix-network, for example, voters. Since all the zero-knowledge techniques we propose can be used in a noninteractive manner in the random oracle model [Fiat and Shamir 1987], those parties can check the information posted on the bulletin board after their messages have been delivered. Similar assumptions are widely used in the research on robust mix-networks [Ogata et al. 1997; Jakobsson and Juels

2001; Abe 1998, 1999].

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 $x \xleftarrow{R} \mathbb{Z}_q^*$. The public key, y , is the value $g^x \in \mathcal{G}_q$. To encrypt a message $m \in \mathcal{G}_q$, one chooses $\gamma \xleftarrow{R} \mathbb{Z}_q^*$ and evaluates the ordered pair $(g^\gamma, m y^\gamma)$. 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 $\delta \xleftarrow{R} \mathbb{Z}_q^*$ and evaluating $(G g^\delta, M y^\delta)$. The decryption method remains the same. Decryption is a homomorphism from the pair-wise 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). [Chaum et al. 1987] 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 Chaum et al. [1987] is one modular exponentiation for \mathcal{P} and two modular exponentiations for \mathcal{V} .

Proof of Correct Re-encryption (PCR). [Chaum and Pedersen 1993] 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 two modular exponentiations for \mathcal{P} and four modular exponentiations for \mathcal{V} .

Discrete Logarithm Proof Systems. [Camenisch and Stadler 1997; Cramer et al. 1994; Santis et al. 1994] An efficient zero-knowledge proof can be constructed for any monotone boolean formula whose atoms consist of the protocols KDL or PCR . This article 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 5.3).

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:

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

Verifiable Mixing. Verifiable mixing protocols [Furukawa and Sako 2001; Groth

2002; Neff 2003] allow a mix-server to prove to a verifier that $L' = \text{MIX}_\pi(L)$. 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 [Groth 2002] 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 $L'_0 = \text{MIX}_{\pi_0}(L_0)$ and $L'_1 = \text{MIX}_{\pi_1}(L_1)$. 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 pair-wise product of ciphertexts of both mixes (see Golle and Jakobsson [2003] for details). 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. MIX-AND-FLIP

4.1 Overview

Mix-and-flip transforms decompose the mixing of an input batch into two steps. The mix step individually mixes the top and bottom input halves. The flip step permutes the aggregate outputs of the two halves using the identity (a 0-flip) or transposition (a 1-flip). Thus, a single input-to-output correspondence for a mix-and-flip transform reveals the transform as a 0-flip or a 1-flip: If the input and output positions are in the same half, it is a 0-flip, otherwise a 1-flip. Note that the privacy protection this approach offers to individual messages is an anonymity set involving half of the batch.

Collateral Key Commitment (Section 4.2). The collateral key commitment (KC) protocol is run only once in a set-up 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. Specifically, the prover broadcasts the public key and makes information theoretically secure commitments to the bits of the secret key; that is, numbers of the form $a_i = g^{x_i}h^{r_i}$ for $x_i \in \{0, 1\}$ and random r_i .

Mix-and-Flip Transformation (Section 4.3). The protocol allows a server to randomly permute at the mix stage, but forces the flip action to match its collateral key commitment: The verifier maps the aggregate-ciphertext flip to every bit on the collateral key, and then the prover shows its equivalence with the commitment of that bit.

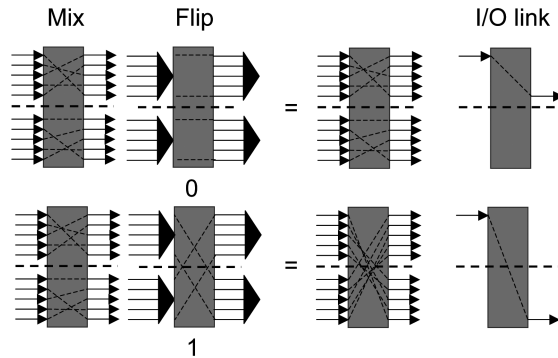


Fig. 1. Mixing is applied randomly to both halves. The first row shows the action of a 0-flip on the halves, while the second row illustrates a 1-flip. The link of the first input reveals a 0-flip when its output is in the top half, or a 1-flip when in the bottom half.

