Closed-Circuit Unobservable Voice Over IP.

Carlos Aguilar Melchor
XLIM-DMI
Université de Limoges
123, av. Albert Thomas
87060 Limoges Cedex
FRANCE

Yves Deswarte
LAAS-CNRS
Université de Toulouse
7, avenue du Colonel Roche
31077 Toulouse Cedex 4
FRANCE

Julien Iguchi-Cartigny
XLIM-DMI
Université de Limoges
123, av. Albert Thomas
87060 Limoges Cedex
FRANCE

Abstract

Among all the security issues in Voice over IP (VoIP) communications, one of the most difficult to achieve is traffic analysis resistance. Indeed, classical approaches provide a reasonable degree of security but induce large round-trip times that are incompatible with VoIP.

In this paper, we describe some of the privacy and security issues derived from traffic analysis in VoIP. We also give an overview of how to provide low-latency VoIP communication with strong resistance to traffic analysis. Finally, we present a server which can provide such resistance to hundreds of users even if the server is compromised.

Index Terms
Unobservability, Anonymity, Voice over IP, Low-Latency

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 will be 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 the content of each packet. 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 will be 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*.

Strong traffic analysis resistance is hard to obtain. It is commonly accepted that the only practical way to achieve it is by the usage of relays that hide the meta-data of the communications.[1] There exists an extensive literature on how to use multiple relays sequentially to obtain traffic analysis resistance. Some of the proposals are based on an usual server-client model [15, 10], and others are peer-to-peer [8, 17], but only two finalized implementations are currently widespread and operational: JAP [7] the Java Anon Proxy, and Tor [6], the second generation onion routing network.

In order to respect the latency constraints of VoIP communication it is not possible (at least over the Internet) to use multiple relays sequentially, and therefore it is preferable that all the users communicate with only one relay. In this paper, this relay is called the communication server. We consider that packets can be routed through this server with a reasonable round-trip time for VoIP communication (including at most 100 ms of processing time in the server). We have not tested or implemented the servers we propose in this paper. We aim to present a theoretical overview of what performance we can achieve with different techniques.

We have also decided to focus on the communication stream. We do not considered signaling issues. How to define a practical signaling protocol that avoids traffic analysis is well beyond the scope of this paper, as well as how

[1]In 1985 a relay-independent approach was proposed [14, 16]. However the anonymous communication systems derived from this technique with practical implementations have been based on relay usage [12].

SIP or another standard signaling protocol can be modified or encapsulated to avoid leaking information to an attacker. Obtaining a low-latency VoIP anonymous communication system is expensive and complex and will probably require dedicated protocols and servers. With current latency, bandwidth and processing power it is unrealistic to consider that an anonymous VoIP communication service will provide strong resistance to attackers and be compatible with standard VoIP protocols, allow conferences, codec negotiation, etc. The service provided will much more likely be a closed-circuit system with medium size sets of users, strong limitations on the number of simultaneous communications and based on protocols dedicated to ensure that no traffic analysis can be done. These servers are intended to be used on highly secured environments like embassies, defense contractors, intelligence agencies, military tactical communications, high-tech research facilities, etc.

How to provide very low-latency unobservability has been scarcely studied in the literature. In [13], a large survey is done of the different approaches to obtain such a service based on a technique originally presented in [5] called superposed sending. This approach will be discussed in section 5. In [3] we proposed a thorough study of how to combine different primitives to obtain low-latency unobservable communication.

The contribution of this paper is twofold. First, we present how traffic analysis can result in privacy and security issues in VoIP. Second, we adapt the techniques of [3] and propose a set of servers providing strong traffic analysis resistance for VoIP communications. In particular, we propose the first server able to provide this service to hundreds of users over the Internet, even if the attackers monitor the whole network or control the VoIP server. The only assertions done to limit the attackers' strength is that they are unable to break the cryptographic primitives used, and that they do not control the computers of the users they want to obtain information from.

