Guarantees for Customers of Incentive Anonymizing Networks

Timothy Atkinson and Marius Silaghi

Florida Tech
{tatkinso,msilaghi}@fit.edu

Abstract. We raise and propose solutions to the problem of guaranteeing that a user of incentive remailing services for anonymization cannot lose money if he does not get full service, i.e., if his message does not reach its destination. Applications such as voting over the Internet or reviewing of articles require anonymous delivery of messages. An anonymizing technique was proposed several decades ago by Chaum and is based on a group of volunteer agents called *mixnet*. However, mixnets are not yet widely known and used today, and one often mentioned reason is the lack of incentives for volunteers. A recently proposed solution is based on adding digital coins to messages, such that each volunteer can extract only the digital coin designated as a payment for her. However, registered volunteers can sabotage the system by extracting and using their coins without performing their task — which consists of forwarding anonymized messages. The main improvement we propose is to guarantee that no money is lost by the user without getting his message at the destination. This is an essential property for a viable service. Solutions described are based on handshaking mechanisms where each volunteer gets her payment (or key to decrypt the payment) from the agent to which she is expected to forward the message, or from the destination using a public board or a reply message. This ensures that a volunteer gets her financial support only if she fulfills her task. We discuss how techniques for non-repudiation of receipt of a message, together with reputation systems, can address the remaining problems.

1 Introduction

Citizens of many countries voting or signing citizens' petitions overseas pay postage to have their vote delivered. Often they do not get any confirmation that the envelopes containing their votes reached the Electoral Commission. As another common application of anonymization services [11, 2], police can also offer a tip hot-line that guarantees the tipster's anonymity.

Getting the vote to the Electoral Commission (EC) so that no observer could discover the originator is not easy with the current Internet. If a citizen, Bob, sends his vote directly to the EC, then Mallory, a malicious person that gained access to the server of the EC, can learn how Bob votes by associating his IP address with the received message. Even if Bob decides to use a third party server to forward his message to the EC, he would have to explicitly trust that the third party server is not controlled by Mallory.

A previously proposed solution to this problem, is (given the availability of a collection of volunteers agents $S = \{S_1...S_M\}$) to set up a Chaumian mixnet [7]. Bob picks a random subset $\{A_2, A_3, ..., A_{N-1}\} \subseteq S$ of servers from the collection of available servers and chooses a random ordering on them, $P_S = A_2, A_3, ..., A_{N-1}$, called a *path*. The sender will be denoted A_1 and the destination will be denoted A_N.

Each server A_i makes available a public key, E_i, for some asymmetric[1] crypto-system. Then, to force the message to visit each server along the path in turn; Bob will encrypt the message as follows:

$$E_2(A_3, E_3(A_4, ...E_N(message)...)). \tag{1}$$

The straightforward implementation where each message is encrypted repeatedly using this key scheme is called *onion encryption*. The message is then sent along the agents on the path P_S, each agent A_i removing his level of encryption E_i to retrieve the identity of the next volunteer and the message to be send to him. Agents are supposed to not forward a message as soon as it was received, but to delay it randomly and to mix it with other messages it forwards. The destination A_N obtains the message. This mechanism solves the problem if and only if there exists at least one server which is not controlled by Mallory. Note that each server is assumed to belong to a different owner, and it makes no sense to have one person or company set up more than one server. We assume here that other problems, such as losing messages or duplication of messages are solved at the application level (techniques that address these issues inside certain types of mixnets are detailed in [18]).

However, servers cost money to operate (network, rack space, power, operating costs, etc), which means someone must pay for the servers. Bob cannot pay to volunteers for these servers directly, since such payments would provide information about his vote, whenever Mallory can gather data (e.g., from routers) on the pattern of communication of the volunteers paid by Bob. Also, political and economical reasons may make a state-sponsored or foundation-sponsored mixnet non-viable, so another method needs to be discovered to compensate servers.

The idea of use market competition to enhance anonymity has been raised in [29, 33]. The idea is to modify the original message to include digital coins at each layer of encryption. These coins will then be distributed as the message passes from server to server. Using this solution, Bob can choose his own level

[1] i.e., public key based

of security (given by the number and identities of the volunteers on his path) without being tied down by the state's budget or suffer the security weaknesses of paying the servers directly. Because of competition and low operation costs, the prices charged by individual servers can be expected to be fair. Those servers that try charging too much will be eliminated by those servers which will charge less. The viability of the system in [33] is based on the assumption that if a server has poor performance, it will eventually be discovered. Then the server would lose reputation, would be eliminated from future mixnets and this would create a self-regulating system. To allow payments to servers along the message path, the structure of the message is modified as follows:

$$E_2(coin_2, A_3, E_3(coin_3, A_4, ...E_N(message)...)), \tag{2}$$

where $coin_i$ is a digital coin to be paid to A_i.

The above method solves the problem of the lack of incentive associated with setting up a server without sacrificing the original anonymity of a Chaumian mixnet. However, a problem is still posed by fraudulent and intermittent servers. The discovery of a fraudulent volunteer (attacker) is difficult and uncertain. Until the attacker is discovered, many users can lose money without getting service. The justice system is too expensive to invoke [23] and, for example, the users will simply abandon such a voting means. In the worst case the server may never be discovered and still get paid for every message it does not deliver [2]. If users lose money without getting the service, they will also lose their confidence and stop using the service, and the system will not survive.

