Reputable Mix Networks

Philippe Golle

Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
E-mail: pgolle@parc.com

Abstract. We define a new type of mix network that offers a reduced form of robustness: the mixnet can prove that every message it outputs corresponds to an input submitted by a player without revealing which input (for honest players). We call mixnets with this property *reputable mixnets*. Reputable mixnets are not fully robust, because they offer no guarantee that distinct outputs correspond to distinct inputs. In particular, a reputable mix may duplicate or erase messages. A reputable mixnet, however, can defend itself against charges of having authored the output messages it produces. This ability is very useful in practice, as it shields the mixnet from liability in the event that an output message is objectionable or illegal.

We propose three very efficient protocols for reputable mixnets, all synchronous. The first protocol is based on blind signatures. It works both with Chaumian decryption mixnets or re-encryption mixnets based on ElGamal, but guarantees a slightly weaker form of reputability which we call near-reputability. The other two protocols are based on ElGamal re-encryption over a composite group and offer true reputability. One requires interaction between the mixnet and the players before players submit their inputs. The other assumes no interaction prior to input submission.

Keywords: Mix Network, Privacy, Anonymity.

1 Introduction

The motivation for this paper lies in the following question: Why is it so difficult to recruit volunteers willing to operate remailers? We identify two answers to that question:

1. The *overhead* of setting up and operating a remailer deters potential volunteers. This issue appears on the verge of resolution. Newly developed infrastructures for anonymous communication such as Mixminion [DDM03] offer remailing clients that are fast becoming as easy to set-up and operate as some hugely popular Peer-to-Peer clients.

2. The *risk and liability* (both real and perceived) of operating a remailer deters potential volunteers. Remailers may unwittingly relay illegal or objectionable messages (e.g. a death threat, child pornography), but can neither filter those messages (since they are encrypted) nor defend themselves effectively against accusations of having authored them. Potential volunteers thus shy away from the risks and liability that come with operating a remailer or mix server.

This paper addresses the second problem (the risk and liability of operating a remailer), which we now examine in more detail. We note first that any legal protection that remailers might enjoy, while useful as an ultimate recourse, will not be sufficient to convince a lot of volunteers that it is safe to run a remailer. Few volunteers will run remailers if they must face the threat and inconvenience of lawsuits, even if they were guaranteed to win in court every time. To spur the deployment of remailers on a large scale, what is needed instead is a simple technological solution that allows a remailer to prove beyond a doubt that it did not author a certain message.

We review first briefly two possible approaches to this problem and explain why they are unsatisfactory:

- **Robust mixnets.** A first approach is to base anonymous communication on robust mix networks [OKST97,Nef01,JJR02]. A robust mixnet can prove that the set of output messages it produces is exactly a permutation of the set of input messages it received from players, without revealing anything about the correspondence between inputs and outputs. Thus in particular, a robust mixnet can prove that it did not produce any output that was not already present in the input. This proof is given without compromising the privacy of any player. But robust mixnets are computationally expensive and require many rounds of interaction between mix servers. This makes them unsuitable for a general-purpose anonymous communication system.
- **Keeping Logs.** Another approach is for each mix server (or remailer) to keep logs of the relationships between the input messages it has received and the output messages it has produced (if the mix derives its randomness from a seed, it only needs to remember that seed, since the log can be reconstructed from it). If a mix server is accused of delivering an offensive or illegal email, it can defend itself by breaking the privacy of the message in question and revealing the corresponding input message. The mix servers are exonerated, but at the price of exposing the player who submitted the message. This approach is extremely undesirable. Indeed, the existence of logs leaves mixnets vulnerable to threats and makes them maybe more willing to compromise the privacy of their users to avoid trouble for themselves. Keeping logs thus weakens the privacy of all users.

The contribution of this paper is to propose a new type of mixnet, called *reputable mixnet*, which offers a limited form of robustness. A reputable mixnet can prove that every message it outputs corresponds to an input submitted by a player, without revealing which input (at least as long as players obey

the protocol: see discussion below). Reputable mixnets thus offer a simple and effective defense against accusations of authoring objectionable outputs.

Reputable mixnets are much more computationally efficient than even the fastest robust mixnets [JJR02,BG02]. However, reputability is a weaker property than robustness, because the mix can not prove that distinct outputs correspond to distinct inputs. In particular, a malicious server in a reputable mix can erase or replicate inputs with impunity. In Chaumian mixnets, servers can easily detect and eliminate multiple fraudulent copies of the same input. In re-encryption mixnets on the other hand, the ability to make copies of inputs goes undetected and potentially allows a malicious server to trace selected inputs [PP89,Pfi94].