This paper is divided in two parts. The first part (from section 2 to section 5) defines the issues of traffic analysis in the VoIP context, and presents the systems that can be implemented based on classic techniques. The second part (section 6) presents our proposal. We have three main reasons that justify the choice of having a very large introductory part, and then a much smaller section presenting our system. First, obtaining a low-latency anonymous communication service is expensive, and therefore it is important to justify why, especially in sensitive contexts, the cost of such systems is justified. Second, one may be tempted to use lesser forms of traffic analysis resistance as it can be done with higher latency applications such as web browsing or electronic mail. We therefore introduce the different levels of traffic analysis resistance and show why only the stronger properties can ensure anonymity in interactive low-latency communication systems. Third, obtaining such a system with a single server is a pretty unexplored research domain. It is therefore important to present a thorough analysis that provides comparative performance overviews and justifies the complexity of the final solution.

Section 2 is devoted to the basic concepts related to traffic analysis resistance. In Section 3 we present the privacy and security issues related to traffic analysis. The usage of a trusted third party will be studied in Section 4, and the performance achievable with classical approaches is presented in Section 5. We present our server in Section 6 and conclude in Section 7.

2 Basic notions

Most research in anonymous communication deals with single messages and the possibility to link them to users. This approach comes from the fact that the first papers on this domain were oriented towards mailing systems. Indeed, in these systems a communication is usually composed of a single unidirectional message, with possibly a reply (with a very large latency between the two messages).

In a VoIP context, communications are bidirectional and usually formed of large sets of packets with very low latency between them. We will therefore 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 Definitions

Informally, let S by a set of **communicating** users and C a communication in which at least one user of S takes part. If for any communication C, an attacker $\mathcal{A}$ is unable to link it to a specific user in S, we will say that this set is an *anonymity set* (with respect to $\mathcal{A}$). A stronger property of traffic analysis resistance is unobservability. We will say that a set of users is an *unobservability set* (with respect to an attacker $\mathcal{A}$) if $\mathcal{A}$ can see the communications 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 internal communication in the set or not we will say that it is a *completely unobservable set*.

It is important to remark 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. A, B are communicating together and C is communicating with an external user F. D and E are not communicating. The anonymity set cannot be larger than $\{A, B, C\}$ as it just concerns communicating users. If the communication system used provides communication unobservability the unobservability set can be as large as

$\{A, B, C, D, E\}$ as the attacker will be unable to know if a given user is communicating or not. If $\{A, B, C, D, E\}$ is a completely unobservable set, the attacker may be able to see that a user of this set is communicating with F, but the communication between A and B will remain unnoticed.

The anonymity sets provide a fair amount of traffic analysis resistance as long as global observers (attackers able to observe simultaneously the whole set of users) are not considered. Indeed, keeping the example proposed in the previous paragraph, 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 will observe that the set of communicating users will be reduced to $\{D\}$ and therefore he will conclude that there has been an internal communication between A and B. Similarly if D and E begin a communication the attacker will observe that the set of communicating users has suddendly increased from $\{A, B, C\}$ to $\{A, B, C, D, E\}$ and will conclude 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 therefore correlate these data to learn who is communicating with whom.

In the case of unidirectional communications, anonymity and unobservability can be related just to the act of sending or to the act of receiving. In VoIP, 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 VoIP, 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 VoIP systems.

As we just deal with systems that resist to strong attackers (and specially to global observers) in this paper, we will not consider the approaches providing just anonymity sets. In the following section we present the general idea behind unobservability primitives.

2.2 Sender and recipient unobservability

We suppose that attackers detect any message in a link they eavesdrop on. The only way to obtain sender (resp. recipient) unobservability is therefore to send (resp. receive) regularly *dummy traffic* among which there may be *information messages*. If it is not possible for the attacker to distinguish between information messages and dummy traffic he will not be able to say whether the user is really communicating or not.

A user can easily decide to send dummy traffic or information messages without anybody knowing (except possibly the receiver). On the other hand, obtaining recipient unobservability has proven to be an more complex issue. The different alternatives that have been proposed are based on a trusted third party (see Section 4) or introduce either large communication costs [16] or computational costs [2].

2.3 Performance bounds

In order to provide correct VoIP communications, an anonymous system must respect some minimal performance bounds. First, the round trip latency must be lower than 250ms (as recommended by the International Telecommunications Union). Second, the throughput must be at least of 8 Kbits/s, and it would be preferable if this throughput can be raised up to 32 Kbits/s (G729-EV codecs). To simplify the examples given, we will suppose that the throughput used for a VoIP communication is 10 Kbits/s in this paper.

