RC22214 (W0110-054) October 22, 2001
Computer Science

IBM Research Report

The Past, Present, and Future of Network Anonymity

Dogan Kesdogan, Charles Palmer
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

The Past, Present and Future of Network Anonymity

D. Kesdogan, C. Palmer

IBM Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
{kesdogan, ccpalmer}@us.ibm.com

Abstract

The Internet promises an ever-increasing variety of services available anytime, almost anywhere, to anyone of just about any experience level. Thus, in many respects, the virtual world has become a viable alternative to our real world, where we can buy anything from a dishwasher to personal services, or publish any information we choose on a personal web site. With all of the convenience and freely-available information that this virtual world provides, it has one major problem: in the real world people can nearly always exert some control over their privacy. If they choose, they can study in absolute solitude or meet with others in private rooms, or they can anonymously buy a magazine. However, on the Internet, users have few controls, if any, over the privacy of their actions. Each communication leaves trails here or there and there is always someone who can follow these trails back to the user. Thus, should we just forget about having privacy in the Internet as others have proclaimed?

The one path towards enabling true network privacy is to provide anonymity. Anonymity services can allow users to carry out their activities anonymously and unobservably on the Internet. In this work we investigate the following technical questions: what is network anonymity, what are the various techniques for reliably achieving anonymity, and what are their associated impacts on network performance and user experience. Our emphasis is on deployable systems for the Internet that provide strong anonymity against a strong attacker model. We present the network anonymity techniques (algorithms) suggested in the past and the ones currently in use, and then we discuss possible anonymity techniques of the future. We present the accepted terminology for discussing anonymity, and the definition and different measures of anonymity. However, throughout our discussions we pay particular attention to the analysis of network performance in the presence of anonymity mechanisms.

1. Introduction

Privacy and Anonymity are essential parts of today's society. In some environments, such as medical treatments and banking, the privacy and anonymity of individuals are protected by law. Other aspects of modern life may not have such clear protections. Today, people are increasingly using computer networks to accomplish activities that were always assumed to be private or anonymous in the non-virtual world. The current protections provided by law and law enforcement are slowly being driven to adapt to this new life in the virtual world.

A complicating factor in this move is that computers and computer networks make it quite easy to maintain and distribute digital information. Millions of bits can be transferred without error, stored, and analyzed in seconds. People are aware of the potential danger of this information processing power and are afraid of losing control of their personal data in both their private and business lives. Yet, their behavior is often inconsistent with that fear. At the 2000 RSA Security Conference, privacy law expert Stewart Baker observed that there are four basic rules of privacy and anonymity opinion and behavior today:

1. Each individual firmly believes that he or she has a right to complete anonymity in all situations.

2. Each individual also firmly believe that *other* people do not have that right. Messages or letters from anonymous callers are always seen as suspicious, if not outright threatening. Individuals are also concerned that while they would never do so, others might employ anonymity to commit crimes.

3. If an individual chooses to give up some private personal information, that information cannot be recovered.

4. Most individuals choose to give up private information in exchange for trivial things, such as access to a web site or to "register" a new purchase.

While disheartening, these attitudes do not constitute a reason to abandon privacy and anonymity research. Rather, it is due to the lack of good alternatives that these rules have evolved.

The main problem is how to deal with the aforementioned information processing power posed by modern computer networks. Thus, the challenge of private information may be simply stated: you can restrict it totally or you give it all:

- If privacy was guaranteed then every individual would have the capability to act anonymously in the virtual world. Therefore, the individuals could use the great power of the information processing systems for their own benefits. The danger here is that everyone, including criminals and terrorists, could use that power for their activities.

- If law enforcement agencies are given the capabilities to undo anonymity, then they can undo it for all people and have this great power to observe everyone in greatest detail. One of the famous scenarios drawn here is from G. Orwell "1984" [Orwe90].
- There are also "complete non-anonymity" proponents. In his "Transparent Society" [Brin99], D. Brin imagines a society in which all aspects of everyone's public life are subject to viewing by web cameras. He then proposed two variants: one with the government and law enforcement being the only ones who can view the images from the omnipresent cameras, and another where *everyone*, individuals as well as government, all have full, immediate access to any of the images via the Internet.

There may be no general solution to this all or nothing problem that satisfies everyone. Even so, we should begin to address this problem now. We suggest that initially focusing on application-specific anonymity techniques may help. For instance, not all people should have the same ability to use privacy techniques, just as not all people have access to prescription drugs (restriction to some persons). Similarly, absolute anonymity should be guaranteed in electronic elections (restriction of application).

Thus, we have the real challenge of: how to achieve practical privacy through anonymity, i.e., while maintaining acceptable network performance. The rest of the paper will address this technical question and will present the important results of the last twenty-plus years of research in this area. We first provide the terminology used to define the anonymity task and then present and critique the various research efforts up to the present day. Finally, we provide our own view of the direction of future anonymity techniques.

2. The Terminology of Anonymity

We begin with a concise definition of the goals of anonymity and the terminology used by anonymity researchers. The terminology in this work is mostly taken from A. Pfitzmann [Pfit90, PfKö01]. In these works, the foundations for the formal definition and terminology is given. Extensions to this terminology can be found in [ReRu98, SySt99]. The three key privacy (anonymity) terms and definitions[1] are:

PT 1. *Anonymity* is the state of being not identifiable within a set of subjects, known as the *anonymity set*. Anonymity in communications can be further distinguished as sender and recipient anonymity.

PT 2. *Unobservability* is the state of an item of interest being indistinguishable from any other item of interest.

PT 3. *Unlinkability* of two or more items or actions means that these items are no more and no less related than they were previously (attacker gains no information).

Unobservability can be reduced to a set of data items, senders or receivers. For example a concrete requirement for messages is that each message cannot be linked to any potential

[1] In this paper we will use the more vague term *anonymity* when speaking of these three key terms.

sender or receiver from the set. At a higher level, relationship unobservability requires that it is not discernable whether anything is sent from a set of potential senders to a set of potential recipients.

A definition of anonymity is incomplete if an attacker model (opponent model) is not specified. The attacker model describes the demands placed on the anonymity techniques and is also for the evaluation and comparison of proposed solutions. In general, a direct relation exists between the strength of the attacker model and the quality of the protection provided by a given solution.

Thus, in this work we assume that anonymity is to be provided in the presence of a powerful attacker. In choosing this attacker model we follow the examples of other works which aimed for strong anonymity protection over an insecure network [Chau81, Chau88, Pfit90, CoBi95, CGKS95]. Thus, the capabilities of an attacker A may vary[2] [Chau81]:

A 1. *Passive attacker*: attacker can observe all communication links.

A 2. *Passive attacker with sending capabilities*[3]: attacker may take part in the anonymity technique (i.e., attacker can send messages) if participation has not been explicitly forbidden for him

A 3. *Active attacker*: attacker can control all communication links, switches, etc., and can attack all messages with delete replay and send, or delay actions.

In [Pfit90] the anonymity requirements are described in a more concrete form by using the above attacker model[4]. For example, consider the more concrete definition for unobservable communication:

> An event E is *unobservable* for an attacker A (i.e., A1-A3) if for each observation *O* that A can make, the probability of *E* given *O* is greater than zero and less than one: $0 < P(E \mid O) < 1$.
>
> If for each possible observation *O* that A can make, the probability of an event *E* is equal to the probability of *E* given *O*, that is $P(E) = P(E|O)$, then *E* is called *perfectly unobservable*.

To evaluate anonymity techniques according to the above definitions, the following notion of *anonymity set* and *anonymity size* can be used as suggested in [KeEB98]:

> **Anonymity set**: Let A be an attacker model and ψ represent a finite set of all users. Let R be a role for the user (sender or recipient) with respect to a message M. If, for an attacker according to model A, the a-posteriori probability p that a user *u* has the role R with respect to M is non-zero ($p > 0$), then *u* is an element of the *anonymity set* $U \subseteq \psi$.

[2] For the sake of simplicity, it is assumed here that cryptography used is unbreakable. However, it is good to keep in mind that it is inappropriate to provide or demand more anonymity protection than the underlying cryptography can provide.

[3] The A2 attacker is not much stronger than A1, yet the A2 attacker poses a bigger threat than A3 because it is by definition undetectable.

[4] Compare also with [FiHu01].

A technique (method) provides an anonymity set R of size n if the cardinality of U is n ($n \in IN$).
An algorithm provides *perfect protection* if n is always the same as the cardinality of the organized additional traffic (attacker gains no information [Shan49]).

The condition $0 < P(E \mid O) < 1$ or $p>0$ is often too vague. In [ReRu98], further degrees of trustworthiness are introduced, providing a more fine-grained model: *beyond suspicion*, *probable innocence*, *possible innocence*, *provably exposed.* Unfortunately, the descriptions of these terms are less quantitative than those in [Pfi90] because the authors provide no mapping function between the probability definition of [Pfi90] and their new terms, e.g., $0.4 < P(E \mid O) < 0.6 \rightarrow$ *possible innocence*.
In [SySt99], degrees comparable to [ReRu97] are introduced, with the difference that these degrees are described in a formal language. However, it is also true that they cannot be used for quantitative evaluations.

Finally, [PfKö01] defines anonymity as a subset of unobservability, due to the fact that when a method provides a sender with unobservability then it automatically provides the sender with anonymity.

3. The Requirements of the Anonymity Task

By choosing the powerful attacker model (omnipresent attacker model A) it follows that a single transmission by a single person can be neither anonymous nor unobservable. The omnipresent attacker can observe the sender of a message (the sending act) and follow the message to the receiver, thereby detecting the communication relation without needing to read the content of the message. Hence, our observation that anonymity techniques require additional traffic, called *cover traffic*. Having the additional traffic, we employ an *embedding function* for the subject traffic in order to confuse the adversary and conceal the particular sender, recipient, and their communication relationship.

The following tasks result:

- **Group function (cover traffic)** Since single transmissions are observable in the network, additional traffic is organized by the group function. It is essential that the attacker not be able to gain control over this additional traffic.
- **Embedding function**: The traffic generated by a particular user must be efficiently and untraceably embedded into the cover traffic.

