Fragile Mixing

Michael K. Reiter*

Xiaofeng Wang†

ABSTRACT

No matter how well designed and engineered, a mix server offers little protection if its administrator can be convinced to log and selectively disclose correspondences between its input and output messages, either for profit or to cooperate with an investigation. In this paper we propose a technique, *fragile mixing*, to discourage an administrator from revealing such correspondences, assuming he is motivated to protect the unlinkability of other communications that flow through the mix (e.g., his own). Briefly, fragile mixing implements the property that any disclosure of an input-message-to-output-message correspondence discloses *all* such correspondences for that batch of output messages. We detail this technique in the context of a re-encryption mix, its integration with a mix network, and incentive and efficiency issues.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*security and protection*

General Terms

Security

Keywords

privacy, anonymity, mix, unlinkability

1. INTRODUCTION

A mix [5] is a server that accepts input messages, transforms them, and outputs them in a permuted order. The intention of a mix is to prevent an observer from linking which output message corresponds to which input message, a property often called *unlinkability*. When no single mix server can be trusted to behave correctly and hide the correspondences between its input and output messages, it is common to route messages through a chain of mix servers (a "mix network", or "mix-net"), so that if any of the mixes is trustworthy, unlinkability of the inputs and outputs of the mix-net can still be achieved. A number of mechanisms have

*Department of Electrical and Computer Engineering, Department of Computer Science, and CyLab, Carnegie Mellon University, Pittsburgh, PA, USA; reiter@cmu.edu
†School of Informatics and Computer Science Department, Indiana University at Bloomington, Bloomington, IN, USA; xw7@indiana.edu

been developed to enhance the robustness of this paradigm to servers actively deviating from the prescribed protocol, e.g., [18, 24, 1, 17, 2, 23, 22, 11, 14, 7, 18].

In contrast, the threat with which we are primarily concerned in this paper is a purely passive attack, in which a mix administrator is convinced to reveal the mapping between an input message and an output message, e.g., to an investigator or buyer. Such a passive disclosure is imminently undetectable by the users of the mix. Moreover, in a world of self-interested mix administrators, it is not clear that lengthening the mix chain, the typical defense against outright compromises, will be effective. That is, without some means of discouraging mix administrators from selling input-to-output message correspondences, it is conceivable that entire mix chains could be "bought out" or otherwise coerced.

Here we introduce a technique designed to better motivate a mix administrator to resist divulging correspondences between input messages and output messages. Our approach is premised on the assumption that the mix administrator is motivated to protect unlinkability for *certain* messages that traverse the mix, though not necessarily all the messages. For example, if the organization hosting the mix sends its own messages through the mix-net, starting at its own mix— a setup that others argue is likely due to the assurances it provides to the mix owner [3]—then the mix administrator will be motivated to protect his organization's communication. The contribution of our approach is to leverage the administrator's desire to protect *certain* information, magnifying it into a desire to protect *all* information. We do so by making the secrecy of many input-output message correspondences dependent on, e.g., the secrecy of each of them; so, if one is revealed, they all are, and in this sense the mixing is *fragile*.

In this paper we provide implementations of fragile mixing for mixes that work in *batch* mode, i.e., that collect a batch of n input messages before emitting them (transformed and permuted). In this context, we implement fragile mixing that makes the secrecy of the input-output mappings for all messages in the batch depend on the secrecy of each of them. A consequence of this goal is that a sender who sends a message through a mix and who can recognize the output of the mix that corresponds to his message—as could occur if the mix transform is private-key decryption [5]—would learn *all* input-output correspondences for that batch. As such, here we focus on re-encryption mixes so as to eliminate this threat, and specifically detail a fragile mix implementation in which the message transform is ElGamal re-encryption. A technical challenge that remains in this context, however, is that the mix administrator could modify the mix to permute "normally", i.e., in a non-fragile way, so as to reserve the right to sell input-output correspondences without risk of disclosing those he wants to protect. To overcome this

challenge, we detail protocols by which the mix proves, non-interactively and in zero-knowledge, that the mix operates as specified, permitting anyone to detect (with overwhelming probability) if the mix cheats.

Finally, we discuss the use of a fragile mix in a larger mix-net. For the implementations of a fragile mix that we offer, we do not recommend that all mixes in the mix-net be fragile, as this would render the end-to-end permutation fragile, as well. As such, we explore a mix-net that interleaves fragile mixes and normal mixes, and show that it works effectively against passive threats without significant degradation of other properties offered by typical mix-nets. We also discuss the incentive structure that this establishes for mix server administrators, and argue that it should lead to the preservation of privacy.

2. RELATED WORK

In this section we review mix-nets and then discuss known countermeasures to defend against misbehaving mixes.

2.1 Mix-nets

As originally introduced by Chaum [5], a mix transforms each input message before forwarding it, by decrypting it with a private key known only to the mix. Consequently, in such a *decryption mix*, the inputs must be encrypted under the mix's public key, leading to a situation in which a message prepared to traverse a mix-net is multiply encrypted and is thus typically of length proportional to the number of hops in the mix-net. More importantly for our work, however, the transmission of a decrypted input permits the creator of that input to learn the position in which his message was output by this mix; as we will see, this would be problematic for our scheme.

For these reasons, the form of mix that we employ here is a *re-encryption mix* [26]. In a re-encryption mix-net, every mix randomly re-encrypts its input messages, and a group of mixes collaboratively decrypts the messages later using a secret key shared among them. The size of the message in a re-encryption mix-net typically does not depend on the number of mixes. And, most importantly for our work, a re-encryption mix can conceal the input/output mapping of each message in a batch, even from its original sender.

In decryption mix-nets, it is generally possible for a user to choose mixes arbitrarily to form her message-forwarding path. Re-encryption mixes, on the other hand, are typically designed for all traffic to traverse a single fixed path, or *mix cascade*, which is a specific form of mix-net. Existing mix-nets that employ cascades include Flash Mixes [15, 20], hybrid mixes [25] and real-time mixes [19].

Mix-nets have been proposed as a foundational technique for electronic voting schemes [5, 10, 26, 30, 18]. They have also been used to build anonymous email systems, e.g., Mixmaster [21], Mixminion [6], and Babel [13]. More recently, mix-nets have been used for connection-based, low-latency communication (e.g., SSH connections and web browsing). Examples include Onion Routing [27], Web MIXes [4], and Morphmix [28].