For this attack to be successful, the adversary must control a non-negligible fraction of remailers. This is easier to achieve when the total number of remailers is small. If reputable mixnets lead to a large increase in the number of volunteers who operate remailers, reputable mixnets will offer much better anonymity in practice than systems with fewer remailers, in spite of the attack just described. Indeed, we believe that the small number of remailers currently constitutes a much graver threat to players' privacy than the risk of a dishonest remailer tracing a message by making copies of it.

We propose three very efficient protocols for reputable mixnets, all synchronous. The first protocol is based on blind signatures. It works both with Chaumian decryption mixnets or ElGamal-based re-encryption mixnets, but guarantees a slightly weaker form of reputability which we call near-reputability (defined below in section 1.1). The other two protocols are based on ElGamal re-encryption over a composite group and offer true reputability. One requires interaction between the mixnet and the players before players submit their inputs. The other assumes no interaction prior to input submission.

Organization of the paper. In the rest of this section, we present our model, define near-reputable and reputable mixnets and review related work. In section 2, we present a near-reputable mixnet protocol based on blind signatures. We present our two reputable protocols in section 3, and conclude in section 4.

1.1 Model and definition

We review first briefly how mix networks are used for private communication. To send an anonymous message to Bob, Alice encrypts her message (including Bob's email address) under the public key of the mixnet. The mixnet collects ciphertexts submitted by different senders to form a batch. It decrypts (or re-encrypts, then decrypts) all these ciphertexts and delivers them (in a random order) to their recipients. The identity of the sender of the message is hidden from the recipient of the message.

We define a reputable mixnet as follows. Let $\mathcal{A}$ be the set of input ciphertexts submitted to a mixnet, and let $\mathcal{B}$ be the corresponding set of output plaintexts produced by the mixnet (for re-encryption mixnets, we assume that the outputs are jointly decrypted after having been re-encrypted by ever server, hence the

outputs are always plaintext). Since we consider synchronous mixes throughout this paper, $\mathcal{A}$ and $\mathcal{B}$ are simply the inputs and outputs of a batch.

Definition 1. (Reputable mixnet) *A mixnet $\mathcal{M}$ is f-reputable if for every batch output $\mathcal{B}$ there exists a subset $f(\mathcal{B}) \subseteq \mathcal{B}$ such that the mixnet can prove to a third party that unless all mix servers collude, every output in $f(\mathcal{B})$ is a decryption of an input in $\mathcal{A}$ without revealing which one.*

Example. Let f_0 be the function that maps every set to the empty set. Every mixnet is trivially f_0-reputable. Let f_1 be the identity function. A mixnet that is f_1-reputable can prove that every output it produces is the decryption of an input received from a player.

Definition 2. (Near-reputable mixnet) *A mixnet $\mathcal{M}$ is f-near-reputable if for every batch output $\mathcal{B}$ there exists a subset $f(\mathcal{B}) \subseteq \mathcal{B}$ and a set of players $\mathcal{P}_\mathcal{B}$ such that the mixnet can prove to a third party that unless all mix servers collude, every output in $f(\mathcal{B})$ was authored by one of the players in $\mathcal{P}_\mathcal{B}$ without revealing which one.*

Example. If we define $\mathcal{P}_\mathcal{B}$ to be the set of players who submitted the inputs in $\mathcal{A}$, then f-near-reputable and f-reputable are the same. But near-reputation is more general than reputation, because the set of players $\mathcal{P}_\mathcal{B}$ need not be the players who submitted the inputs in $\mathcal{A}$. For example, the set $\mathcal{P}_\mathcal{B}$ could be a set of players who committed themselves to submit at a later time.

Our definitions of reputation and near-reputation make no assumption about whether the players execute the submission protocol correctly or about whether the servers execute the mixing protocol correctly (although, as we shall explain, the function f depends on the behavior of players and servers). The only assumption we make is that the number of servers that collude is below a certain threshold. As long as the number of colluding servers is below the threshold, servers gain nothing by colluding. But if the number of colluding servers reaches or exceeds the threshold, they may falsely convince a third party that they are f-reputable or f-near-reputable even though they are not.