We will also suppose that the users do not want to use neither more than ten percent of their available bandwidth nor over 1 Mbit/s for the communication system. The available bandwidths will be set in a local network to 100 Mbits/s and in the case of an Internet connection we will suppose that the users have a 1 Mbit/s upload 128 Kbits/s download xDSL line. The server is supposed to have a 100 Mbits/s connection dedicated to the VoIP traffic whether the users are in a local area network or are connected through the Internet.

3 Privacy and security issues

As indicated in the previous section, three sorts of traffic analysis resistance can be distinguished:

- prevent an attacker to know who is communicating with whom: the communicating users form an anonymity set,
- prevent an attacker to know if a user is communicating or not: the users form an unobservability set,
- prevent an attacker to know whether there is ongoing communications or not in an unobservability set: the users form a completely unobservable set.

Even if in this paper we do not deal with systems just allowing the users to form anonymity sets, it is important to separate the issues associated with each of these cases, as it shows what security and privacy enhancing features result of the usage of unobservability providing systems (besides resistance to global observers).

3.1 Anonymity issues

Sometimes, knowing who is communicating with whom is much more important than knowing what is being said. For example, inferring a society strategies and alliances from the phone-calls made by its CEO is a clear security risk. From the privacy protection point of view, if a user makes a phone-call to Alcoholic Anonymous, it does not matter much whether the conversation is encrypted. It would be a clear privacy leak if somebody learnt about this communication.

3.2 User unobservability issues

If an attacker eavesdrops on a communication link used by a single user (for example the link directly connected to a computer in an office or in a home connection), the existence of a communication on this link will probably reveal the presence of the user in front of his computer and therefore his location. Knowing that a user is communicating, even without knowing with whom is indeed an information leading to various location and information inference issues.

Location issues can be very important for security. For example, in a military context, the presence of a communicating officer is a critical information. In a company, knowing whether a given person is in his office or not can make a burglary easier. In an embassy, the presence of a high personality in his office can engage a distant attack.

Inference from traffic analysis is also a major issue in security. Again, in a military context, the detection of an air-strike by a mobile radar can be inferred from a sudden communication burst from the radar to a command center. By detecting who is communicating in the different wings of a sensitive building, much information can also be inferred. For example, an outbreak of communications among the users in the middle-east section of an intelligence agency may reveal the discovery of an incoming threat to an eavesdropper able to know who is communicating and who is not.

Of course location and inference issues are also critical for privacy. For example, learning if a user is communicating from his home or his office can lead to serious privacy leaks. Unacceptable inference of a user's habits can be obtained from cross-referencing location data with communication timings.

3.3 Set unobservability issues

Knowing the number of communications in a set can be a critical information by itself. In a military context, a sudden rise on the enemy communications can for example reveal the imminence of an attack.

Another major issue is that user observability depends on set observability. Indeed, if an attacker wants to know if a user U is communicating or not and observes that there is no communication in the set he will learn with probability 1 that the U is not communicating. Likewise the attacker will know that the larger the number of communications in the set is, the more probable is that U is communicating.

Remark that this is a very important issue, but as sets grow larger the attacker will obtain less and less information about a given user. Indeed, granularity will decrease and therefore the influence a user has on the number of communications will be relatively smaller and smaller.

4 Trusted third party servers

To obtain low latency unobservable communications, one can use a trusted third party generating cover traffic and therefore providing recipient unobservability.

4.1 General construction

The idea is to have a server which routes all the communications between the users connected to it. The usage of cover traffic will ensure that all the users connected form a completely unobservable set. We will call such a server a *tMIX* (for *trusted MIX*[2]). When a user gets connected to the tMIX, they both follow protocol 1.

Protocol 1 tMIX connection.

1. The user sets an encrypted link with the tMIX.
2. The user sets an upload cover traffic channel by sending every tenth of a second a one kilobit packet of encrypted garbage to the tMIX.
3. The tMIX sets a download cover traffic channel by sending every tenth of a second a one kilobit packet of encrypted garbage to 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 (for example through a special identifier at the beginning of the decrypted message). When a user A wants to have a communication with another user B they both follow protocol 2.