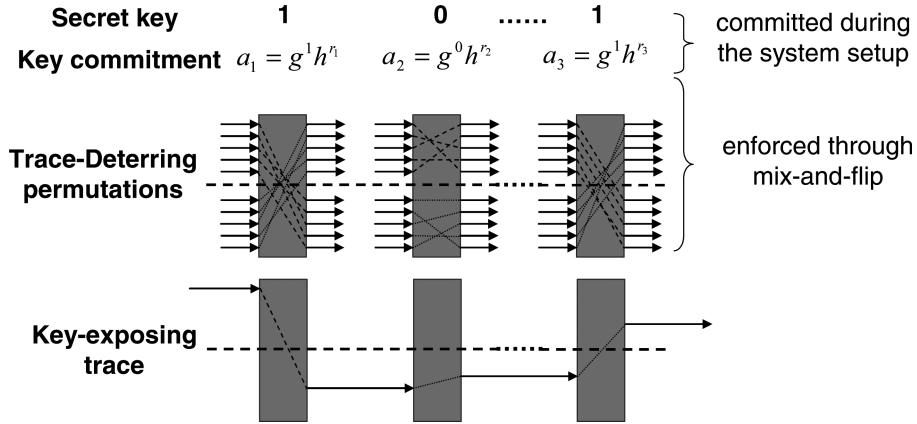


Fig. 2. Mix-and-flip. In the figure, a secret key string $10 \dots 1$ is committed bit by bit. A mix server proves this correspondence to a verifier through the mix-and-flip protocol. When mixing a batch of $10 \dots 1$ messages, the server performs the mix-and-flip transformations according to bit commitments: the first mix server performs a 1-flip, the second performs a 0-flip, \dots , the last performs a 1-flip. Later on, if the mix server helps trace a message, it has to expose the sequence: top \rightarrow bottom \rightarrow bottom \dots bottom \rightarrow top, which represents $10 \dots 1$, the secret bit string.

Trace-Deterring Mix-Network (Section 6). Since there is only 1 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 an end-to-end correspondence of a message to a third party without revealing the correspondences of intermediate transformations. 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.

4.2 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 $(y_e, [a_i]) \leftarrow \text{KC}(g, q, y)$. We note that the KC protocol only needs to be executed once during the system bootstrap stage.

PROTOCOL 1. *Key commitment protocol* $(y_e, [a_i]) \leftarrow \text{KC}(g, q, y)$

- \mathcal{P} generates a secret key $s_e \xleftarrow{R} \mathbb{Z}_q^*$, 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 $h \xleftarrow{R} \mathcal{G}_q \setminus \{1\}$ 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 $r_i \xleftarrow{R} \mathbb{Z}_q^*$ 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 $(\text{KDL}_h \{a_i\} \vee \text{KDL}_h \{a_i/g\})$. As noted in Section 3.2, an efficient proof can be constructed for this boolean formula.
- (2) \mathcal{P} computes $A = (a_0)^{2^0} \cdot (a_1)^{2^1} \cdot \dots \cdot (a_{k-1})^{2^{k-1}}$. Note that $A = g^{s_e} h^R$, where $R = \sum 2^i r_i$.
- (3) \mathcal{P} proves $\text{KDL}_h \{A/y_e\}$ to \mathcal{V} using knowledge of R .

The commitment scheme used in Protocol 1 is well known to be binding and zero-knowledge (see Pedersen [1992]).

4.3 One Round of Mix-and-Flip

Each server e performs a sequence of mix-and-flip (or M & F) transformations on a batch of input messages according to the sequence of flip commitments. Since a single input-to-output correspondence discloses a secret bit, the administrator of the mix-server will then be deterred from helping a third party to trace even a single message.

PROTOCOL 2. *Mix-and-Flip* $M\&F(y, a_i, [(G_j, M_j)])$

Stage 1. *Mixing the input batch.* A mix-server e chooses a permutation π_i , which randomly permutes messages within a half and maps input halves to the output halves according to b_i . Then, mix-server e generates an output batch $L' = [(G'_j, M'_j)]$ ($0 \leq j < n$) from the input batch $L = [(G_j, M_j)]$ such that $L' = \text{MIX}_{\pi}(L)$.