If the attacker can control the cover traffic, all anonymity is lost. For example, in the *masquerading attack* [Denn82], the attacker sends known transmissions (n-1) times as an offer of the cover traffic, and then tries to observe the target user's transmissions by picking them out from among the known traffic. To avoid this and other attacks wherein the attacker exerts some control over the cover traffic, the *CUVE* requirements must be met (see also the discussion of secure elections in [Schn95]):

- ***C****ompleteness*: all users can verify that their messages have been correctly sent, received, or transmitted.
- ***U****n-reusability*: within any given session[5], no user can participate more than an allowed number of times.
- ***V****erifiability*: an adversary cannot change another user's message without being discovered by the system.
- ***E****ligibility*: only authorized users can participate in a given session.

The embedding function has to be applied in an environment[6] where we will not encounter any attackers[7]. This is similar to the case of encryption, wherein the application of the encryption algorithm has to be performed in a trusted environment. Otherwise, security cannot be guaranteed, since the attacker will have full knowledge of the process. However, unlike encryption the users depend on others[8] (due to the requirement of cover traffic). Thus, while the security point of view recommends having the embedding function applied only within the trusted domains of the sender and receiver (the end-to-end solution), from a practical point of view it may be preferable for the group function to use central Trusted Third Parties (TTPs).
Next, we need to specify the participants of the anonymity techniques and their trust relationships, in order to derive requirements for the network of trust.

3.1 The Participants and Their Trust Requirements

The participants and their trust environment are shown in Figure 1. The desired goal of anonymity is only achievable if at least two honest participants work together. In general, we will always assume that $n > 1$ users participate in the application of a given anonymity technique. We consider the network itself to be unsecure and the trusted domains of the users to be secure. Trusted Third Parties, or TTPs, sit between these two trusted endpoints and must fulfill some special trust requirements:

TR 1. A single point of outside trust should be avoided; the trust has to be distributed equally over all used *N* TTPs.

TR 2. TTPs should be as transparent to the user as possible, i.e., the correct functionality of the environment should be controlled by the user.

TR 3. TTPs should be independently designed and produced and have independent operators.

If (*n*-1) of the users providing the cover traffic are dishonest, then obviously the technique cannot provide any protection. Many of the previous works in anonymity neglect to consider the corrupt user, and make the assumption that all participating users are honest. Certainly in open environments like the Internet, the attacker could be an

[5] A session is the transmission of a packet using an embedding function and sent in the presence of cover traffic.
[6] The only exception to this statement is in the case of recipient anonymity for broadcasted messages.
[7] We make this simplifying assumption in order to focus only on the anonymity techniques.
[8] In general, encryption is a unilateral function and anonymity a multilateral function, i.e., a user can encrypt his electronic diary on his machine but to act anonymously he always needs additional users.

alliance of (*n*-1) dishonest persons. Unfortunately, there is no technical means to test the honesty of people and, thus, no way of providing perfect protection[9] in the technical sense.

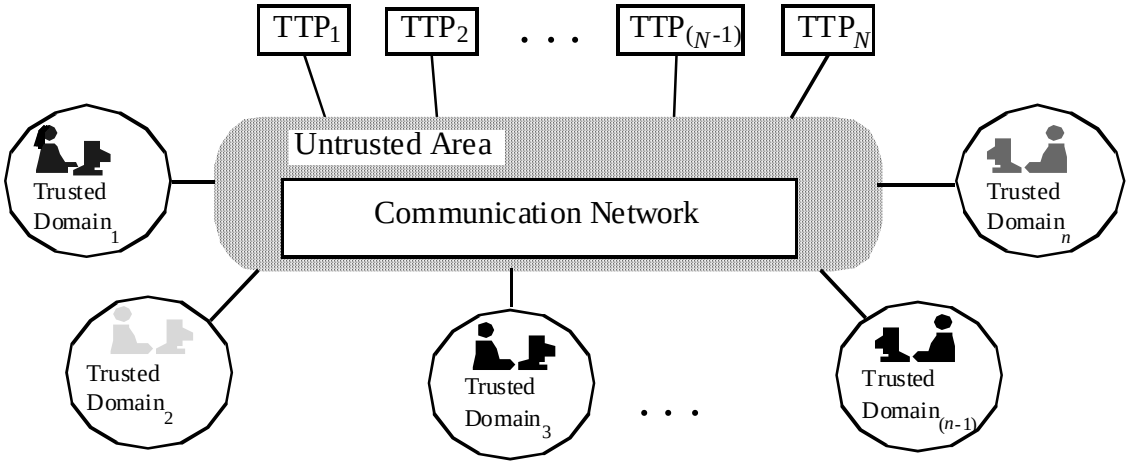

***Figure 1**: The General Model*

The following trust requirements can be added to increase the overall trust and decrease the probability of participating dishonest users:

TR 4. The size of the anonymity set should be increased as much as possible, i.e., the required number of participating dishonest users to successfully attack the method should be as large as possible.

TR 5. The number of all possible participating users should be fixed and not much greater than *n* (the number of required participants for the anonymity technique). This will ensure that an attacker has little chance of finding (*n*-1) dishonest users in the set of all users.

4. Weak Points and Attacks

In the literature [BeFK01, GuTs96, Raym01 etc.] several attacks are defined that an anonymous system should protect against. While this is true, we will not address these attacks in this paper. Most often these attacks exploit the weak points of a particular technical design. We are more interested in the fundamental weak points and the resulting attacks upon all anonymity systems.

There are two primary weak points of the anonymity systems when anonymity is defined as above:

1. Trust and group function: Anonymity techniques depend on other users.

[9] Note that using one time pad cryptography can provide perfect security [Stin95].

2. Unlinkability: this requirement, if realized, ensures that the attacker cannot gain any information by observing the system. The information gain would be through finding some patterns in the system and using them for an attack.

The first weak point leads to the well-known *(n-1)-Attack* [Pfit90]. If the attacker is somehow able to contribute or control *(n-1)* messages of the overall *n* messages, then this attack is successful. To thwart this attack and to ensure that only genuine traffic is processed, users who wish to act anonymously must identify themselves. This presumes the existence of a global PKI, which does not yet exist (for more information about this topic see [Gutt00]).

The so called *intersection attack* uses the second weak point. This attack searches for a pattern of user behavior as sender or recipient by observing him over a long time and collecting information about his communication behavior. It is hard, or perhaps even impossible, to thwart such an attack in reality, since this would mean that all actions should be independent of all other actions in past and future of the same user. This strong independency requirement is even dropped for random number generators, since only pseudo-random generators are possible [Chai74]. So, just as in the field of random numbers, we relax this strong requirement to the weaker requirement of pseudo unlinkability. Pseudo unlinkabilty requires that while linkability is possible in general, it should be computationally hard to determine it.

5. Performance Parameters for Anonymity Techniques

Anonymity protocols belong to the family of group communication protocols. Typically, networks (e.g., the Internet) are not designed to handle the huge amount of traffic produced by these protocols. In general, the situation is even worse for anonymity techniques since they depend upon dummy messages.

To evaluate the performance of the anonymity techniques, we use a model as shown in Figure 1, where we abstract from a concrete network structure and consider it as a black box. In this network model we assume that the cost of sending a message without anonymity providing technique is one and the transmission time is also one. Now, we are interested in the overhead that comes with the anonymity techniques. We describe this overhead using two parameters: *System Effectiveness* and *Time Efficiency* [Kesd99].

5.1 System Effectiveness

Since privacy techniques need cover traffic in order to provide unobservable communication, it is important to maximize the amount of real messages sent in a session. Consider a technique with a group function that handles $n>1$ messages from n distinct users. If k of the messages are real and m are dummy ($m = n-k$) then we define

System effectiveness[10] as $\eta = k/n$. η is always less than 1, since there cannot be more real messages sent than all of the messages sent. If the message is sent via several TTPs, then these additional reroutings count as re-sending all of the messages again. The formula is:

$$\eta = \frac{number\ of\ \ real\ messages\ sent\ directly\ without\ anonymity\ technique}{number\ of\ \ all\ messages * (number\ of\ re\ routes + 1)}$$

5.2 Time Efficiency

Clearly, it is desirable to have an anonymity technique with high system effectiveness ($\eta \approx 1$). Assuming that such a technique exists, it would mean that there exists *n* people who want to send *n* real messages. Since there are not always *n* people who want to send a real message within a given time period, the technique has to wait for enough real messages before beginning. The waiting time interval could be chosen that with high probability *n* people will want to send something. Thus, the people may have to wait until the specified time interval has elapsed. This time cost can be measured as **time efficiency** τ. Additionally, if the message is sent over several TTPs, then this time has to be counted also. The resulting formula for time efficiency is:

$$\tau = \frac{time\ needed\ \ to\ send\ a\ message\ directly}{time\ needed\ \ to\ send\ a\ \ message\ using\ the\ method}$$

System effectiveness and time efficiency represent measures for evaluating the basic system characteristics [Kesd99].

6. Basic Techniques

The basic techniques in this area date back to the 1970's and 1980s when David Chaum and others suggested some revolutionary techniques: broadcast and implicit addresses, MIXes, and DC-Networks. In [PfWa85] these works are first presented as basic techniques and in [Pfit90] various enhancements and extensions in theory and technique were suggested. These seminal contributions have proven to be the basis for many of today's anonymity techniques. In more recent times, a new technique providing perfect protection was discovered by two different groups [CGKS95, CoBi95]. Known as *Private Information Retrieval* (*PIR*), this new technique has similarities to DC-Networks. In this section we will present these basic techniques, but not all of the details of their variants since we are more interested in the basic features of the techniques and their resulting effects on networks. For a more comprehensive and detailed presentation see [Pfit01].

[10] The given definition is the same for *information rate* in information theory or system effectiveness in physics. In both of these cases, as with anonymity, something more (bits or energy) has to be provided to meet the goal.

6.1 Broadcast Networks and implicit addresses

Receiving a message can be made completely anonymous to observers on the network by delivering the same message (possibly end-to-end-encrypted) to all *n* stations (broadcast). If the message has a specific intended recipient, an *addressee*, the message must contain some attribute by which the addressee alone can recognize the message as being addressed to him [FaLa75]. This message attribute is called an *implicit address*. It is only meaningful to a recipient who can determine whether he is the intended addressee. In contrast, an *explicit address* describes the specific place in the network to which the message should be delivered and, therefore, cannot provide anonymity. Implicit addresses can be further distinguished according to their visibility, i.e., whether they can be tested for equality or not. An implicit address is called **invisibl**e, if it is only visible to its addressee, and is called **visible** otherwise [PfWa85].

