Trustable Relays for Anonymous Communication

Carlos Aguilar Melchor[*], **Yves Deswarte**[**]

[*] XLIM-DMI laboratory, 123 av. Albert Thomas, 87060 Limoges Cedex, FRANCE.

[**] LAAS-CNRS laboratory, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, FRANCE.

E-mail: carlos.aguilar@unilim.fr, yves.deswarte@laas.fr

Abstract. Current systems providing anonymous interactive communication [15, 22] are based on networks of anonymity-providing relays called MIXes. An important issue with such systems is that a MIX is able to betray its users, and thus it is necessary to use several MIXes sequentially for each communication, which distributes the trust among them. This increases the complexity of the protocols as well as the latency. On the other side, such distributed systems are resilient and scalable, and they provide good enough performance for web browsing.

An ideal relay should be unable to betray its users (we will say in this case that the relay is *trustable*). In such a setting, using multiple relays to distribute trust is not necessary, which simplifies design and reduces costs. Superposed sending provides an approach to construct trustable relays, the DC-net relays. Straightforward usage of classic protocols leads to other approaches and recently we proposed a set of trustable relays, based on Private Information Retrieval protocols, that provide new alternatives.

Independently of their practical applications, these relays are interesting from a theoretic point of view. In this paper, we present a survey that gathers the different trustable relays we have been able to identify and gives a unified view of their construction.

1 Introduction

In an IP network, a communication is composed of packets. Each packet has a set of headers and a content. When confidentiality is a concern, in particular with respect to eavesdropping, it is usually assumed that the eavesdropper is interested in the contents of the packets. However, sometimes, an eavesdropper is just interested in the packet headers (to learn who are the sender or the recipient for example) or in their presence (to detect an ongoing communication or changes in the amount of traffic on the network).

The existence of a communication, when it does begin or end, the users taking part in it, or the amount of information exchanged, is part of the *meta-data* defining a communication. Hiding the contents of a communication is easy to achieve through the encryption of each packet's content. Hiding the meta-data, on the other side, can be very difficult. End-to-end encryption cannot be used on the packet headers as they are needed by the intermediate nodes of the network for routing purposes. Moreover, in most of the networks over which IP is implemented an attacker eavesdropping on a communication link is able to observe the existence of all the transiting packets. The systems that try to hide the meta-data associated to a communication are called *anonymous communication systems* and the act of trying to discover this meta-data is called *traffic analysis*.

There exists an extensive literature on how to use multiple relays sequentially to obtain traffic analysis resistance for interactive communication. Some of the proposals are based on a usual client-server model [31, 23], and others are peer-to-peer [18, 34], but only two finalized implementations are currently widespread and operational: JonDo [22] (previously known as JAP, the Java Anon Proxy); and Tor [15], the second generation onion routing network.

Using a single trustable relay instead of multiple untrusted relays sequentially seems to be reasonable in some cases. Indeed, a single relay brings various advantages among which the most evident is simplicity. On the other side, note that if trustable relay systems are simpler than multiple relay systems, they are usually expensive from a communication and computation point of view. The trustable relays do not replace multiple relay systems, they are an alternative on some settings in which multiple relay systems fail. Trustable relays may be preferred for low-latency applications, small isolated networks, applications with a small group of users, etc.

Which are the different approaches to create a trustable relay for interactive anonymous communication is an interesting question not only for its practical applications, but also from a theoretical point of view. However, the literature on trustable relays is scarce and for a long time only one approach has been considered in scientific publications [12]. This technique, led in particular to an implementation and a very interesting performance study [26]. In [2] and [3] we introduced new trustable relays which we proposed to apply to Voice over IP in [4]. We do not discuss in this paper implementations or tests results of the relays we present. The aim of this paper is to survey the different techniques able to provide trustable relays and compare their relative advantages from a theoretical point of view. In order to do this we present representative trustable relays with a unified categorization. Apart from the inherent interest of a survey on such relays, the unified categorization has a useful side-effect: it highlights a new relay that has not been presented in the literature and outperforms some of the other relays, the *recursive pMIX*.

Section 2 is devoted to the basic notions and Section 3 to the primitives we will use in this paper. In Section 4 the setting under which the relays will be studied is given before presenting the basic, trusted third party, relay model in Section 5, and the trustable relays in Section 6. Finally, we comment globally the performance and unobservability results in Section 7, and conclude in Section 8.

2 Basic notions

We place ourselves in an (interactive) communication-oriented setting. A given user will be able to wait a given time in $O(\tau_s)$ to switch between two communications or to start a new one. After, this time an important amount of messages will be exchanged with another user with an expected round-trip time in $O(\tau)$ which will be significantly smaller than the switch time. We define the traffic analysis resistance properties directly over communications and not over messages as it is traditionally done for high latency anonymous communication systems.

2.1 Anonymity

Three attacker models will be considered in this paper: the global observer (who is able to eavesdrop in all the communication links simultaneously and to do precise timing attacks), the relay administrator,[1] and the attacker controlling a subset of the users. Of course an attacker can be a combination of these models too (for example an administrator controlling some users).

[1] This attacker can be in fact an official relay administrator or a hacker who has gained administrative rights. See Section 4 for further details.

Formal definitions of the anonymity properties that are desirable when communicating are given in [30]. In order to improve the readability of our paper we will use more informal definitions in this paper. Let S be a set of **communicating** users and denote by C a type of communications. Examples of types of communications can be the "internal communications" between two users connected to the anonymous communication service or the "external communications", between a user connected to the communication service and an external user. If an attacker $\mathcal{A}$ is unable to link the communications of type C to specific users in S, we say that this set is an *anonymity set* (with respect to $\mathcal{A}$ for communications of type C). A stronger property of traffic analysis resistance is unobservability. We say that a set of **communicating and non-communicating** users is an *unobservability set* (with respect to an attacker $\mathcal{A}$ and for communications of type C) if $\mathcal{A}$ can learn how many communications of type C are associated to this set, but is unable to know for any user of the set if he is communicating or not. If moreover $\mathcal{A}$ is unable to know if there is any communication of type C in the set or not we say that it is a *completely unobservable set*.

Note that n completely unobservable singletons (users) are equivalent to a completely unobservable set S of n users. We will thus sometimes just say that each user is completely unobservable, the notion of set disappearing. On the other hand, this is not true for unobservability sets. For example, having two unobservability sets S_1 and S_2 is not equivalent to have an unobservability set $S_1 \cup S_2$. In the former case an attacker can learn n_1, the number of communications associated to S_1, and n_2, the number of communications associated to S_2. In the latter case the attacker can only learn $n_1 + n_2$, the number of communications associated to $S_1 \cup S_2$.

It is important to note the differences between anonymity and unobservability sets. In order to illustrate them, let us present an example. Let $\{A, B, C, D, E\}$ be a set of five users using an anonymous communication relay. A and B are communicating together and C is communicating with an external user F which is not connected to the anonymous communication service. D and E are connected to the service but are not communicating.

If a relay hides from an attacker $\mathcal{A}$ who is communicating with whom but not whether a user is communicating or not, $\mathcal{A}$ will be able to notice that A, B, and C are communicating, D and E are not, and that one of the former users is communicating with an external user F. However, $\mathcal{A}$ is not able to decide who among A, B and C is communicating with F nor which are the two users communicating together. The set $\{A, B, C\}$ is an anonymity set with respect to $\mathcal{A}$ for both internal and external communications. If the relay also hides from $\mathcal{A}$ who (among the users connected to it) is communicating but not the total number of internal or external communications, the attacker will just be able to learn that one user among A, B, C, D and E, is communicating with F and that there is one internal communication. He will not be able to learn who are the two users having the internal communication or who are the users actually communicating. The set $\{A, B, C, D, E\}$ is an unobservability set with respect to $\mathcal{A}$ for internal and external communications. Finally, if the relay also hides from $\mathcal{A}$ the number of internal communications, the attacker will just be able to learn that one of the users is having a communication with F but not whether there are any ongoing internal communications in the set or not. The set $\{A, B, C, D, E\}$ is a completely unobservable set with respect to $\mathcal{A}$ for internal communications, and just an unobservability set for external communications.

The anonymity sets provide a fair amount of traffic analysis resistance as long as global observers are not considered. Indeed, using the previous example, suppose that an attacker can observe simultaneously A, B, C, D and E. The attacker can see that three users $\{A, B, C\}$ are communicating and two $\{D, E\}$ are not. When the communication between A and B is over he is able to observe that the set of communicating users will be reduced to $\{C\}$ and therefore he concludes that there has been an internal communication between A and B. Similarly if D and E begin a communication the attacker is able to observe that the set of communicating users is increased suddenly from $\{A, B, C\}$ to $\{A, B, C, D, E\}$ and concludes that D and E have probably started a communication. More generally, if a global observer is able to know whether the users communicate or not, he can identify when they start and stop communicating and correlate these data to learn who is communicating with whom. As we just deal with single relay systems it is specially important to resist to global observers. We will therefore just consider primitives providing unobservability properties.

In the case of unidirectional communications, unobservability can be related just to the sent streams or to the received streams. In an interactive setting, as communications are bidirectional, detecting that a user is receiving or sending messages is equivalent as either he is doing both, or none. Even if this distinction between sending and receiving properties is unnecessary in interactive communication, the techniques used to obtain each of them are different: some primitives provide sender unobservability, while others provide recipient unobservability. Of course, both sender and recipient unobservability are necessary to obtain communication unobservability in an interactive setting. In Section 3 we present the different primitives that can be used to obtain sender and recipient unobservability.

Finally, besides external attackers and the anonymity properties that can be obtained against them we will also consider whether the initiator and responder of a communication are able to learn who is their interlocutor or not. In all the relays we present, either both can remain anonymous or none of them. We will therefore introduce an anonymity property linked to this issue, *initiator/responder anonymity*.

2.2 Cryptography

The different primitives used to provide anonymous properties in this paper are often based on cryptography. We provide here the basic definitions of the techniques used and their security properties.

Definition 1 (Public Key Encryption Scheme)**.** A public key encryption scheme $PKE = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ consists of three PPT algorithms:

- The key generation algorithm $\mathcal{KG}$ which takes as input a positive integer k (the security parameter), and outputs a pair (pk, sk) of public and secret keys.
- The encryption algorithm which takes as input a public key and a plaintext message from the plaintext space $\mathcal{M}$ and outputs a ciphertext on the ciphertext space $\mathcal{C}$.
- The decryption algorithm which takes as input a secret key and a ciphertext and outputs a plaintext message.

A common secret key encryption scheme follows the same definition except that $pk = sk$ is a secret key. We say that a public key or secret key encryption scheme is randomized if $\mathcal{E}$ is probabilistic.