When a set of users is connected to the tMIX, an attacker will only be able to see that they all send every tenth of a second an encrypted one kilobit packet to the tMIX and that they all receive every tenth of a second a one kilobit packet from the tMIX. As long as the attacker cannot decrypt the packets, he is unable to say whether they are encrypted garbage or encrypted information and therefore he

[2]A MIX is a node used to relay messages while hiding the relation between incoming and outgoing messages [4]

Protocol 2 tMIX communication.

1. A replaces the one kilobit encrypted garbage packets on his upload channel by one kilobit encrypted packets encapsulating the information he wants to send to B.
2. Upon reception of these packets the tMIX decrypts them and recognizes that they are not encrypted garbage.
3. The tMIX forwards these messages to user B's download channel by replacing the encrypted garbage he is sending him by an encryption of the packets received from A.
4. B decrypts the packets received from the tMIX and recognizes an incoming communication from A.
5. B replies to A following the same protocol.

is unable to distinguish whether a user is communicating or not.

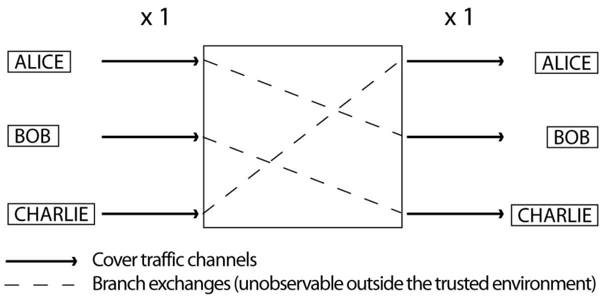
x 1
x 1
ALICE
BOB
CHARLIE
ALICE
BOB
CHARLIE
Cover traffic channels
Branch exchanges (unobservable outside the trusted environment)

Figure 1. tMIX description.

Likewise the view the attacker has from the whole system is the same whether there are communicating users or not. Therefore the attacker is not only unable to say whether a user is communicating or not but also whether there are ongoing communications or not. The set of users forms a completely unobservable set.

4.2 Implementations

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.

In classical systems providing anonymity sets for medium or large latency applications, the usual way to avoid this issue is to share the trust among a set of relays that are used one after another. In this set, the relays must all betray the users (or be compromised by the attacker) to be able to obtain some traffic analysis information.

This cannot be done in our context for various reasons. The most important of them is that recipient unobservability is provided by the cover traffic sent to the users by the last relay used, and therefore it is enough to compromise it to defeat recipient unobservability. More complex trust distribution techniques can be used to avoid this attack, but they generally lead to unsatisfactory performance. The second main reason not to use trust distribution among different servers is that the latency introduced by this approach is proportional to the number of servers used and in most of the cases it is impossible to respect the latency restrictions if more than one relay is used.

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

The trust will not be in this case placed on the server administrators and on the server resistance to intrusions, but on the trusted hardware device. This trust consists in mainly two assumptions. First, that the administrators are unable to tamper 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 ?*

Indeed, 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.

4.3 Performance

Every user needs a 10 Kbits/s duplex channel for its unobservability. A trusted server can therefore handle a completely unobservable set of ten thousand users using completely its 100 Mbits/s connection.

For the approach based on trusted hardware, the main limitation will be the device I/O throughput. Indeed, last generation trusted hardware devices have a USB2 communication interface, however, due to obfuscation needs, the I/O throughput is usually very low. Using the IBM 4758 high performance secure co-processor, this value is limited

Context	Trusted administrators	Trusted hardware
Max users	thousands	hundreds
Communications	n/2	n/2
User up/down	1 | 1	1 | 1
Server up/down	n | n	n | n

Figure 2. tMIX performance overview.

to 8 Mbits/s. With such devices, the tMIX will be able to handle a completely unobservable set of eight hundred users per co-processor used. Figure 8 presents the performance results, n being the number of users connected to the tMIX. As each user has a 10 Kbits/s upload and download cover traffic channel at his disposal he can only deal with one communication at a time which limits the maximum number of simultaneous communications this server can deal with to $n/2$. The users' expansion factors are one both for the upload and download channels. The server will receive and send $n \times 10$ Kbits/s which is indicated in the last line of the performance overview.

5 Broadcast-based servers

In many situations it will not be tolerable to have a single point of failure for traffic analysis or to consider a trusted entity, and it will be mandatory to have a communication system in which even if the server is compromised, no traffic analysis can be done.