2.2 Mix-nets robust against misbehaving mixes

The forms of misbehavior considered in prior works have been active in nature. Active deviation refers to the misbehavior of a mix so as to not conform to the protocol (Byzantine failure). For example, the mix may deliberately drop some messages from an input batch. Countermeasures to this threat have been studied under the rubric of mix-net *robustness*. Roughly, robustness means that each mix is asked to provide a proof or strong evidence for its honest behavior. Some robust mix-nets are also capable of successfully delivering messages even when $k < \ell$ out of the ℓ mixes on a user's path do not follow the protocol. Most of the proposed approaches have been built upon zero-knowledge proofs and secret sharing in re-encryption mix-nets. For example, Ogata et al. present a robust mix-net based on cut-and-choose methods [24]. Both Abe [1] and Jakobsson and Juels [17] propose more efficient zero-knowledge proofs which achieve *universal verifiability* [1, 2]. This property allows a third-party to verify the proof of correct behavior. *Verifiable shuffling* schemes [23, 22, 11] and Millimix [16] also have this property. Other proposals include *repetition robustness* [14], *layer redundancy* [7], and *random partial checking* [18].

When a mix's misbehavior is detectable, other approaches can be used to help users avoid misbehaving mixes and thereby to encourage individual mixes to behave honestly. For example, Dingledine et al. [8, 29] propose to use a reputation system to record every mix's previous performance, in order to help users to choose a reliable path through a mix-net. Our work differs by addressing an *undetectable* attack, i.e., a mix administrator's selective disclosure of input-to-output message correspondences.

Finally, Acquisti et al. built the first economic model for mix-nets [3]. Their research argues principles for the behavior of individual mixes based on self-interest and rationality. This has been influential in our research, as described below.

3. PRELIMINARIES

In this section, we describe the assumptions and concepts on which our approach rests.

3.1 Attack model

Here, we list several assumptions we employ:
Mixes (i.e., mix administrators) value the anonymity of their own messages over benefits to be gained by disclosing input-to-output message correspondences.

Research on the economics of anonymity argues that a strong incentive for one to run a mix is to keep its own communication private [3]. This suggests that a mix's valuation of anonymity could be higher than other users'. Therefore, it is reasonable to expect that a fragile mix that ties the anonymity of its own messages to that of others', as we propose here, will be motivated to keep all input-to-output correspondences secret, and that this will counterbalance any benefits the mix might be offered for divulging an input-output correspondence.

This assumption is not intended to rule out the possibility of mix compromise, after which we presume it discloses all input-to-output correspondences to its attacker. As with other mix-nets, mix-nets that employ fragile mixes (Section 6) still offer anonymity provided that compromised mixes do not make up too much of the mix-net.

In addition, in our security analysis we will permit mixes that trade a small degradation of its anonymity for profit. For example, a normal (non-fragile) mix may choose to expose to a third party some of its input-to-output correspondences, though within limits so as to not reduce the privacy of its own messages too substantially.

Mixes send out their own messages frequently. In other words, there is significant fraction of mixes in a mix cascade transmitting their own messages in every message batch.

As pointed out by previous research [3], acting as a mix incurs costs. To justify such costs, it is reasonable to believe that the mix has high demand from the organization running it for the anonymity services it provides.

That said, our approach is still effective when a small number of mixes do not send their own messages in some batch, and thus are willing to divulge their permutations completely; we will discuss this in Section 6. Note that this can hurt these mixes' own privacy because it signals to others the batches in which this mix sent messages.

Adversaries are capable of observing all communications between any two mixes in a mix cascade, submitting messages to the cascade, and controlling (compromising) some number of mixes.

3.2 Bulletin board model

A "bulletin board" model is a widely-used model for re-encryption mix-nets, and we also adopt it here. In the bulletin board model, there are three types of participants: users, a bulletin board and mixes. The users post encrypted messages to the bulletin board. After the number of messages reaches a predetermined quota n, these messages are transmitted as a batch through a mix-cascade. We assume that not only the output batch of the mix-cascade, but all the intermediate batches forwarded by every mix to its neighbor, appear on the bulletin board. This models our assumption of the adversary's capability: It observes all communication in the mix-cascade.

An individual mix can be a user of the mix-net, and typically is [3]. Here, we describe two models that allow mixes to insert their own messages into a batch.

- **Common mode:** In common mode, every mix in a cascade is required to write κ messages to the bulletin board before transmission starts. A mix can either post messages it wants to dispatch, or dummy messages, to the bulletin board.

- **Attach mode:** In the attach mode, the initial batch reserves $m\kappa$ slots for mixes, where m is the number of mixes. During transmission, every mix adds κ messages to the batch it receives, i.e., emitting κ more messages than it takes in.

The common mode can be less efficient than the attach mode, as individual mixes must route their messages to the head of the cascade for transmission. However, in the common mode, an adversary must convince a mix that a message the adversary is asking the mix's help to trace is not the mix's own message, because the mix cannot recognize its own messages in a re-encryption mix-net [1].

Unless we specify otherwise, in the rest of the paper we will assume that common mode is employed. However, our scheme can also work in the attach mode.

3.3 ElGamal encryption

Let $\mathcal{G}_q$ be a multiplicative group of prime order q in which the Decisional Diffie-Hellman problem is hard. Let g be a generator of $\mathcal{G}_q$. In the ElGamal encryption scheme, a secret key is $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$ (" $\stackrel{R}{\leftarrow} S$" denotes selection uniformly at random from the set S), and the corresponding public key h is computed as $h \leftarrow g^x$ (where multiplication is performed in the group).

The encryption of a message $m \in \mathcal{G}_q$ is $(G, M) = (g^\gamma, mh^\gamma)$ where $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_q$ is chosen anew per encryption. Decryption of the ciphertext (G, M) returns $M(G)^{-x}$. A mix with private key x and public key $h = g^x$ re-encrypts an ElGamal ciphertext (G, M) by choosing $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and forming (Gg^γ, Mh^γ).

Tsiounis and Yung [31], and Jakobsson [14] independently propose an extension to ElGamal encryption to render it non-malleable, by using γ as a secret key to generate a Schnorr signature on (G, M); this signature can be verified using G as the public key. This construction prevents an adversary from using a copied or mauled ciphertext to trace an honest user's message through a mix. We omit these details here, but we note that this technique is compatible with our protocols.

3.4 Zero-knowledge proofs

Our fragile mixing protocols require zero-knowledge proofs. A zero-knowledge proof is a protocol in which a computationally unbounded *prover* P convinces a probabilistic polynomial time *verifier* V of an assertion while giving the verifier no additional information beyond the validity of the assertion. Formal definitions are available in, e.g., [12]. Here we remind the reader of their definition informally, consisting of completeness, soundness and zero-knowledge.

- *Completeness* A proof protocol is *complete* if an honest prover convinces the verifier with overwhelming probability.

- *Soundness* A proof protocol is *sound* if any prover succeeds in convincing the verifier of a false assertion with negligible probability.

- *Zero-knowledge* A proof protocol is *zero-knowledge* if there exists a probabilistic polynomial-time algorithm (a *simulator*) whose output (without interacting with P) is indistinguishable from the transcript of V's interaction with P.

We assume that the verifier is honest, i.e., it always follows the protocol.