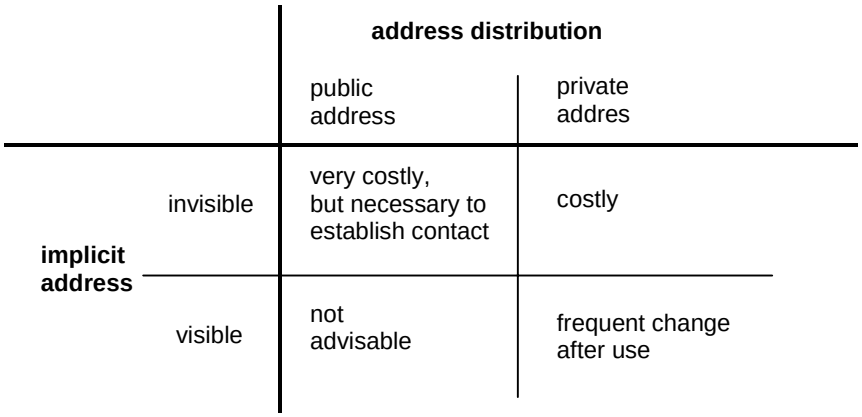

***Figure 2**: Combination of implicit addressing modes and address distribution [Pfit90]*

*Invisible implicit addresse*s, which are unfortunately very costly in practice, can be realized using a public key cryptosystem. *Visible implicit addresses* can be realized more easily by having users select arbitrary names for themselves, which are then be prefixed to messages.

Another criterion of implicit addresses is their distribution. An implicit address is called *public* if it is known to every user (like telephone numbers today) and *private* if the sender received it secretly from the addressee. This private distribution can be accomplished in several ways, including outside the network, as a return address, or by a generating algorithm that the sender and the addressee agreed upon [FaLa75, Karg77].

Public addresses should not be assigned using visible implicit addresses in order to avoid the linkability of the visible public address of a message and the addressee.

Private addresses can be realized by visible addresses but then each of them should be used only once. While this approach can produce perfectly-secure addresses, similar to the one-time pads of cryptography, both schemes suffer from difficulties with initialization and distribution.

Recipient anonymity seems to be easily obtained, providing that a secure broadcast functionality can be guaranteed. Such a network guarantees that all possible recipients receive the same message in the same time frame. Here is an example of a recipient anonymity scheme using *private visible implicit addresses* and the simplifying assumption that the network is time slotted with the parameter *t*:

1. **Initialization**: Each user securely shares two secret keys: one for end-to-end encryption, k_{enc}, and another for generating random numbers using a strong **R**andom **N**umber **G**enerator (RNG), k_{RNG}.
2. **Message Transmission**: The message *M* is end-to-end encrypted, a random number $x = \mathrm{RNG}(k_{RNG}, t)$ is generated, and the message $[x, k_{enc}(M)]$ is sent.
3. **Cover Traffic**: The packet is copied exactly (*n*-1) times and broadcasted at the same time slot to all other (*n*-1) users.

The embedding function of this trivial method simply copies the message and broadcasts it to all participants. Thus the technique assumes a reliable and secure broadcasting function. An active attacker capable of deleting, delaying and adding new messages is able to attack this scheme and disclose the recipient (see for more information [Waid89]). In [Pfit90, Waid89] physical and algorithmic countermeasures, such as using physical links like optical fiber as the medium which cannot be easily attacked, were suggested for use against this kind of attack.

6.1.1 Example

If user A wants to keep the recipient B of a message secret, he chooses additional pseudo recipients (e.g., C and D)[11]. Together with the real recipient B, these additional recipients form the anonymity set. The message is then broadcasted to all members of the anonymity set.

[11] Of course each additional recipient has to keep it secret, that the received message was not intended to him. Thus, the method has to organize a group of trustworthy recipient beforehand agreeing this condition.

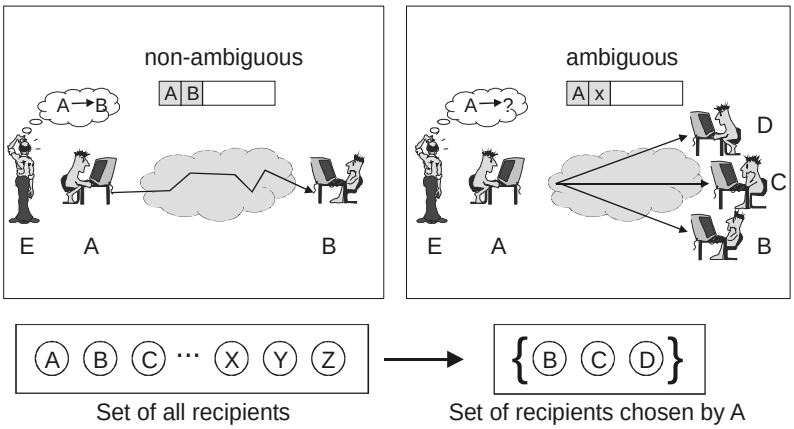

***Figure 3**: The idea of general recipient anonymity by broadcast and addressing attributes (E is the attacker)*

6.1.2 Performance

All messages are replicated *n* times and sent directly to *n* participants. Thus, the system effectiveness η is $1/n$ and the time efficiency τ is 1.

6.2 DC-Network

While the previous method provides recipient anonymity, all observers know the identity of the sender. In [Chau88] a powerful technique called DC-Networks is suggested for sender anonymity. The DC-network is time slotted and in each time slot all participating senders send a message, although for successful transmission only one of them has to be the real message and the others have to be dummy messages. Now the task is to hide the real message in the cover of the dummy messages. For this task *n* users exchange secret keys along a given key graph, each ($sender_i$) then locally exclusive-ors (XORs)[12] all of the keys they have with the dummy or real message M_i they are to send[13], and finally all of the local results of the *n* users are XOR'ed globally.

In [Pfit90] it is shown that this technique provides perfect protection if the computation is carried out over an Abelian (commutative) group. Here we formally present the technique for the simple case using binary numbers (bits):

1. **Initialization**: *n* users exchange secret keys (random numbers as long as the messages) along a given key graph *G*, i.e., a graph with the users as nodes and the secret keys as edges. If a node *x* (i.e., user *x*) is connected to node *y*, then they both share the same symmetric secret key, k_{xy}, such that all keys appear twice in the technique.

[12] Using the XOR function: $1 \oplus 1 = 0 \oplus 0 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 1$.
[13] Similar to the use of a Vernam cipher with one time pad in [Denn82].

2. **Message Transmission**: To send a message *M* including recipient address, a node *X* XORs *M* with all the keys k_{xj} that its shares with its "key graph neighbors": $M \oplus \sum k_{xj}$. The result is sent as a communication packet.
3. **Cover Traffic**: All other nodes XOR all the keys they share with their key graph neighbors and send the results as communication packets.

If all packets are XOR'ed in this manner, then only the message *M* will remain since all keys occur twice and will neutralize each other. If more than one message sent in the same slot, then each user will receive the XOR summation of these messages. Since no user will be able to decode it, the senders can recognize this "collision" and can apply a collision detection method in the next phase (e.g., ALOHA[14] protocol [Tane96]).

The embedding function here is that each of the packets contains the message, it is invisible to the attacker. Only the superposition of *all* packets can recover the hidden message.

6.2.1 Example

To send a message ("110101"), user A XORs the message with the previously exchanged secret key. Other users XOR in the same manner (Figure 4). All sums of all users are XOR'ed successively. Because every secret key is added twice, the distributed message is the message of A.

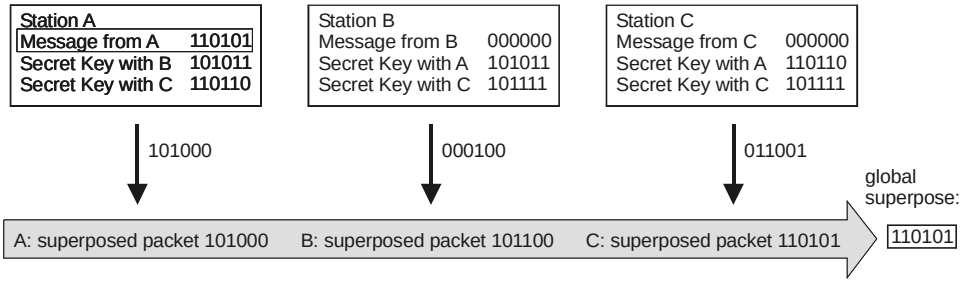

***Figure 4**: DC-Network (taken from [Pfit90])*

6.2.2 Performance

The DC-Network can be realized in several ways. We assume that we have a logical ring with two phases per slot, a sending and receiving phase. In the sending phase all users apply the protocol as given in the above example and in the receiving phase all users can read the result. Thus, η is $1/(2n)$ and $\tau=1/(2n)$. If a "collision" occurs, because two or more people attempted to send a message during the same sending phase, then η and τ have to be increased by the factor e^G (assuming slotted ALOHA[15]).

[14] Slotted Aloha refers to a simple communication scheme in which the network is time slotted and each user chooses a slot to send a data whenever he needs to. We can adapt this protocol like this: if there is no collision (can be checked after all messages XOR'ed globally) the next message can be sent. If there is a collision (sender does not see the sent own message after global superposition), he sends it again in the next slot with a given probability.

[15] The performance of slotted ALOHA is $\exp(G)$, where G is the overall traffic (i.e., the new traffic and the failed traffic).

6.3 MIX

The anonymity-providing techniques presented so far suffer from the same major drawback: they require the pre-existence of a closed anonymity set (e.g., through the exchange of secret keys). The members of the anonymity set have to establish the anonymity set themselves and are then involved in every communication. This severely limits the flexibility of the involved users.

The MIX-Method [Chau81] avoids this drawback by shifting the task of generating anonymity sets from the user to special trusted third parties (TTPs) called MIX nodes or MIXes. MIXes enable much larger networks of users because they eliminate the constraint that each user has to participate in every communication. By providing such flexible access to an anonymity service, the MIX approach is the most interesting for open networks like the Internet. Here is a more formal description of the MIX approach:

1. **Initialization**: A Public Key Infrastructure (PKI) provides users with the public keys of the MIX nodes, where PK_i is the public key of the i^{th} MIX node and SK_i the secret key.
2. **Message Transmission**: Assume a data packet $M = [A_{recipient}, k(message)]$ containing recipient's address and an end-to-end encrypted message with a constant message length (messages are split or padded out to the specified length). The user performs the following recursive cryptographic operation (starting from the end of the message):

$$\begin{aligned} M_{N+1} &= M \\ M_i &= PK_i\left(A_{i+1}, D_{i+1}, r_{i+1}, M_{i+1}\right) \quad for\ i = N, N-1, \ldots, 1 \end{aligned} \tag{1}$$

   A_i is the address of MIXi , D_i is special information for the MIX station for application-specific use, and r_i is a random number, which is included to cause any attempt at decryption to be non-deterministic[16]. The user sends the message, e.g., $M_1 = PK_1(A_2, r_2, M_2)$, to the MIX node with the address A_1. Since the packet contains all the hop-by-hop routing information (source routing) as readable only by the specified MIX node, the message can be routed to the destination without revealing the complete address to any sub group of the MIX nodes.
3. **Cover Traffic**: All other users send similarly prepared packets, which can be real, dummy, or a mix of real and dummy messages, all with equal length, to the MIXes.