One of our solutions is based on paying the volunteers only after the message is delivered. For this purpose, the payments (or keys to decrypt them) are sent to the destination with the message. The destination distributes them via a public board, or with a receipt (aka reply block) that can be returned by the destination in inverse direction along the path of the message. The reply block can be prepared by the sender itself in his message, similar to existing return channel techniques [29, 11, 17]. However, we also show how the task of constructing the reply block can be performed by volunteers, to let them control the payment path.

Also, to encourage volunteers to forward receipts, we propose that the receipt be prepared such that a volunteer cannot extract his coin directly. Instead volunteers need the cooperation of the next agent on the path of the receipt, the one closer to the sender. As such, an agent will know that he has to get a receipt with a payment for himself whenever he is requested to help another agent to

[2] It should be noted here that only discovered fraudulent servers will have their reputation adversely affected. Extending the reputation impact on suspected servers is unfair and can be exploited by Mallory to discredit uncorrupted servers.

retrieve a coin. This makes identification of fraudulent volunteers easier, and they can be reported to a reputation system.

The two techniques can be used separately and help in easily discovering fraudulent volunteers, and in guaranteeing that a customer of the system cannot lose money if he gets no service.

Because it is possible to construct the message so that there is no difference between Bob and some S_i along the path P_S, Bob can receive confirmation that his message was successfully delivered and remains anonymous.

We then other incentives for correct behavior in different circumstances, and describe how a reputation system can be used to address remaining problems.

Outline. In Section 2, we present a more detailed examination of the background, with a formal background in Section 3. Sections 4 to 7 detail the algorithms proposed in this paper.

2 Related Work

There have been many attempts to solve the problem of creating a guaranteed anonymous message service. Chaum's mixnet method [7] is the first. It solves the problem given a set of volunteer servers, but offering no incentives to create any such servers. Later, [16] presents a technique based on Chaum's mixnet which uses a proxy to hide the involvement of a mixnet from the application layer. It shows how to hide the destination and source by making them look like servers. An application of mixnets to voting was discussed in [25]. The use of digital currency in a mixnet is proposed in [29, 33]. Several techniques for digital anonymous cash and digital coins are known [28, 3, 4, 32, 8, 6, 5]. Some of these techniques require that cash givers answer a unique challenge, in order to avert double spending by merchants, and this is possible by publishing the challenge with the volunteer data (together with an expiration time).

In [20], it is shown how to improve reliability of servers by probabilistic checking. Techniques for replying to an anonymous message have been provided by existing implementations [29, 11, 17, 19]. They are based on the use of *reply blocks*, an attachment piggy backed on the original message and which provides a path for a reply message. The reply block is built as a regular mixnet message, but oriented from the target backward to the source, and it may visit a different path then the original message. Some of the most general existing versions are unfortunately subject to the spam attack.

[31, 13] study how to avoid timing analysis by generating false traffic. Many recent approaches use versions based on homomorphic encryption, and several mechanisms are known for zero knowledge proofs for the correctness of servers in that setting [10, 18]. Such correctness proofs require simultaneous submission

and handling of all votes and do not apply in the setting of paid volunteers, and specially for applications such a police tipping and petition signing.

Voting over the Internet is practiced in several countries such as Estonia and Switzerland [12, 27], while in other countries it is discouraged due to potential threats, such as denial of service attacks [21] on election day. Petition signing over the Internet is common in other countries, in particular in UK [24, 1].

3 Framework

Let us start with an initial system based on Chaum's work [7]. Let us recall that the set of all available volunteers is denoted $S = \{S_1, ..., S_M\}$, while the participants that get involved in a path through the mixnet are $A = \{A_2, ..., A_{N-1}\}$, $A \subseteq S$. In such a system, an end user A_1 (message sender to a destination A_N) decides to ask help from a subset $P_S = \{A_2, ..., A_{N-1}\}$ of the available service providers. Each server A_i makes available a public key, E_i, for some asymmetric crypto-system. As end user of the system, A_1 may pick the order it wishes on P_S. To send a message $message$ to A_N, A_1 prepares and sends the message in Equation 1 to the first server in P_S, A_2. A_2 decrypts the message, and forwards it to the next server in the list, A_3, who then decrypts the message and forwards it to the next server in the list, A_4, who again decrypts the message and again forwards the message, etc. When the packet reaches A_N, A_N retrieves the message. This algorithm solves the problem of anonymity of the source at the destination so long as A_1 believes that at least one of the servers in P is reliable. If A_N would like to learn the identity of the source, it needs to trace backward the path of the message. A_N must go through every server to figure out who gave them the message. If any one of the servers refuses to cooperate, then it is impossible for the destination to find the source.