4. FRAGILE MIXING

The basic idea of *fragile mixing* is to bind the anonymity of a batch of messages together: As soon as the permutation has been revealed partially, it is disclosed completely. This gives individual mix administrators a disincentive to reveal input-to-output message correspondences, since doing so reveals the correspondences for all the messages (including the mix adminstrators', for example). In this section, we more carefully define a fragile mix and its properties.

Let $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$. A *permutation* π on $\mathbb{Z}_n$ is a bijection $\pi : \mathbb{Z}_n \to \mathbb{Z}_n$. A *permutation family* is a set of permutations with the same domain and range $\mathbb{Z}_n$.

DEFINITION 1. **Fragile permutation**. *A permutation family $\mathcal{F}$ is* fragile *if for any $\pi, \pi' \in \mathcal{F}$ and $x \in \mathbb{Z}_n$, $\pi(x) = \pi'(x)$ only when $\pi = \pi'$. We call every member of $\mathcal{F}$ a* fragile permutation.

In the space $\mathbb{Z}_n \times \mathbb{Z}_n$, a fragile permutation will be uniquely identified in its family by revealing any single point. Intu-

itively, knowing any input/output pair of a fragile permutation, one can determine the rest of permutation. Some fragile permutation families are listed below:

- **Loop permutation ($\mathcal{LP}$):** $\mathcal{LP} = \{\pi^b \mid \pi^b(x) = x + b \bmod n; x, b \in \mathbb{Z}_n\}$. For example, a permutation $\{0,1,2,3,4,5\} \mapsto \{5,0,1,2,3,4\}$ is a member of $\mathcal{LP}$. It is easily to see that the cardinality of $\mathcal{LP}$ is n.

- **Multiplicative permutation ($\mathcal{MP}$):** $\mathcal{MP} = \{\pi^a \mid \pi^a(x) = ax \bmod n; a, x \in \mathbb{Z}_n \setminus \{0\}, n \text{ prime}\}$. $\mathcal{MP}$ is a fragile permutation family on $\mathbb{Z}_n \setminus \{0\}$. Its cardinality is $n-1$.

- **Key permutation ($\mathcal{KP}$):** $\mathcal{KP} = \{\pi^K \mid \pi^K(x) = x \oplus K; x, K \in \mathbb{Z}_{2^\eta}\}$. $\mathcal{KP}$ is a fragile permutation family on $\mathbb{Z}_{2^\eta}$, where $\eta \in \mathbb{N}$. Its cardinality is 2^η, the size of its domain.

Since any two members of a fragile permutation family $\mathcal{F}$ map any input of their domain to different output, the cardinality of $\mathcal{F}$ is at most n, i.e., a fragile permutation family contains no more than n permutations. We can increase the number of permutations in the permutation family at the cost of fragility, by generalizing to a k-fragile permutation.

DEFINITION 2. k-**fragile permutation**. *A permutation family $\mathcal{F}^{(k)}$ is a k-fragile permutation if for any $D \subseteq \mathbb{Z}_n$ with $|D| = k$ and $\pi, \pi' \in \mathcal{F}^{(k)}$, $\pi(x) = \pi'(x)$ for all $x \in D$ only when $\pi = \pi'$.*

For example, the **pair-wise permutation family** $\mathcal{PP} = \{\pi^{a,b} \mid \pi^{a,b}(x) = ax+b \bmod n; a \in \mathbb{Z}_n \setminus \{0\}; b \in \mathbb{Z}_n; n \text{ prime}\}$ is a 2-fragile permutation.

k-fragile permutations may be useful in permitting a limited number (less than k) of input-output message correspondences to be exposed without imposing on the privacy of other messages not being investigated. The cardinality of a k-fragile permutation family is bounded by the number of ordered subsets of size k in the set of n elements, which is $\frac{n!}{(n-k)!}$. In the rest of this paper, we focus our attention on (1-)fragile permutations. However, k-fragile permutations also can be utilized in the protocols we propose.

Many fragile permutation families form groups under composition ($\circ$), and it is easy to verify that the above examples do. In particular, when a fragile permutation family is closed under composition, a mix-cascade completely made by fragile mixes could be vulnerable: An active adversary only need insert a message to the batch, find the output corresponding to its input message, and use this to determine the combined permutation performed by the cascade. We address this problem in Section 6.

5. PROOF OF FRAGILE MIXING

Unless a mix is forced to utilize a fragile permutation when reordering a batch of messages, it may simply use any permutation and therefore gain flexibility to disclose certain input-to-output message correspondences. In this section, we therefore show how a mix server can prove in zero knowledge that it has utilized a fragile permutation, which it should be required to do by its clients.

Specifically, let $0 \le j < n$. The input batch of the mix is $\{(G_j, M_j)\}$ and its output batch is $\{(G'_j, M'_j)\}$. Let $\mathcal{F}$ be a public fragile permutation family. An honest mix knows

values $\{\gamma_j\}$ in $\mathbb{Z}_q$ and a permutation π such that for all $0 \le j < n$

$$\pi \in \mathcal{F} \tag{1}$$
$$(G'_{\pi(j)}, M'_{\pi(j)}) = (G_j g^{\gamma_j}, M_j h^{\gamma_j}) \tag{2}$$

The mix (the prover) is required to convince a verifier V of the existence of such a π and $\{\gamma_j\}$ without revealing any information about $\{\gamma_j\}$ and π. In this section, we present two zero-knowledge proof protocols by which the mix can prove this.

5.1 A general proof

We first describe a general proof that works when the fragile permutation family $\mathcal{F}$ forms a group under composition. This protocol is based on the well-known cut-and-choose technique.

Protocol 1

- Repeat the following steps σ times.

    1. The mix chooses $\bar{\gamma}_j \overset{R}{\leftarrow} \mathbb{Z}_q$ for $0 \le j < n$ and $\bar{\pi} \overset{R}{\leftarrow} \mathcal{F}$. Then, the mix publishes

    $$\bar{G}_{\bar{\pi}(j)} = G_j g^{\bar{\gamma}_j} \tag{3}$$
    $$\bar{M}_{\bar{\pi}(j)} = M_j h^{\bar{\gamma}_j} \tag{4}$$

    2. The verifier V flips a public coin: $b \overset{R}{\leftarrow} \{0, 1\}$

    3. If $b = 0$, then:
        - Mix publishes $\bar{\pi}$ and $\{\bar{\gamma}_j\}$.
        - V accepts iff $\bar{\pi} \in \mathcal{F}$ and (3) and (4) hold.