5.1 The bMIX

A very simple way to avoid placing trust in anyone is first to encrypt the upload channel cover traffic with a key shared with the recipient instead of the tMIX when the user is communicating and with a random key when he is not. Second, the server will not create a download cover traffic channel for each user and instead of it he will broadcast all the users upload cover traffic channels.

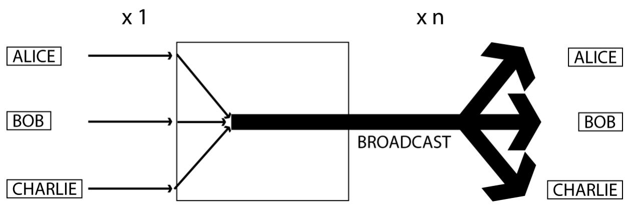

Figure 3. bMIX description.

With such changes, the server, which will be called a *bMIX* (for *broadcast MIX*), is unable to know when a user is communicating or not. Indeed, he cannot know if the user is using his upload channel to communicate as he is unable to decrypt it. The bMIX cannot either know when a user is receiving information as he sends the same encrypted information to all the users, and he is unable to know which users are able to decrypt a stream and which are unable. When a user A gets connected to the bMIX he follows protocol 3.

Protocol 3 bMIX connection.

1. A exchanges a secret key K_{AX} with each user X connected to the bMIX.
2. He sets an upload cover traffic channel by sending every tenth of a second a one kilobit packet of encrypted garbage to the tMIX.
3. The bMIX broadcasts every tenth of a second all the packets received from the upload cover traffic channels of the users.
4. For each user X, A tries to decrypt the packets from X's upload channel with the secret key K_{AX}. On failure he drops the result.

When A wants to communicate with another user B he just encrypts his upload channel with K_{AB}, the secret key they share. When B will try to decrypt the broadcasted channels he will discover that the one associated to A gets decrypted into an incoming communication. He will then replace the garbage of his upload channel with his reply and encrypt it with K_{AB}.

Context	LAN	ADSL	UMTS
Max users	100	10	3 - 4
Communications	n/2	n/2	n/2
User up/down	1 | n	1 | n	1 | n
Server up/down	**n** | n	n^2 | n	n^2 | n

Figure 4. bMIX performance overview.

If n users are connected to the server, each will receive $n \times 10$ Kbits/s quickly saturating his download bandwidth. This technique is therefore unusable over the Internet for more than a few users, but can be used in a local area network to form completely unobservable sets of up to one hundred users (in which case each user will use 1 Mbit/s of his download bandwidth). Figure 4 resumes the performance values. Remark that on a LAN the bMIX will broadcast $n \times 10$ Kbits/s (which is represented by a bold **n** in Figure 4). If the users are distributed over the Internet, the bMIX will be unable to broadcast and will have to unicast $n \times 10$ Kbits/s to each of the n users and therefore have an upload expansion factor of n^2.

5.2 The sMIX

To limit bandwidth usage, it is possible to use superposed sending [5]. Suppose that at a given time there is m internal communications in a set of users (which implies that $2m$ users are actively using their upload channels). Let us note $U_1, \cdots, U_{2m}$ the set of communicating users. Superposed sending is a turn-based collaborative technique such that when implemented in our context can ensure that at any time:

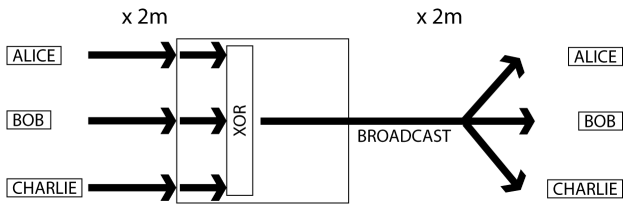

Figure 5. sMIX description.

- every user knows the number $2m$ of users actively transmitting (i.e. not sending garbage), but not who they are,
- every user has $2m$ active upload channels $C_1, \cdots C_{2m}$, using therefore $2m \times 10$ Kbits/s of bandwidth,
- each transmitting user U_i sends his communication mixed with scrambling data on the upload channel C_i, and just scrambling data on the others,
- the non-transmitting users send scrambling data on all the channels,
- the server xors all the users' sets of $2m$ channels into a single set of $2m$ channels,
- the scramblings cancel themselves when xored and the result is the set of $2m$ unscrambled transmissions from $\{1, \cdots, 2m\}$.