To offer digital currency to the servers along the path, the message constructed by A_1 will look as in Equation 2 where $coin_i$ represents the digital cash for A_i [33]. From now on, for simplicity, the coin for A_i will be denoted C_i. As previously mentioned, the issue we want to raise and address is that, from the sender's perspective, this system poses financial risks that can discourage usage. In particular, any failure means not only lost effort but also lost money. As with many other mixnets, it does not provide a mechanism to verify whether the message was received by the destination and does not provide easy fraudulent server detection.

4 Proposed techniques

Here we raise awareness about an important property recommended for techniques allowing a user, Bob, to send a message to Alice such that no one knows that Bob has sent the message. Namely, some of the techniques proposed next guarantees that if Bob spends any money paying mixnet volunteers, then his message has made it to the destination. If the message fails to read the destination, then the volunteers cannot cash the digital coins within a deadline set by Bob, and therefore Bob can reuse them.

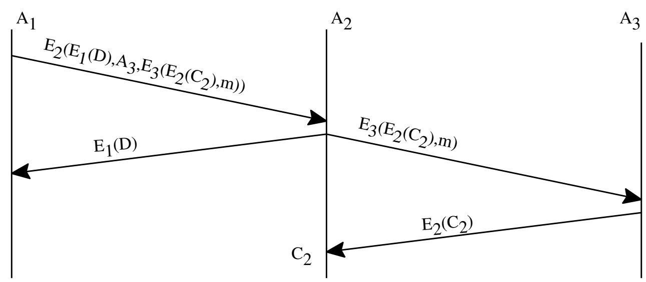

Fig. 1. Mixnet with payment by handshake.

when *receiveMessage*(M_i) **do**
  if *message contains dummy content for fake traffic* **then**
    if *(handshaking)* **then** generate fake reply;
    exit;
  end
  decrypt next hop A_{i+1};
  decrypt coin C_i, if available;
  if *(handshaking)* **then**
    extract $E_{i-1}(C_{i-1})$;
    send $E_{i-1}(C_{i-1})$ to A_{i-1} on a non-repudiated reception channel;
  end
  decrypt M_{i+1};
  send M_{i+1} to A_{i+1} on a non-repudiated reception channel, in fix-size chunks and with random delays;
end do.

Algorithm 1: Algorithm of A_i for forwarding messages to destination.

4.1 Utilizing the underlying network structure to encourage nodes to forward their data (handshaking reputation driven mixnet)

In a network, when a server A_i receives a communication c from the server A_{i-1}, by the nature of the socket connection, it knows that A_{i-1} has sent the message. Conversely, A_{i-1} must know who to transmit the message to, and so knows that the message must be sent to A_i. Hence for any node along the path A_2, ..., A_{N-1}, every server A_i knows (and only knows) the subpath A_{i-1}, A_i, and A_{i+1}. Since every server along the path in previously used mixnets already knows this information; we can modify the structure of the message slightly to provide a strong guarantee that no fraudulent servers can get paid. To do this, we doubly encrypt each coin C_i as

$$E_{i+1}(E_i(C_i)).$$

This forces every node along the path to at least forward the message if they want to be able to decrypt their money. Since every server knows who they have sent the message to, they can use the reputation system (and proof of transmission) should the next server refuse to remove their encryption from the money and return it back to the sender. The obtained protocol is called a *handshaking reputation driven mixnet*. An example with $N = 3$ is shown in Figure 1. Users can build such messages according to the recursion in the Equations 3 and 4,

$$M_N = E_N(E_{N-1}(C_{N-1}), P) \tag{3}$$

$$M_i = E_i(E_{i-1}(C_{i-1}), A_{i+1}, M_{i+1}); i \in (1, N) \tag{4}$$

where M_i specifies the message that will be received by A_i for transmitting payload P. The operations to be performed by a volunteer are described in Algorithm 1. This algorithm supports optional handshaking as well as direct payment, while volunteers generate fake traffic and exchange messages in fixed-size chunks as common with Type III remailers [11, 15]. The first operation in this algorithm (Line 1) consists of verifying whether the incoming message is just a fake message, generated between volunteers to make it harder to analyze the network traffic. If that is the case, the message is simply discarded (Line 3). However, if the mixnet protocol implements our method for returning digital coins as reply to the previous hop, then a fake reply will have to be generated (Line 2), otherwise traffic analyzers have a way to differentiate between real traffic and fake traffic.

If the message is not fake, the volunteer decrypts it. Each message contains the address of the next hop, extracted at Line 4. If some coins are also delivered immediately (as in [33]), then it is extracted at Line 5. If handshaking is implemented for paying the previous hop, then at this point the agent of the volunteer will extract the coin of the previous hop, $E_{i-1}(C_{i-1})$, and will send the coin with a reply (see Lines 6 and 7). The coin should be sent on a channel with non-repudiation of receipt (to allow the sender to defend himself on a reputation system).

Further the agent of the volunteer can decrypt the message to be sent to the next hop, M_{i+1} (Line 8), and sends it on a non-repudiated reception channel (Line 9). As in type III remailers [15], messages should be sent in fix-size chunks and with random delays to make traffic analysis harder.