A reputable or near-reputable mixnet makes no claim about how messages are processed between the time they are input and the time they are output. In particular, reputable mixnets do not guarantee that messages are correctly mixed, nor even that messages will go through every server along the mixing route that was set for them. Recall that the goal of reputable mixnets is to let servers defend themselves against accusation of having authored their output, without breaking the privacy of the outputs. This indirectly enhances the privacy of players, but it does not offer privacy where there is none (e.g. if servers do not mix correctly).

Finally it should be made clear that, naturally, an f-reputable mixnet does not prove that messages in $f(\mathcal{B})$ did not originate with a mix server. Indeed, servers may submit messages in the input of the mix like any other player. What a reputable mixnet can prove however is that every message in $f(\mathcal{B})$ was

submitted by a player in the input (or a player in $\mathcal{P}_{\mathcal{B}}$ for near-reputable mixnets). That fulfills our stated goal of shielding the servers from liability related to the outputs they produce. We do not and can not shield servers from liability related to other activities they may choose to engage in, such as for example being a player as well as a server.

An analogy might help clarify this last point. A reputable mixnet is akin to a postal mail sorting facility where workers are searched before reporting for work to make sure they do not smuggle letters from outside into the sorting facility. This search procedure shields the sorting facility from liability in case illegal pieces of mail are found to have passed through the facility. Indeed, those pieces of mail may have been sorted at the facility, but were provably not created there. Of course, nothing can prevent a worker at the facility from mailing illegal material from home while off-duty. The search procedure proves the innocence of the facility and of the workers while on-duty at the facility, but need not and does not account for what workers do when they are off-duty. Similarly, a reputable mixnet proves only the innocence of the mixing operation (which is all we care about).

1.2 Related Work

Chaum defined mix networks in his seminal paper [Cha81] in the context of an anonymous email system. Chaum's mixnet, based on RSA decryption, is not robust. His paper inspired a long line of work on non-robust private communication, from Type I Cypherpunk remailers [Par96] to Onion Routing [GRS99], Babel [GT96] and most recently Mixminion [DDM03]. We refer the interested read to [FHP] for a complete annotated bibliography of the most significant results in anonymity.

A parallel but separate line of work is the development of robust mixnets based on ElGamal re-encryption. The first techniques for robust mixing were based on cut-and-choose zero-knowledge proofs [SK95,OKST97] that are very computationally expensive . A number of recent schemes [FS01,Nef01,JJR02] offer more efficient proofs of robust mixing but they remain very expensive compared to Chaumian mixes. As already noted, robust mixnets can prove to a third party that the set of output they produce is a permutation of the set of inputs without revealing the correspondence between inputs and outputs. This implies that robust mixnets are Id-reputable (where Id is the identity function), but the converse is not true. An Id-reputable mixnet does not guarantee that distinct messages in the output correspond to distinct messages in the input. The near-reputable and reputable mixnet protocols described in this paper are much more computationally efficient than robust mixnets.

In our protocols, we use blind signature to authenticate messages. This approach is similar to the work on hybrid mixes by Jakobsson and Juels [JJ01].

2 Near-Reputable Mixnet with Blind Signatures

In this section, we present a protocol for a near-reputable mixnet based on blind signatures [Cha82]. Our protocol works both with decryption or re-encryption mixnets. We assume throughout that messages are mixed synchronously in batches.

Recall first that a blind signature scheme allows a player to obtain a signature S on a message m without revealing m to the entity that produces the signature. The RSA signature scheme and the Schnorr signature scheme for example have blind variants. For concreteness, we base our presentation on RSA blind signatures. Let N be an RSA modulus, e a public verification key and d a private signing key such that $ed = 1 \mod \varphi(N)$. To obtain a blind signature on message m, a player chooses at random a blinding factor $r \in \mathbb{Z}_N$ and submits $mr^e \mod N$ to the signer. Let S' be a signature on mr^e. It is easy to see that the signer learns nothing about m, yet $S = S'/r$ is a signature on m.

We show how to use blind signatures to convert a mixnet into a near-reputable mixnet. Let $\mathcal{M}$ be a mixnet protocol (either decryption or re-encryption mix) that processes messages synchronously. The description of $\mathcal{M}$ consists of a public encryption function E and a mixing protocol MIX. Players submit encrypted inputs $E(m)$ to the mixnet. The protocol MIX takes a batch of encrypted inputs and produces as output the corresponding plaintexts in random order.