   MIX functionality: Each MIX node waits until *n* packets of equal length from *n* distinct users arrive. All the packets are compared to the previously processed packets and any duplicates will be deleted (in order to thwart replay attacks). Then the MIX node decrypts the packets with its private key, removes their random numbers, and outputs the updated packets in a different order than that in which they arrived (e.g., lexicographically sorted).

[16] Otherwise, the deterministic change of encoding does not prevent tracing messages through MIXes; an attacker would simply encrypt the output of the MIX with its public key and compare it with the input.

This mode of a MIX wherein it collects *n* packets from *n* distinct users before beginning operation is known as *batch* mode.

If all packets flow through all MIXes in a static order then it is called *mix-cascade*. In this case no address information for inter-MIX routing is needed. Otherwise, if the route for each packet is chosen individually, then this mode is called *mix-network* (or *mix-net*) [Chau81].

The general MIX functionality hides the relationship between a sender and a recipient of a message from everyone but the MIX and the sender of the message. This is because each MIX in the chain (cascade or network) hides the following features of a packet by retransmitting it:

M 1. the *appearance* (bit structure) of the packets are irreversibly changed through decryption (an attacker would have to guess the random number).

M 2. the packets can not be distinguished by their *length*, since all packets have the same length (agreement).

M 3. any *time correlation* (like FIFO) is avoided as the MIX reorders the packets.

The only remaining problem is that an attacker can participate in this protocol too. If he is able to contribute (*n*-1) of the *n* packets, then the one remaining packet is observable. This attack can be thwarted by a *loop back* scheme. The first step is that every MIX must know the sender anonymity set (i.e., the sending users). Thus, the first MIX can apply direct identification techniques to ensure the requirement of *n* distinct users. The following MIXes have to ensure this security functionality without weakening the anonymity (i.e., untraceability) of the former MIXes. One way to achieve this is to employ an anonymity-preserving *loop back* function: the MIX signs the received packets and performs a secure broadcast of this signature information back to the users in the anonymity set. Each user determines whether or not his message is included and transmits a secure *yes* or *no* reply. The MIX proceeds if it receives *yes* messages from all members of the anonymity set.

By applying this protocol the MIX method fulfills the CUVE requirements. The relationship between the sender and the recipient is hidden from an attacker as long as the attacker neither controls every MIX through which the message passes nor cooperates with the other $(n-1)$ senders.

6.3.1 Example

Assume that *A* wants to send a message *M* to *Z* over the MIX cascade (Figure 5). *A* must encrypt the message two times with the public keys, PK_i, of the respective MIXes, including the random numbers r_i:

$$PK_1(r_1, PK_2(r_2, Z, M)).$$

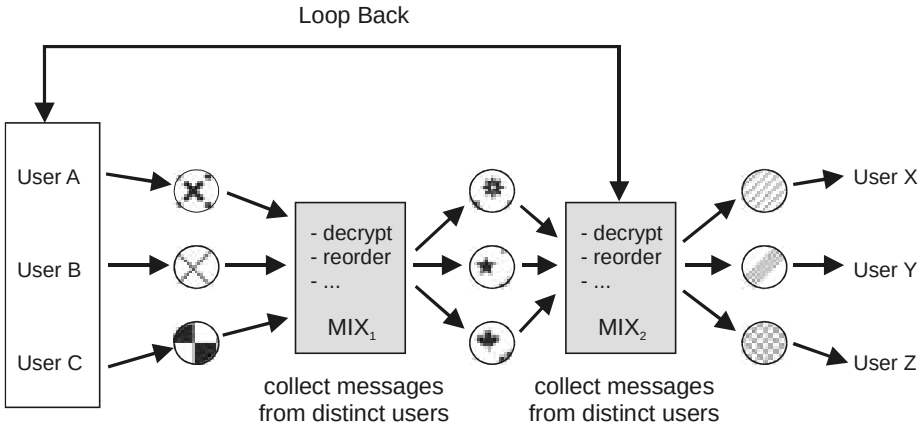

***Figure 5**: Cascade of two mixes*

6.3.2 Performance

MIXes can be operated synchronously or asynchronously, and the performance of these differs:

> **synchronous**: The network is slotted and all participants send a message in each slot. If the overall rate for real messages is λ then the mean number of real messages in a given time slot t_{slot} is $\lambda * t_{slot}$. Thus, $\eta = \lambda * t_{slot} / n$ for one MIX and $\lambda * t_{slot} / [n*(N+1)]$ for N MIXes (using reroutes). In view of the loop-backs, η has to be further adjusted to account for these $N*(2*n)$ messages (see Table 1).
> **asynchronous**: The mix-net waits till n messages are collected. We can assume here that all messages are real messages. Thus, η is just the number of reroutes and the loop-backs. But, the time efficiency is dependent on the mean packet rate λ and the performance of the system μ_D (see below and [Kesd01]).

6.3.2.1 Mean Delay using an a asynchronous MIX node

To estimate the delays incurred for messages passing through an asynchronous MIX node, we model a MIX node as a 2-server system (see Figure 6). The first server decrypts the messages and performs other tasks such as replay attack detection. The service times of this server are dependent only on server power and message size and therefore can be assumed to be constant. The appropriate model is then an $M/D/1$[17] server. The second server queues the messages until transmission. Because it does very little computing compared to the first server, it can be viewed independently even if both servers are in reality one computer.

[17] This notion is used in queuing systems and is known as Kendall notation [Klei75]. It describes the stochastic system characteristics of the form *A* / *S* / *s*: *A* stands for the description of the arrival process, (e.g., *M* stands for Markovian arrivals), *S* stands for the service time distributions, (e.g., *D* stands for Deterministic service), *s* stands for the number of servers in the system, and can be any integer larger than 0.

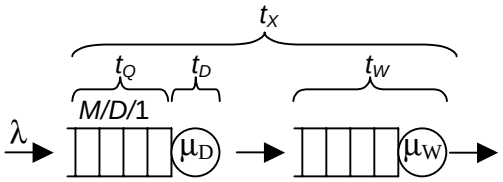

***Figure 6**: Anonymity providing node seen as a 2-server system*

We will now derive the mean time a message spends in the server, that is, the expectation of t_X . Let $1/\mu_D$ be the time the first server needs to process a message and λ the rate of the message arrival process. Assuming that the system is stable, i.e., $\rho=\lambda / \mu_D <1$, we can determine the expectation for the delay in the first server using the Pollaczek-Khintchine formula and Little's Theorem [Klei75, King90]:

$$E(t_Q + t_D) = \frac{1}{\mu_D}\left(1 + \frac{\rho}{2(1-\rho)}\right)$$

For a MIX with constant batch size *n*, the time spent in the second server is the time it needs to fill the batch. In a stable server system, the rate with which messages leave the server equals the arrival rate. Therefore, the mean interarrival time at the second server is $1/\lambda$, the expectation of the delay is

$$E(t_W) = \frac{n-1}{2\lambda},$$

and we get the resulting expression (see Figure 7)

$$E(t_X) = \frac{1}{\mu_D}\left(1 + \frac{\rho}{2(1-\rho)} + \frac{n-1}{2\rho}\right).$$

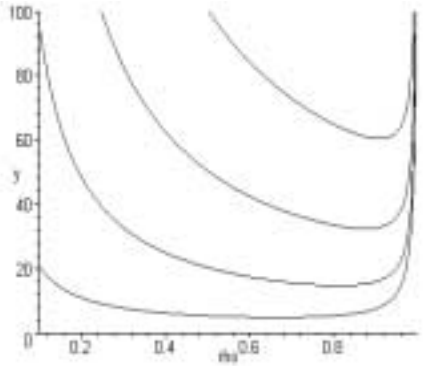

***Figure 7**: Mean total message delay $y=E(t_X)\cdot \mu_D$ in a MIX with constant different batch sizes n (bottom-up) n=5, n=20, n=50, and n=100 and (μ_D=1)*

The typical mean delay behavior of a MIX node has a hyperbola form. If the utilization is low ($\rho \to 0$) then the packets have to spend a long time in the second server waiting for *n* packets to be collected. If the utilization is increased then the waiting time decreases in

the second server but the service time increases in the first server due to the higher load situation. Between these hyperbola arms lies the practical operation range of a MIX node. As the batch size n increases, the second server needs more time to "collect" *n* packets. For *n*=50 the mean delay of the packets is high even for high utilization (ρ=0.4, 0.5, 0.6). For n→∞ the hyperbola "arms" converge to each other and to the border of ρ=1 (the stability border).

For batch size *n*≥100 the probability is low that a MIX node can send a packet in a reasonable time. The practical operation mode is in the range ρ=(0.6, …, 0.9). Thus, in open environments the best operational mode from the performance view is for *n*=20 or less.

6.4 Private Information Retrieval (PIR)

Private Information Retrieval [CGKS95], also known as *private message service* [CoBi95], assures that an unbounded attacker[18] is not able to discover the information that a user has requested (perfect protection from a database. The goal is to request exactly one datum that is stored in a remote memory cell of a server without revealing which datum is requested (protection of *interest data*). If the user requests all cells, then the interest data is easily protected using an efficient method suggested in [CoBi95]:

1. **Initialization**: *N* replicated servers, each with an identical copy of the database having *n* cells.
2. **Message Request**: In order to read a cell at a position *i*, the user generates a vector V_i that has a "1" at the i^{th} position and "0" otherwise.
3. **Cover Traffic**: To protect his interest data the user randomly generates a second vector V_{random} (i.e., for each i^{th} position of the vector a "1" or a "0" is chosen randomly) and XORs the two to form another vector:

   $$V_1 = V_i \oplus V_{random}$$

   Sender then transmits V_1 via a secure channel (i.e., an end-to-end encrypted channel) to server$_1$.