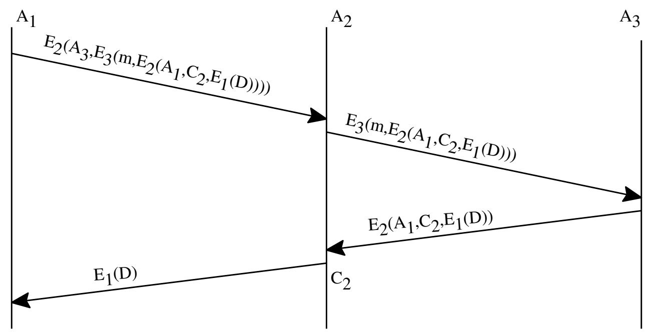

Fig. 2. Mixnet with payment during reply.

4.2 Forcing service delivered before payment (the reply-pay mixnet)

Our first solution, proposed above, provides good incentives for volunteers to perform the tasks for which they enrolled. However, it cannot give guarantees of message delivery or guarantees that no payment is made if the message is not delivered. An attacker having a big stake in disrupting the system prefers to lose some coins to make sure that messages are not delivered. We believe that, for the system to be successful, no client should pay anything without getting the service due. We assume that it is better to risk that a server does not get paid rather than risk that a client pays for nothing. We propose to withhold payment until after the message has been delivered. For this purpose, we are going to initially construct a reply block message R as if it came from A_N using the mixnet with payment or handshaking reputation driven mixnet described above, then append R to the payload of a standard Chaumian mixnet message. The new payload P for sending a message m along the Chaumian mixnet now looks like in Equation 5, where the dummy payload D is used to hide that A_1 is the message originator (if he can also be perceived as a volunteer).

$$P = m, E_{N-1}(A_{N-2}, E_{N-2}(E_{N-1}(C_{N-1}), ...A_1, E_1(E_2(C_2), D)...)) \qquad (5)$$

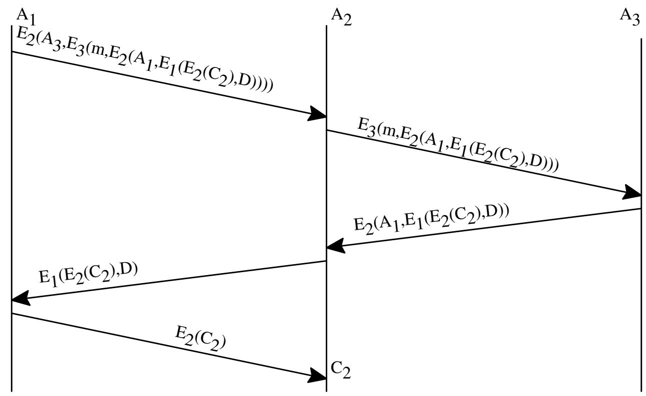

Fig. 3. Mixnet with handshaking payment during reply.

The full message constructed by A_1 is:

$$E_2(A_3, E_3(...(A_N, E_N(P)))).$$

An example with length $N = 3$ and without handshaking payment on the reply path is shown in Figure 2. A similar example but requiring handshaking for coins on the reply path (as in the previous technique) is shown in Figure 3.

The message of the sender can be built according to the recursions in Equations 6 to 10. R_i denotes the message to be received on the reply path by A_i and, as previously, M_i is the message to be sent on the forward path to A_i. The same algorithm can be used for returning reply blocks as the algorithm described for sending messages in our first solution.

$$R_1 = E_1(E_2(C_i), D) \tag{6}$$
$$R_i = E_i(E_{i+1}(C_i), A_{i-1}, R_{i-1}); i \in (1, N{-}1) \tag{7}$$
$$P = m, E_{N-1}(A_{N-2}, R_{N-2}) \tag{8}$$
$$M_N = E_N(P) \tag{9}$$
$$M_i = E_i(A_{i+1}, M_{i+1}); i \in (1, N-1) \tag{10}$$

We can still maintain the guarantee that as long as there exists a trustworthy secure agent along the path from A_2 to A_{N-1}, the anonymity of the sender is maintained. This is achieved because the backward message is in fact a Chaumian mixnet message, as well as the forward message (by the definition of the algorithm). As with previous mixnets enabling replies, volunteers need to change and discard their secret keys frequently, to make sure that they cannot be forced by an attacker to decrypt old reply blocks. No server A_i gains any more information about the structure of the message than they already had in a traditional Chaumian or reputation based mixnet. Also, because the message is making its way back to the original sender, the sender now has a built-in means to learn that the message was received by the destination, and knows that if the message is not received then they did not spend their money on a wasted message. The confirmation may indeed still be lost on the return path, e.g., if a server disappears, but no payment is done without service. Also, several applications (like voting and citizen's initiative signing) have alternative application-level means to confirm reception. E.g., the destination can post the value of vote on the Internet together with a verification ID, a large random number attached to the vote by the sender. Fast expiration of digital coins together with a guarantee of fast confirmation by the destination can ensure that a client does not attempt to resend, paying twice, if a message is just too slow. Expiration deadlines can be sent with the message (appended to the name of the next hop) to warn volunteers that they can discard a message whose coins have expired.

4.3 Proof of operational correctness

Our main goal in this paper is to guarantee that, while the shufflers have an incentive to participate, the user of a mixnet cannot lose money without getting service.