The security property that is usually required for randomized encryption schemes is *indistinguishability*, which ensures that an attacker is unable to distinguish between the numerous[2] encryptions of any two messages. More formally, we say that a function is negligible in k if it approaches zero (as $k \to \infty$) faster than any inverse polynomial, and that an attacker is computationally bounded if its time complexity is polynomial in k. Consider the following game,

[2] Indeed, in a randomized public key cryptosystem, to each cleartext corresponds a huge set of different ciphertexts.

Protocol 1 Indistinguishability challenge.

1. The challenger generates a key pair (pk, sk) for a security parameter k, and gives pk to the attacker.
2. The attacker may perform a polynomial number of encryptions or other operations.
3. Eventually, the attacker submits two distinct chosen plaintexts m_0, m_1 to the challenger.
4. The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, and sends the challenge ciphertext $c \longleftarrow \mathcal{E}[pk, m_b]$ back to the attacker.
5. The attacker can perform a polynomial number of additional computations or encryptions. Finally, it outputs a guess for b.

The public key encryption scheme has the indistinguishability property if any computationally bounded attacker can only obtain a negligible advantage in k (with respect to guessing) in this game.

Depending on the attacker capacities two levels of indistinguishability are usually defined. *IND-CPA*, or indistinguishability against *chosen plaintext attacks* supposes that the attacker has access to the encryption function, as in the challenge above. If moreover the attacker has access to a decryption oracle in steps 2 and 5 which can be called on any ciphertext but c we talk about *IND-CCA2* security, or indistinguishability against *adaptive chosen ciphertext attacks*.

A second notion of security that is important in our work is the one of *indistinguishability of keys*. The game to be considered in this case is

Protocol 2 Key Indistinguishability challenge.

1. The challenger generates two key pairs (pk_0, sk_0) (pk_1, sk_1) for a security parameter k, and gives pk_0, pk_1 to the attacker.
2. The attacker may perform a polynomial number of encryptions or other operations.
3. Eventually, the attacker submits a plaintext m to the challenger.
4. The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, and sends the challenge ciphertext $c \longleftarrow \mathcal{E}[pk_b, m]$ back to the attacker.
5. The attacker can perform a polynomial number of additional computations or encryptions. Finally, it outputs a guess for b.

Again, we say that the scheme has the property of *indistinguishability of keys against chosen plaintext attacks* (or *IK-CPA*) if any computationally bounded attacker can only obtain a negligible advantage in k (with respect to guessing) in this game. If moreover, the attacker has access to a decryption oracle in steps 2 and 5 which can be called on any ciphertext but c for both key pairs we talk about *indistinguishability of keys against adaptive chosen ciphertext attacks* (or *IK-CCA2*).

El Gamal [17] is an encryption scheme that achieves both IND-CPA and IK-CPA. Abadi and Rogaway in [1] define *type-1 security* for symmetric encryption schemes as the equivalent of having IND-CPA and IK-CPA for public key encryption schemes and prove that AES in CBC and CRT modes is type-1 secure under the canonical assumption that AES is a good pseudo-random permutation.

3 Primitives

Different primitives are usable in our context: superposed sending (Section 3.1) allows to be unobservable while sending, encrypted padding (Section 3.2) while sending or receiving, and broadcast with implicit addresses (Section 3.3) and private information retrieval (Section 3.4) allow to be unobservable while receiving.

3.1 Superposed Sending

Superposed sending as an anonymous communication tool was first introduced by David Chaum in 1985 [11]. The basic idea was presented through an allegory of three cryptographers on a dinner wishing to know if one of them had paid the dinner, without knowing whom (i.e., they wanted to be able to say ”I have paid” with sender unobservability). Computer networks that implement the resulting protocol are called dining cryptographers networks (DC-nets); the protocol itself is named the DC-net protocol.

Most of the literature on superposed sending is dedicated to the detection of disrupters [40, 39, 7, 38, 41, 21] (i.e., system users that disrupt other users' communications), except a few publications [16, 36] which study how to implement better superposed sending protocols, focusing in key distribution and the use of multiple-level hierarchical topologies.

3.1.1 Description

In the DC-net protocol we distinguish two sorts of users: the *senders* are the users that actually want to send a message, and the *non-senders* which participate on the round but do not want to send anything. Each user (sender or non-sender) must participate in each DC-net round by the emission of a scrambled message. The scrambled messages can be sent to a single user that we will call the *aggregator*; or broadcast, in which case each user in the network is an aggregator.

A DC-net protocol is characterized by the period τ between each round and the length ℓ of the emitted messages. The protocol described below shows how a round from a (τ, ℓ) DC-net protocol is performed.

Protocol 3 ℓ-bit DC-net Round.

Let $U_1, ..., U_n$ be a set of users such that each couple of users (U_i, U_j) shares a unique ℓ bit long secret, $S_{i,j} = -S_{j,i}$.
Each user U_i encodes a message *Message(i)* as a string of ℓ zeroed bits if he is a non-sender, or as an ℓ-bit message if he is a sender.

Round progress:
(1) Every user U_i sends *Emission(i) = Scrambling(i) + Message(i)* with:

$$\textit{Scrambling(i)} = \sum_{j=1, j\neq i}^{n} S_{i,j}$$

to the aggregator.

(2) The aggregator computes the result:

$$\textit{Result} = \sum_{i=1}^{n} \textit{Emission(i)}.$$

Since every $S_{i,j}$ appears twice (once added by U_i and once subtracted by U_j), the scramblings cancel each other and the aggregator obtains *Result* = $\sum_{i=1}^{n}$ *Message(i)*.

Each couple, of users (U_i, U_j) is thus supposed to share an ℓ bit long secret, the $S_{i,j}$, which must be renewed each round. They can be randomly generated and stored in large storage devices (DVDs, for instance), that are exchanged physically by the users, or be pseudo-randomly generated from a secret seed known only to users U_i and U_j. In the former case the security of the protocol is unconditional, in the latter, it can be reduced to the one of the pseudo-random generator used [40].

3.1.2 Collisions

In a given round, if only one user has attempted to transmit, the result of the round is his message (as all the other messages are composed of zeroed bits). If more than one user has attempted to transmit, there is a *collision*. The easiest way to deal with collisions is to wait for a random number of rounds before trying to transmit again, but there are more efficient alternatives. In particular, superposed receiving and channel reservation provide simple solutions to this issue.

Superposed receiving [29] (see [39] for a reference in English) allows solving a collision of s messages in s rounds. Superposed receiving has a very small communication and computation overhead, and is therefore a much better solution than waiting for a random number of rounds if collisions are frequent.

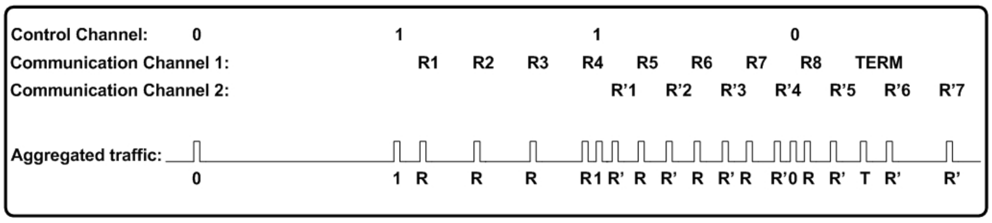
Control Channel: 0 1 1 0
Communication Channel 1: R1 R2 R3 R4 R5 R6 R7 R8 TERM
Communication Channel 2: R'1 R'2 R'3 R'4 R'5 R'6 R'7
Aggregated traffic:
0 1 R R R R1 R' R R' R R' R R'0 R R' T R' R'

Figure 1: Channel reservation.

In [3] we proposed the use of multiple independent superposed sending channels when dealing with low-latency and communication-oriented systems, and called this approach channel reservation. This technique is based on three parameters τ_s, τ, ℓ and is described below.

Protocol 4 Channel Reservation Protocol.

Let $U_1, ..., U_n$ be a set of users and (τ_s, τ, ℓ) the protocol parameters.

Control channel:

```
Every $\tau_s$
(1) Users        : Participate to a $\lceil \log n \rceil$-bit DC-net round sending 1 if requesting a channel creation or 0 if not.
(2) Aggregator   : Send the result of the round to all the users.
(3) Users        : If the result of the round is
      0  : Wait for the next control channel round.
      1  : Start a new communication channel.
    $> 1$ : Solve the collision with a collision resolution protocol.
```

Communication channel:

```
Every $\tau$
(1) Initiator    : Participate to an $\ell$-bit DC-net round acting as a sender when desired.
    Other Users  : Participate to an $\ell$-bit DC-net round acting as non-senders.
(2) Aggregator   : If the result of the round is "TERM" broadcast it to the users. Else, process it as required.
(3) Users        : On reception of a "TERM" message stop participating to the communication channel.
```

In this approach there is a *control channel* in which a superposed sending round is executed every τ_s. When a user wants to begin a communication, he first sends a message using the superposed sending protocol through the control channel (see the example given in Figure 1). This message results in the creation of a *communication channel*, an independent DC-net protocol with parameters

(τ, ℓ). In this channel only transmits the user who has requested its creation, and therefore no collision occurs. When he finishes his transmission, he sends through the communication channel an agreed message for channel termination and users stop the rounds associated to it. Collisions may occur in the control channel, but as channel creation requests are generally much scarcer than message sending, the number of collisions to be resolved is greatly reduced.

Note that if an attacker can relate even a single message of a communication channel to a user, all the other messages will automatically be also related to the same user as the communication channel is known to be linked to a single user (its creator). This is not true if the superposed sending rounds are run independently without any control channel. On the other hand, in this work anonymity is related to communications, not messages, and thus this loss is acceptable specially considering the performance improvement in collision handling.

3.1.3 Unobservability properties

The effective key graph of a DC-net protocol against an attacker $\mathcal{A}$ is the graph $G = (V, E)$, the set of vertices V being the users participating to the round not controlled by the attacker and the set of edges E being defined by the couples of users sharing secrets unknown to $\mathcal{A}$.

If this graph is not partitioned, the set V forms a sender unobservability set against $\mathcal{A}$. If it is partitioned, each subset of users associated to each partition forms a different sender unobservability set against $\mathcal{A}$. For example, if the graph is partitioned in two, $\mathcal{A}$ will be able to determine how many messages are sent from each partition instead of from the whole set of users. Note that in the protocol we have described each couple of users shares a secret, which results in a complete key graph. The effective key graph against a given attacker will therefore never be partitioned and V will always be a sender unobservability set, independently of the number of controlled users. Indeed, each non-controlled user (*i.e.* each element of V) has an edge to each other non-controlled user and thus the only way for a non-controlled used to be an isolated vertex is if the attacker controls all the other users. In such a case the graph is a singleton (and the sender unobservability set too).