Stage 2. *Generating the proof for a correct mixing.*

- (1) \mathcal{P} proves to \mathcal{V} of the correct mixing of the input batch by running $PVM\{L \rightsquigarrow L'\}$.
- (2) \mathcal{P} proves to \mathcal{V} of the correct mapping from the permutation π to the bit commitment a_i : Formally, let

$$F = PCR \left\{ \left(\prod_{0 \leq j < \frac{n}{2}} G_j, \prod_{0 \leq j < \frac{n}{2}} M_j \right) \rightsquigarrow \left(\prod_{0 \leq j < \frac{n}{2}} G'_j, \prod_{0 \leq j < \frac{n}{2}} M'_j \right) \right\}$$

$$F' = PCR \left\{ \left(\prod_{0 \leq j < \frac{n}{2}} G_j, \prod_{0 \leq j < \frac{n}{2}} M_j \right) \rightsquigarrow \left(\prod_{\frac{n}{2} \leq j < n} G'_{j+1}, \prod_{\frac{n}{2} \leq j < n} M'_{j+1} \right) \right\}$$

\mathcal{P} proves to \mathcal{V} the following formula.

$$(KDL_h \{a_i\} \wedge F) \vee (KDL_h \{a_i/g\} \wedge F').$$

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

In practice, it may not be enough to verify the correct operation of the mix as we propose to do earlier in the text by computing the product of all inputs in each partition and confirming the validity of the mix's operation on the products. A cheating server could exploit same-product subsets in the partitions to subvert this check. However, finding these subsets appears to be hard in the random case¹ (it is equivalent to the NP-complete knapsack problem), and we can also extend the naive check mentioned earlier to eliminate potential attacks against same-product verification (this extension, however, complicates the protocol and adds to its computational cost). Aside from this concern, the completeness, soundness, and zero-knowledge properties of Protocol 2 are directly derived from the primitives it uses, which include KDL, PCR, and the discrete logarithm proof system. This is possible due to the serial nature of the composition of the building blocks, and the fact that the building blocks of the composite proof are independent but for relating to the same collection of input and output variables associated with the statement to be proven. (For serially composed zero-knowledge proofs not to be zero-knowledge, they need to share what we may think of as internal variables.)

Computational Complexity of the M & F Protocol. Let n denote the number of ciphertext inputs (i.e., the number of elements in the list L_0).

- The cost of Step 1 of the M & F protocol is the same as that of a normal round of mixing (Section 3.3), which is $2n$ modular exponentiations.
- The cost of Step 2 of the protocol is the cost of one verifiable mixing, which includes $18n$ modular exponentiations for both the prover and the verifier, plus the cost of the proof for the Boolean formula, which includes seven exponentiations for the prover and eight for the verifier. Altogether, the computational complexity of the M&F protocol is $18n + 7$ for the prover and $18n + 8$ for the verifier.

¹Note that the mix-server does not have control of the content of the messages and needs to collude with all its predecessors to change the product of G_j .

Encoding More than One Bit in One Mixing Round. In the M&F protocol, the flip step of a mix-and-flip transformation reveals one bit of secret information at a time. This technique can generalize to reveal more information per step by increasing the number of partitions; for example, the independent mixing of four equal-sized sets followed by their aggregate permutation encodes two bits; L equal partitions encodes $\log L$ bits. In this setting, the flips are cyclic permutations over the number of partitions. Cyclic permutations can be proved in zero knowledge using the technique in Reiter and Wang [2004]. This treatment, however, reduces the anonymity set for every additional bit.

4.4 An Attack and Mitigation

A serious threat to the mix-and-flip network is anonymity tracing: An attacker may request the simultaneous trace of a target message in the upper half and a comessage in the lower half. Assuming that the partitions are the same for each server, for example, upper and lower halves, the simultaneous trace of the two messages will show one message in both halves at each step along the way. This eliminates the disclosure of secret bits by concealing the flip, while reducing the anonymity set from n to 2.

A simple countermeasure is to let the bulletin board randomly reorder the messages output from every mix-server before the next mix-server processes them.² In this scheme, the target message and comessage will end up in the same partition with a high probability before the end of the network. Once the two share the same partition, the next step in the trace reveals the cheating server's choice of flip. To counter this, the attacker chooses two more cover messages in the empty half (for a total of four messages), and requests their output correspondences. The server can comply without revealing the flip; however, the anonymity set has doubled. In this way, the size of anonymity set could increase quickly.