Near-reputable mixnet. In a setup phase, the near-reputable mixnet publishes the encryption function E. In addition, the mix servers jointly generate an RSA modulus and a shared signing key d and they publish the corresponding verification key e (see [BF01] for detail on shared generation of RSA parameters). These parameters may be replaced at any time, but they are meant to be long-lived and reused to mix many batches. In what follows, the symbol $||$ denotes string concatenation. The mixnet processes a batch in two phases:

1. **Signature phase.** All mix servers jointly generate a random (say, 160-bit) value b. This value serves as a unique identifier for the current batch. The value b is published. To submit a message m for processing in batch b, a player obtains from the mixnet a blind signature S on $m||b$ and submits as input $E(m||b||S)$. We denote by $\mathcal{P}_b$ the set of players who request a blind signature during batch b. Note that a player may cheat and obtain a signature on an improperly formatted message (for example, a message with the wrong batch identifier).
2. **Mixing phase.** At some point (e.g. when enough inputs have been submitted), the mixnet stops issuing signatures and starts mixing the batch. The mixnet executes the protocol MIX on the batch of inputs and outputs in random order the set of plaintexts $m||b||S$. The mixnet need not verify that the signature S on $m||b$ is valid.

When mixing is over, the mixnet returns to the signature phase. It generates a new random batch identifier b' and starts accepting messages for that new batch.

Proposition 1. *We define the function f as follows. The set $f(\mathcal{B})$ contains every message in $\mathcal{B}$ that was (1) submitted by an honest player and (2) processed*

correctly by the mix. The mixnet defined above is f-near-reputable for the set of players $\mathcal{P}_b$.

Given the definition of the set $f(\mathcal{B})$, it comes as no surprise that every output in $f(\mathcal{B})$ was authored by a player. The property of being near-reputable lies not in the truth of that statement, but in the ability to *prove* to a third party that the statement is true.

Proof. (Proposition 1) By definition of the set $f(\mathcal{B})$, every output $m||b||S$ in $f(\mathcal{B})$ contains a valid signature S on $m||b$. This signature allows the mixnet to convince a third party that the author of message $m||b$ belongs to the set of players $\mathcal{P}_b$ who requested signatures during batch b. Indeed, the only window of opportunity for a player to obtain with non-negligible probability a valid signature for a message submitted in batch b is after the value b is published, and naturally before submission is closed for batch b. Furthermore since the signature is blind, it reveals nothing about the correspondence between outputs in $f(\mathcal{B})$ and players in $\mathcal{P}_b$. Note that a regular signature would afford players no privacy precisely because it would expose the correspondence between $f(\mathcal{B})$ and $\mathcal{P}_b$. □

If accused of authoring a message that is not in $f(\mathcal{B})$ (e.g. a message that contains an invalid signature, or no signature at all), the mixnet has no other option but to break the privacy of that message and trace it back either to an input, or to a malicious server who introduced the input into the batch fraudulently. In either case, honest servers are exonerated. It is consequently in the best interest of players and servers to execute the protocol correctly.

Efficiency. The protocol is efficient: the only additional overhead comes from the shared signing of messages, at a cost of one exponentiation per server per message.

3 Reputable Mixnet Protocols

In this section, we propose two reputable mixnet protocols. These protocols are based on synchronous ElGamal re-encryption mixnets over composite groups. The section is organized as follows. We review first briefly the properties of the ElGamal cryptosystem. We present next a reputable mixnet protocol. We end with a variant protocol that eliminates the need for interaction between mix servers and players prior to input submission.

3.1 Preliminaries: The ElGamal Cryptosystem

We review briefly the ElGamal cryptosystem and introduce two properties of this cryptosystem that underpin the design of our reputable mixnets: the re-encryption and ciphertext-signing operations. The first property, re-encryption, is well-known and has long been used in the design of mixnets [OKST97]. The

second property, ciphertext-signing, is used here for the first time to guarantee the reputation of the mixnet.

ElGamal is a probabilistic public-key cryptosystem, defined by the following parameters: a group G of order n, a generator g of G, a private key x and the corresponding public key $y = g^x$. To encrypt a message $m \in G$, a player chooses a random element $s \in \mathbb{Z}_n$ and computes the ciphertext $E(m) \in G \times G$ as the pair $E(m) = (g^s, my^s)$. To decrypt, one uses the private key x to compute $my^s/(g^s)^x = m$. ElGamal is semantically secure under the assumption that the Decisional Diffie Hellman (DDH) problem is hard in the group G.