4. **Additional Requests**: Generate (*N*-2) further random vectors $V_2,\ldots,V_{N-1}$ and calculate the N^{th} vector:

   $$V_N = \sum_{i=1}^{N-1} V_i$$

   Send each random vector over a secure channel to the corresponding server.

5. **Server Algorithm**: If V_i has a "1" at position *j*, then the j^{th} cell is read. After XOR'ing all the read cells together to obtain one cell, the i^{th} server responds with the result C_i using again a secure channel:

[18] An unbounded attacker is an omnipresent attacker with unbounded computational capacity.

$$C_i = \sum_{if\ j^{th}\ position of\ V_i=1} cell_j \pmod 2$$

6.4.1 Example

Figure 8 shows an example of this method. Assume the cells are (X, Y, Z). If the user wants to read X then he creates the vector (1,0,0) and a random vector, say (1,0,1) and calculates his first request vector V_1=(0,0,1). Then he chooses another random vector, say V_2= (0,1,1), and calculates the third vector as V_3= (1,0,1) ⊕ (0,1,1) = (1,1,0). He sends the request vectors V_1, V_2, and V_3 end-to-end encrypted to the corresponding servers. Each server retrieves the cells corresponding to the 1's in its vector and XORs them to create one cell. The cells from all servers are returned (end-to-end encrypted) to the requestor who XORs all the cells and obtains the requested data cell X.

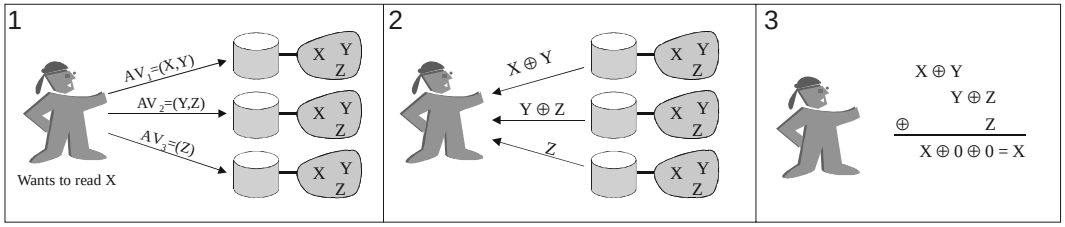

***Figure 7**: Basic concept of Private Information Retrieval and message service.*

6.4.2 Performance

All requests are sent to *N* replicated databases and each of them responds with one XOR'ed cell. Thus, η=1/*N* and the time efficiency is 1 because we only consider the network cost here.

Of course, the performance load of each database is increased by a factor proportional to the number of stored elements, since on average each database query requires *n*/2 read operations. If we assume the database can read an average of μ cells/sec, then τ=(*n*/2)/μ and η=2/*n*.

6.5 Comparing the Basic Techniques

All of the basic techniques presented provide an anonymity set of size *n* and, when implemented correctly, can provide perfect anonymity and unobservability [Pfit01].

Although MIXes can provide perfect protection, they are based upon public key cryptography, which is known to provide at best a complexity theoretic or probabilistic security [MeOV97]. The overall security of a system is determined by the weakest part. Thus, it is pointless to claim to provide more protection strength, or even to demand it, if the basic building blocks cannot provide that strength.

The other methods can use secret key cryptography and, thus, could conceivably use a one-time pad for encryption, thereby achieving perfect security. However, this is

impractical, so we assume, for all the methods, that the secure channel is at least initially protected with public key cryptography (e.g. [RiSA78]).

	Broadcast	DC-Net	PIR	mix-cascade	mix-net
System effectiveness η	$\frac{1}{n}$	$\frac{1}{2 \cdot n \cdot e^{G}}$	$\frac{1}{N}, \frac{2}{n}$	$\frac{\lambda t_{slot}}{n \cdot (N+1) + N \cdot (2 \cdot n)}$	$\frac{1}{(N+1) + 2 \cdot N}$
Time efficiency τ	1	$\frac{1}{2 \cdot n \cdot e^{G}}$	$1, \frac{n}{2\mu}$	$\frac{1}{(N+1) + 2 \cdot N}$	$\frac{1}{(N+1) + N\mu_s \frac{n-1}{2\lambda} + 2 \cdot N}$
Sender Anonymity	No	Yes	No	Yes	Yes
Recipient anonymity	Yes	No	No	Yes	Yes
Flexible access	No	No	Yes	No	Yes

***Table 1**: Performance Parameters*

Table 1 shows the performance parameters for the basic techniques. To evaluate the methods according their performance characteristics, we have to consider the requirements from section 3. There is no requirement for the required number of TTPs. We consider a system with N=3 TTPs to be sufficient to provide anonymity.

TR 4 requires that the number of participants *n* should be as large as possible. For all methods this maximum number has a different meaning that has to be considered when designing a system.

- *Broadcast*: if all packets would be broadcasted, the approach would be prohibitively expensive on an open network, e.g., the Internet. However, for broadcast networks like LANs, satellite networks, mobile communications, etc., this method applicable and introduces no additional cost. Thus, *n* could be much larger than 1000.
- *DC-Network*: Although there are useful solutions suggested for DC-Networks [DoOs97], they are only efficient on certain network topologies (e.g., rings).
- *MIX*: MIXes appear to be more practical for general circumstances. The above analysis for asynchronous communication shows that $n \leq 20$ is practical (for $\mu_D=1$).
- *PIR*: PIR presumes an existing replicated database infrastructure. Thus, the algorithm as suggested requires a new infrastructure. Even for a small number of replicated databases, e.g., N=3, the databases have to be designed carefully in order to manage the potentially huge number of query operations.

Therefore, we can conclude for broadcast, DC-Network, and PIR, that they can be applied for a large number of users if the network structure is chosen appropriately and the initialization phase realized efficiently.

The last row of the table compares the methods' spontaneous communication capability. A method provides a spontaneous communication if a user can send a message immediately (without a long initialization phase) whenever he has to send something. When considering this capability, only PIR and mix-net (asynchronous MIX) can provide this useful functionality.
When comparing these basic methods from a practicability point of view, the following drawbacks can be found with broadcast, DC-Network, and PIR:

1. Group function: the users are directly involved in building the group (e.g., DC-Network key graph)
2. The system effectiveness is proportional to 1/*n*, i.e., to send or receive a bit, *n* dummy bits has to be sent, with *n* being as large as possible.
3. To provide overall protection (i.e., sender and recipient anonymity), DC-Network and Broadcast have to be combined. PIR cannot protect the sender or the recipient, just their interests.

The MIX technique can provide sender and recipient anonymity and, as a result, can protect data about their interests as well:

1. Group function: senders control their own messages. Centralized MIX stations handle the bookkeeping associated with the group function.
2. Centralized MIX stations provide flexible access to an anonymity service and can therefore serve a large number of users: i.e., not all users have to be involved in each transmission.

The second property of the MIX may be advantageous from the practical point of view, but from security point of view it is risky since an intersection attack is possible (see Section 4). We will discuss this and other problems of MIXes by discussing the general problem of anonymity in open environments in Section 8.

7. Applications of the Basic Techniques for Closed Environments

The team of Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner, and their students, produced a large body of work regarding the theoretical foundation and implementation of anonymity techniques [PfPW01]. In this chapter, we will briefly discuss their results and contributions for unobservable communication.

7.1 Anonymity Providing Networks

As presented in Section 6.5 most of the techniques assume a certain network topology. In addition, given the number of dummy messages that may need to be sent in order to send one message, it is clear that such techniques can be best applied in closed environments. In his work A. Pfitzmann [Pfit90] suggested a hierarchical network structure and showed

that real time communication is possible for a narrowband network like Integrated Services Digital Networks (ISDN)[19]. As shown in Figure 9 the network is divided into two parts: local area and wide area.

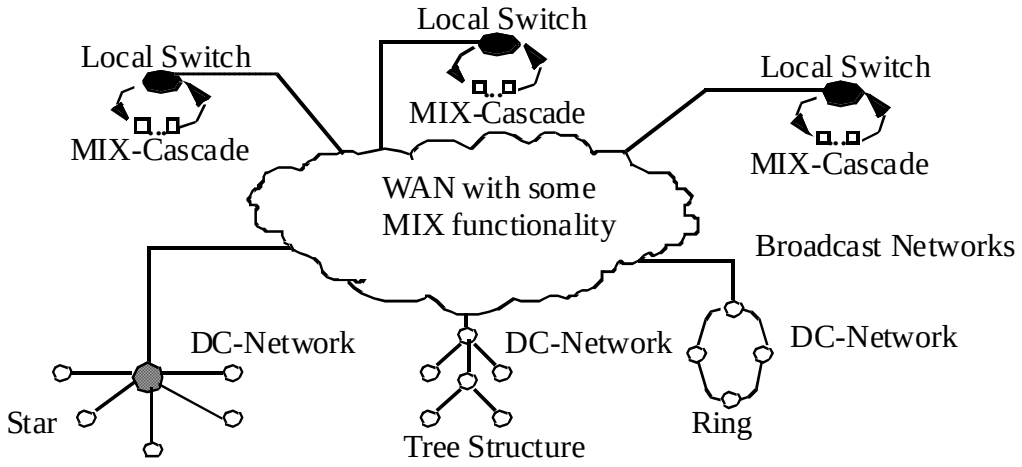

***Figure 8**: Switched/broadcast network providing anonymity*

To guarantee unlinkability and prevent intersection attacks, wherein an attacker can observe the persons in the different anonymity sets and can collect statistics about them, the following strategies can be used:

> **Constant Pattern**: Time slots are defined by the network and all users have to contribute a message in each slot.

Since the attacker should not gain any information by observing the network, all users have to behave absolutely the same and the method should treat every station indistinguishably (compare with [Pfit01], Section 5.4.6).
The design of the network in Figure 8 demonstrates the fundamental framework for the solution for closed environments[20]. It is now an engineering task to fulfill the technical requirements of the basic anonymity techniques. Further discussions of other security requirements such as secure payments, robustness and fault tolerance, etc., are also considered in [Pfit01]. Next we will sketch the solution for the requirement of real time communication for telephony.

7.2 Real-Time MIXes for ISDN

Mix-mediated techniques for efficient anonymous communication with real-time constraints were first described in [PfPW89, PfPW91] and [JMPP98]. They examined methods to anonymously transmit a continuous data stream as required for services like

[19] The political dimension of his work should be emphasized here, since at that time the German PTT planned the ISDN network with the aim to provide multimedia services (e.g., radio, TV, electronic news paper). Providing access to all these services and, thereby, having the ability to observe all citizens, it was an optimal monitoring tool that would violate all privacy regulations.
[20] To the best of our knowledge, this question is not yet fully answered for open environments.

telephony. The main reason for this work was to demonstrate anonymity-providing real-time communication over a bandwidth-limited channel.