When there are more partitions, a mix-server participating in the anonymity tracing attack must reveal equal number of messages in every partition to protect the confidential bits of its collateral key. The anonymity set in this case could be expanded even faster by randomly reordering output messages between two mixings. To evaluate the efficacy of this simple countermeasure, we did a simulation with Matlab, which demonstrates that the average size of the anonymity set increases quickly. For example, given an input batch of 1,000 messages, a two partition mix-and-flip cascade yielded an anonymity set with more than 740 messages on average after mixing the batch 80 times; in case of 1,000,000 input messages and 16 partitions (which encodes 4 bits), an anonymity set of 278,288 was reached in 160 mixings. Therefore, we tend to believe that treatment could effectively mitigate the threat of anonymity tracing in most practical applications, including electronic voting. The detailed results of the simulation are presented in Table I.

²This can be done, for example, by using a random hash function to compute a hash value from the output batch of a mix-server and according to that value, picking up a permutation from a public permutation set. Given the published output batch, permutation set and hash function, the correctness of such a reordering can be trivially verified by message senders (e.g., voters).

Table I. Efficacy of Random Reordering Against the Anonymity Tracing Attack

Size of the inputs	Number of partitions	Number of mixings	Size of anonymity set
1,000	2	40	220
1,000	2	80	743
1,000	8	30	1,000
100,000	8	80	24,839
100,000	8	160	73,552
1,000,000	16	80	78,560
1,000,000	16	160	278,288

The results were averaged over 100 independent trials.

5. TRACE-DETECTING MIXING

5.1 Overview

In this section, we present another technique to eliminate the threat of subset-tracing at the cost of the performance overhead and the capability to encode multiple bits in one mixing round. Here, we give a broad overview of this approach, which is called *trace-detecting mixing*. 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-detecting 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 leftover permutations that are never used by the mix-server.

Definition 1 (Trace-detecting 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-detecting 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, nonempty) 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 nonempty 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

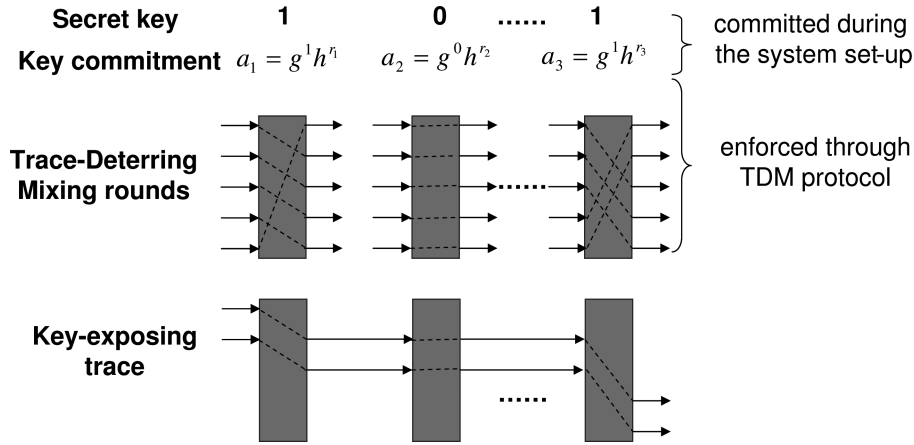


Fig. 3. Trace-detering protocol.

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, nonempty) 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 [Reiter and Wang 2004]), 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 3 illustrates the idea. In the figure, a collateral secret key string $10\dots 1$ is committed through a key commitment scheme (Section 4.2). 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 5.2). 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.

All-or-Nothing Partition (Section 5.2). This article defines a specific TD-partition, all-or-nothing, 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. Formally, $|\mathcal{T}_0 \cup \mathcal{T}_1| \geq \Gamma$, where Γ is a threshold for the number of permutations. Note that the larger the threshold is, the more difficult an adversary will be in identifying the output of an input message.

- (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 in zero-knowledge 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 4.2). The collateral key commitment protocol is identical to that of the mix-and-flip mixing.