3.2 Encrypted padding

3.2.1 Description

A second approach to send messages unobservably is to use *encrypted padding*. Encrypted padding was first suggested in computer networks in [27] and consists in the introduction of dummy encrypted messages to lower the capacity of the side channel resulting from the analysis of packet sizes and timings. Different statistical approaches can be used to generate the dummy encrypted messages. In order to limit the number of variants for the relays we present, we will just focus on full encrypted padding, letting the reader evaluate other padding techniques by himself.

If a user `A` wants to set a full encrypted padding link between her and `B` she sends him a fixed-size encrypted message every τ seconds. Each of these messages' associated cleartext is garbage as long as `A` has no information to send to `B`. When `A` wants to send a message to `B` she encapsulates it inside the encrypted padding messages replacing the garbage with the information to be sent. `B` decrypts all the messages received from `A`. As long as the resulting cleartexts are garbage `B` dumps them. When the resulting cleartext contains useful information (which can be revealed by a given format or marker), he reads it.

The basic idea is that, if we consider that it is not possible to distinguish between the encrypted messages containing garbage and the encrypted messages actually containing information for `B`, an attacker always sees the same thing: a constant rate flow of fixed-size encrypted messages between `A` and `B`.

3.2.2 Unobservability properties

A randomized public key encryption scheme with IND-CPA security ensures that an attacker, even with knowledge of the public key used to encrypt the padding channel cannot distinguish between the encryptions of any two plaintexts. Thus if such an encryption scheme is used an attacker cannot distinguish encrypted garbage from encrypted messages, and `A` is a completely unobservable sender. Of course, `A`'s unobservability cannot hold against `B`, as he can decrypt the messages. Similarly, if the attacker is not able to decide whether `B` receives encrypted padding or not, `B` is a completely unobservable recipient (except with respect to `A`).

Note that this situation is different in multi-relay systems with encrypted padding in which the different relays used iteratively decrypt data and forward the decrypted data over unsecured channels. In such cases, the relays can be used as decryption oracles and thus the encryption schemes used must ensure IND-CCA2 security. In the primitive we present in this section there is only one encrypted link, and the decrypted data is never accessible to an attacker (not controlling `A` or `B`) and thus IND-CPA security is enough.

When using encrypted padding together with other primitives (as we do in the relays we present in this paper), the encryption scheme should have the IK-CPA property in order to avoid leakage about the recipient's key. As noted in Section 2.2 El Gamal is a public key encryption scheme that has both IND-CPA and IK-CPA security, and AES in CBC or CTR modes is a symmetric encryption scheme that guarantees equivalent properties (type-1 security). Both schemes are thus usable for encrypted padding.

3.3 Broadcast with implicit addresses

3.3.1 Description

Sending messages that everybody receives (or can receive) such that only the real recipient is able to decrypt them is a classical mean to ensure recipient unobservability. For example, coded messages were broadcast by radio during World War II to the resistance. Of course, nobody but the recipients could say which radio listeners were able to decrypt the messages and which not and therefore recipients were unobservable. On a computer network, broadcast allows sending a message to all the addresses of a given network or sub-network. Its usage is however constraining as the communication links of all users are encumbered, and even if we can broadcast in Wide Area Networks it is not possible to do it at large scale (for example over the whole Internet).

When users receive a broadcast message they must be able to distinguish whether they are the intended recipient or not. In order to avoid providing information to the attackers that may be used to break unobservability, implicit addresses are used. Different approaches exist [32], but the easiest (as in World War II radio broadcasts) is for each user to try to decrypt the message and conclude, depending on whether he is able to decrypt it to a meaningful cleartext or not.

3.3.2 Unobservability properties

As for encrypted padding, the encryption scheme used for broadcast with implicit addresses should have IND-CPA and IK-CPA security (or type-1 security if symmetric). IND-CPA ensures that it is not possible to distinguish whether a given message contains data or garbage, and IK-CPA ensures that an attacker is unable to distinguish the ciphertexts of any two different recipients.

With these two properties the recipients are thus completely unobservable against any of the considered attackers.

3.4 Private Information Retrieval

Private Information Retrieval (PIR) is a field of research dissociated from anonymous communication even if a few connections between these two fields have been done in the past. The first link between PIR and communication systems was done by [14] to provide a message service with location privacy for mobile users. In [2] and [3] we proposed to use PIR protocols as a much more communication-efficient alternative to broadcast with implicit addresses which will be presented in this section. Finally, in [35] the usage of PIR over a distributed database was proposed to retrieve mail anonymously.

In this section, first we provide a description of the PIR research field and then we describe how to use it for anonymous communication.

3.4.1 Description

Usually, to retrieve an element from a database, a user sends a request pointing out which element he wants to obtain, and the database sends back the requested element. Which element a user is interested in may be an information he would like to keep secret, even from the database administrators. For example, the database may be :

- an electronic library, and which books we read may provide information about our politic or religious beliefs, or details about our personality we may want to keep confidential,
- stock exchange share prices, and the clients may be investors reluctant to divulge which share they are interested in,
- a pharmaceutical database, and some client laboratories wish that nobody may learn which are the active principles they want to use,

To protect his privacy, a user accessing a database may therefore want to retrieve an element without revealing which element he is interested in. A trivial solution is for the user to download the entire database and retrieve locally the element he wants to obtain. This is usually unacceptable if the database is too large (for example, an electronic library), quickly obsolete (for example, stock exchange share prices), or confidential (for example, a pharmaceutical database).

Private Information Retrieval schemes aim to provide the same confidentiality to the user (with regard to the choice of the retrieved element) than downloading the entire database, with sub-linear communication cost. PIR was introduced by Chor, Goldreich, Kushilevitz, and Sudan in 1995 [13]. In their paper, they proposed a set of schemes to implement PIR through replicated databases, which provide users with information-theoretic security as long as some of the database replicas do not collude against the users.

In this paper, we focus on PIR schemes that do not need the database to be replicated, which are usually called single-database PIR schemes. Users' privacy in these schemes is ensured only against computationally-bounded attackers. It is in fact proved that there exists no information-theoretically secure single-database PIR scheme with sub-linear communication cost [13]. The first single-database PIR scheme was presented in 1997 by Kushilevitz and Ostrovsky, and since then improved schemes have been proposed by different authors [37, 8, 25, 9, 24, 20, 5].

Generally, in a PIR protocol a user generates a query (using a randomized algorithm) for the element he wants to retrieve and sends it to the database. The database mixes the PIR query to the database elements (using a deterministic algorithm) and obtains a result which is sent back to the user. The user is able to recover the element he wanted to retrieve out of the PIR reply. User privacy is ensured as obtaining any information about which element was retrieved from the PIR query or the PIR reply implies breaking a well-known cryptosystem (such as Pailler's IND-CPA encryption scheme [28]). Current PIR schemes are very efficient from a communication point of view. In [20], Gentry and Ramzan proposed a scheme in which the user sends a fixed size query (2048 bits for current security standards) and obtains the database element he is interested in with a database reply expansion factor of 4, independently of the number of elements contained on the database. The computational cost of user operations (query generation and reply decoding) is constant whereas the computational cost for the server is linear in the number of database elements. We will use this scheme as a reference for communication and computational complexity.

3.4.2 Usage with an anonymous communication relay

A pretty important fact about PIR schemes is that in all of the existing protocols the request is independent of the database contents. The same request can therefore be used to generate different replies as the database evolves. Besides, a relay providing an anonymous communication service receives the different communication streams sent by the users (whether they use encrypted padding or superposed sending to obtain sender unobservability). This relay can be seen as a database, with a slot for each of these streams, that evolves very quickly. If the relay generates a PIR reply every time he updates the stream slots, the user doing the PIR query retrieves a communication stream without the database being able to know which stream the user has selected. In figure 2 a user has sent a query to retrieve the contents of the second slot of the relay (which is receiving three streams). Each time the relay updates the slots she generates a PIR reply. The user decodes the replies sent by the relay recovering all the messages of the second stream.

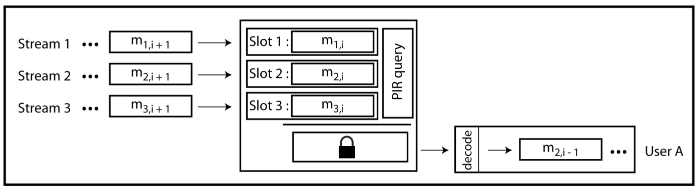

Figure 2: Private stream retrieval.

One major issue is the fact that a given user does not know, a priori, whether one of the incoming communications is intended to him or not, nor which. The users must obtain this information somehow while remaining untraceable. This issue will be dealt with in Section 4.1.3 as the techniques used are interesting (for different reasons) for other primitives too. Until then, we just assume that the users are aware of when they receive a communication and which is its index among all the streams the relay deals with.

3.4.3 Unobservability properties

If a set of users follows the approach described in the previous section they form a recipient anonymity set even with respect to the relay administrator, as he is able to know if a user is receiving a communication or not but unable to learn which communication corresponds to which user. The reason why the relay administrator is able to know if a user is receiving a communication is that he is aware of which users send PIR queries and receive a reply stream, and which not. To obtain recipient complete unobservability, a user must therefore follow either a full padding or a superposed sending approach to send its PIR queries. The full padding approach is described below. On the other hand, the superposed sending approach is just presented in Section 6.3 as its relevance is highlighted better then.

If the users follow a full padding approach, each of them sends to the relay a PIR query every τ_s seconds. When a user wants to receive a communication he sends PIR queries for the slot corresponding to the communication he is interested in, and when he has nothing to receive he sends queries for a random slot and drops the PIR replies received. As the relay is unable to learn anything about the users' choices from the PIR queries or from the PIR replies he generates, each user is a completely unobservable recipient. The value of τ_s is pretty important as it defines the maximum time a user may have to wait when trying to switch from one communication to another. Imagine a user $\mathtt{A}$ learns (no matter how) that he is going to receive a communication with index 2 but he has just sent a fake query for another index. $\mathtt{A}$ has to wait for τ_s seconds before sending another query and being able to receive the communication. If τ_s is too large the delay may be unacceptable. On the other hand, the smaller is τ_s the larger the overhead induced by the fake PIR queries is, and therefore a trade-off must be found for each application.

4 Setting

In this paper we describe relays that provide anonymous communication to their users. We do not deal with sets of users that communicate among them and, through a distributed protocol, obtain some anonymous communication property. Such approaches exist [33, 12] but are only considered in this paper when usable as primitives to build trustable relays. In this section we will first describe the generic model of the presented relays, and then the security setting.