Theorem 1. *The protocol allows a user to send an anonymous message to a destination, and guarantees that the user does not spend money if the message is not delivered and the destination is honest.*

Proof. In order to prove this statement, we make the following *assumptions*:

1. There exists some A_i where $1 < i < N$ who drops the message before it reaches the next hop.
2. No A_i where $i \neq j$ can decrypt E_j.

Therefore we can *infer* that:

1. The message has successfully passed from A_1 to A_{i-1} without a problem by assumption 1.
2. Because A_i drops the message without forwarding it to A_{i+1} and due to being refused connection from A_{i+1}
   (a) We know that, by construction, all the coins are encrypted with E_N until the message reaches A_N. Therefore by assumption 2, no server $\{A_2$,..., $A_i\}$ will be able to decrypt their coin and get paid.

(b) Since A_i did not forward the message, we also know that servers $\{A_{i+1}$,..., $A_{N-1}\}$ did not get paid.

If the message is not delivered then no money is spent by the user.

If the destination is not honest and does not register the message but sends the payments to the mixnet, it will be discovered after the returning message reaches the sender. A sender receiving his receipt back and not seeing his message confirmed using application level mechanisms can infer that the destination is faulty.

5 Analysis of remaining attacks

The main advantage of the proposed method was discussed above. Namely no money is lost by the user without getting the message to the destination. We believe that this already outweights the new weakness consisting of the fact that operators may fail to get payment for messages that they handle. This new weakness (that we believe less critical) can be at least partly compensated by reputation systems and by setting prices that offset costs given experimented rates of payment failures. It would be nice to have a technique that avoids this risk for operators, and is a good research topic, but we believe that the improvement made by our approach is essential for the survivability of the service.

The main new strategies that attackers can now use against operators are:

- A denial of service attacker Malory can register as a volunteer and send messages on a chain that contains himself

$$\{Malory, A_2, A_3, ..., Malory, dummy\}.$$

  If Malory would never return money in the reverse direction, the intermediary operators are swamped relaying messages that will never be paid for. This is only partly a bad attack, if we take into account that fake messages are recommended for the security of the system, and as long as the generated traffic does not turn off the network (in which case standard approaches against DOS attacks can be used). Currently the only approach we know for fixing the mild version of this problem is based on reputation systems collecting information about operators accused of not returning payment. Employed communication can offer non-repudiation of reception (NRR) to help the reputation system [34, 22, 26, 30, 9].
- A fraudulent user may register a volunteer and have this always as the last one in his chain. This server would then never return a receipt to earlier operators. This is nothing more then a version of the previous attack. However, an user does not have incentives to apply this attack more than the previous

one. This is because it does not necessarily improve his anonymity (anybody able to trace messages would reach him on the shortest path used by the reply block). To claim that he received the message from somebody else he only needs to create the dummy part of the message as such.

- Users may send the message without a reply block (or with a reply block containing some invalid coins). Using the volunteer-based reply block construction proposed in the next section, operators can cooperate to trace such attackers, and can report them to a reputation system. Volunteers would cooperate in this reconstruction only if one proves to them that the message was badly formed, which can be done by revealing the cyphertexts. To avoid that the sender fakes to be a volunteer in this case to pass the accusation on an innocent volunteer, communications can offer non-repudiation of sending (e.g., based on digital signatures).
- Operators may fail to forward the reply block after extracting their coin. Incentives against this behavior are proposed in the next section.
- The *spam attack* (consisting of resending a reply block many times) cannot be detected from coin expiration if handshaking is used on the reply path. Volunteers have therefore to store hash values for reply blocks that they already forwarded (together with NRR receipts), and refuse to forward again messages with the same hash value.

While users can never rely on courts to defend them since *they are too expensive* [23], proofs that are court-strong make a good foundation for a successful reputation system.

It may be worth reminding some weaknesses that were already present in earlier incentive mixnets, and which are not yet solved. First, an entity node can register several operators with false names. If a user happens to employ K consecutive operators of the same entity on a path, the operator can take K coins while needing less bandwidth than other correct volunteers. The weakness is that such a volunteer offers only the trustworthiness of a single volunteer for a higher cost and a false sense of security. Second, as for any type III mixnet supporting replies, an attacker may use courts to force agents to help in decrypting reply blocks. Agents need therefore to frequently let their secret keys expire and to discard them.

5.1 Coin encryption with decryption key published by destination

Another solution, related to handshaking, is to encrypt coins with a key, K_T, used only once. K_T is sent to the destination and published by it on a website:

$$M_N = E_N(K_T, m) \tag{11}$$

$$M_i = E_i(E_T(C_i), A_{i+1}, M_{i+1}); i \in (1, N) \tag{12}$$

No payment is made if the message is not delivered. If the coin obtained by a volunteer is not valid, the use of NRR and of a public key scheme for E_T can help volunteers to prove it in order to get help from the previous hops in identifying the sender.