All-or-Nothing Mixing Round (Section 5.3). One round of all-or-nothing mixing ($A|N$) binds one bit of a server’s collateral key to one permutation applied to a batch of messages. The $A|N$ mixing protocol does not consume any keying information and can be run a polynomial number of times for KC. In $A|N$, 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 $A|N$ protocol correctly.

5.2 All-or-Nothing Partition

Throughout this section and the rest of this article, 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 $\text{ld} \in \mathcal{T}$ 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 article, 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 $\mathcal{T}_* = \mathcal{T} - (\mathcal{C} \cup \text{ld})$ denote permutations that are neither circular nor the identity. The partition $(\{\text{ld}\}, \mathcal{C}, \mathcal{T}_*)$ 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 $\text{ld}(M) \neq \pi(M)$. We have $\text{ld}(M) = M$. We also

have $\pi(M) \neq M$, otherwise, π would have a cycle of length strictly less than n , contradicting the assumption that π is a circular permutation. It follows that $\text{ld}(M) \neq \pi(M)$. \square

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 $(\{\text{ld}\}, \mathcal{C}, \mathcal{T}_*)$ 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 $(\{\text{ld}\}, \mathcal{C}, \mathcal{T}_*)$ is of maximal size in the following sense: Any TD-partition $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ such that $\mathcal{T}_0 = \{\text{ld}\}$ 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 $\nu(i) = i + 1 \pmod n$. 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 4 (Section A) shows $\mathcal{C} = \{\pi^{-1} \circ \nu \circ \pi \mid \pi \in \mathcal{T}\}$. 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 5.3 to let a mix-server prove that a batch of inputs was mixed correctly according to our TD-partition.

5.3 One Round of All-or-Nothing Mixing

This section introduces a protocol to perform one round of A/N mixing. In $A|N$, 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 to the TD-partition $(\{\text{ld}\}, \mathcal{C}, \mathcal{T}_*)$ of Section 5.2. Finally, the server outputs the permuted list and proves to an honest verifier that it executed the $A|N$ 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 $A|N$ as $A|N(y, b_i, L_0)$, 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 3. *All-or-Nothing Mixing $A|N(y, b_i, L_0)$*

- (1) *The mix-server e chooses a permutation π_i uniformly at random from \mathcal{T} . The mix-server computes $L_1 = \text{MIX}_{\pi_i}(L_0)$ and outputs L_1 .*
- (2) *The server outputs a list L_2 defined as follows.*
 - *If $b_i = 0$, the server defines $L_2 = \text{MIX}_{\text{ld}}(L_1)$.*
 - *If $b_i = 1$, the server defines $L_2 = \text{MIX}_{\nu}(L_1)$, where ν is the shift permutation defined in Section 5.2.*
- (3) *The server computes $L_3 = \text{MIX}_{\pi_i^{-1}}(L_2)$ and outputs L_3 .*

Soundness. We prove first that the protocol $A|N$ is sound. More precisely, we prove that $A|N$ 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

$$L_3 = \text{MIX}_{\pi_i^{-1}} \left(\text{MIX}_{\text{Id}} (\text{MIX}_{\pi_i} (L_0)) \right)$$

and, thus, $L_3 = \text{MIX}_{\text{Id}} (L_0)$. 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

$$L_3 = \text{MIX}_{\pi_i^{-1}} \left(\text{MIX}_{\nu} (\text{MIX}_{\pi_i} (L_0)) \right).$$

By Proposition 4, 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 $A|N$ protocol correctly. The proof proceeds as follows:

PROTOCOL 4. *Generating a Proof of Correct Execution of $A|N$*

- (1) *To prove correct operation in Steps 1 and 3 of the $A|N$ protocol, \mathcal{P} first proves to \mathcal{V} the correctness of the mixing.*

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

\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

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

- (2) *\mathcal{P} proves to \mathcal{V} correct operation in Step 2 as follows: Recall from Section 4.2 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 $j = 0, \dots, n-1$,

—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 $j = 0, \dots, n-1$ (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

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

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

\mathcal{P} proves to \mathcal{V} the following formula.

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

This protocol can be converted into a noninteractive version in the random oracle model by using the Fiat-Shamir heuristic [Fiat and Shamir 1987].