4.1 Relays

For each relay that we present, many variants are possible, each of them possibly having different performance, security properties etc. We do not aim to present thoroughly all the possible alternatives, but just a credible representative of each family of relays which can illustrate an approach more than a given optimal relay (which in any case would depend much on the application aimed).

4.1.1 Global view

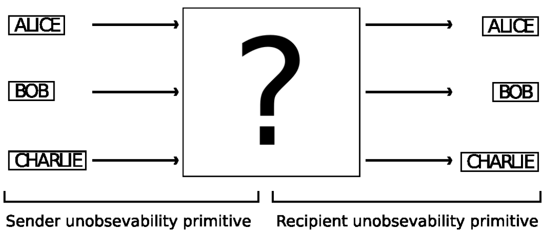
ALICE
BOB
CHARLIE
?
ALICE
BOB
CHARLIE
Sender unobsevability primitive
Recipient unobsevability primitive

Figure 3: Trustable relay.

In a relay-based communication, the users are connected to the relay in a client-server model and use different privacy-enhancing techniques to communicate through it. As noted in Section 2, users must have both sender and recipient unobservability to resist to traffic analysis attacks over interactive communications. Therefore, each user will send a communication flow to the relay using a sender unobservability primitive, and receive another communication flow from the relay with a recipient unobservability primitive (see Figure 3). The relay is thus considered to be a physical server which in a real world model will have one or more administrators. The administrators are supposed to install the required software in the server and act in case of malfunction. They may be curious and even do some active attacks withing the setting defined in Section 4.2.2. The basic idea to obtain a trustable relay is that the primitives used for communication are such that no usable information is sent (for a polynomial-time attacker) independently of the actions of the relay.

4.1.2 Communication links

Beyond the black-box model derived from the client-server relationship between users and the relay, we can also infer some properties of the communication links themselves. More specifically, a lower bound can be given for the amount of traffic needed to obtain the different degrees of unobservability. Indeed, when an attacker eavesdrops on a communication link, he can observe the traffic on that link. This implies that in order to have an unobservability property, the probability distribution of the traffic parameters (load, message sizes, timings) on a user's link must be similar (in terms of distinguishability) whether the user is communicating or not.

For some primitives it will be possible to generate just as much traffic as in a given communication in which case we will say that the expansion factor of the primitive is 1. In other cases this expansion factor may be proportional to n, the number of users, or m the number of communications. Thus, to a given relay will be associated two expansion factors, one derived from the sender unobservability primitive and one derived from the recipient unobservability primitive.

4.1.3 Signaling rounds

For most of the relays that we present, having special rounds in which the users send signaling information to the other users is either necessary, or improves greatly the performance. For example, if broadcast is used for recipient unobservability, users unable to know the moment in which communications begin have to decrypt all the messages from all the communications. On the other hand if they know when a communication does begin (for example, because there is a signaling round and communications can only start after a signaling round), they just have to decrypt the messages broadcast once every few seconds, lowering considerably the computational cost. For PIR protocols, it is not just better, but absolutely compulsory to have signaling rounds, as a user must know if one of the incoming streams is associated to him and which, in order to select it with his PIR queries.

The basic idea of signaling rounds is for every user to publish whether he wants to establish a call or not, and with whom. Depending on the information published and on the unobservability primitives used to publish it, different levels of anonymity are obtained. Which sort of signaling rounds are done depends on the primitives used for the communication flow and therefore on the relay chosen. In any case, there are two ways to publish the signaling information, corresponding to the two primitives providing sender unobservability: encrypted padding and superposed sending. In an encrypted padding signaling round, every user sends a message to the relay that will be either encrypted garbage, or an encrypted message for a given recipient. The relay broadcasts all the messages, and the users try to decrypt every broadcast message. On success, a user deducts he is going to receive a call and possibly retrieves an index indicating which of the communication flows corresponds to the incoming call. In a superposed sending signaling round, the users execute one or more rounds (to solve collisions) and obtain a set of messages with sender unobservability. The advantage of this approach is the reduction on the number of messages published. The drawbacks are the ones associated to superposed sending: a more complex and less robust protocol, and a leakage on the number of new communications.

When deciding the periodicity τ_s of signaling rounds, there is a trade-off between performance and waiting time to establish a communication, as for the query sending periodicity when using PIR protocols. If signaling rounds are too sparse the delay to start a communication may be unacceptable. If the delay between each signaling round is too small, there is an important communication overhead.

4.2 Security

4.2.1 Unobservability properties and stream secrecy

Two different levels of unobservability can be reached in our setting against the external attackers (observer, administrator, controller). A user may be in an unobservability set (usually the set of users connected to the trustable relay) or completely unobservable.

It is well known that the composition of two secure primitives does not lead necessarily to a secure system. In this case it is also true, composing a primitive providing sender unobservability with another one providing recipient unobservability is not enough.

Sender unobservability ensures that the output of a user will not indicate that he is sending information or not and recipient unobservability ensures that his input will not reveal whether he is receiving information or not. When combining the primitives, we must verify that there is a step in the process that ensures unlinkability between the senders output and the recipients input. This will be noted for each relay.

A second question is whether the relay provides initiator/responder anonymity or not. Using pseudonym lists the relay administrators can always provide this property to the users, but the real question will be if this property can resist to a user-administrator collusion.

If a user receives a given stream which he can link to a communication, he can inform the administrator. If the administrator is able to link this stream to a sender, initiator/responder anonymity will not resist to such a collusion. Indeed, in some cases the administrator will be able to link a sender's stream to an intermediate stream and the recipient the intermediate stream to himself. Combining both associations it will be possible to link a sender to a recipient and break initiator/responder anonymity. Whether this is possible or not will be noted for each server.

The relays will provide thus anonymity properties to their users but, on the other hand, a new security issue can sometimes be raised. Indeed, in a normal relay, the administrators are trusted to correctly switch the different communication channels. In trustable relays the administrators are unable to learn how the communications are switched (or they would learn who is communicating with whom) and thus cannot ensure that users are not getting communication streams that they were not supposed to get.

Depending on the primitive used some users may inconspicuously get streams that they should not have accessed: sometimes they will be able to get one, and sometimes all of them. In the relays we present these streams are encrypted, but this may still be unacceptable in some settings. A simple example is the one of a common user that stores unauthorized streams hoping that one day the encryption used will be broken and the information obtained still valuable.

Thus, whether other users' streams[3] can be obtained inconspicuously or not and how easily it can be done will be taken into consideration for each relay.

4.2.2 Active attacks

The attackers, specially the relay administrator, can modify the streams and do any sort of active attack trying to obtain some information from the reactions of the users.

The unobservability properties of the presented primitives remain unchanged even in the presence of such attacks. Indeed, modifying a stream of encrypted padding gives no information to the attacker of whether the stream contained useful information or not. The same is true about superposed

[3] Note that stream secrecy only concerns users, a global observer or relay administrator always has the possibility to store the encrypted streams.

sending, in which modification or delay introduction will just result in disruption of the protocol, but will not provide the attacker with information about who was sending. Again, with broadcast modifying what some recipients receive does not provide any information to the attacker. What can leak information is the reaction of the users to such an attack. If a user stops executing his part of the protocol or reacts in any way to a disrupted communication he may reveal he was communicating.

There is therefore a simple way to avoid any leakage: each user must always follow the same protocols sending messages at the same pace and ignore any error on the incoming streams.

Trustable relays can only handle small groups of users which usually are static and in a closed environment. Most of the active attacks are conspicuous or can be revealed through the usage of traps (see for example [40]). In such an environment, an action (eventually legal, or just an eradication) can be undertaken against such detected attackers. Active attacks should thus be scarce enough not to represent an issue from a disruption point of view and ignoring errors is therefore an acceptable defense against such attacks. Of course if such errors are frequent whistle-blowing is possible through an anonymous protocol (for example superposed sending), and proof can be given to the users that there has been an active attack.

5 Trusted third party relays

The first relay we will present is the trust based MIX or *tMIX*. As its name suggests this relay is based on user trust and thus does not correspond to the definition we have given (in the abstract) of a trustable relay. It is presented as a reference in order to answer to the question: what is the price to pay to resist to relay administrators and hackers controlling the relay?

The tMIX is based on encrypted padding channels both from the users to the relay (to ensure sender unobservability against eavesdroppers) and from the relay to the users (to ensure recipient unobservability). An alternative would have been to use superposed sending to ensure sender unobservability, but it leads to worse performance results and as we do not aim to do a survey of all trust-based relays we only describe the tMIX.

5.1 Relay description

When a user gets connected to the tMIX, he sets an upload encrypted padding channel. The tMIX sets a download encrypted padding channel towards the user. Both the tMIX and the user decrypt all the data they receive, and dump the result as long as it is recognized as garbage.

When a user `A` (or a user with pseudonym `A`) wants to have a communication with another user `B` (or a user with pseudonym `B`) he replaces the garbage packets on his upload encrypted padding channel by the information he wants to send to `B`. When the tMIX decrypts the contents of the upload padding channel of `A` and detects a communication towards user `B`, he forwards it to user `B`'s download channel by replacing the garbage on the encrypted padding channel by the information received from `A`. When `B` decrypts the contents of the download padding channel and recognizes an incoming communication, he replies to `A` following the same protocol.

Chaum proposed the idea of using mixing relays (that he called *mixes*) to obtain untraceable mail in 1981 [10]. The first encrypted padding based mixes were proposed by Pfitzmann et al. in [31] and were called ISDN-MIXes.[4] These relays were intended to be used in low-latency systems (for telephony) but as in Chaum's work they were supposed to be used iteratively in order to distribute trust among the different relays.

A simple signaling technique can be used in this relay. In parallel to the upload encrypted padding channel used for the communication a second encrypted padding channel can be created with a larger

[4] Which justifies the uppercase notation we have chosen despite the fact that Chaum's original mixes were noted lowercase.

period τ_s for signaling purposes. A user only starts sending information through the communication channel once he has alerted the relay through the signaling channel that a communication is going to be initiated. This allows the relay to decrypt the incoming communication channels only when needed.

5.2 Unobservability and stream secrecy

Users connected to the tMIX have no unobservability at all against the relay administrators or against a hacker with administrative privileges on the relay. On the other hand encrypted padding channels ensure that each user is a completely unobservable sender and recipient against global observers and attackers controlling a subset of the users (if the encryption scheme is IND-CPA). Re-encryption by the relay ensures that the input and output streams of the users are unlinkable. The users form therefore a completely unobservable set against any attacker except the relay administrators.

Property	**Unobservability**		**Initiator/Responder Anonymity**		**Stream Secrecy**
Attacker	**Administrator**	**Other Attackers**	**Administrator+User**	**User**	**User**
tMIX	None	Complete	No	Yes	Yes

The tMIX can hide from the initiator/responder the identity of its communication partner if pseudonyms are used. On the other hand, a user colluding with the relay administrators will break initiator/responder anonymity as administrators know who is communicating with whom.