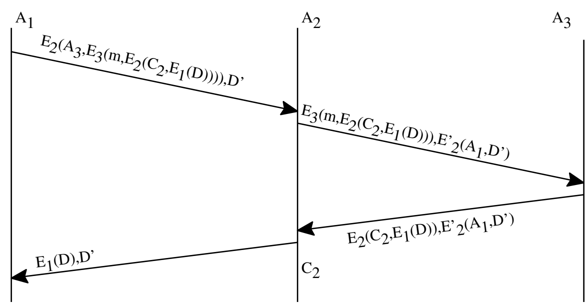

Fig. 4. Mixnet with reply-block built by volunteers.

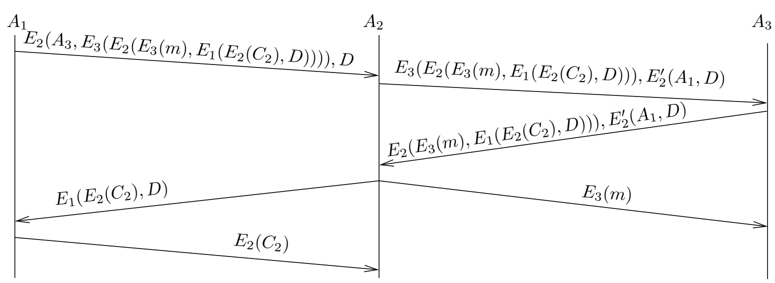

Fig. 5. Volunteer-based reply path construction for reply-pay with handshaking and volunteers trusted more than the destination. C stands for coin and D for dummy and m for message.

6 Other incentives

Volunteers not trusting the sender. A sender can format its reply path to bypass some of the volunteers on the forward path, avoiding to pay them. We propose a mechanism to guarantee that the reply path that will traverse the same set of volunteers. The idea is to have the reply path be constructed by volunteers themselves. This can be done by composing the message from two parts. One

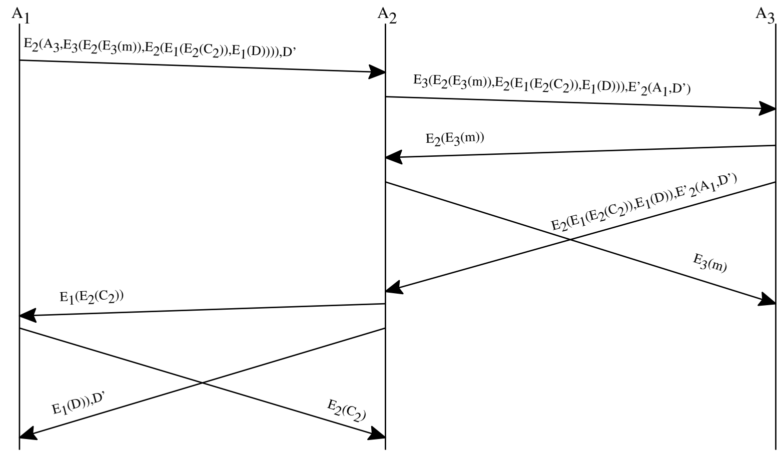

Fig. 6. Creating a non-trusting reply-path mixnet (based on the fair mechanism for simultaneous message exchange).

contains the payload from which each volunteer removes a layer of encryption. The second is a reply path, to which each volunteer adds the name of the previous hop with a new layer of encryption (to know to whom to send the reply on its return). Each volunteer A_i uses a separate symmetric cryptosystem E'_i for this purpose. The return path message is built according to the recursive Equations 13 to 14,

$$M'_1 = D' \tag{13}$$

$$M'_i = E'_i(A_{i-1}, M'_{i-1}); i \in (1, N) \tag{14}$$

where M'_i is the reply block built by volunteer A_i. A_1 uses a dummy D' value that can serve for hiding from A_2 the fact that A_1 is the sender. On the return path, each layer i of encryption on the message M'_N is removed by the agent creating it, A_i, for retrieving the identity of next hop, before forwarding the reply further.

Since the reply path is built by the volunteers, there is no need to include it with the coins prepared for the backward phase. Therefore Equations 7 and 8 can be replaced with Equations 15 and 16

$$R_i = E_i(E_{i+1}(C_i), R_{i-1}); i \in (1, N-1) \tag{15}$$

$$P = E_{N-1}(E_N(message), R_{N-2}) \tag{16}$$

which do not specify the next hop identity. An example with $N = 3$ and simple payment on reply is shown in Figure 4. An example of the obtained protocol with handshaking payment on reply is shown in Figure 5.

With this scheme, volunteers can include in M_i' a hash value H_i, of the message to be able to verify complains about badly formed messages that they can receive from other participants. This hash value also helps to avoid double processing of a return path (now composed of two parts), which would enable known spam attacks on anonymity. Whenever a reply block with a tag H_i is processed, the volunteer will mark the hash H_i stored during the forward process (tagged with the receipt of the NRR transmission). If a volunteer receives a reply block with a value H_i that is already marked, then he can infer that he has already cashed his coins for this message and should refuse to send it further.

Alternative mechanism for untrusted volunteers. Volunteers can prevent payment of each other either by nor forwarding the reply message (which was prevented by handshaking in the previous mechanism), or by not returning the coin for their predecessors. Each of these faults can be detected easily by neighbors on the path and can be proven to a reputation system. Additionally a handshaking mechanism can offer incentives against either one of the two failures.