The solution is a combination of the MIX technique, time slices (time slot), dummy traffic, and broadcast in the local exchange.

The main idea of the ISDN-MIXes is if all communication happens at the local exchange then the network can be synchronized in time slices. In each time slice all users send a special call setup message and data messages via *N* MIX stations arranged in a cascade.

Connection Setup: The setup message from each MIX station stores a symmetric key, K_i , and a virtual channel identifier in D_i (see Formula 1). Thus, only the user and the MIX station know this shared secret. All MIX stations store this vector for use in the next time slice for message processing.

Message processing: the user encrypts the message with the symmetric key, *Ki*, and adds the appropriate channel identifier recursively, i.e., starting with the last MIX up to the first MIX[21] . Upon receiving a message the MIX station can determine from the virtual channel identifier which symmetric key to use for decryption, decrypt the message, and forward the result.

In principle if only local communications are considered, then the above technique would provide total anonymity (sending, receiving, and relationship), with the assumption that all transmitted messages will be broadcasted locally to all stations on the local exchange.

A complete solution to the original problem[22] requires additional techniques like anonymous receiving channels (see also untraceable return addresses [Chau81]) in order to enable global communication.

8. MIXes for Open Environments

Today, anonymity has become a hot topic on the Internet, as illustrated by several recent publications and implementations [e.g. BeFK01, Fede01, Free00, Comm99, Cott01, GuTs96, IEEE98, KeEB98, Mart99, Raym01, ReSG98, ReRu98, Zero01]. However, as shown in the previous sections, anonymity is not a new topic at all. While the main goal of the former, mostly theoretical, works was to provide full, or even perfect, protection, the focus of current research has moved towards practical anonymity.
Of the suggested techniques, the MIX concept is the most often used. As shown in previous sections, centralized MIX stations can provide flexible access to an anonymity service and can therefore serve a large number of users.
Unfortunately, not all of the features of the MIX method can be directly used in an application. Therefore most new works and deployments use a modified approach: In open environments a specific topology for the underlying network cannot be assumed. Furthermore, it cannot be assumed that the network is slotted in time intervals or that all users will necessarily participate in the application. Thus, the following strategies can be found in the literature:

[21] This is the same recursive encryption idea as in the classical MIX method, only performed with symmetric keys.
[22] All users send and receive in each time slice, therefore the network can even not determine whether a message is "really" sent or received.

Strategy A. Only users who want to send a message are allowed into the anonymity system. The anonymity system will integrate users subject to the network capacity and maintaining the anonymity requirements. These users agree to participate for a specified minimum period, during which they will send real or dummy messages.

Strategy B. The anonymity size of the technique continuously increases over time. This is a theoretical assumption of these techniques. When implemented, these techniques can be more time consuming.

Both strategies have the goal to make the analysis for the attacker (intersection attack) as hard as possible. Unfortunately, at this time, there has been no mathematical basis presented to determine if the measures are sufficient.

8.1 Review of the Anonymity Requirements and the Suggested Techniques

Several anonymity services[23] are available today on the Internet which are based upon the MIX method: MIXmaster [Cott01], BABEL-MIX [GuTs96], Onion Routing [ReSG98], and Web MIX [BeFK01]. Unfortunately, none of the existing systems provides protection against the omnipresent attacker model (see also [BeFK01, Raym01]).

Even they cannot provide complete protection against the omnipresent attacker model, the investigations are still important in reference to the general requirements:

1. Ensuring the group functionality: In a global environment with millions of potential users, the task of building a secure group is very difficult. To build a secure group each MIX has to ensure that all packets are sent from distinct senders. One solution for this task is to use a Public Key Infrastructure (PKI).
2. Increasing the trust:
   - The Internet is an open network, thus everyone can take part in it. It is much easier in this environment for an attacker to find and organize (n-1) corrupt users if n is small. Therefore, n, the anonymity size, should be as large as possible (Requirement TR 4).
   - The security of the technique should not be based upon the trust of one specific node in the Internet. To increase the trust, several mixes should be used and each should be transparent to the user (TR 2).

It should be emphasized here that for the distribution of the public keys of the MIXes, a full PKI is not required. A minimal PKI only including the MIX nodes would be sufficient. However, the public keys of the MIX nodes could also be simply published in

[23] We do not discuss other approaches like CROWDS here [ReRu98], since such techniques provide protection against a weaker attacker model. This does not mean, however, that the techniques presented in this chapter automatically provide more protection.

newspapers or journals (secure broadcast using a reliable medium) and controlled by the owners (the MIX nodes).

All the MIX suggestions rely on the same scheme as suggested from D. Chaum in that they apply the same cryptographic function (see Formula 1). The only difference between them is how they build the anonymity set, increase the anonymity size (requirement TR 2), ensure the *CUVE* requirements are met, and increase transparency of the MIXes. Thus, we will focus on techniques that employ these extensions to the classical MIX.

8.2 MIXmaster

The MIXmaster uses a pool instead a batch; each incoming message is added to the pool and one of the pooled messages is randomly selected and forwarded.

Algorithm: Always maintain n packets in the pool. If a new packet arrives then choose randomly one of the n+1 packets and forward it.

The advantage of the above algorithm is that an attacker can never be sure whether or not a message received by the MIXmaster has been forwarded. This comes at the cost of longer mean delays and if the message is not delivered, the sender cannot determine whether the message is in the pool or has been attacked.
Therefore, the anonymity size increases with each arriving packet. However, the drawback is that the MIXmaster is not transparent to the user (contradicting requirements TR 2 and TR 4). Because the MIXmaster cannot ensure that the received packets are from distinct users, it provides little protection against attacker A2 (see [Kesd01]) and no protection against attacker A3.
To estimate the longer mean delays introduced by MIXmaster, the same model (2-server systems, see Section 6.3.2.1) can be applied here. Let Y be a random variable describing the number of messages that must have arrived before a specific message is sent. Since this would entail independent experiments with two outcomes, Y is a geometrically distributed random variable with the probability density function:

$$f_Y(i) = \frac{1}{n+1} \cdot \left(\frac{n}{n+1} \right)^m$$

The mean delay of a packet, assuming that m messages previously arrived, is: $E(W \mid Y=m) = m/\lambda$. Using the theorem of total probability the general mean delay is:

$$E(W) = \sum_{m=1}^{\infty} E(W \mid Y = m) \cdot f_Y(m) = \frac{1}{\lambda} \sum_{m=1}^{\infty} m \cdot f_Y(m) = \frac{1}{\lambda} E(Y) = \frac{n}{\lambda} = \frac{1}{\mu_D} \frac{n}{\rho}$$

Thus, the mean delay for a pool of constant size n is:

$$E(t) = \frac{1}{\mu_D} \left(1 + \frac{\rho}{2(1-\rho)} + \frac{n}{\rho} \right)$$

The expected delay is twice as long as the classical MIX node.

8.3 BABEL and T-MIX

BABEL MIX operates like the classical MIX in batch mode without ensuring the group function. That is, it collects a number of packets and applies the MIX technique. Several "enhancements" are made in this work to decrease the "guess factor" of the attacker, which is the ability of an adversary to match messages passing through the BABEL MIX with their original senders. A BABEL MIX :

1. randomly decides to send the packet via other BABEL MIXes (a *detour*) before it forwards to the specified addressee.
2. may randomly delays a message until the next batch.
3. can guarantee a maximum transmission time using *interval batching*: if in given interval *T* not enough real packets could be collected, then dummy messages are supplied and the batch is sent.

The detours do not increase the anonymity set if the attacker model is not changed (i.e., we assume that at least one of the used MIXes is honest). The detours just delay non-transparently the message to the sender. The random delay of the real message can be determined by a similar measure to that shown in the above analysis of the MIXmaster approach.

Interval batching is the most interesting measure, because it is transparent for the user. The worst case waiting time is *T* time units and the most important feature is that the security is also enhanced:

> The probability of a message being successfully attacked by the attacker models A1 and A2 is simply the probability that no other real message arrives in the same interval. If the number of real messages is assumed to be Poisson distributed with the mean rate λ, then the probability of successful attack is $\exp(-\lambda T)$ against attacker models A1 and A2 [Ross72]. Increasing or deceasing T will have an exponential effect on this probability.

Because this simple measure has so many good features, we call it as T-MIX [Kesd99]:

Algorithm: A T-MIX operates with a fixed time interval of length T between the outputs of two batches.

Obviously, the expectation of t_W (see Figure 6) for a MIX with constant time intervals of length *T* is *T*/2 and, thus, the mean total delay according to the above analysis is:

$$E(t_X) = \frac{1}{\mu_D}\left(1 + \frac{\rho}{2(1-\rho)}\right) + \frac{T}{2}$$

8.4 Onion Routing and WEB MIXes

Onion Routing[24] and WEB MIXes are designed for real time communication and are very similar to ISDN MIXes. They have two phases of communication like the ISDN MIXes: connection establishment and data sending phase. The performance characteristic is directly comparable to MIX cascade with synchronous sending.

In order to guarantee sufficient cover traffic, Onion Routing uses dummy traffic between the routers. Due to the fact that the endpoints of the network are not covered by this measure, the traffic of a particular user can still be linked at the endpoints. Furthermore, in the data exchange phase a correlation of a message by length is possible, as the system allows messages of individual lengths.

WEB MIXes use end-to-end dummy traffic, constant packet lengths, and other measures fulfilling the MIX requirements M1, M2 and M3 as required in [Chau81, Pfit90]. Of course, there are implementation differences, since ISDN MIXes are proposed for closed environments. The main differences are:

1. WEB MIXes use strategy A defined earlier.
2. A PKI is assumed so that a secure ticketing system can be applied (similar to anonymous e-cash [Bran99]) to ensure that the *CUVE* requirements are met.