In a tMIX a user can just get one stream at a time, and there is no way for him to store a communication in which he does not participate, as the tMIX just forwards communications to the intended recipients. Stream secrecy is thus preserved. Note again that stream secrecy is just considered against users as global observers and relay administrators can always store all the encrypted communications, whatever the relay is.

5.3 Trusted hardware and tMIX relays

The most evident way to obtain a tMIX is to trust its administrators not to betray the users and to consider (or hope!) that the attackers are unable to intrude in it. Indeed, as well the administrators as an intruder are able to collect all the traffic analysis information in this single point of failure.

Another approach is for the relay to use a trusted hardware device inside of which the tMIX is implemented, so that the relay administrators or an attacker intruding on the relay are unable to obtain more information than observing the communication links.

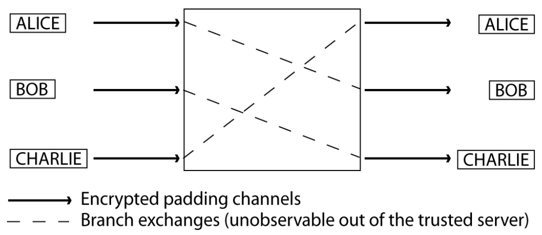
ALICE
BOB
CHARLIE
ALICE
BOB
CHARLIE
Encrypted padding channels
Branch exchanges (unobservable out of the trusted server)

Figure 4: tMIX schematic view.

The trust is not in this case placed on the relay administrators and on the relay resistance to intrusions, but on the trusted hardware device. This trust consists in mainly two assumptions. First, that the administrators are unable to tamper with the device to obtain traffic analysis information. Second, that the software installed on the trusted device is secure and does not contain any backdoor. The first assumption is inherent to the usage of a trusted hardware device. The second is a more complex issue. Indeed, the main question is *who installs the tMIX software in the trusted device ?*

When using a trusted hardware device, instead of placing the users trust on the administrators of the tMIX it is placed on the entity who installed the tMIX software in the trusted device. There is thus not such a big difference between the two situations except for one important point. The software installation can be done off-line, in a controlled environment and it may be supervised or certified.

It is however important to keep in mind that in a tMIX we cannot avoid to place some trust in an entity. This can be the tMIX administrators or the entity providing a programmed trusted device. By no means the usage of trusted hardware ensures a security based uniquely on its tamper-proof capabilities.

5.4 Performance

Every user needs two encrypted padding channels for communication and one for signalling. Communication channels result in a communication cost in $O(B)$, B being the bandwidth available for a communication. The signalling channel results in a communication cost in $O(\tau_s^{-1})$. Relay computational cost is split in two: signalling channels induce a cost in $O(\tau_s^{-1} \times n)$, and active communication channels a cost in $O(B \times m)$, n being the number of connected users and m the number of simultaneous communications. User computational cost is scarcely an issue and can be easily computed from the figures we give. We therefore don't include it on our performance results.

Relay	**Bandwidth Usage (User → Relay | Relay → User)**		**Relay Computational Cost**
	Communication	**Signalling**	
tMIX	$O(B)\vert O(B)$	$O(\tau_s^{-1})\vert 0$	$O(mB) + O(n\tau_s^{-1})$

Figure 5: tMIX performance overview.

Of course, there is an expansion factor on both the upload and download channels inherent to the size of the headers and to the randomized encryption of the padding channels in which the communication is encapsulated. Similarly the exact complexity of the computational operations leads to implementation dependent constants and to hard to compare results. We do not make explicit such constants and choose big O notation to present performance results in this paper.

6 Trustable relays

We have identified two primitives able to provide sender unobservability: encrypted padding and superposed sending. Similarly, we dispose of three primitives able to provide recipient unobservability: encrypted padding, broadcast with implicit addresses, and private information retrieval. As seen in the previous section, using encrypted padding for recipient unobservability implies trusting the relay administrators and thus we only use the other two primitives for trustable relays. Combining sender and recipient unobservability primitives we would obtain theoretically four different relays. However, as we will see in the following sections, some of these combinations can lead to multiple communication relays.

In any case, we classify the relays by the primitive used for recipient unobservability: in Section 6.1 we deal with the broadcast based relays, in Section 6.2 we introduce the relays that result from the usage of Private Information Retrieval protocols. Section 6.3 is dedicated to alternatives to the previous relays.

6.1 Broadcast-based relays

6.1.1 bMIX relays

The *bMIX* (for *broadcast based MIX*) is a variant of the tMIX which, as its name suggests, uses broadcast to ensure recipient unobservability. Although Chaum cited the possibility of using encrypted padding and broadcast on mixes[10], the first interactive relays based on these primitives were the ISDN-MIXes proposed by Pfitzman et al. [31]. More precisely, ISDN-MIXes being used iteratively, intermediate ISDN-MIXes were tMIX relays and the last ISDN-MIX was a bMIX relay.

Description In a bMIX, encrypted padding channels are still used for sender unobservability (both for the communication and the signaling streams), but the padding channels are not encrypted with a key shared with the relay. Instead, they are encrypted with a key shared with the intended recipient or with a random key if there is no intended recipient. The relay broadcasts all the users encrypted padding channels.

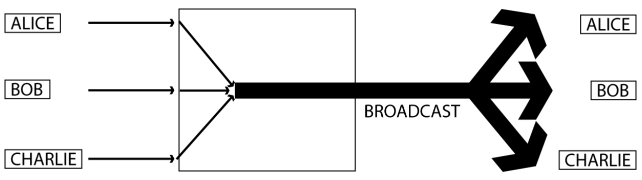

Figure 6: bMIX schematic view.

Given the implicit addressing technique chosen in Section 3.3, a set of public keys is supposed to be known to all users. Each key pk_X is associated to a given user X (or to a pseudonym $\mathtt{X}$ in order to provide initiator/responder anonymity).[5] When a user $\mathtt{A}$ (or user with pseudonym $\mathtt{A}$) gets connected to the bMIX he creates two upload cover traffic channels towards the relay with a random encryption key. One of them is used for communication and the other for signaling (with a single packet every τ_s seconds). For each user X, $\mathtt{A}$ tries to decrypt the packets from X's signaling channel (broadcast by the pMIX) with key sk_A. On failure she drops the result.

If $\mathtt{A}$ wants to communicate with another user $\mathtt{B}$ (or with pseudonym $\mathtt{B}$) she just sends a notification through the signaling channel encrypted with pk_B. Just after this notification, she starts sending the communication data through the communication channel. When $\mathtt{B}$ tries to decrypt the broadcast signaling channels he discovers $\mathtt{A}$'s notification and starts decrypting $\mathtt{A}$'s communication channel with sk_B which results in the incoming communication. He replaces then the garbage of his communication channel with his reply and encrypts it with pk_A.

Each user sends one communication and one signaling upload encrypted padding channels with respective communication costs in $O(B)$ and $O(\tau_s^{-1})$, and receives n encrypted padding channels of

[5] Symmetric cryptography can be also used if each couple of users (U_1, U_2) shares a secret key $K_{U_1U_2} = K_{U_2U_1}$. This key may have been exchanged through an off-line secure channel or obtained with a key exchange protocol with resistance to man-in-the-middle attacks (see for example [6]).

each type. The computational cost for the relay is roughly null as it only forwards the communication streams. [6]

Unobservability and Stream Secrecy The upload encrypted padding channels allow the users to form a completely unobservable sender set. Reciprocally, the broadcast with implicit addressing allow the users to form a completely unobservable receiver set (if the encryption scheme used is IND-CPA). Broadcast with implicit addressing also ensures that the recipients cannot be linked to the senders (if the encryption scheme used is IK-CPA). The users form thus a completely unobservable set.

On the other hand, initiator/responder anonymity does not resist to administrator-user collusions. Indeed, the relay administrators can link an encrypted padding stream to its emitter and the recipient can reveal which encrypted padding stream he is able to decrypt. Thus if they collude the sender-recipient relationship can be revealed.

Stream secrecy is not ensured either. Indeed, a user can retrieve directly all the encrypted communication streams from all the users. This server should be therefore only used if this is tolerable, for example because the information exchanged is quickly obsolete or the encryption used considered strong enough.

6.1.2 DC-net relays

The DC-net relay (the name references the dining cryptographers paradigm which was presented by Chaum in [12] when introducing superposed sending as an anonymity providing protocol), is similar to the bMIX except that encrypted padding is replaced with superposed sending for sender unobservability. Such relays were implemented and deeply studied by David Michael Martin in his PhD dissertation [26].

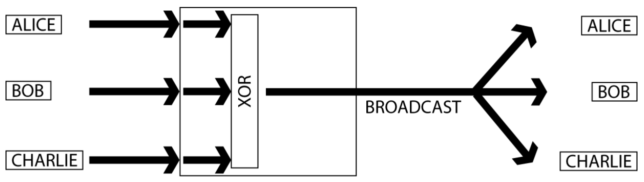

Figure 7: DC-net relay schematic view.

Description Suppose that at a given time there is m internal communications in a set of users (which implies that $2m$ users are communicating). Using the channel reservation technique presented in 3.1, each time a user initiates a communication he creates a superposed sending channel which is used simultaneously by him and his interlocutor. Indeed, as users can transmit in a channel anonymously, if `A` and `B` are having a communication together they can transmit in the same communication channel their respective messages M_A and M_B. Upon reception of the channel's messages, `A` and `B` can obtain each other's message from the channel's output $M_A + M_B$ by subtracting their own transmission. For example, `A` obtains $(M_A + M_B) - M_A = M_B$. Thus, with the usage of superposed sending, the channels can be duplex and only m channels are needed to engage m bidirectional communications.

[6] We do not consider the computational complexity of receiving and sending the data, in order to simplify the comparison between the different servers.

Signaling rounds are implicit to the usage of superposed sending with a control channel, as this channel holds inherently signaling information. The results of this channel are retrieved by the relay and this one informs the users when it is appropriate that another communication channel should be started or stopped. By adding an encrypted message to the channel creation requests, the initiator of a communication can easily inform his interlocutor that he is the intended recipient.

Thus, in this relay, a channel creation request in the control channel may be composed of a tag indicating that a new communication channel is requested and an encrypted message for the intended recipient. When the result of a control channel round is a channel creation request, the DC-net relay broadcasts the associated encrypted message to all the users. The users just have to try to decrypt these messages and upon success they start communicating through the created channel.