If fear of reputation systems is not sufficient to trust volunteers to return coins to their predecessors on the return path, an alternative handshaking can provide them with an incentive. The idea is to not give them the message with their coin until they return the predecessor's coin. This method adds an inefficiency consisting of an additional round-trip delay on the reply path.

To build the corresponding messages, Equations 15 and 16 can be replaced with Equations 17 and 18

$$R_i = E_i(E_{i+1}(C_i)), E_i(R_{i-1}); i \in (1, N-1) \tag{17}$$

$$P = E_{N-1}(E_N(message)), E_{N-2}(R_{N-2}) \tag{18}$$

which encrypt separately the next coin from the rest of the coins. The enabled double handshaking is depicted in Figure 6.

Note that the 4^{th} and 5^{th} messages in Figure 6 can now be sent concurrently. Actually concurrent sending is needed in order to guarantee that the receipt with payments is delivered if and only if the message reaches the destination. Such concurrent transmission guarantees can be offered using the method in [14], based on incremental revelation.

7 Non-repudiation of message reception

As mentioned earlier, users and volunteers can improve the chance of fairness for operators, and timely service for users, by using reputation systems against attackers.

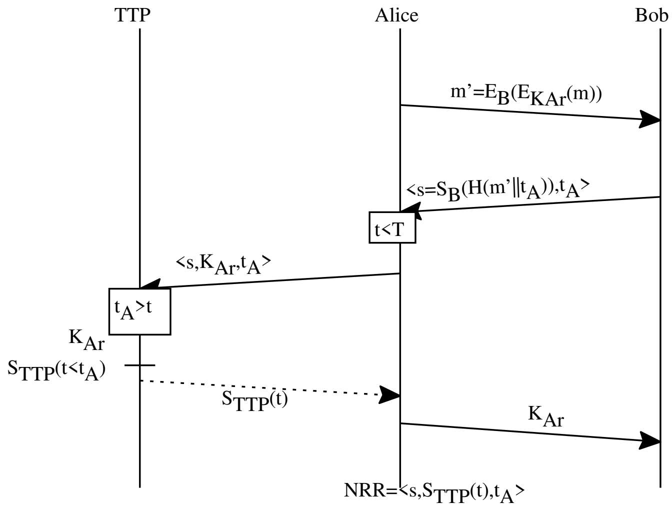

Fig. 7. A NRR protocol. Dashed arrow shows reading a public board.

For the reputation system to be efficient, reports submitted to such a system should be provable with court-strength. Such proof often require non-repudiation of reception of messages, as described with previously discussed attacks in Section 5.

One typically says that a protocol provides non-repudiation of receipt (NRR), if and only if it generates a non-repudiation of receipt evidence, destined to Alice, that can be presented to an adjudicator (the reputation system), who can unambiguously decide whether Bob received a given message or not. Various techniques for NRR are proposed in [34, 22, 26, 30, 9].

The protocols for achieving NRR are often based on a Trusted Third Party (TTP). It was shown possible to avoid the use of a TTP based on probabilistic methods. Let us now describe a version of Zhang and Shi's protocol [22] for the case where Alice needs a receipt for delivering a message m to Bob (see Figure 7). First Alice generates a new random session key K_{Ar} and encrypts the message with it, sending $m' = E_{K_{Ar}}(m)$ to Bob (using a secure channel, such as encryption with Bob's public key). Alice then waits for Bob to reply with a tuple $< s, t_A >$, where $s = S_B(H(m'||t_A))$, S_B is his digital signature algorithm, H an agreed hash function, and t_A is the deadline before which he wants to get K_{Ar}. Alice checks the delay and restarts the protocol if the deadline has already expired. Otherwise, Alice posts the tuple $< s, K_{Ar}, t_A >$ on a trusted public board (TTP) that registers the time t of the posting and where Bob can retrieve it. At fixed intervals of time the TTP also publishes the signed hash of their current content, the signature at time t being $S_{TTP}(t)$. The board publishes the tuple only if its next signature time $t \leq t_A$ otherwise announces Alice about the failure. Alice stores $S_{TTP}(t)$ as her NRR proof and sends K_{Ar} to Bob. If Bob did not receive the key before the deadline t_A then he looks for it on the public board. If the key is not on the public board then Bob stores $S_{TTP}(t_A)$ and discards the message m' (his proof that Alice did not comply). The reference time is made available by the TTP.

The small modifications to the original version of this protocol [22] consist of the fact that the hash of m' is computed by Bob instead of the TTP (making the task of the TTP lighter), and that the TTP generates signed hashes of their content for proof of innocence in the case of failures of the TTP. As in the case of previous techniques, an attack is possible if the TTP colludes with Bob, giving Bob K_{Ar} without posting it. One can further improve the reliability of the system by having several redundant TTP used in parallel to avoid this problem.

If Bob's computer is offline or if his agent does not reply timely, Alice can immediately report him to the reputation system together with the message she was passing to him. The reputation system can contact him on the behalf of Alice (passing his reply to Alice) or exhonerating Alice of her responsibility.