This method, in combination with a time protocol, shifts the main costs from the message transmission phase to the initialization phase of the application [BeFK01]. The protocol enables a user to anonymously buy a ticket with some time restriction from the MIX node in order to use it at the specified time:

1. **Initialization 1**: all users establish a confidentiality and integrity guaranteeing connection to each MIX and an authentication procedure between MIX and user is applied.
2. **Initialization 2**: the user requests an anonymous ticket for a batch that will be processed at time *t*. The procedure is that the usersends a blinded ticket[25] to the MIX for signing. The MIX signs this ticket with a key pair that is only valid for the specified time *t*.
3. **Initialization 3**: the user un-blinds the ticket and can use the ticket at the specified time to access the anonymity service of the considered MIX by encoding the ticket recursively into the message as part of *D* (see Formula 1)).
4. **Application phase**: the MIX receives the message, decrypts it with the private key, and checks the ticket with the special private key. The MIX cannot link the ticket to the user because of the blinding factor.

[24] A company named Zero Knowledge suggested scheme similar to Onion Routing call the Freedom Network [Zero01].
[25] Blinded signatures are suggested first in [Chau85] and enable anonymous communication systems.

Since the ticket can only be used for a specified batch at a given time *t*, the technique ensures the CUVE requirement by not allowing the user to have multiple tickets for a time *t*. After step 2 above, the MIX station will have established an anonymity group by signing the tickets for different users. In the application phase (step 4) the MIX station has only to check the tickets for validity.

The major difference between ISDN MIXes and WEB MIXes is that it is possible that no one, or only one person, buys a ticket for a batch (no traffic guarantee). Even if the MIX ensures that there are at least *n* distinct users requesting tickets, it cannot be sure that (*n*-1) of them are not collaborating with the attacker.

8.5 SG-MIXes

Most of the previous work did not address the CUVE requirements. The WEB MIXes did address them, but assumed the existence of a PKI. Unfortunately, no generally available PKI yet exists. This is the starting point of the SG-MIX suggestion [KeEB98]. An SG-MIX operates in the same way as a classical MIX, but does not collect a fixed number of messages before sending. The suggested solution uses a time protocol and the goal is to provide the group functionality with a given probability (probabilistic approach):

1. Sender A selects *N* SG-MIXes to be used with equal probability.
2. He calculates for every node *i* a time window $\left(TS^{\min}, TS^{\max}\right)_i$ and selects a random delay time T_i from an exponential distribution having a suitable parameter μ (defined later). This information is appended to the packet as general information *D* before encrypting it with the SG-MIX's public key (see Formula 1).
3. The SG-MIX i extracts $\left(TS^{\min}, TS^{\max}\right)_i$ and T_i after decryption. If the arriving time of the packet is earlier or later than that given by the time window, the message will be discarded.
4. After the expiration of the maximum time in the window $\left(TS^{\min}, TS^{\max}\right)_i$, SG-MIX i waits until T_i units of time have elapsed and forwards the packet to the next hop or its final destination.

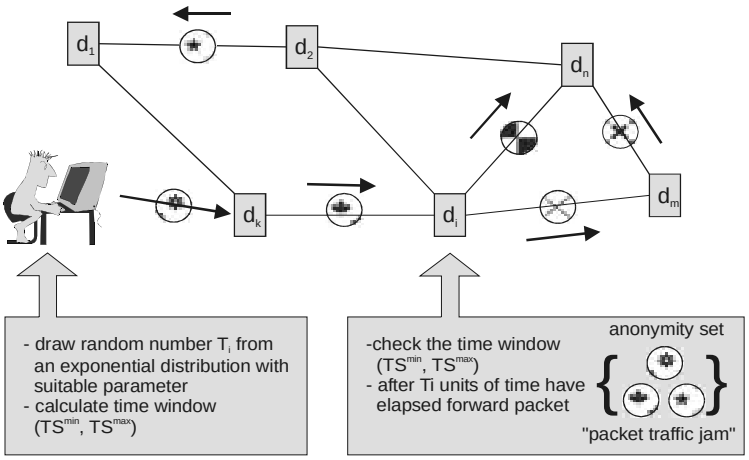

***Figure 9**: SG-MIXes*

Selecting the SG-MIX nodes with equal probability guarantees that a user does not favor certain SG-MIXes over others, which would enable an analysis attack. Furthermore, the anonymity size will always be maximal with a high probability.

8.5.1 The Security of the SG-MIX

The security of a SG-MIX does not rely on shuffling a batch of messages, but on delaying each message individually and independently by a random amount of time. If the delay times are individually drawn from the same exponential distribution with the same parameter μ, the knowledge of the time that a specific message arrived at the SG-MIX node will not help the attacker in identifying the corresponding outgoing message as long as there is at least one other message in the queue at some time during the delay. If n messages are in the queue, it is equally probable for any one of them to be sent next, regardless of their arrival times. Therefore, an attacker can correlate the arrival and departure of a message only if during the whole delay time no other message is in the queue.
The resulting probability that an arbitrary message can be tracked by attacker models A1 and A2 is given[26] by

$$P(success) = \frac{e^{-\lambda/\mu}}{1+\lambda/\mu} \quad ,$$

with λ denoting the rate of message arrivals. Let us consider an example: Assume a SG-MIX node with a mean arrival rate λ = 10 packets/s and parameter μ = 0.2 packets/s, which is a mean delay of 5 seconds. Then the probability of an arriving packet finding the server queue empty is $e^{-50} \approx 1.9 \cdot 10^{-22}$.

8.5.2 (n-1)-Attack

In order to provide probabilistic anonymity a SG-MIX must be able to fend off blocking attacks (attacker A3). When running such an attack the intruder must delay all incoming

[26] The complete derivation of the formula can be found in [Kesd99].

data packets for a certain amount of time in order to ”flush“ the SG-MIX. Therefore, we introduce the time stamps $(TS^{\min}, TS^{\max})$ to detect when an incoming data packet is delayed and then discard such packets. This prevents blocking attacks. The SG-MIX technique allows the calculation of the time windows very accurately, as the user knows the time a message will be delayed in advance.

We define for the pair of time stamps of node i, $(TS^{\min}, TS^{\max})_i$, the time window Δt_i during which a packet must arrive at SG-MIX i. If a Δt value is given, then the success probability of a blocking attack is:

$$P(success) = \exp\left(\frac{-\lambda e^{-\mu \Delta t}}{\mu}\right).$$

Obviously, when Δt is given, a linear decrease of μ leads to an exponentially decreasing probability of a successful blocking attack. The only successful attack is if the adversary blocks the incoming messages of all SG-MIXes for a long time period before the attacked message arrives. This will work due to not knowing which SG-MIX node would be selected from the user. This is usually impossible to do ”on demand“ and, in any case, would block the whole network, i.e., result in the loss of many messages due to time-outs, which would surely not go undetected.

8.5.3 Detection of Packet Loss

The SG-MIX protocol allows the sender to determine an accurate arrival time for each message. SG-MIXes or Trusted Third Parties (TTP) can use this feature to detect blocking attacks [GuTs96]. For this purpose they send a message via an arbitrary path through the SG-MIX network to themselves using the SG-MIX protocol. If the related packets arrive within the calculated time period, the TTP can assume that there are currently no blocking attacks going on. If the packets arrive significantly later than expected or do not arrive at all, this could be an indication for an ongoing blocking attack.
The keep-alive timer (i.e., timeout parameter) for such a feedback control mechanism must be chosen very carefully. If the parameter is chosen too small, then every variation of the transmission delay will result in an alarm (false positive). If the parameter is chosen too large, blocking attacks will remain undetected (false negatives).

8.5.4 Anonymity Size of the SG-MIXes

The size of the anonymity set of the SG-MIXes is obviously zero when the an SG-MIX has no packet in its queue (idle time). Otherwise, the same argument can be used as for the MIXmaster solution: the attacker does not know if a packet has been forwarded or not, and has to consider this packet in the set of the anonymity set until the queue is empty. Thus, the anonymity set U produced by a SG-MIX for a single message M is composed of the recipients of the messages already present at the SG-MIX at the arrival

of M and the messages which are received by the SG-MIX during the same busy period. The mean size of the anonymity set[27] is:

$$E(|U|) = \frac{\lambda}{\mu} + \frac{e^{\lambda/\mu} + 1}{2}.$$

Hence, we can deduce that the size of the anonymity set will always be maximal with high probability. For example, choosing the same parameters for λ and μ as before the size is more than $e^{50}/2$.

8.5.5 Performance

To calculate the mean delay we again use the 2-server model (see Figure 6). The first server is same as the classical MIX and the mean delay in the second server is simply the expectation of the chosen exponential distribution $1/\mu_W$. This results in:

$$E(t_X) = \frac{1}{\mu_D}\left(1 + \frac{\rho}{2(1-\rho)}\right) + \frac{1}{\mu_W}.$$

While these average delay times are of the same order of magnitude for the methods (asynchronous MIX, MIXmaster, T MIX and SG MIX), it is important to note the differences from the practicality point of view: SG-MIX is the only method where the user knows exactly how much delay the message will suffer, apart from the usually rather small network and queuing delays. This knowledge is important not only to compute the time windows mentioned in the previous section, but also to calculate turn-around times needed for congestion control in network protocols.

8.6 Comparing the Techniques

Table 2 shows the comparison of the techniques with respect to the requirements. Again, none of the suggested techniques can guarantee the privacy requirement of unlinkability on the Internet. This is only possible in closed environments. However, the underlying cryptography provides at most probabilistic security, thus it would be sufficient if this could also be shown for unlinkability. A good starting point to increase unlinkability is to increase the size of the anonymity set (requirement TR 4) as suggested in MIXmaster and SG-MIX, without losing the transparency (Request TR 2). Web-MIXes suggest the use of end-to-end dummy messages.

The second problem is the well known (*n*-1) attack (variants are flooding, trickle attack etc.), wherein the attacker is able to contribute the cover material ((*n*-1) of the *n* overall messages). For this a secure group function is needed and can only be provided by SG-MIXes or by Web MIXes if PKI would exist.

[27] The complete derivation of the formula can be found in [Kesd99].