Re-encryption. Re-encryption mixnets exploit the fact that the ElGamal cryptosystem allows for *re-encryption* of ciphertexts. Given an ElGamal ciphertext $C = (g^r, my^r)$ and the public key y under which C was constructed, anyone can compute a new ciphertext $C' = (g^{r+s}, my^{r+s})$ by choosing a random s and multiplying the first and second element of the pair by g^s and y^s respectively. Note that C and C' decrypt to the same plaintext m. Furthermore, without knowledge of the private key x, one cannot test if C' is a re-encryption of C if the DDH problem is hard in G.

Signatures. In typical implementations of ElGamal, the group G is a multiplicative subgroup of $\mathbb{Z}_p$ of prime order q. We will instead be working with ElGamal over a composite group. Let p, q be two large primes (say, 1024 or 2048 bits) and let $N = pq$. We define G to be the group $\mathbb{Z}_N^*$. The ElGamal encryption, decryption and re-encryption operations are exactly the same over a composite group as over a group of primer order. See [McC88,FH96] for more detail on the security of ElGamal over composite groups.

We can define an RSA signature scheme for ElGamal ciphertexts over a composite group G as follows. We choose a public exponent e and compute the corresponding secret key d such that $ed = 1 \mod \varphi(N)$, where $\varphi(N) = (p-1)(q-1)$. Let $E(m) = (g^r, my^r)$ be an ElGamal ciphertext over G. We define the signature on $E(m)$ to be $S = ((g^r)^d, (my^r)^d) = (g^{rd}, m^d y^{rd})$. The signature S is itself an ElGamal encryption of the plaintext m^d in the cryptosystem defined by the public parameters (G, g^d, y^d) and private key x. The signature S can therefore be re-encrypted like any other ElGamal ciphertext.

Signatures and re-encryption. Let S be a signature on $E(m)$. We have already noted that both S and $E(m)$ can be re-encrypted. It remains to show how to verify signatures. The common strategy to verify a signature fails: raising both elements of the ciphertext S to the power e yields nothing recognizable after S or $E(m)$ have been re-encrypted. Instead, we verify the signature by first decrypting S (we get plaintext a) and $E(m)$ (we get plaintext b) and verifying that $a^e = b$. This verification is compatible with re-encryption.

3.2 A First Reputable Mixnet

The reputable mixnet protocol presented here is based on a re-encryption mixnet with ElGamal over a composite group. While our presentation is self-contained, readers unfamiliar with re-encryption mixnets may consult [OKST97] for more detail.

Setup. The mix servers jointly generate an RSA modulus N, a shared signing key d and the corresponding public verification key e. See [BF01] for an efficient protocol to generate shared RSA parameters. The mix servers publish the modulus N and the public key e. The mix servers then jointly generate the parameters of an ElGamal cryptosystem over the composite group $\mathbb{Z}_N^*$. Each mix server holds a share of the private key x. The corresponding public key $y = g^x$ is published. See [Ped91,GJKR99] for efficient protocols to generate shared ElGamal keys. Finally, the mix servers jointly compute and publish g^d and y^d. All these parameters can be changed at any time, but they are meant to be long lived and reused to process a number of batches.

Creation of a batch of inputs. The creation of a batch of inputs proceeds in 2 steps.

1. **Submission of inputs.** All mix servers jointly generate a random (say, 160-bit) value b. This value serves as a unique identifier for the current batch. The value b is published. Players submit their input m to the batch as follows. A player encrypts $m||b$ under the ElGamal public key of the mixnet to create the ciphertext $E(m||b) = (g^r, (m||b)y^r)$ and posts this ciphertext to a bulletin board. Note that players may submit inputs that are improperly formatted.
2. **Signing.** When all ciphertexts have been posted to the bulletin board, the mix servers jointly compute for every ciphertext $E(m||b)$ a threshold signature $S = ((g^r)^d, ((m||b)y^r)^d)$ and append this signature to the ciphertext. The bulletin board now contains pairs of the form $(E(m), S)$.