    4. If $b = 1$, then:
        - Mix publishes $\pi_d = \pi \circ \bar{\pi}^{-1}$ and $\{d_j = -\bar{\gamma}_{\bar{\pi}^{-1}(j)} + \gamma_{\bar{\pi}^{-1}(j)} \bmod q\}$.
        - Verifier V accepts iff

        $$\pi_d \in \mathcal{F} \tag{5}$$
        $$G'_{\pi_d(j)} = \bar{G}_j g^{d_j} \tag{6}$$
        $$M'_{\pi_d(j)} = \bar{M}_j h^{d_j} \tag{7}$$

This protocol can be made noninteractive in the random oracle model by using the Fiat-Shamir heuristic [9].

THEOREM 1. *For any fragile permutation family $\mathcal{F}$ that forms a group under composition, Protocol 1 is an honest verifier zero-knowledge proof of the existence of π and $\{\gamma_j\}$ satisfying (1) and (2).*

Proof.

- *Completeness.* Since $\mathcal{F}$ forms a group under composition, $\mathcal{F}$ is closed under composition and inverses exist. Thus, the prover (mix) can compute $\bar{\pi}^{-1}$ and $\pi_d = \pi \circ \bar{\pi}^{-1}$ in $\mathcal{F}$, and so can complete the proof.

- *Soundness.* Suppose that a prover is capable of answering both $b = 0$ and $b = 1$ queries, i.e., of publishing $\bar{\pi} \in \mathcal{F}$ and $\{\bar{\gamma}_j\}$ satisfying (3) and (4), and $\pi_d \in \mathcal{F}$ and $\{d_j\}$ satisfying (6) and (7). Define $\pi = \pi_d \circ \bar{\pi}$ and $\gamma_j = d_{\bar{\pi}_j} + \bar{\gamma}_j \bmod q$. It is easily verified that π and $\{\gamma_j\}$ satisfy (1) and (2), and so the assertion that such a π and $\{\gamma_j\}$ exist is true. By the contrapositive, if this

assertion is false, then any prover is capable of answering at most one of the $b = 0$ and $b = 1$ queries, and so convinces the verifier with probability at most $1/2$. Since this is repeated σ times, the prover convinces the verifier with probability at most $1/2^\sigma$.

- *Zero-knowledge.* The simulator selects $b \stackrel{R}{\leftarrow} \{0, 1\}$. If $b = 0$, it selects $\bar{\pi} \stackrel{R}{\leftarrow} \mathcal{F}$ and $\bar{\gamma}_j \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and outputs $(\{(\bar{G}_{\bar{\pi}(j)}, \bar{M}_{\bar{\pi}(j)})\}, 0, (\bar{\pi}, \{\bar{\gamma}_j\}))$, where $\{(\bar{G}_{\bar{\pi}(j)}, \bar{M}_{\bar{\pi}(j)})\}$ are computed as in (3)–(4). If $b = 1$, the simulator selects $\pi_d \stackrel{R}{\leftarrow} \mathcal{F}$ and $d_j \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and outputs $((\{(G'_{\pi_d(j)}g^{-d_j}, M'_{\pi_d(j)}h^{-d_j})\}), 1, (\pi_d, \{d_j\}))$. Repeating σ times, the simulator generates a transcript that is perfectly indistinguishable from that of the real protocol. □

We denote Protocol 1 by $PFG(\{(G_j, M_j)\}, \{(G'_j, M'_j)\})$.

5.2 A more efficient proof for $\mathcal{LP}$

We now present a more efficient protocol by which a mix can prove in zero knowledge that it employed a fragile permutation. This protocol requires that the fragile permutation family be $\mathcal{LP}$ (the loop permutation family). Our approach builds upon protocols for proving *verifiable shuffling* [23, 11]. The problem of verifiable shuffling is as follows:

Let $0 \le j < n$. Let Π be the set of all permutations on $\mathbb{Z}_n$. There are two sequences of pairs $\{(X_j, Y_j)\}$ (input) and $\{(\bar{X}_j, \bar{Y}_j)\}$ (output), and the prover knows values $\{\gamma_j\}$ from $\mathbb{Z}_q$ and $\pi \in \Pi$ such that for all $0 \le j < n$

$$(\bar{X}_{\pi(j)}, \bar{Y}_{\pi(j)}) = (X_j g^{\gamma_j}, Y_j h^{\gamma_j}) \quad (8)$$

The prover is required to convince a verifier V of the existence of $\{\gamma_j\}$ and π satisfying (8), without revealing any information about $\{\gamma_j\}$ and π. We denote a proof for this problem by $PRS(\{(X_j, Y_j)\}, \{(\bar{X}_j, \bar{Y}_j)\})$.

Our proof protocol is built upon a proof protocol for verifiable shuffling, e.g., [23, 11].

Protocol 2: Proof for loop permutations

1. The mix knows a permutation $\pi \in \mathcal{LP}$ and a sequence $\{\gamma_j\}$ which satisfy (2). Using this knowledge, the mix runs $PRS(\{(G_j, M_j)\}, \{(G'_j, M'_j)\})$ to V.

2. Verifier V generates $a_j \stackrel{R}{\leftarrow} \mathcal{G}_q$ and $b_j \stackrel{R}{\leftarrow} \mathcal{G}_q$ for $0 \le j < n$. V gives $\{a_j\}$ and $\{b_j\}$ to the mix.

3. The mix and V each compute the new list:

$$(\hat{G}_j, \hat{M}_j) = (G_j a_j, M_j b_j) \quad (9)$$

   Then, the mix generates secret values $\hat{\gamma}_j \stackrel{R}{\leftarrow} \mathbb{Z}_q$ for $0 \le j < n$ and uses these and the loop permutation π (the permutation on $\{(G_j, M_j)\}$) to compute the following list:

$$(\hat{G}'_{\pi(j)}, \hat{M}'_{\pi(j)}) = (\hat{G}_j g^{\hat{\gamma}_j}, \hat{M}_j h^{\hat{\gamma}_j}) \quad (10)$$

   The mix gives (10) to V.

4. The mix runs a zero-knowledge proof for shuffling $PRS(\{(\hat{G}_j, \hat{M}_j)\}, \{(\hat{G}'_j, \hat{M}'_j)\})$ to V.

5. The mix and V each compute the following lists:

$$(\bar{G}_j, \bar{M}_j) = (\hat{G}_j/\hat{G}_{j+1 \bmod n}, \hat{M}_j/\hat{M}_{j+1 \bmod n}) \quad (11)$$

$$(\bar{G}'_j, \bar{M}'_j) = (\hat{G}'_j/\hat{G}'_{j+1 \bmod n}, \hat{M}'_j/\hat{M}'_{j+1 \bmod n}) \quad (12)$$

6. The mix uses its knowledge of π and $\{\hat{\gamma}_j - \hat{\gamma}_{j+1 \bmod n}\}$ to run a proof $PRS(\{(\bar{G}_j, \bar{M}_j)\}, \{(\bar{G}'_j, \bar{M}'_j)\})$ to V.