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 A|N 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 A|N 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 Camenisch and Stadler [1997], 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 A|N protocol is, thus, $24n$ modular exponentiations for the prover \mathcal{P} and $22n$ modular exponentiations for the verifier \mathcal{V} .

6. A TRACE-DETECTING MIX-NETWORK

In this section, we discuss how to construct a complete trace-detecting mix-network using as a building block the $M\&F$ protocol (Section 4.3) or the A|N protocol (Section 5.3). Our trace-detecting 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 4.1, a mix-server must perform a sequence of $M\&F$ -mixing or A|N-mixing operations over a batch of input messages, each corresponding to one bit of its collateral secret key. If all these mixings are executed consecutively, the mix-server can disclose to a third party the input of a message to the first mixing and its output of the last 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 4 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 $11 \dots 0$ in the first loop and $10 \dots 1$ 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

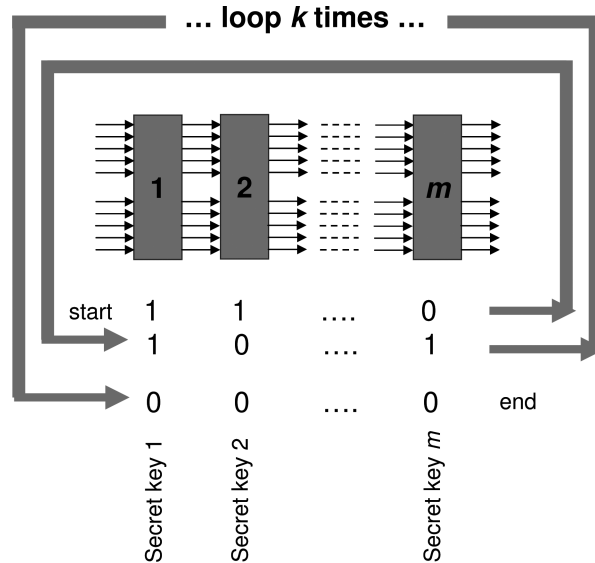


Fig. 4. A trace-detering mix-cascade.

from colluding.

We describe a trace-detering mix-cascade protocol built on Abe’s scheme [Abe 1998]. In an initial set-up 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: (i) re-encryption and mixing and (ii) 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 $M&F$ protocol or the $A|N$ protocol, and proves correct execution of the protocol, as discussed in Section 4.3 and Section 5.3. 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 5. *TD Mix-Cascade*

System Initialization. The mix-servers jointly generate an ElGamal private/public key pair $(x, y = g^x)$ using a threshold protocol [Gennaro et al. 1999]. 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 $\gamma_j \xleftarrow{R} \mathbb{Z}_q^*$ 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 $0 \leq \ell < k$), mix-server e processes its inbound message batch

$[(G_{\ell m+e-1,j}, M_{\ell m+e-1,j})]$ by running

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

or

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

and proves correct execution of the protocols, as described in Sections 4.3 and 5.3.

- (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-detering mix-net. For example, mix-servers may precompute 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 that 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.

Our attack model assumes away the conspiracy attack in which all mix-servers participate in an MPC to extract an end-to-end trace (Section 3.1). Such a threat can actually be mitigated by proper construction of mix-cascades. If we are only worried about the passive adversary who cannot send test messages through the mix-network, a cascade can be built in a way that it transmits a message within a single loop, based on the commitment of a single secret bit from each mix-server.³ On such a cascade, individual servers will have no incentives to join an MPC conspiracy as the attack also jeopardizes the secrecy of their own key bits: Consider Server i ($1 < i < m$) that helps reveal a trace; the same trace can be used to identify its key bit through an MPC between Server 1 and $i - 1$ that discloses i 's input, and one between $i + 1$ and m that reveals i 's output. In the presence of the active adversary, a loop-based cascade is needed to protect servers' key bits. In this case, we can use a long cascade to make an MPC very expensive, as the collusion needs to involve at least two loops to ensure that each server will be encountered twice. This is necessary for convincing individual servers that their bits will not be exposed, as described earlier in the text.