Mixing. Each server in turn mixes and re-encrypts the set of messages in the batch. The first server takes as input the list of pairs $(E(m), S)$ posted on the bulletin board. Let $E(m) = (a, b)$ and $S = (\alpha, \beta)$ be one particular input. To re-encrypt this input, the server chooses independently at random $r, r' \in \mathbb{Z}_N$ and computes two new pairs (ag^r, by^r) and $(\alpha g^{dr'}, \beta y^{dr'})$. Every pair is re-encrypted in the same way. Finally, the first server outputs these pairs of values in a random order. These outputs become the input to the second server, who processes them in the same way. More generally, mix server M_i receives as input the set of pairs of ElGamal ciphertexts output by mix server M_{i-1}. Server M_i permutes and re-randomizes (i.e. re-encrypts) each element of all these pairs of ciphertexts, and outputs a new set of pairs of ciphertexts, which is then passed to M_{i+1}.

Decryption. The mix servers jointly decrypt the final output. Note that the servers need not provide a zero-knowledge proof of correctness for decryption

(we are not aiming for robustness). The servers need not verify the validity of signatures either.

Proposition 2. *We define the function f as follows. The set $f(\mathcal{B})$ contains every message in $\mathcal{B}$ that was (1) submitted by an honest player and (2) processed correctly by the mix. The mixnet defined above is f-reputable.*

Proof. Signatures ensure that every output in $f(\mathcal{B})$ corresponds to an input, assuming that the number of colluding servers is below the threshold required to generate a fake signature. □

Efficiency. The computational and communication cost of our reputable mixnet is exactly twice that of the mixing cost of a standard plain-vanilla re-encryption mixnet, since every input consists of a pair of ElGamal ciphertexts. All exponentiations for re-encryption depend only on fixed public parameters and can therefore be pre-computed before the batch is processed.

3.3 A Reputable Mixnet With Non-Interactive Submission of Inputs

The near-reputable and reputable protocols described so far both rely on the ability of the mix servers to interact with players *before* they encrypt and submit their inputs. Indeed, the mix servers must communicate the batch identifier b to the players. In some circumstances, this requirement may be too restrictive. A mix server may be given as input a set of ciphertexts, without the ability to interact with the players who created these ciphertexts.

Consider for example the last k servers in a mix cascade that consists of $K > k$ servers. These k servers may decide to form a reputable sub-cascade and prove that every ciphertext they output corresponds to one of their input ciphertexts. The protocol of the previous sections do not work in this case. In this section, we describe a variant of the reputable protocol of section 3.2 that allows a reputable mixnet to work with already encrypted inputs.

Variant protocol. In a setup phase, the mix servers jointly generate and publish an RSA modulus N, then jointly generate the parameters of an ElGamal cryptosystem over the composite group $\mathbb{Z}_N^*$. The secret key x is shared among the servers while the corresponding public key $y = g^x$ is published. The submission of inputs involves the following 2 steps:

1. **Generation of a signing/verification key pair.** All mix servers jointly generate a shared RSA signing key d and the corresponding public verification key e over the group $\mathbb{Z}_N$. New keys d and e are generated for each batch.
2. **Submission and signing of inputs.** Players submit their input m encrypted under the ElGamal public key of the mixnet as $E(m) = (g^r, my^r)$. The mix servers jointly compute for every ciphertext $E(m)$ a threshold signature $S = ((g^r)^d, (my^r)^d)$ and append this signature to the ciphertext. The bulletin board now contains pairs of the form $(E(m), S)$.

From here onwards, the mixing and decryption proceed as in the protocol of section 3.2. In essence, the signatures in this protocol are tied to a batch not via a batch identifier appended to the plaintext, but rather by using a new and different signing/verification key pair for every batch.

Proposition 3. *We define the function f as follows. The set $f(\mathcal{B})$ contains every message in $\mathcal{B}$ that was processed correctly by the mix. The variant mixnet defined above is f-reputable.*

Observe that here the set $f(\mathcal{B})$ depends only on the honesty of the servers, not that of the players. This is arguably a small difference, since we have already shown that it is in the best interest of players to execute the protocol correctly. Nevertheless, this variant protocol offers the strongest guarantee of reputability.

4 Conclusion and Future Work

We defined a new property of mix network: reputability. Reputable mixnets can prove to a third party that every output they produce is a decryption of an input submitted by a player without revealing which input for honest players (dishonest players forfeit their own privacy). In practice, reputable mixnets offer a twofold advantage.