7. The mix and V each compute:

$$(a'_j, b'_j) = (\hat{G}'_j/G'_j, \hat{M}'_j/M'_j) \quad (13)$$

8. The mix uses the knowledge of π and $\{\hat{\gamma}_j - \gamma_j\}$ to run $PRS(\{(a_j, b_j)\}, \{(a'_j, b'_j)\})$ to V.

9. Verifier V accepts if Steps 1, 4, 6, 8 are all correct.

Intuitively, this protocol proceeds as follows. In Steps 2-3, the mix generates a new list $\{(\hat{G}_j, \hat{M}_j)\}$ with the random challenges from verifier V and permutes this new list in the same way as the list $\{(G_j, M_j)\}$ using another set of random exponents $\{\hat{\gamma}_j\}$. Then, in Steps 4-6, it proves that the permutation on the new list is a loop permutation. Finally, using Steps 1, 7-8, the mix proves that this permutation, in fact, is the permutation mapping $\{(G_j, M_j)\}$ to $\{(G'_j, M'_j)\}$.

LEMMA 1. *A permutation $\pi \in \mathcal{LP}$ if and only if for all $x \in \mathbb{Z}_n$, $\pi(x + 1 \bmod n) = \pi(x) + 1 \bmod n$.*

Proof. The "only if" direction is straightforward, and so we detail only the "if" part. With the above property, we have:

$$\pi(1) = \pi(0) + 1 \bmod n$$
$$\pi(2) = \pi(1) + 1 \bmod n$$
$$= \pi(0) + 2 \bmod n$$
$$\cdots$$
$$\pi(x) = \pi(0) + x \bmod n$$

By the definition of a loop permutation, $\pi \in \mathcal{LP}$. □

THEOREM 2. *The above protocol is a honest-verifier zero-knowledge proof of the existence of π and $\{\gamma_j\}$ satisfying $\pi \in \mathcal{LP}$ and (2).*

Proof (sketch).

- *Completeness.* For an honest mix that performs a loop permutation on the input batch, the protocol will be completed given the completeness of PRS. Both [23] and [11] offer schemes with proved completeness.

- *Soundness.* We prove the soundness in the following way. We first prove that the permutation $\hat{\pi}$ on the new list $\{(\hat{G}_j, \hat{M}_j)\}$ is a loop permutation. Then we show that that for each j, $(G'_{\hat{\pi}(j)}, M'_{\hat{\pi}(j)})$ is an encryption of the same plaintext as (G_j, M_j), and so $\hat{\pi}$ demonstrates the existence of the needed permutation.

  If the verifier V accepts the proof in Step 4, the mix proves

$$(\hat{G}'_{\hat{\pi}(j)}, \hat{M}'_{\hat{\pi}(j)}) = (\hat{G}_j g^{\hat{\gamma}_j}, \hat{M}_j h^{\hat{\gamma}_j}) \quad (14)$$

  After completing Step 6, the mix proves there exists a permutation $\bar{\pi}$ and a sequence $\{\bar{\gamma}_j\}$, such that

$$(\bar{G}'_{\bar{\pi}(j)}, \bar{M}'_{\bar{\pi}(j)}) = (\bar{G}_j g^{\bar{\gamma}_j}, \bar{M}_j h^{\bar{\gamma}_j}) \quad (15)$$

According to (15), we know:

$$(\hat{G}'_{\bar{\pi}(j)}/\hat{G}'_{\bar{\pi}(j)+1 \bmod n}, \hat{M}'_{\bar{\pi}(j)}/\hat{M}'_{\bar{\pi}(j)+1 \bmod n}) \quad (16)$$
$$= (\hat{G}_j g^{\bar{\gamma}_j}/\hat{G}_{j+1 \bmod n}, \hat{M}_j h^{\bar{\gamma}_j}/\hat{M}_{j+1 \bmod n})$$

From (16) and (14), we have:

$$\log_g \left(\frac{\hat{G}_j}{\hat{G}_{j+1 \bmod n}} \cdot \frac{\hat{G}_{\hat{\pi}^{-1}(\bar{\pi}(j)+1 \bmod n)}}{\hat{G}_{\hat{\pi}^{-1}(\bar{\pi}(j))}} \right) = \quad (17)$$
$$\log_h \left(\frac{\hat{M}_j}{\hat{M}_{j+1 \bmod n}} \cdot \frac{\hat{M}_{\hat{\pi}^{-1}(\bar{\pi}(j)+1 \bmod n)}}{\hat{M}_{\hat{\pi}^{-1}(\bar{\pi}(j))}} \right)$$

There are four cases to consider in (17):

1. $j \neq \hat{\pi}^{-1}(\bar{\pi}(j))$ and $j + 1 \bmod n \neq \hat{\pi}^{-1}(\bar{\pi}(j) + 1 \bmod n)$

2. $j = \hat{\pi}^{-1}(\bar{\pi}(j))$ and $j + 1 \bmod n \neq \hat{\pi}^{-1}(\bar{\pi}(j) + 1 \bmod n)$

3. $j \neq \hat{\pi}^{-1}(\bar{\pi}(j))$ and $j + 1 \bmod n = \hat{\pi}^{-1}(\bar{\pi}(j) + 1 \bmod n)$

4. $j = \hat{\pi}^{-1}(\bar{\pi}(j))$ and $j + 1 \bmod n = \hat{\pi}^{-1}(\bar{\pi}(j) + 1 \bmod n)$

Recall that $\hat{G}$s and $\hat{M}$s in (17) are randomly and independently distributed in $\mathcal{G}_q$ due to the random challenges $\{a_j\}$ and $\{b_j\}$. Thus, in case 1, given a pair $(\hat{G}_j, \hat{G}_{j+1 \bmod n})$, the probability that there exists another pair $(\hat{G}_{\hat{\pi}^{-1}(\bar{\pi}(j)+1 \bmod n)}, \hat{G}_{\hat{\pi}^{-1}(\bar{\pi}(j))})$ such that the corresponding random M tuple

$$(\hat{M}_j, \hat{M}_{j+1 \bmod n}, \hat{M}_{\hat{\pi}^{-1}(\bar{\pi}(j)+1 \bmod n)}, \hat{M}_{\hat{\pi}^{-1}(\bar{\pi}(j))})$$

satisfies (17) is at most $\frac{n^2}{q}$, which is negligible given $n \ll q$. In a similar way, in both case 2 and case 3, the probabilities that (17) holds are upper-bounded by $\frac{n}{q}$ which is also negligible. The only case which allows (17) to always hold is case 4. Therefore we have:

$$\hat{\pi}(j) = \bar{\pi}(j) \quad (18)$$
$$\hat{\pi}(j + 1 \bmod n) = \bar{\pi}(j) + 1 \bmod n \quad (19)$$

This gives us $\hat{\pi}(j+1 \bmod n) = \hat{\pi}(j) + 1 \bmod n$. Therefore, Lemma 1 applies, and we have $\hat{\pi} \in \mathcal{LP}$.