The DC-net relay has to broadcast m communication channels where a bMIX had to broadcast n. This represents a drastic reduction as usually the number of simultaneous communications m in a group is much smaller than the size of the group n. On the other side the DC-net relay receives $m \times n$ streams whereas the bMIX only received n. The relay must XOR all the incoming streams which has a computational cost in $O(mnB)$.

Unobservability and Stream Secrecy In opposition to the bMIX relay setting, superposed sending only allows the users to form an unobservability sender set and thus the number of active communications m can be learned through eavesdropping. Broadcast with implicit addressing ensures recipient unobservability. Unlinkability of recipients and senders is in fact ensured twice, superposed sending ensures unlinkability between the users and the resulting streams (after the XOR), and broadcast with implicit addressing ensures unlinkability between these streams and the recipients (if the encryption scheme used ensures IK-CPA). The users form therefore an unobservability set against the considered attackers, but not a completely unobservable set.

Initiator/Responder anonymity is ensured even against an administrator-user collusion. Indeed, even if a user reveals to which broadcast stream he is linked, superposed sending ensures that the administrator is unable to link this stream to a sender.

Finally, as for the bMIX, all the communication channels are broadcast, and therefore any user can store the encrypted communication streams inconspicuously. Stream secrecy is thus not ensured.

It is possible for the administrators of the relay to hide m as well from eavesdroppers as from the users. Indeed, to do this, the users can maintain a constant number of channels m_{max} active. When the number of used channels is smaller than this value, the unused channels result in streams of zeroes and they are replaced by the relay with encrypted padding before broadcast. With such a technique, eavesdroppers and users always see the same number of superposed sending channels on the relay input, and of broadcast channels on the relay output.[7]

Of course, this approach is simple but more expensive from a communication point of view than the basic one. The upload and download expansion factor is m_{max} for each user. On the other hand the users are completely unobservable against all the attackers except the relay administrators, against which they form an unobservability set. Users may learn the value of m by trying to start new communications. Thus, if the relay refuses to initiate a communication, the user can conclude that $m = m_{max}$. However, even if this allows to punctually learn the value of m it is an active attack and therefore conspicuous. Of course, trying to follow the evolution of m is even more conspicuous and countermeasures can thus easily be introduced (see [7]).

[7] In order to avoid that a global observer verifies which streams XOR to a communication and which to zero, the streams should be encrypted between the users and the DC-net relay.

6.2 PIR-based relays

Without a trusted relay, the alternative to broadcast with implicit addresses is the usage of Private Information Retrieval protocols. As noted in Section 3.4, the streams that the communication relay gets from the primitive used to ensure sender unobservability can be seen as a rapidly evolving database. The user can therefore send PIR queries to recover one of these streams. If encrypted padding is used for sender unobservability we call such a relay a pMIX (for *PIR based MIX*). If superposed sending is used we call such a relay a pDC-net relay (for *PIR based DC-net relay*).

The authors introduced pMIX relays, proposed setup and communication establishment protocols, and studied their performance in [2]. On the other hand, pDC-net relays have not been presented formally in previous work. The possibility to build such a relay was noted by the authors in [3], but it was not presented as the *apMIX* (which will be described in the next section) seems more interesting from a practical point of view with comparable computational costs and better communication costs. However, from a theoretical point of view this relay is interesting and thus we present it in this paper side by side with the other relays.

6.2.1 pMIX relays

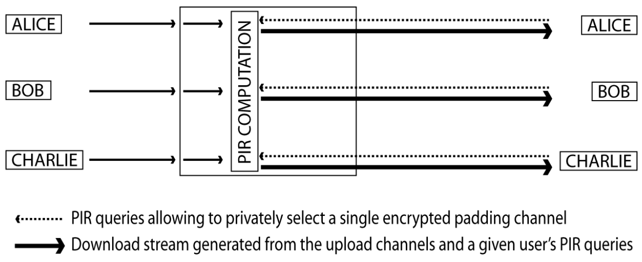
ALICE
BOB
CHARLIE
PIR COMPUTATION
ALICE
BOB
CHARLIE
PIR queries allowing to privately select a single encrypted padding channel
Download stream generated from the upload channels and a given user's PIR queries

Figure 8: pMIX schematic view.

Description A pMIX is basically a bMIX that, instead of broadcasting all the user communication-oriented encrypted padding channels, defines them as a set of streams among which the users privately choose one by sending PIR queries (signaling-oriented encrypted padding channels are unchanged). Thus, upon connection, a user sets two encrypted padding channel towards the pMIX, one for communication and one for signaling, with a random encryption key. Every τ_s seconds, he also sends a PIR query for a random stream among n. The pMIX generates a download channel by using the user's PIR queries over the set of the n communication-oriented encrypted padding channels he receives and broadcasts the signaling encrypted padding channels. The authors introduced pMIX relays, proposed setup and communication establishment protocols, and studied their performance in [2].

A and B initiate a communication as in a bMIX using the signaling channels. If a user A is having a communication with another user B he chooses B's stream with his PIR queries. If a user is not communicating he randomly chooses a stream and sends PIR queries for it.

Each user generates two encrypted padding channels and a PIR query stream, and receives a PIR reply stream and the broadcast signaling channels. If Gentry and Ramzan's scheme is used (as suggested in Section 3.4) the PIR query stream throughput is in $O(\tau_s^{-1})$ and the reply stream has a constant expansion factor. Thus user upload and download communication streams are in $O(B)$. If we include the PIR queries in signaling costs the total bandwidth usage for signaling is in $O(\tau_s^{-1})$ on upload and $O(n\tau_s^{-1})$ on download. The communication costs are thus almost the same as with the tMIX. On the other hand, generating a PIR reply stream has a cost with current PIR protocols in

$O(nB)$ and thus the computational cost for the relay is in $O(n^2B)$ in order to generate all the PIR reply streams, which is much worse than for all the previous relays.

Unobservability and Stream Secrecy PIR ensures unlinkability between the encrypted padding streams and the PIR replies. Sender and recipient complete unobservability are ensured by encrypted padding and PIR. The users of a pMIX form therefore a completely unobservable set with respect to any of the considered attackers.

Initiator/Responder anonymity cannot be ensured against an administrator-user collusion as a recipient can link himself to a given encrypted padding stream and the relay administrator can link the encrypted padding stream to a sender.

Stream secrecy is ensured in two ways. First, the users can only retrieve one stream through the PIR protocol, and not all of them as in broadcast based servers. Second, in a pMIX there are n streams to choose from and only $2m$ of them contain actual unidirectional communications. The user must choose one of these streams and as usually $m \ll n$ most of the time the stream is just encrypted garbage. Moreover, as the streams are unidirectional, if he chooses right he just is able to store half of the communication. Note that on the other hand, a user could always store the contents of a given slot, thus choosing a user to spy on. This can be avoided by hiding the user-slot associations and changing them frequently, for example every τ_s seconds.

Thus a pMIX provides very good stream security as most of the time nothing is stored, and the remaining time only a unidirectional encrypted stream is stored, and the stream obtained changes every τ_s seconds.

6.2.2 pDC-net relays

Description In this variant, the users act as with a DC-net relay with channel reservation. At each round, the users send to the relay the scrambled messages for every active bidirectional channel, and the relay gathers the results of the superposed sending round for every channel, to obtain m communication slots. Every τ_s seconds each user also sends a PIR query for an m element database.

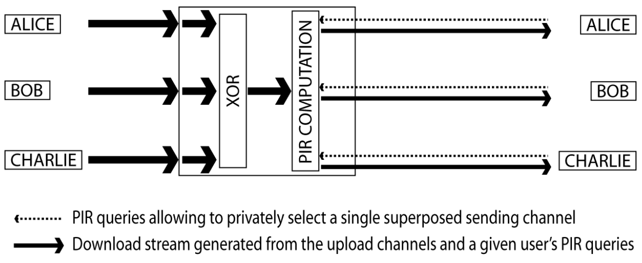
ALICE
BOB
CHARLIE
XOR
PIR COMPUTATION
ALICE
BOB
CHARLIE
PIR queries allowing to privately select a single superposed sending channel
Download stream generated from the upload channels and a given user's PIR queries

Figure 9: pDC-net relay schematic view.

As in the case of the pMIX, the relay generates PIR reply streams. However, the PIR replies are computed over the m communication streams resulting from the superposed sending protocol, instead of computed over the data sent by the users.

Each user uploads m communication streams with a cost in $O(mB)$, a control channel stream which contains signaling information and a PIR query stream in $O(1/\tau_s)$ as for the pMIX, and downloads a PIR reply stream. As PIR replies are computed over an m slot database, the computational cost for the relay is in $O(mnB)$.

Unobservability and Stream Secrecy The usage of PIR ensures that the users are completely unobservable recipients against any of the considered attackers and that the streams resulting from the superposed sending are unlinkable to the recipients. On the other hand, as superposed sending is used for sender unobservability the users will just form a sender unobservability set. Of course, superposed sending also ensures unlinkability between the users and the streams resulting from the protocol. The relay administrators can use the technique proposed in Section 6.1.2 in order to hide the number of active communications. This technique does not improve unobservability against relay administrators but ensures that the users will be completely unobservable against other attackers.

Initiator/Responder anonymity is ensured even against an administrator-user collusion (as for the DC-net relay) as the administrators cannot link the senders to the streams resulting from the superposed sending protocol.

Stream secrecy is not as good as in the pMIX as an eavesdropper is always sure to store a bidirectional communication for the basic server, and with high probability if the administrators include padding channels on the output. On the other side it is better than for the bMIX as only one communication can be stored and not all of them.

6.3 Tweaking PIR-based relays

At a given time, many of the users of a pMIX may not be communicating and therefore most of the PIR replies may not be read by the clients. To reduce the computational cost, the pMIX should only compute PIR replies for the users that are actually communicating. However, if only the communicating users were sending PIR requests, the pMIX administrator would know which users would be communicating and which would not. We propose in this section two relays solving this problem, the apMIX (for *anonymous PIR queries based MIX*), and the rpMIX (for *recursive pMIX*).

The authors introduced apMIX relays in [3] as an alternative to pMIX relays and their important computational costs. On the other hand, rpMIX relays are presented here for the first time.

6.3.1 apMIX relays

Description To prevent the relay administrators to learn who is sending PIR requests and who is not, the users can send their PIR requests through a superposed sending protocol, which allows them to form an unobservability set with respect to the publication of their queries. If at a given time, the users are having m bidirectional communications, $2m$ users will send a PIR query. To treat collisions, superposed receiving can be used and $2m$ rounds of the protocol result in the $2m$ PIR queries generated by the communicating users. The relay is unable to link the requests to specific users, and therefore it is not possible for the apMIX to know who is interested by which reply. The relay will thus broadcast the replies to all the users.