7. CONCLUSION

This article makes several contributions. We introduce a model for a coercive, but stealthy, adversary. We abstract the notion of a customizable disclosure penalty with collateral keys. Collateral keys may guard an arbitrary resource of value and need not expose other messages from the network. Moreover, they behave as a publicly verifiable proof of disclosure of trace information. We submit a general strategy, trace-detering partitions, for encoding collateral key bits into partial trace information. Two specific transformation strategies, *mix-and-flip* and *all-or-nothing*, embed collateral key bits into partial trace information. Although mix-and-flip is vulnerable to subset-tracing attacks, a random and publicly visible re-ordering of messages on the bulletin board makes penalty-free end-to-end subset tracing highly improbable. We further show how to embed multiple key bits into mix-and-flip traces. All-or-nothing transformations completely eliminate the subset-tracing attack. Finally, we illustrate a full trace-detering mix-network, which mixes messages in rounds of interleaved servers. The computational costs are well within reach of current technology for high-assurance privacy applications.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments.

APPENDIX

A. CIRCULAR PERMUTATIONS

PROPOSITION 3. *Let $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ be a $A|N$ partition as defined in Section 5.2. If $\mathcal{T}_0 = \{\text{Id}\}$, then $\mathcal{T}_1 \subseteq \mathcal{C}$, where \mathcal{C} denotes the subsets of permutations on n elements that are circular.*

³Other bits are committed when the servers are sending other batches of messages.

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 $\text{ld}(C) = \pi(C) = C$ and, therefore, $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_*)$ cannot be a TD-partition. \square

PROPOSITION 4. *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 5. *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 $i, j \in \{1, \dots, n\}$ 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 previous equation 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. \square

LEMMA 6. *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 $i \in \{1, \dots, n\}$, 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 $i, j \in \{1, \dots, n\}$ 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 $\{1, \dots, n\}$.

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

REFERENCES

- ABE, M. 1998. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'98)*. Springer-Verlag, Berlin, 437–447.
- ABE, M. 1999. Mix-networks on permutation networks. In *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT'99)*. Springer-Verlag, Berlin, 258–273.
- ACQUISTI, A., DINGLEDEINE, R., AND SYVERSON, P. 2003. On the economics of anonymity. In *Proceedings of the 7th Annual Financial Cryptography (FC'03)*. Springer-Verlag, Berlin, 84–102.
- BOLDYREVA, A. AND JAKOBSSON, M. 2002. Theft-protected proprietary certificates. In *Proceedings of the 2002 Digital Rights Management Workshop*. 208–220.