The only information that can be obtained when such a protocol is used is m, the number of active communications. It is not possible to know if a user is communicating or not except if all the other users betray him. With such a server (that we will call *sMIX*), the users must have $2m$ upload channels and therefore will use $2m \times 10$ Kbits/s of their upload bandwidth. As for the bMIX recipient unobservability is ensure by the broadcast of the $2m$ resulting channels.

Context	LAN	ADSL	UMTS
Max users	hundreds		
Communications	2000/n		
User up/down	2m | 2m		
Server up/down	**2m** | 2n*m		

Figure 6. sMIX performance overview.

Using a sMIX to communicate, the users will therefore use $2m \times 10$ Kbits/s both of their upload (for the superposed sending protocol) and their download (as the sMIX broadcasts the resulting channels) bandwidth. This represents a drastic reduction of the download bandwidth when compared to the $n \times 10$ Kbits/s of the bMIX as usually $m \ll n$. For example, in a university with one thousand phone sets the number of simultaneous communications will scarcely raise over ten. To simplify the provided formulas we will consider that usually $m \leq n/100$, if this is not the case the reader is invited to adapt the performance evaluation we provide to its specific context.

The server receives $2n \times m \times 10$ Kbits/s. Supposing that $m \simeq n/100$ the 100 Mbits/s bandwidth will be saturated for eight hundred users. However, in practice, handling more than two or three hundred users is difficult because of the collaborative nature of the superposed sending protocol. In the Internet the sMIX is not usable as the latency constraints are too strong for this protocol to be used (with current latency performance in Internet connections). The sMIX server has to XOR the incoming traffic, which has a computational cost linear in $m \times n$. This limits the maximum number of simultaneous communications, but not enough to be a practical issue.

6 PIR-based servers

Private Information Retrieval (PIR) is a cryptographic primitive that allows a user to download an element from a database without revealing to anybody, even the database administrators, what element is being retrieved. Current PIR schemes are very efficient from a communication point of view. In [11], a scheme is proposed in which the user sends a small query and obtains the database element he is interested in with an expansion factor on the communications of 2.

Instead of using PIR protocols over a database we propose in [2] to use them over a set of streams (which can be seen as a database evolving very quickly) to select one stream among many without the streaming server noticing which stream is being selected.

6.1 The pMIX

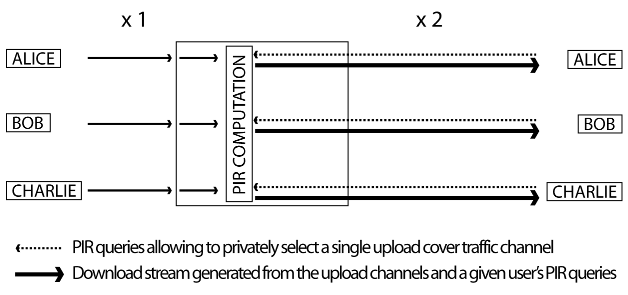
x 1
x 2
ALICE
BOB
CHARLIE
PIR COMPUTATION
ALICE
BOB
CHARLIE
PIR queries allowing to privately select a single upload cover traffic channel
Download stream generated from the upload channels and a given user's PIR queries

Figure 7. pMIX description.

We define a *pMIX* as a bMIX that instead of broadcasting all the user upload cover channels defines them as a set of streams among which the users privately choose one by sending every few seconds a PIR query. Each user must send PIR queries at the same pace. If a user A is having a communication with another user B he will choose B's stream with his PIR queries. If a user is not communicating he will randomly choose a stream and send queries for it. The security properties of the PIR scheme used ensures that all the queries are indistinguishable both for the database administrators and for global observers whatever the queried streams are. Protocol 4 shows the protocol followed between a user and the pMIX server upon connection.

Protocol 4 pMIX connection.

1. The user sets an encrypted link with the pMIX.
2. The user sets an upload cover traffic channel by sending every tenth of a second a one kilobit packet of encrypted garbage to the pMIX.
3. The user sends every five seconds a PIR query for a random stream among n.
4. The pMIX sets a download channel by using the user's PIR queries to generate a stream out of the set of the n upload cover traffic channels.