We now prove that for each j, $(G'_{\hat{\pi}(j)}, M'_{\hat{\pi}(j)})$ is an encryption of the same plaintext as (G_j, M_j), completing the proof of soundness. Let π be the permutation on $\{(G_j, M_j)\}$ that is proved in Step 1, i.e.,

$$(G'_{\pi(j)}, M'_{\pi(j)}) = (G_j g^{\gamma_j}, M_j h^{\gamma_j}) \quad (20)$$

Given (14) and $(a'_j, b'_j) = (\hat{G}'_j/G'_j, \hat{M}'_j/M'_j)$, we have $(a'_j, b'_j) =$

$$\left(\frac{G_{\hat{\pi}^{-1}(j)} a_{\hat{\pi}^{-1}(j)} g^{\hat{\gamma}_{\hat{\pi}^{-1}(j)}}}{G_{\pi^{-1}(j)} g^{\gamma_{\pi^{-1}(j)}}}, \frac{M_{\hat{\pi}^{-1}(j)} b_{\hat{\pi}^{-1}(j)} h^{\hat{\gamma}_{\hat{\pi}^{-1}(j)}}}{M_{\pi^{-1}(j)} h^{\gamma_{\pi^{-1}(j)}}} \right)$$

If Step 8 holds, we know that $(a'_j, b'_j) = (a_k g^{\tilde{\gamma}_k}, b_k h^{\tilde{\gamma}_k})$ for some $0 \leq k < n$ and some $\tilde{\gamma}_k$. Therefore, we have

$$\log_g \left(\frac{G_{\hat{\pi}^{-1}(j)} a_{\hat{\pi}^{-1}(j)}}{G_{\pi^{-1}(j)} a_k} \right) = \log_h \left(\frac{M_{\hat{\pi}^{-1}(j)} b_{\hat{\pi}^{-1}(j)}}{M_{\pi^{-1}(j)} b_k} \right)$$

Recall that each a_j and b_j is drawn randomly from $\mathcal{G}_q$. So, for any $(G_{\hat{\pi}^{-1}(j)}, M_{\hat{\pi}^{-1}(j)})$ and $(G_{\pi^{-1}(j)}, M_{\pi^{-1}(j)})$, the probability that there exist pairs $(a_{\hat{\pi}^{-1}(j)}, b_{\hat{\pi}^{-1}(j)})$ and (a_k, b_k) that satisfy the above equation is negligible, bounded from above by $\frac{n^2}{q}$, unless $k = \hat{\pi}^{-1}(j)$. So,

$$\log_g(G_{\hat{\pi}^{-1}(j)}/G_{\pi^{-1}(j)}) = \log_h(M_{\hat{\pi}^{-1}(j)}/M_{\pi^{-1}(j)}),$$

i.e., $(G_{\hat{\pi}^{-1}(j)}, M_{\hat{\pi}^{-1}(j)})$ and $(G_{\pi^{-1}(j)}, M_{\pi^{-1}(j)})$ are encryptions of the same plaintext, for each j. Since we additionally know that (G'_j, M'_j) and $(G_{\pi^{-1}(j)}, M_{\pi^{-1}(j)})$ are encryptions of the same plaintext (by Step 1), permutation $\hat{\pi}$ demonstrates the existence of a loop permutation meeting the requirements of the proof.

- *Zero-knowledge.* The zero-knowledge property of this proof depends on the zero-knowledge of the *PRS* protocol. Informally, the simulator randomly generates the outputs for steps 2 and 3, and then runs the simulator for *PRS* to produce those transcripts. If the *PRS* protocol is zero-knowledge, the transcript produced is indistinguishable from the real protocol run.

□

The complexity of the proof protocol depends on the complexity of the underneath *PRS* protocol. For example, running Neff's verifiable shuffling protocol [23], we need $32n$ group exponentiations for generating a proof and $48n + 8$ for verifying the proofs.

We denote Protocol 2 by $PLP(\{(G_j, M_j), (G'_j, M'_j)\})$.

6. A MIX NETWORK USING FRAGILE MIXES

In this section, we discuss the impact of fragile mixing in the context of a larger mix-net. Ideally our techniques could be combined with a standard mix-net so as to retain the properties of the original mix-net construction, and to add the additional properties that fragility is intended to provide (i.e., discouraging mix administrators from disclosing input-to-output message correspondences).

As mentioned previously, however, a mix-cascade built solely from fragile mixes can significantly constrain the end-to-end permutation implemented by the mix-net, if the fragile permutation family is closed under composition. In this case, an active adversary capable of inserting messages to the input can easily reveal this end-to-end permutation. Although this problem does not happen when mixes are not constrained to perform fragile permutations, these mixes are less trustworthy, in the sense that they are less motivated to protect *all* their input-to-output message correspondences.
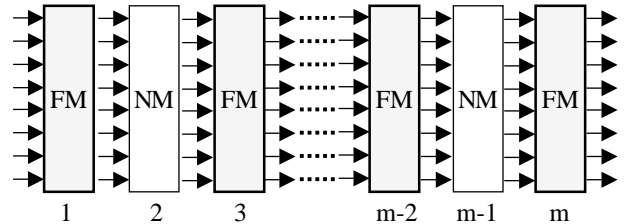

Figure 1: *Fragile mix cascade. FM denotes a fragile mix, and NM denotes a normal mix.*

We observe that these two types of mixes complement each other, which motivates us to propose the cascade construction illustrated in Figure 1. This mix-cascade interleaves two types of mixes, fragile mixes and normal mixes; we call this a *hybrid cascade*. This design results from the observation that adjacent mixes of the same type may not significantly improve the security of the cascade: On one hand, messages can presumably be traced through two adjacent normal mixes roughly as easily as tracing through one normal mix, since these mixes, not being motivated to keep private all of their input-to-output message mappings, may easily be convinced to disclose them. On the other hand, successive fragile mixes (assuming a fragile permutation family closed under composition) implement the same permutation as a single fragile mix could. Moreover, note that both the head and the tail of the cascade are fragile mix servers, in order to provide greater strength at the head and tail against these servers being convinced to divulge input-to-output mappings. This hybrid cascade construction thus strikes a balance between defending against passive attacks to disclose mixes' input-to-output message correspondences, and ensuring that the end-to-end permutation implemenated by the mix is an arbitrary permutation.

For completeness, here we illustrate a mix-cascade protocol built on this idea, using Abe's scheme [1]. The mix-cascade operates in two stages, an encryption stage and a decryption stage. During the encryption stage, the mix cascade performs re-encryptions and permutations on the input batch; fragile mixes also run a proof protocol for their permutations, as discussed in Section 5. In the decryption stage, all mixes in the cascade collaborate to decrypt the output batch and forward these decryptions to the receivers. The protocol is presented below, in which m denotes the length of a cascade. We assume that mix i has a private key $x_i \in \mathbb{Z}_q$, and that $h = g^x$ where $x = \sum_{1 \leq i \leq m} x_i \bmod q$.