- CAMENISCH, J. AND STADLER, M. 1997. Proof systems for general statements about discrete logarithms. Tech. rep. TR 260. Dept. of Computer Science, ETH Zurich.
- CANETTI, R. AND OSTROVSKY, R. 1999. Secure computation with honest-looking parties: What if nobody is truly honest? In *Proceedings of the ACM Symposium on the Theory of Computing (STOC'99)*. ACM, New York, 255–264.
- CHAUM, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2, 84–88.
- CHAUM, D. 2004. E-voting: Secret-ballot receipts: true voter-verifiable elections. *IEEE Secur. Privacy* 2, 1, 38–47.
- CHAUM, D., EVERTSE, J.-H., VAN DE GRAAF, J., AND PERALTA, R. 1987. Demonstrating possession of a discrete logarithm without revealing it. In *Proceedings of the Cryptology Conference (CRYPTO'86)*. Springer-Verlag, Berlin, 200–212.
- CHAUM, D. AND PEDERSEN, T. P. 1993. Wallet databases with observers. In *Proceedings of 12th Annual International Cryptology Conference (CRYPTO'92)*. Springer-Verlag, Berlin, 89–105.
- CRAMER, R., DAMGÅRD, I., AND SCHOENMAKERS, B. 1994. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference (CRYPTO'94)*. Springer-Verlag, Berlin, 174–187.
- DAMAGARD, I. AND ISHAI, Y. 2005. Constant-round multiparty computation using black-box pseudorandom generator. In *Proceedings of the 25th Annual International Cryptology Conference (CRYPTO'05)*. Springer-Verlag, Berlin, 378–394.
- DANEZIS, G. AND SERJANTOV, A. 2004. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of the 6th Information Hiding Workshop*. Springer-Verlag, Berlin, 293–308.
- DESMEDT, Y. AND KUROSAWA, K. 2000. How to break a practical MIX and design a new one. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'00)*. Springer-Verlag, Berlin, 557–572.
- DWORK, C., LOTSPIECH, J., AND NAOR, M. 1996. Digital signets: Self-enforcing protection of digital information. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. ACM, New York, 489–498.
- FELTEN, E. W. AND SCHNEIDER, M. A. 2000. Timing attacks on web privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*. ACM, New York, 25–32.
- FIAT, A. AND SHAMIR, A. 1987. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of the Cryptology Conference (CRYPTO'86)*. Springer-Verlag, Berlin, 186–194.
- FUJIOKA, A., OKAMOTO, T., AND OHTA, K. 1992. A practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (AUSCRYPT'92)*. Springer-Verlag, Berlin, 244–251.
- FURUKAWA, J. AND SAKO, K. 2001. An efficient scheme for proving a shuffling. In *Proceedings of the 21st Annual International Cryptology Conference (CRYPTO'01)*. Springer-Verlag, Berlin, 368–387.
- GENNARO, R., JARECKI, S., KRAWCZYK, H., AND RABIN, T. 1999. Secure distributed key generation for discrete-log based cryptosystems. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'99)*. Springer-Verlag, Berlin, 295–310.
- GOLLE, P. AND JAKOBSSON, M. 2003. Reusable anonymous return channels. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*. ACM, New York, 94–100.
- GROTH, J. 2002. A verifiable secret shuffle of homomorphic encryptions. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography*. Springer-Verlag, Berlin, 145–160.
- JAKOBSSON, M. AND JUELS, A. 2001. An optimally robust hybrid mix network. In *Proceedings of the 2001 Conference on Principles of Distributed Computing*. ACM, New York, 284–292.
- JAKOBSSON, M., JUELS, A., AND NGUYEN, P. Q. 2002. Proprietary certificates. In *Proceedings of the Cryptographer's Track at the 2002 RSA Conference on Topics in Cryptology*. Springer-Verlag, Berlin, 164–181.
- JAKOBSSON, M., JUELS, A., AND RIVEST, R. 2002. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of USENIX 2002*. USENIX, Berkeley, CA, 339–353.

- KATZ, J., OSTROVSKY, R., AND SMITH, A. 2003. Round efficiency of multi-party computation with a dishonest majority. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'03)*. Springer-Verlag, Berlin, 578–595.
- KESDOGAN, D., AGRAWAL, D., AND PENZ, S. 2002. Limits of anonymity in open environments. In *Proceedings of the 5th Information Hiding Workshop*. Springer-Verlag, Berlin, 53–69.
- LEVINE, B. N., REITER, M. K., WANG, C., AND WRIGHT, M. K. 2004. Timing attacks in low-latency mix-based systems. In *Proceedings of Financial Cryptography 2004*. Springer-Verlag, Berlin, 251–265.
- NEFF, A. 2001. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 2002 ACM Conference on Computer and Communication Security*. ACM, New York, 116–125.
- NEFF, A. 2003. Verifiable mixing (shuffling) of ElGamal pairs. Tech. rep. <http://courses.csail.mit.edu/6.897/spring04/Neff-2004-04-21-ElGamalShuffles.pdf>.
- OGATA, W., KUROSAWA, K., SAKO, K., AND TAKATANI, K. 1997. Fault tolerant anonymous channel. In *Proceedings of Information and Communications Security 1997*. Springer-Verlag, Berlin, 440–444.
- PARK, C., ITOH, K., AND KUROSAWA, K. 1993. All/nothing election scheme and anonymous channel. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'93)*. Springer-Verlag, Berlin, 248–259.
- PEDERSEN, T. P. 1992. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the Annual International Cryptology Conference (CRYPTO'91)*. Springer-Verlag, Berlin, 129–140.
- REITER, M. AND WANG, X. 2004. Fragile mixing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*. ACM, New York, 227–235.
- SAKO, K. AND KILIAN, J. 1995. Receipt-free MIX-type voting scheme: A practical solution to the implementation of a voting booth. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'95)*. Springer-Verlag, Berlin, 393–403.
- SANTIS, A. D., CRESCENZO, G. D., PERSIANO, G., AND YUNG, M. 1994. On monotone formula closure of SZK. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS'94)*. IEEE, Los Alamitos, CA, 454–465.

Received October 2007; revised May 2009; accepted September 2009