In a bMIX the users use 10 Kbits/s for their upload channel and $n \times 10$ Kbits/s for their download channel. In a pMIX, using an instantiation of the protocol proposed in [11], the users will use 10 Kbits/s for the upload channel and 20 Kbits/s for their download channel. The cost of sending the PIR queries does not change the order of magnitude of the upload bandwidth.

Context	LAN	ADSL	UMTS
Max users	10*CPUS$^{1/2}$	10*CPUS$^{1/2}$	10*CPUS$^{1/2}$
Communications	n/2	n/2	n/2
User up/down	1 | 2	1 | 2	1 | 2
Server up/down	2n | n	2n | n	2n | n

Figure 8. pMIX performance overview.

This server provides a full unobservability set to its users with almost optimal bandwidth usage and does not require any trust from them. On the other side, the computational cost of the operations done is so large that it can just handle very small sets of users. Moreover, the scalability of pMIXes is reduced as the computational cost is proportional to n^2, n being the number of users. A server with a high-end processor will not be able to go over $n^2 \simeq 100$ using all its processing power. Thus a mono-processor server will not be able to handle much more than a dozen users, and even a strong initial investment in a multi-processor server or a cluster of computers will not allow to have more than some dozens of users at a reasonable price. For example, a cluster with twenty-five processors will lead to $n^2 \simeq 2500$ and therefore to a set of just fifty users.

Obtaining a server that is both efficient from a communication and computational point of view is a bit more complicated. We present such a server in the next section.

6.2 The apMIX

The idea we propose to limit the computational cost of a PIR-based server is to reduce the number of PIR queries that must be treated. Instead of sending directly their PIR queries to the server, the users will use a superposed sending protocol increasing the communication cost of the emission by $2m$ but enabling the server to recover after the xoring phase just $2m$ PIR queries instead of n.

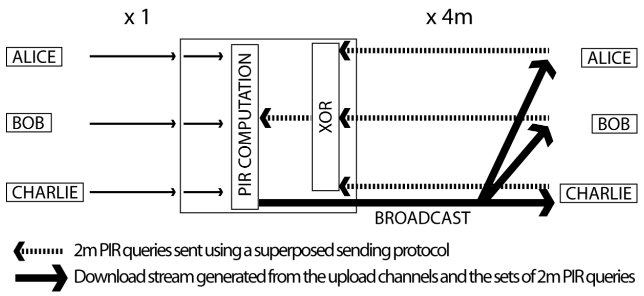
x 1
x 4m
ALICE
BOB
CHARLIE
PIR COMPUTATION
XOR
ALICE
BOB
CHARLIE
BROADCAST
2m PIR queries sent using a superposed sending protocol
Download stream generated from the upload channels and the sets of 2m PIR queries

Figure 9. apMIX description.

As the server has only to reply to $2m$ PIR queries the computational cost is proportional to $n \times 2m$ instead of n^2. The number of simultaneous communications in a set of users is generally at least one or two orders of magnitude smaller than the set size and therefore this optimization results in a much smaller computational cost. Instead of handling a few dozen users we can form up with this server unobservability sets of a few hundred users. Scalability is not improved as m is proportional to n and therefore $O(n \times m) = O(n^2)$. Nevertheless, it is important to remark that in the different contexts (embassies, sensitive laboratories, military communications, etc...) in which a completely unobservable system would be used, the set of users will almost never go over some hundred users and will not need to scale up to tens of thousands of users.

Context	LAN	ADSL	UMTS
Max users	50*CPUS1	50*CPUS1	50*CPUS1
Communications	min(1*CPUS2, 50)	min(1*CPUS2, 3)	min(1*CPUS2, 1)
User up/down	1 | 4m	1 | 4m	1 | 4m
Server up/down	**4m** | n	4m*n | n	4m*n | n

#CPUS required by the apMIX = CPUS1 * CPUS2

Figure 10. apMIX performance overview.

Using the protocol proposed in [11] to generate the PIR replies, a server with a high-end processor will be able to handle a set of users as long as $n \times 2m \simeq 100$. This roughly means the initial investment will be of one processor per fifty users and per communication. For example, a server with four processors will be able to handle one hundred users having at most two simultaneous communications ($n \times 2m \simeq 400$). A rack of six servers with four processors each will be able to handle two hundred users having up to six simultaneous communications.