Protocol 3: Fragile mixing cascade protocol

- **Encryption stage:**

  1. A user (who could be a mix) chooses $\gamma_j \xleftarrow{R} \mathbb{Z}_q$ and posts an encryption of her message M_j to the bulletin board:

  $$G_{0,j} = g^{\gamma_j} \tag{21}$$
  $$M_{0,j} = M_j h^{\gamma_j} \tag{22}$$

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

  2. Upon receiving $\{(G_{i-1,0}, M_{i-1,0}), \ldots, (G_{i-1,n-1}, M_{i-1,n-1})\}$, mix i first draws $\gamma_{i,j} \xleftarrow{R} \mathbb{Z}_q$ for $0 \leq j < n$. Then, if mix i is a fragile mix, it draws a fragile permutation $\pi_i \xleftarrow{R} \mathcal{F}$, where $\mathcal{F}$ is a public fragile permutation family. Otherwise, it chooses $\pi_i \xleftarrow{R} \Pi$ where Π is the space of all permutations on $\mathbb{Z}_n$. Finally, the mix re-encrypts the batch as:

  $$G_{i,\pi_i(j)} = G_{i-1,j} g^{\gamma_{i,j}} \tag{23}$$
  $$M_{i,\pi_i(j)} = M_{i-1,j} h^{\gamma_{i,j}} \tag{24}$$

  where $0 \leq j < n$, and sends out the batch to mix $i + 1$. This process continues until the batch has been re-encrypted by the last mix m.

  3. Every fragile mix i runs either $PFG(\{(G_{i-1,j}, M_{i-1,j})\}, \{(G_{i,j}, M_{i,j})\})$ or $PLP(\{(G_{i-1,j}, M_{i-1,j})\}, \{(G_{i,j}, M_{i,j})\})$ as approporiate, to prove its correct performance of fragile mixing. Any verifier can check these proofs.

- **Decryption stage:**

  1. The mix cascade starts decryption from the first mix. Every mix i computes $W_{i,j} = W_{i-1,j} G_{m,j}^{x_i}$ for each j, where $W_{0,j} = 1$.

  2. Mix m computes the batch $\{M_{m,j}/W_{m,j}\}$, and deliveries the batch.

In the attach mode, the input batch contains $n - m\kappa$ messages. Every mix i generates κ ciphertexts $\{(g^{\gamma_l}, M_l h^{\gamma_l})\}$ $(l = 0, \ldots, \kappa - 1)$, and attaches them to the end of the message batch it receives. Then, i permutes and re-encrypts all $n - (m - i)\kappa$ messages as Step 2 in the encryption stage of the above protocol.

6.1 Security

In this section, we discuss the security of the hybrid cascade described above. For passive adversaries who monitor the bulletin board, we simply note that due to the use of normal mixes, the end-to-end permutation implemented by the hybrid cascade is not constrained to be a fragile permutation. Thus, the strength of the hybrid cascade should be similar to that of a traditional cascade against passive adversaries.

The more interesting scenarios result from various active adversaries, particularly when we combine the possibility of some mixes being compromised, and other normal mixes being convinced to divulge a portion of their input-to-output message correspondences. More generally, in a hybrid mix cascade, not only will the compromised mixes divulge their full permutations to the attacker, but so may normal and fragile mixes that did not submit a message in the batch the adversary is interested in. We call these *exposed mixes*. For the purpose of our analysis for a single batch of messages, we can remove exposed mixes from the cascade, combine two consecutive mixes of the same type (normal or fragile) together and remove the head or tail of the cascade if they are normal mixes. This results in a mix cascade with the same construction as in Figure 1. We call this cascade the *base cascade*.

To trace a message through the base cascade, the adversary must determine the fragile permutations performed. It could persuade normal mixes to give out part of their permutations. However, since these mixes have their own messages in the batch (else they are exposed and thus removed from the base cascade), it is reasonable to assume that they will reveal only a portion of their input/output pairs, in order to ensure adequate anonymity for their own messages.[1] Using such disclosures, the adversary can conceivably insert a pool of test messages to the input batch, and then determine whether the output pattern of his own messages at the end

---

[1] In common mode, the existence of the fragile mixes renders the adversary incapable of convincing a normal mix that the message being traced is not its own, though the mix may nevertheless cooperate if it trusts the adversary or is forced to. This, however, is not a problem in the attach mode because each mix knows exactly where in the batch its own messages are.

of the cascade is consistent with a hypothesized set of fragile permutations used by the fragile mixes.

Somewhat more specifically, in a base cascade with m mixes, mixes $1, 3, \ldots, m$ are fragile mixes. Let $(\pi_1, \pi_3, \ldots, \pi_{m-2})$ be a vector of permutations representing the fragile permutations that all but the last fragile mix performed on a single batch.[2] Let τ be the cardinality of the fragile permutation family. The size of the permutation vector space is $\tau^{\frac{m-1}{2}}$.

Now suppose that the adversary can convince each normal mix to divulge ρ of its input-to-output message pairs for the batch of interest. Assume pessimistically (optimistically for the attacker) that the attacker is able to confirm whether a hypothesized path passing through a divulged mapping at each normal mix is, in fact, a path traveled by one of his test messages through the cascade. We stress that this seems to be a generous advantage to give the attacker; we do not know how to confirm this ourselves. We note, however, that such a confirmation immediately determines the vector $(\pi_1, \pi_3, \ldots, \pi_{m-2})$—what the adversary wants to determine—since each fragile mix is uniquely determined by any single input-output mapping.

We informally claim in this setting that if $\rho < \tau$, the probability that the adversary can find the vector $(\pi_1, \pi_3, \ldots, \pi_{m-2})$ is negligible in m, the length of the base cascade. Intuitively, this follows from the fact that the total number of paths through normal mixes that the adversary can consider is at most $\rho^{\frac{m-1}{2}}$, which correspond to at most $\rho^{\frac{m-1}{2}}$ permutation vectors of a total of $\tau^{\frac{m-1}{2}}$ such vectors. Therefore, the probability that these vectors include a correct one is at most by $\left(\frac{\rho}{\tau}\right)^{\frac{m-1}{2}}$.

For example, consider a hybrid mix-net utilizing loop permutations. Suppose that the size of a batch is 1000, the length of a base cascade is 9 and a normal mix is willing to give up at most 10% of its input-to-output mappings. By the reasoning above, the probability that an adversary determines the correct fragile permutation vector to trace a message is no more than $0.1^4 = 0.0001$.