Signaling information can also be sent by adding one superposed sending round to the PIR query generation phase instead of using encrypted padding as in the pMIX. This reduces the communication cost of signaling as there is less information to broadcast.

The apMIX allows to reduce the computational cost for the relay from $O(n^2B)$ to $O(mnB) + O(mn\tau_s^{-1})$ (computing n PIR reply streams over an m element database and XORing the superposed sending streams). On the other hand, it increases the users' download communication cost factor from $O(B)$ to $O(mB)$, and the upload signaling cost from $O(\tau_s^{-1})$ to $O(m\tau_s^{-1})$ (again by including PIR query emission in signaling costs).

Unobservability and Stream Secrecy Encrypted padding ensures that senders are completely unobservable. PIR and superposed sending ensure that the recipients form an unobservability set as the number of PIR queries reveals the number of communications. PIR also ensures that the

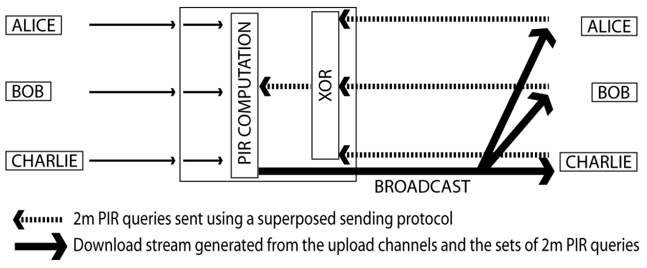
ALICE
BOB
CHARLIE
PIR COMPUTATION
XOR
ALICE
BOB
CHARLIE
BROADCAST
2m PIR queries sent using a superposed sending protocol
Download stream generated from the upload channels and the sets of 2m PIR queries

Figure 10: apMIX schematic view.

encrypted padding streams and the broadcast streams are unlinkable, and broadcast ensures that the streams and the recipients are unlinkable too. The users form therefore an unobservability set with respect to any attacker. If moreover the approach described in Section 6.1.2 is used for the PIR query publication, they also form a completely unobservable set with respect to any attacker except the relay administrators.

Initiator/Responder anonymity is not ensured against an administrator-user collusion as a user can link himself to an encrypted padding stream (for which he has sent a PIR query through the superposed sending protocol), and the administrators can link the encrypted padding stream to a sender.

In this relay, the communication streams are broadcast to all the users, but the unintended recipients cannot access directly the encrypted communication streams as they are encapsulated in PIR reply streams. Thus, not just the cryptosystem used to encrypt the communications must be broken but also the public key cryptosystem used for the PIR protocol. The problem for the attacker is that he does not even have the public key associated to this cryptosystem, as this key is sent with the PIR query and only the server has access to it.

The PIR protocols we propose to use in this paper are based on RSA-like cryptosystems, and if it is probable that the public keys used today will be factorizable in the upcoming decades this will not suffice to break the protocol without the public key. Informally, the idea is that a few ciphertexts (as the PIR query and thus the public key changes every few seconds) will never provide significative information on what is the modulus the server is computing with, and therefore even if an attacker is able to factor big numbers, he will be unable to decrypt the PIR queries.

Thus, stream secrecy is not as good as in a pMIX as the server broadcasts the information, but it is better than with a bMIX or a DC-net server as the encrypted streams are not directly accessible to the attacker.

6.3.2 rpMIX relays

Description The idea behind the rpMIX is that the output of an apMIX is a set of $2m$ streams, and therefore the users can do PIR queries to retrieve one of the resulting streams as if these $2m$ streams were the input of a pMIX. Thus the n initial streams are reduced to $2m$ intermediate streams through the PIR queries sent anonymously by the users (just as for the apMIX), and these $2m$ streams are reduced to a single stream for each user through usual PIR queries (see Figure 11).

Every τ_s, each user participates to the superposed sending rounds resulting in the $2m$ PIR queries and the signaling information, and sends one PIR query directly to the relay. When a user receives a notification of an incoming call and an index of the associated stream through a signaling round, he sends a PIR query for this channel through the superposed sending protocol, and a direct PIR query to retrieve the desired stream out of the $2m$ streams generated. When a user is not communicating, he just participates to the superposed sending rounds without sending any PIR query through this

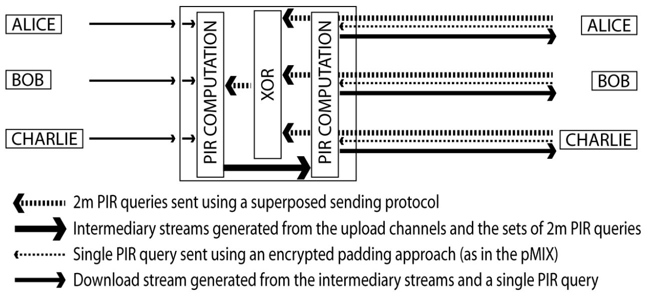
ALICE
BOB
CHARLIE
PIR COMPUTATION
XOR
PIR COMPUTATION
ALICE
BOB
CHARLIE
2m PIR queries sent using a superposed sending protocol
Intermediary streams generated from the upload channels and the sets of 2m PIR queries
Single PIR query sent using an encrypted padding approach (as in the pMIX)
Download stream generated from the intermediary streams and a single PIR query

Figure 11: rpMIX schematic view.

protocol, sends directly a PIR query for a random slot among $2m$, and drops the packets from the reply stream he gets from the rpMIX.

The communication cost on upload download for the communication stream is in $O(B)$ and for the signaling in $O(m\tau_s^{-1})$ on upload and in $O(\tau_s^{-1})$ on download. The relay computational cost is in $O(mnB)$ for computing the PIR replies and in $O(mn\tau_s^{-1})$ for XORing the signaling and PIR query streams of the superposed sending protocol.

Unobservability and Stream Secrecy The users form an unobservability set with respect to any attacker, using the same arguments as with the apMIX. Again, as in the apMIX, an approach such as the one presented in Section 3.1.3 can be used to form completely unobservable sets with respect to any attacker except the server administrators.

Initiator/Responder anonymity is not ensured against an administrator-user collusion, as for the apMIX, as users know which encrypted padding stream they are obtaining.

Stream secrecy with this server is, as for the pMIX, very good. Again, the user can get at most one stream as results are not broadcast but retrieved as a PIR reply stream. A user can retrieve inconspicuously one of the intermediate streams generated from another user's PIR query. As for the apMIX the stream will be encrypted with an unknown public key. Of course, the difference with the apMIX is that only one PIR reply stream can be obtained, and not all the streams.

7 Global view

7.1 Unobservability and Stream Secrecy

All the proposed relays allow the users to form a completely unobservable set against global observers and attacker controlling users, and initiator/responder anonymity against users not colluding with the administrators. The main difference is the unobservability degree provided against the relay administrators and the initiator/responder anonymity against administrator-user collusions. Against these attackers the tMIX relay provides no anonymity at all; the DC-net, pDC-net, apMIX and rpMIX relays allow to form an unobservability set; and only the bMIX and pMIX allow to be completely unobservable. Initiator/Recipient anonymity against administrator-user collusions is only ensured in the relays that use superposed sending for sender unobservability, which are the DC-net and pDC-net relays.

In Figure 12 we present the unobservability degree and initiator/responder anonymity provided by the relays together with the stream secrecy properties: the number of streams that can be simultaneously stored by a curious user and the additional protection layers beyond encryption of the streams.

Relay	**Unobservability**	**Initiator/Responder**	**Stream Secrecy**	
		Anonymity	**Accessible Streams**	**Protection**
bMIX	COMPLETE	NO	ALL	NONE
pMIX	COMPLETE	NO	m/n	NONE
DC-net	SET	YES	ALL	NONE
apMIX	SET	NO	ALL	UNKNOWN PK
pDC-net	SET	YES	1	NONE
rpMIX	SET	NO	$1/2$	UNKNOWN PK
tMIX	NONE	NO	NONE	N/A

Figure 12: Anonymity and security overview.

Of course, all the streams can be stored simultaneously by a curious user if the relay is based on broadcast of the communication streams (bMIX, DC-net, apMIX relays). If the communication streams are retrieved by PIR a curious user can retrieve at most one encrypted communication. In the case of the rpMIX we note $1/2$ as the communication is just unidirectional, and in the case of the pMIX m/n as an unidirectional communication is retrieved only with probability $2m/n$ (the rest of the time only encrypted garbage is retrieved). Finally, the retrieved streams are always encrypted, but in the case of the apMIX and rpMIX relays there is an additional layer of protection as the stream has a second layer of encryption (with a short-lived unknown public key, noted UNKNOWN PK in Figure 12).

7.2 Performance

Relay	**Bandwidth Usage (User → Relay | Relay → User)**		**Relay Computational Cost**
	Communication	**Signalling**	
bMIX	$O(B)\|\boldsymbol{O(nB)}$	$O(\tau_s^{-1})\|\boldsymbol{O(n\tau_s^{-1})}$	$\simeq 0$
pMIX	$O(B)\|O(B)$	$O(\tau_s^{-1})\|\boldsymbol{O(n\tau_s^{-1})}$	$O(n^2B)$
DC-net server	$O(mB)\|\boldsymbol{O(mB)}$	$O(\tau_s^{-1})\|0$	$O(mnB) + O(n\tau_s^{-1})$
apMIX	$O(B)\|\boldsymbol{O(mB)}$	$O(m\tau_s^{-1})\|\boldsymbol{O(\tau_s^{-1})}$	$O(mnB) + O(mn\tau_s^{-1})$
pDC-net	$O(mB)\|O(B)$	$O(\tau_s^{-1})\|\boldsymbol{O(\tau_s^{-1})}$	$O(mnB) + O(n\tau_s^{-1})$
rpMIX	$O(B)\|O(B)$	$O(m\tau_s^{-1})\|\boldsymbol{O(\tau_s^{-1})}$	$O(mnB) + O(mn\tau_s^{-1})$
tMIX	$O(B)\|O(B)$	$O(\tau_s^{-1})\|0$	$O(mB) + O(n\tau_s^{-1})$

Figure 13: Asymptotic performance overview.

Figure 13 compares the asymptotic performance of the different trustable relays to the one of the tMIX. The first two columns concern communication performance and the last computational costs.

Communication performance is split in two. The first column deals with the communication traffic, and the second one with signaling traffic. These figures are in boldface when they represent broadcasts.

The first remark is that the two relays that provide complete unobservability against relay administrators are very expensive from the communication or computational point of view. On the other hand, we can note that none has both drawbacks, the bMIX relay (resp. the pMIX relay), which is very inefficient from a communication (resp. computational) point of view is optimal from a computational (resp. communication) point of view.