Latency is not a problem for superposed sending in this server as it was for the sMIX because is not used to send the data exchanged during the communication (which must have an RTT lower than 250 ms), but only to send the PIR queries. The latency for PIR query sending has an impact on call establishment times for which the acceptable latency is of some seconds.

The price to pay with this approach, when compared to a pMIX, is that the server is not able to know which query corresponds to which user and must therefore broadcast the $2m$ 20Kbits/s resulting streams to all the users. The upload cover traffic channel needs only 10 Kbits/s, but the download cover traffic channel will need $4m \times 10$ Kbits/s. In a local area network this limitation is minor but if the users are connected through the Internet with 1 Mbit/s of download bandwidth, the number of simultaneous communications cannot be more than three if the users limit their bandwidth usage to ten percent.

7 Conclusion

Over the Internet, without relying on a trusted third party, the only server that would be able to provide unobservable VoIP communications to more than a few users is the apMIX (as long as there are not many simultaneous communications). In a local area network, a bMIX or a sMIX can both provide unobservability in VoIP communications to one hundred users or more. However, we believe the apMIX approach is also preferable specially as it has better scalability. Indeed, for the sMIX, scalability is simple in theory but very hard to achieve in practice. In the case of the bMIX it is not possible to scale up except if dedicated lines for the VoIP system are used (in which case the bMIX is the best option without any doubt).

The investment needed for an apMIX may seem unreasonable, but this should be moderated by other costs, specially the cost per user of the VoIP infrastructure. Indeed, the level of security provided by the systems presented in this paper is inconsistent with the usage of softphones. A network in which such strong unobservability properties would be expected each user will probably have a secured hardphone. These hardphones must moreover be able to encrypt and decrypt the communications as traffic analysis resistance is senseless if the communications are unencrypted. This means that the cost to set up such a VoIP network will be at least some hundred dollars per user on the network. The cost per user with an apMIX is roughly (if we suppose each processor costs five hundred dollars) of ten dollars per simultaneous communication that the server can handle. We can therefore conclude that the cost introduced by the apMIX is reasonable when compared with the other costs, as long as the number of simultaneous communications does not reach many dozens.

In this paper we have just considered anonymizing the communication data. As future work we plan to deal with the communication signaling and try to build a set of consistent modules for an Asterisk IP PBX. We hope that this work will motivate research in this field.

8 Acknowledgements

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`.

References

[1] A. Acquisti, R. Dingledine, and P. F. Syverson. On the Economics of Anonymity. In *Financial Cryptography*, pages 84–102, 2003.

[2] C. Aguilar Melchor and Y. Deswarte. pMIX: Untraceability for Small Hiding Groups. In *Fourth IEEE International Symposium on Network Computing and Applications*, pages 29–40, 2005.

[3] C. Aguilar Melchor and Y. Deswarte. From DC-nets to pMIXes: multiple variants for anonymous communications. In *Fifth IEEE International Symposium on Network Computing and Applications*, pages 163–172, 2006.

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

[5] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[6] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[7] JAP, Anonymity and Privacy. `http://anon.inf.tu-dresden.de`.

[8] M. J. Freedman and R. Morris. Tarzan: a Peer-to-Peer Anonymizing Network Layer. In *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206, 2002.

[9] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Onion Routing. *Communications ACM*, 42(2):39–41, 1999.

[10] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, and M. Waidner. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[11] H. Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In J. Zhou, J. Lopez, R. H. Deng, and F. Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2005.

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

[13] A. Pfitzmann. How to Implement Isdns Without User Observability - some Remarks. Technical report, Karlsruhe, 1985.

[14] A. Pfitzmann. *Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz*, volume 234 of *Informatik-Fachberichte*. Springer, 1990.

[15] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.

[16] A. Pfitzmann and M. Waidner. Networks without User Observability—Design Options. In F. Pichler, editor, *Advances in Cryptology—EUROCRYPT 85*, volume 219 of *Lecture Notes in Computer Science*, pages 245–253. Springer-Verlag, 1986, April 1985.

[17] M. Rennhard and B. Plattner. Practical Anonymity for the Masses with MorphMix. In *Financial Cryptography*, pages 233–250, 2004.