6.2 Incentives

The privacy guarantees of a hybrid mix network hinge on the assumption that a fragile mix will not disclose any information about its permutation for batches that include its own messages. This could be problematic in the presence of *strategic players* who speculate about others' strategies and adapt their behavior accordingly. That is, assuming that others will not disclose their permutations and thus also protect the anonymity of its own messages, a strategic fragile mix may decide to sell its permutation. If every fragile mix in a mix-net adopted this strategy, the system would collapse. Here, we sketch a simple model based on game theory by which we can mitigate this problem.

From the last section, we know that the adversary's chance to successfully trace a message relates to the number of fragile mixes in the hybrid cascade. A mix may expect its own messages to be protected by enough mixes. For example, assume that each fragile mix will be confident of the privacy of its own messages in a base cascade of $L \geq T$ fragile mixes,

where T is a threshold. Suppose further that each mix sends a message in each batch with probability p, and that both p and T are public knowledge.

Every mix can estimate $L = \bar{m}p$, where $\bar{m}$ is the number of fragile mixes in the hybrid cascade. To encourage fragile mixes to protect their own messages, we set $\bar{m} = T/p$. Thus, a mix will find that its optimal strategy is to keep its permutation secret for batches in which it sends messages, and to disclose (e.g., sell) its permutations for other batches. The mix must do the former, since otherwise the expected number of servers L in the base cascade will go below the threshold T. Via this strategy, individual fragile mixes will lock their behavior to a *Nash equilibrium*, in which no one benefits from unilaterally deviating from this strategy.

6.3 Efficiency

The efficiency of our protocol depends primarily on the complexity of zero-knowledge proofs, which we presented in Section 5. Since every fragile mix runs a proof protocol, the verifier's work is linear in the number of fragile mixes in a mix cascade. With a long mix-cascade, such a workload could be substantial.

One approach to improve verifier efficiency builds from the feature of our approach that checking a fragile mix's proof protects not only the verifier but also every other sender's privacy. This allows us to explore a verification strategy in which every fragile mix posts (in a non-interactive manner) its proof. Every verifier, instead of checking all these proofs, only randomly checks a constant number of proofs. Given sufficiently many verifiers, the probability that a proof has not been left unverified will be small.

7. CONCLUSION AND FUTURE WORK

In this paper we have addressed a threat in the context of mix networks that has, to date, been largely ignored: the possibility that a mix administrator can be convinced to log and selectively disclose certain input-to-output message mappings. Our approach to mitigating this threat is *fragility*, i.e., requiring that the secrecy of all input-to-output mappings depends on the secrecy of each of them. Thus, this approach leverages the presumption that the mix administrator is interested in protecting the privacy of *some* messages (e.g., his own); our approach requires him to protect all the messages to do so. We defined the concept of fragile mixing and identified some fragile permutation families; we proposed zero-knowledge proofs that permit any verifier to confirm that a mix used a fragile permutation; and described the integration of fragile mixing into a larger mix cascade.

Our work introduces a new research direction, and offers several opportunities for future research. For example, one may ask if a similar approach can be implemented in decryption mix-nets. Unfortunately, it seems that a straightforward extension is not likely: re-encryption mix-nets allow a mix to conceal its permutation on a batch of messages from any observers, including the senders of these messages, while decryption mix-nets do not—and this fact is fatal to a fragile permutation. A way around this problem is to use link encryption to hide the communication flows between two decryption mixes, though since the receiving mix may collaborate with a message originator to expose its predecessor's fragile permutation, this solution is not especially effective. Another potential direction is to improve the effi-

---

[2] Here, we do not consider the permutation of the last mix m. Since the adversary knows the output of its test messages, the last fragile permutation is completely determined by all previous fragile permutations.

ciency of the zero-knowledge proofs proposed, or use *strong evidence* [18] instead of a proof to make the protocol easier to implement.

8. REFERENCES

[1] M. Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[2] M. Abe. Mix-networks on permutation networks. In *Proceedings of ASIACRYPT 1999*, volume 1716 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[3] A. Acquisti, R. Dingledine, and P. Syverson. On the Economics of Anonymity. In *Financial Cryptography (FC '03)*, 2003.

[4] O. Berthold, H. Federrath, and M. Kohntopp. Project anonymity and unobservability in the internet. In *Computers Freedom and Privacy Conference 2000 (CFP 2000) Workshop on Freedom and Privacy by Design*, April 2000.

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

[6] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[7] Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. In *Proceedings of EUROCRYPT 2000*, volume 1803 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[8] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A Reputation System to Increase MIX-net Reliability. In *Proc. Financial Cryptography (FC '02)*. Springer-Verlag, 2002.

[9] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, pages 186–194, New York, 1987. Springer-Verlag.

[10] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proc. 1992 AUSCRYPT*, 1992.

[11] J. Furukawa and K. Sako. An efficient scheme for proving a shuffling. In *Proc. 2001 CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[12] O. Goldreich. *The Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[13] C. Gülcü and G. Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.

[14] M. Jakobsson. A practical mix. In *EUROCRYPT 98*, 1998.

[15] M. Jakobsson. Flash mixing. In *Symposium on Principles of Distributed Computing*, pages 83–89, 1999.

[16] M. Jakobsson and A. Juels. Millimix: Mixing in small batches. In *DIMACS Technical Report*, June 1999.

[17] M. Jakobsson and A. Juels. An optimally robust hybrid mix network (extended abstract). In *Proceedings of Principles of Distributed Computing - PODC '01*. ACM Press, 2001.

[18] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX'02*, 2002.

[19] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, and M. Waidner. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 1998.

[20] M. Mitomo and K. Kurosawa. Attack for flash mix. In *Proc. 2000 ASIACRYPT*, Lecture Notes on Computer Science. Springer-Verlag, 2000.

[21] U. Möller and L. Cottrell. Mixmaster Protocol — Version 2. Unfinished draft, January 2000.

[22] A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. 2002 ACM conference on computer and communication security*, 2001.

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

[24] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *Information and Communications Security — First International Conference*, volume 1334, pages 440–444, Beijing, China, 11–14 1997. Springer-Verlag.

[25] M. Ohkubo and M. Abe. A Length-Invariant Hybrid MIX. In *Proceedings of ASIACRYPT 2000*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[26] C. Park, K. Itoh, and K. Kurosawa. All/nothing election scheme and anonymous channel. In *Proc. 1993 EUROCRYPT*, 1993.

[27] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection*, 1998.

[28] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society*, Washington, DC, USA, November 2002.

[29] P. S. Roger Dingledine, Nick Mathewson. Reliable MIX Cascade Networks through Reputation. In *Proc. Financial Cryptography (FC '03)*. Springer-Verlag, 2003.

[30] K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *Proc. 1995 EUROCRYPT*. Springer-Verlag, 1995.

[31] Y. Tsiounis and M. Yung. On the security of ElGamal based Encryption. In *Proceedings of PKC*, 1998.