Among the relays that provide set unobservability against relay administrators the communication performance is better as we go downwards on the figure. However, even if the theoretical figures seem to show similar computational performance for the four relays it must be noted that the number of bit operations done in the first relay (XOR-based) is several orders of magnitude lower than on the other three (PIR-based). Thus, we have a computationally efficient relay and three communication efficient and computationally more expensive variants.

Finally, the tMIX is both efficient from a communication and a computational point of view but provides no unobservability against the relay administrators.

8 Conclusion

This survey highlights the diversity of options that are available to build a low-latency anonymous communication relay. Such relays can provide a simple solution for small sets of users with high traffic analysis resistance, even if the relay is compromised.

The approaches providing complete unobservability even against relay administrators remain however expensive either from a computational or from a communication point of view. On the other hand, it is possible to obtain set unobservability with more reasonable costs while remaining completely unobservable against attackers other than the relay administrators.

Computational costs are an issue for some of these relays, specially those based on PIR protocols. However for small groups real world applications seem possible [4]. Moreover, fast PIR protocols have been proposed recently [19, 5]. Used on the presented relays, these PIR protocols would reduce by several orders of magnitude the cost per user for the relay processing power. However, these protocols are not mature enough and, in particular, PIR queries are too large for them to be considered on these relays. We hope this recent research will lead to more optimized protocols.

Research in anonymous communications has been during a few years focused on multi-relay protocols, and the idea of using a single relay with strong privacy-enhancing properties in some applications has not been much explored. In this paper we have just considered these relays theoretically, we hope this work will motivate real implementations and tests as well as research on practical applications.

9 Acknowledgments

This work has been partially supported by the European project PRIME (Privacy and Identity Management for Europe, Integrated Project IST-507591) `http://www.prime-project.eu` and the European Network of Excellence ReSIST (Resilience for Survivability in IST), funded by the European Commission Framework Program 7, Contract Number 026764 `http://www2.laas.fr/RESIST/index.html`

We would like to thank the editors and reviewers for the fast editorial process and the interesting suggestions done to improve this paper.

References

[1] ABADI, M., AND ROGAWAY, P. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology 15(2)* (2002), 103–127.

[2] AGUILAR MELCHOR, C., AND DESWARTE, Y. pMIX: Untraceability for Small Hiding Groups. In *Fourth IEEE International Symposium on Network Computing and Applications (NCA'05), Boston, MA, USA* (2005), IEEE Computer Society Press, pp. 29–40.

[3] AGUILAR MELCHOR, C., AND DESWARTE, Y. From DC-nets to pMIXes: multiple variants for anonymous communications. In *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06), Boston, MA, USA* (2006), IEEE Computer Society Press, pp. 163–172.

[4] AGUILAR MELCHOR, C., DESWARTE, Y., AND IGUCHI-CARTIGNY, J. Closed-Circuit Unobservable Voice Over IP. In *23rd Annual Computer Security Applications Conference (ACSAC'07), Miami, FL, USA* (2007), IEEE Computer Society Press, pp. 119–128.

[5] AGUILAR MELCHOR, C., AND GABORIT, P. A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol. In *Western European Workshop on Research in Cryptology (WEWoRC'2007), Bochum, Germany. Book of Abstracts* (2007), pp. 50–54, Extended version available on IACR eprints `http://eprint.iacr.org/2007/446`.

[6] BARKER, E., JOHNSON, D., AND SMID, M. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised). Tech. Rep. SP800-56A, National Institute of Standards and Technology, 2007.

[7] BOS, J., AND DEN BOER, B. Detection of Disrupters in the DC Protocol. In *8th Annual Eurocrypt Conference (EUROCRYPT'89), Houthalen, Belgium* (1990), vol. 434 of *Lecture Notes in Computer Science*, Springer, pp. 320–328.

[8] CACHIN, C., MICALI, S., AND STADLER, M. Computationally Private Information Retrieval with Polylogarithmic Communication. In *18th Annual Eurocrypt Conference (EUROCRYPT'99), Prague, Czech Republic* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 402–414.

[9] CHANG, Y.-C. Single Database Private Information Retrieval with Logarithmic Communication. In *Information Security and Privacy: 9th Australasian Conference (ACISP'04), Sydney, Australia* (2004), vol. 3108 of *Lecture Notes in Computer Science*, Springer, pp. 50–61.

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

[11] CHAUM, D. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM 28*, 10 (1985), 1030–1044.

[12] CHAUM, D. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology 1*, 1 (1988), 65–75.

[13] CHOR, B., GOLDREICH, O., KUSHILEVITZ, E., AND SUDAN, M. Private Information Retrieval. In *46th IEEE Symposium on Foundations of Computer Science (FOCS'95), Pittsburgh, PA, USA* (1995), IEEE Computer Society Press, pp. 41–50.

[14] COOPER, D. A., AND BIRMAN, K. Preserving Privacy in a Network of Mobile computers. In *1995 IEEE Symposium on Security and Privacy (S&P'95), Oakland, CA, USA* (1995), IEEE Computer Society Press, pp. 26–38.

[15] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. F. Tor: The Second-Generation Onion Router. In *11th USENIX Security Symposium (USS'04), San Diego, CA, USA* (2004), USENIX, pp. 303–320.

[16] DOLEV, S., AND OSTROVSKY, R. Xor-Trees for Efficient Anonymous Multicast and Reception. *ACM Transactions on Information and System Security 3*, 2 (2000), 63–84.

[17] ELGAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory 31*, 4 (1985), 469–472.

[18] FREEDMAN, M. J., AND MORRIS, R. Tarzan: a Peer-to-Peer Anonymizing Network Layer. In *ACM Conference on Computer and Communications Security (CCS'02), Washington, DC, USA* (2002), ACM Press, pp. 193–206.

[19] GASARCH, W., AND YERUKHIMOVICH, A. Computational inexpensive PIR, 2006. Draft available online at `http://www.cs.umd.edu/~arkady/pir/pirComp.pdf`.

[20] GENTRY, C., AND RAMZAN, Z. Single-Database Private Information Retrieval with Constant Communication Rate. In *32nd Annual International Colloquium on Automata, Languages and Programming (ICALP'05), Lisbon, Portugal* (2005), vol. 3580 of *Lecture Notes in Computer Science*, Springer, pp. 803–815.

[21] GOLLE, P., AND JUELS, A. Dining Cryptographers Revisited. In *23rd Annual Eurocrypt Conference (EUROCRYPT'04), Interlaken, Switzerland* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 456–473.

[22] JonDo, Privacy needs Anonymity. `http://anon.inf.tu-dresden.de`.

[23] JERICHOW, A., MÜLLER, J., PFITZMANN, A., PFITZMANN, B., AND WAIDNER, M. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications 16*, 4 (1998), 495–509.

[24] LIPMAA, H. An Oblivious Transfer Protocol with Log-Squared Communication. In *8th Information Security Conference (ISC'05), Singapore* (2005), vol. 3650 of *Lecture Notes in Computer Science*, Springer, pp. 314–328.

[25] MANN, E. Private Access to Distributed Information, Technion Master's Thesis, Israel, 2004.

[26] MARTIN, D. M. *Local anonymity in the Internet.* PhD thesis, Boston University, 1999.

[27] PADLIPSKY, M. A., SNOW, D. W., AND KARGER, P. A. Limitations of end-to-end encryption in secure computer networks. Tech. Rep. ESD-TR-78-158, The MITRE Corporation: Bedford MA, HQ Electronic Systems Division, Hanscom AFB, MA, August 1978.

[28] PAILLIER, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *18th Annual Eurocrypt Conference (EUROCRYPT'99), Prague, Czech Republic* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238.

[29] PFITZMANN, A. *Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz*, vol. 234 of *Informatik-Fachberichte*. Springer, 1990.

[30] PFITZMANN, A., AND KÖHNTOPP, M. Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. In *Designing Privacy Enhancing Technologies* (Berkeley, CA, USA, July 2000), H. Federrath, Ed., vol. 2009 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–9.

[31] PFITZMANN, A., PFITZMANN, B., AND WAIDNER, M. ISDN-mixes: Untraceable communication with very small bandwidth overhead. vol. 267 of *Informatik-Fachberichte*, Springer, pp. 451–463.

[32] PFITZMANN, A., AND WAIDNER, M. Networks without User Observability—Design Options. In *4th Annual Eurocrypt Conference (EUROCRYPT'85), Linz, Austria* (1986), vol. 219 of *Lecture Notes in Computer Science*, Springer, pp. 245–253.

[33] REITER, M. K., AND RUBIN, A. D. Crowds: Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur. 1*, 1 (1998), 66–92.

[34] RENNHARD, M., AND PLATTNER, B. Practical Anonymity for the Masses with MorphMix. In *8th International Conference in Financial Cryptography (FC'04), Key West, FL, USA* (2004), vol. 3110 of *Lecture Notes in Computer Science*, Springer, pp. 233–250.

[35] SASSAMAN, L., COHEN, B., AND MATHEWSON, N. The pynchon gate: A secure method of pseudonymous mail retrieval. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2005)* (Arlington, VA, USA, November 2005), pp. 1–9.

[36] SHERWOOD, R., BHATTACHARJEE, B., AND SRINIVASAN, A. P5: A Protocol for Scalable Anonymous Communication. In *2002 IEEE Symposium on Security and Privacy (S&P'02), Berkley, CA, USA* (2002), IEEE Computer Society Press, pp. 58–72.

[37] STERN, J. P. A New Efficient All-Or-Nothing Disclosure of Secrets Protocol. In *13th Annual International Conference on the Theory and Application of Cryptology & Information Security (ASIACRYPT'98), Beijing, China* (1998), vol. 1514 of *Lecture Notes in Computer Science*, Springer, pp. 357–371.

[38] VON AHN, L., BORTZ, A., AND HOPPER, N. J. k-Anonymous Message Transmission. In *10th ACM Conference on Computer and Communications Security (CCS'03), Washingtion, DC, USA* (2003), ACM Press, pp. 122–130.

[39] WAIDNER, M. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. In *8th Annual Eurocrypt Conference (EUROCRYPT'89), Houthalen, Belgium* (1990), vol. 434 of *Lecture Notes in Computer Science*, Springer, pp. 302–319.

[40] WAIDNER, M., AND PFITZMANN, B. The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability (Abstract). In *8th Annual Eurocrypt Conference (EUROCRYPT'89), Houthalen, Belgium* (1990), vol. 434 of *Lecture Notes in Computer Science*, Springer, p. 690.

[41] YAO, G., AND FENG, D. A New k-Anonymous Message Transmission Protocol. In *5th International Workshop on Information Security Applications (WISA'04), Jeju Island, Korea* (2004), vol. 3325 of *Lecture Notes in Computer Science*, Springer, pp. 388–399.