- First, reputable mixnets are almost as efficient as non-robust mixnets, yet they offer better privacy than non-robust mixnets because they are not vulnerable to accusations of having authored the outputs they produce. For private communication systems, we believe that such accusations constitute a much graver threat to players' privacy than the risk of a mix server cheating in the execution of the mixing protocol.
- Second, reputable mixnets may spur the development of anonymous email systems as volunteers need not fear the threats and liability to which non-robust non-reputable mixnets are exposed.

We proposed three very efficient protocols for near-reputable and reputable mixnets, all synchronous. An interesting direction for future work would be to look into the design of reputable protocols for asynchronous mixnets.

Acknowledgements

The author would like to thank the anonymous referees for valuable comments that helped improve the presentation of this paper.

References

[BF01] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Journal of the ACM (JACM)*, Vol. 48, Issue 4, pp. 702–722, July 2001.

[BG02] D. Boneh and P. Golle. Almost entirely correct mixing with applications to voting. In *Proc. of the 9th ACM Conference on Computer and Communications Security,* pp. 68–77. ACM Press, 2002.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, 24(2):84-88, 1981.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Proc. of Crypto '82*, pp. 199–203. Plenum Press, N.Y., 1983.

[CFN88] D. Chaum, A. Fiat, M. Naor. Untraceable electronic cash. In *Proc. of Crypto'88*, pp. 319–327. Springer-Verlag, LNCS 403.

[DDM03] G. Danezis, R. Dingledine and N. Mathewson. Mixminion: design of type III anonymous remailer protocol. In *Proc. of the 2003 IEEE Symposium on Security and Privacy*, pp. 2-15. On the web at `http://mixminion.net/`

[FH96] M. Frankling and S. Haber. Joint encryption and message-efficient secure computation. In *Journal of Cryptology*, pp. 217–232, Vol. 9, No. 4, Autumn 1996.

[FHP] The Free Haven Project. On the web at `http://www.freehaven.net/anonbib/`

[FS01] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of Crypto '01*, pp. 368-387. Springer-Verlag, 2001. LNCS 2139.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *Proc. of Eurocrypt '99*, pp. 295-310. LNCS 1592.

[GRS99] D. Goldschlag, M. Reed and P. Syverson. Onion routing for anonymous and private internet connections. In *Communications of the ACM*, 42(2):39-41, 1999.

[GJJS03] P. Golle, M. Jakobsson, A. Juels and P. Syverson. Universal re-encryption for mix networks. In *Proc. of the 2004 RSA Conference, Cryptographer's track.*

[GT96] C. Gulcu and G. Tsudik. Mixing E-mail with Babel. In *Proc. of Network and Distributed Security Symposium - NDSS '96*. IEEE, 1996.

[JJ01] M. Jakobsson and A. Juels. An optimally robust hybrid mix network. In *Proc. of PODC '01*, pp. 284–292. ACM Press, 2001.

[JJR02] M. Jakobsson, A. Juels and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of USENIX'02.*

[McC88] K. McCurley. A key distribution system equivalent to factoring. In *Journal of Cryptology*, pp. 95–105, Vol. 1, No. 2, Autumn 1988.

[Nef01] A. Neff. A verifiable secret shuffle and its application to E-Voting. In *Proc. of ACM CCS'01*, pp. 116-125. ACM Press, 2001.

[OKST97] W. Ogata, K. Kurosawa, K. Sako and K. Takatani. Fault tolerant anonymous channel. In *Proc. of ICICS '97*, pp. 440-444, 1997. LNCS 1334.

[Par96] S. Parekh. Prospects for remailers. *First Monday,* 1(2), August 1996. On the web at `http://www.firstmonday.dk/issues/issue2/remailers/`

[Ped91] T. Pedersen. A Threshold cryptosystem without a trusted party. In *Proc. of Eurocrypt'91*, pp. 522-526, 1991.

[PP89] B. Pfitzmann and A. Pfitzmann. How to break the direct RSA-implementation of mixes. In *Proc. of Eurocrypt '89*, pp. 373-381. Springer-Verlag, 1989. LNCS 434.

[Pfi94] B. Pfizmann. Breaking an Efficient Anonymous Channel. In *Proc. of Eurocrypt'94*, pp. 332–340. Springer-Verlag, LNCS 950.

[SK95] K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *Proc. of Eurocrypt '95*. Springer-Verlag, 1995. LNCS 921.