8 Conclusion

Mixnets offering payments for shuffling servers as an incentive can be applied to large problems, such as anonymous voting, petition signing over the Internet, and police tipping. Some of the proposed solutions guarantees that, if a mixnet fails to deliver the message, then the sender does not lose any money. Destinations have incentives to be honest since any failure is immediately detected by the senders (they being the only possible culprits for a payment without service). We believe that the guarantee offered by this technique is necessary for a viable mixnet service. Also, the return path for the receipt with payments is built by volunteer agents to guarantee that it will be the reverse of the forward path. Several incentives are discussed. We also discuss how a non-repudiation of receiving (NRR) mechanism can support court-strong proofs to be used by a reputation system for aleviating the remaining weaknesses of a mixnet.

References

1. Jean-Luc Abbet. Projet pilote neuchatêlois, 2004.
2. Inc Anonymizer. Privacy is your right. `http://www.anonymizer.com/`, 2007.
3. S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology — CRYPTO'93*, number 773 in LNCS, pages 302–318. Springer Verlag, 1994.

4. Stefan Brands. Off-line electronic cash based on secret-key certificates. In *LATIN'95*, pages 131–166, 1995.
5. Ernest F. Brickell, Peter Gemmell, and David W. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *SODA'95*, 1995.
6. Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient fair payment system. In *ACM'96*, pages 88–94, 1996.
7. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Com. of ACM*, 24(2):84–88, 1981.
8. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology — CRYPTO'88*, pages 319–327. Springer-Verlag, 1990.
9. Michael Conrad. Non-repudiation mechanisms for peer-to-peer networks. `doc.tm.uka.de/2006/conext2006.pdf`.
10. I. Damgard, J. Groth, and G. Salomonsen. *Secure Electronic Voting*, chapter The theory and Implementation of an electronic voting systems. Kluwer, 2002.
11. G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: design of type iii anonymous remailer protocol. In *IEEE Symposium on Security and Privacy*, pages 2–15, 2003.
12. Estonia. E-estonia. `http://www.vm.ee/estonia/kat_175/pea_175/2972.html`, 2005. `http://web-static.vm.ee/static/failid/286/E-Estonia_uus.pdf`.
13. Freedman and Morris. Tarzan: a peer-to-peer anonymizing network layer. In *ACM CCS'02*, 2002.
14. Juan Garay. Fair multi-party computation. In *Workshop on Secure Multiparty Protocols*, Amsterdam, October 2004.
15. Ian Goldberg. On the security of the tor authentication protocol. In *Privacy Enhancing Technologies*, 2006.
16. D. Goldschlag, M. Reed, and P. Syverson. Hiding routing information. In *Information Hiding*, number 1174 in LNCS, pages 137–150, 1996.
17. P. Golle and M. Jakobsson. Reusable anonymous return channels. In *WPES03*, 2003.
18. J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *PKC*, volume 2567 of *LNCS*, pages 145–160, 2003.
19. C. Gulcu and G Tsudik. Mixing e-mail with babel. In *IEEE Network and Distributed Security Symposium (NDSS)*, 1996.
20. M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX*, 2002.
21. David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner. A security analysis of the secure electronic registration and voting experiment (serve). New York Times (`http://www.servesecurityreport.org`), January 2004.
22. Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of non-repudiation protocols. *Computer Communications Journal*, 17(25):1606–1621, 2002.
23. Lawrence Lessig. *Free Culture*. Penguin Press HC, 2004.
24. A.; Macintosh, A.; Malina and S. Farrell. *Advances in Digital Government: Technology, Human Factors, and Policy*, chapter Digital Democracy through Electronic Petitioning, pages 137–148. Kluwer, 2002.
25. M. Merritt. *Cryptographic Protocols*. PhD thesis, Georgia Inst. of Tech., Feb 1983.
26. Refik Molva and Pietro Michiardi. Process for providing non repudiation of receipt (nrr) in an electronic environment. European Patent EP1300980, Patent Application EP20010480093, Sept 2003.
27. Neuchatel. Guichet unique. `www.GuichetUnique.ch`, 2006.
28. T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology — CRYPTO'91*, number 576 in LNCS, pages 324–337. Springer-Verlag, 1992.

29. S. Parekh. Prospects for remailers. *First Monday*, 1(2), August 1996. `http://www.firstmonday.dk/issues/issue2/remailers/`.
30. Dimitrios Petropoulos and Panayiotis Kotzanikolaou. Some more improvements on a fair non-repudiation protocol. *Journal of Internet Technology*, 4, 2004.
31. Reiter and Rubin. Crowds: Anonymity for web transactions. *ACM Trans. on Information and System Security*, 1(1):66–92, 1998.
32. D.R. Simon. Anonymous communication and anonymous cash. In *CRYPTO*, number 1109 in LNCS, pages 61–73, 1996.
33. S. Xu, W. Nelson Jr., and R. Sandhu. Enhancing anonymity via market competition. information assurance and security. In *IEEE ITCC'04*, 2004.
34. Jianying Zhou and Dieter Gollmann. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 55–61, Oakland, CA, 1996. IEEE Computer Society Press.