	MIX master	T-MIX	Onion	Web MIX	SG-MIX
Req. TR 2:	No	Yes	Yes	Yes	Yes
Req. TR 4:	Yes	No	No	No	Yes
Ensuring Group Function	No	No	No	Yes (PKI)	Yes (statistical)
Req. PR 2:	Anonymity size (TR 4)	No	No	End-to-end dummy messages	Anonymity size (TR 4)
System effectiveness η	$\frac{1}{N+1}$	$\frac{1}{N+1}$	$\frac{\lambda t}{(N+1)\cdot n}$	$\frac{\lambda t_{slot}}{(N+1)\cdot n}$	$\frac{1}{N+1}$
Time efficiency τ	$\frac{1}{(N+1)+N\mu_s \frac{n}{\lambda}}$	$\frac{1}{N+1+NT/2}$	$\frac{1}{N+1}$	$\frac{1}{N+1}$	$\frac{1}{(N+1)+N\mu_s \frac{1}{\mu_w}}$

Table 2: Performance Parameters

In order to send a real message the Onion Routing and WEB MIXes systems have to send a huge amount of dummy messages, but they can realize real time communication. Chaum [Chau81] suggested MIXes for applications like e-mail and it is shown by Pfitzmann [PfPW91] that they can also be used for real-time applications if there is a sufficient amount of dummy traffic. In the Internet the amount of dummy messages should be reduced, otherwise the technique is prohibitively expensive. The newer techniques supporting real-time communication address this problem by restricting the number of involved users and the time in which they have to send dummy messages and have to deal with the intersection attack.

9. Future

The privacy and anonymity area will continue to be controversial and the conflict between the demand for personal privacy and the goal of overall security will remain in the future. Therefore, the only productive approach is an iterative one: the problems are addressed through building solutions, evaluating their acceptability and usability, refining the requirements, and beginning again. The sociological and cultural questions of privacy and anonymity are also very important since they specify some of the technical requirements. If, for example, general privacy and anonymity is requested, then this can be best fulfilled in a closed environment as shown in section 7.

Privacy and anonymity researchers have traditionally had only three protection levels as their goals: unconditional, probabilistic, and ad-hoc protection. These very closely parallel some of the protection levels defined for cryptographic systems, but are only a subset of those cryptographic protection levels [MeOV97]. Privacy and anonymity research should define these missing additional levels in order to reach a more fine-grained set of practical protection levels. The addition of these levels, and the associated evaluation techniques, should be the focus of future work.

The technical evaluations in this paper are characterized in terms of time and transmission overhead. If we are willing to accept a certain amount of transmission overhead, then we are able to reduce the time overhead and may increase the protection level and vice versa. We discussed these trade-offs for several approaches. In the future the ratio between these two parameters τ and η will be more interesting, since it relates how efficient the techniques uses a transmission overhead and achieves better time efficiency and protection level.

A fundamental requirement of anonymity techniques is that the attacker cannot take part in an anonymity technique (n-1) times and, therefore, control the cover traffic. In order to guarantee this property most of the techniques require identification and authentication of the users. However, in addition to the lack of any existing means of obtaining user information for this requirement (e.g., no deployed PKI), it is also a contradicting requirement: to anonymize a person requires the use of a strong identification procedure. This property may discourage persons from using the techniques. New techniques are needed that do not have contradiction [KeEB98].

A mathematical foundation for pseudo-unlinkability has not yet been given. Such a result is especially applicable to open environments.

In this paper, we discussed all the techniques in terms of the omnipresent attacker model, which assumes an attacker who can observe users without any time and space restriction. Of course, such a powerful attacker leads to techniques providing a maximum protection level in exchange for unfortunately high costs. In the future, realistic, but weaker, attacker models are needed, both to reduce the network costs and to provide a closer match to realistic situations.

10. Conclusion

This paper has addressed the technical challenges of network anonymity and privacy. First we have provided the formal terminology and have used it to define the anonymity task and to describe the techniques. We then presented and discussed the various research efforts up to the present day. Thus, this paper discusses the important results of the last twenty-plus years of research and their effects upon the underlying networks. We finished by adding our own view of the directions of future anonymity techniques.

Clearly the conflicting goals of anonymity and good performance pose a complex problem that can only be solved by controlling the amount of traffic. This traffic has to be ensured against changes by an attacker, should have the maximum effect on anonymity, and should enable real time communication. Since all requirements cannot be fulfilled at the same time, a finer-grained design that would enable balancing the goals in a practical manner is necessary.

Although this paper focuses on technical issues of privacy, we do not want to neglect the social and cultural parts of the subject. Privacy through anonymity is freedom for all people to act freely in the virtual world. For all those people who claim that this freedom should not be given, because of the security risk that terrorists and criminals will misuse it, we can only refer to Benjamin Franklin, who said 240 years ago: *The man who trades freedom for security does not deserve nor will he ever receive either*.

11. Literature

[BeFK01] O. Berthold, H. Federrath, S. Köpsell: *Web MIXes: A System for Anonymous and Unobservable Internet Access*, International Workshop on Design Issues in Anonymity and Unobservability, Berkley, 2009 LNCS, Springer-Verlag, 2001.

[Bran99] S. Brands: *Rethinking Public Key Infrastructure and Digital Certificates – Building in Privacy*, PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherland, 1999

[Brin99] D. Brin: *The Transparent Society: Will Technology Force Us to Choose Between Privacy and Freedom?*, Perseus Books, Reading MA, 1999, ISBN: 0738201448

[Chai74] G. J. Chaitin: *Information-theoretic computational complexity*, IEEE Transactions on Information Theory IT-20 (1974), pp. 10-15.

[Chau81] D. L. Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, In: Comm. ACM, Feb. 1981, Vol. 24, No. 2, pp 84-88.

[Chau85] D. L. Chaum: *Security without identification: Transaction systems to make big brother obsolete*, CACM, 28(10), 1985.

[Chau88] D. L. Chaum: *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*, Journal Cryptology, Vol. 1, No. 1, Springer-Verlag, 1988, pp 65-75.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan: *Private Information Retrieval*, Proc. of the 36th Annual IEEE symposium Foundations of Computer Science, 1995.

[CoBi95] D. A. Cooper, K. P. Birman: *Preserving Privacy in a Network of Mobile Computers*, 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995.

[Comm99] Communications of the ACM: *Internet Privacy: The Quest for Anonymity*, vol. 42, num. 2, February 1999.

[Cott01] L. Cottrell: *MIXmaster and Remailer Attacks*, http://www.obscura.com/~loki/remailer/remailer-essay.html, 2001.

[DaPr84] D.W. Davies, W. L. Price: *Security for Computer Networks*, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer, John Wiley & Sons, Chichester, New York, 1984.

[Denn82] D. E. Denning: *Cryptography and Data Security*, Addison-Wesley, 1982.

[DiHe76] W. Diffie, M. E. Hellman: *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 10, Juni 1977.

[DoOs97] S. Dolev, R. Ostrovsky: *Efficient anonymous multicast and reception*, Advances in Cryptology – EUROCRYPT'97, LNCS, 1997

[FaLa75] D. J. Farber, K. C. Larson: *Network Security Via Dynamic Process Renaming, Fourth Data Communication Symposium*, 7-9 Oct. 1975, Quebec City, Canada.

[Fede01] H. Federrath (Ed.): *Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, LNCS 2009, Springer-Verlag 2001.

[Free00] M. J. Freedman: *Design and Analysis of an Anonymous Communication Channel for the Free Haven Project*, BACHELOR THESES, MIT 2000, http://theory.lcs.mit.edu/~cis/cis-theses.html.

[FiHu01] S. Fischer-Hubner: *It-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms*, (LNCS 1958), Springer Verlag; ISBN: 3540421424.

[GuTs96] C. Gülcü, G. Tsudik: *Mixing Email with Babel*, Proc. Symposium on Network and Distributed System Security, San Diego, IEEE Comput. Soc. Press, 1996, pp 2-16.

[Gutt00] http://www.cs.auckland.ac.nz/~pgut001/tutorial/index.html

[IEEE98] IEEE Journal on Selected Areas: *Copyright and Privacy Protection*, vol. 16. no. 4, May 1998.

[JMPP98] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, M. Waidner: *Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol*, IEEE Journal on Selected Areas in Communications, 1998.

[Karg77] P. A. Karger: *Non-Discretionary Access Control for Decentralized Computing Systems*, Master Thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Camebridge, Massachusetts 02139, Mai 1977, Report MIT/LCS/TR-179.

[Kesd99] D. Kesdogan: *Privacy im Internet*, Vieweg Verlag, ISBN: 3-528-05731-9 (in German).

[Kesd01] D. Kesdogan: *Evaluation of Anonymity Providing Techniques using Queuning Theory*, The 26th Annual IEEE Conference on Local Computer Networks (LCN 2001), November 15-16, 2001, Tampa, Florida

[KeEB98] D. Kesdogan, J. Egner, R. Büschkes: *Stop-And-Go-MIXes Providing Probabilistic Anonymity in an Open System*, Proc. 2nd Workshop on Information Hiding (IHW98), LNCS 1525, Springer-Verlag 1998.

[King90] P. J. B. King: *Computer and Communication Systems Performance Modelling*, Prentice Hall, 1990.

[Klei75] L. Kleinrock: *Queuing Systems*, Vol. I: Theory. John Wiley & Sons, 1975.

[Mart99] D. Martin Jr.: *Local Anonymity in the Internet*, Ph.D. Dissertation, Boston University, 1999.

[MeOV97] A. J. Menezes, P. v. Oorschot, S. Vanstone: *Handbook of Applied Cryptography*, CRC Press 1997.

[Orwe90] G. Orwell: *1984*, New Amer Library Classics, 1990, ISBN: 0451524934

[Pfit90] A. Pfitzmann: *Dienstintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz*, IFB 234, Springer-Verlag, Heidelberg 1990 (in German).

[Pfit01] A. Pfitzmann: *Datensicherheit und Kryptographie*, TU Dresden, 2001, http://dud.inf.tu-dresden.de/~pfitza/DSuKrypt.html (in German).

[PfKö01] A. Pfitzmann, M. Köhntopp: *Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology*, IWDIAU, LNCS 2009, Springer-Verlag, 2001.

[PfWa85] A. Pfitzmann, M. Waidner: *Networks without user observability – design options*, Advances in Cryptology – Eurocrypt '85, 219 LNCS, Springer-Verlag, 1985.

[PfPW89] A. Pfitzmann, B. Pfitzmann, M. Waidner: *Telefon-MIXe: Schutz der Vermittlungsdaten für zwei 64-kbit/s-Duplexkanäle über den (2·64 +16)-kbit/s-Teilnehmeranschluß*, DuD 13/12, Vieweg-Verlag, 1989.

[PfPW89] A. Pfitzmann, B. Pfitzmann, M. Waidner: *Telefon-MIXe: Schutz der Vermittlungsdaten für zwei 64-kbit/s-Duplexkanäle über den (2·64 +16)-kbit/s-Teilnehmeranschluß*, Datenschutz und Datensicherung (DuD) 13/12, Vieweg-Verlag, 1989.

[PPW91] A. Pfitzmann, B. Pfitzmann, M. Waidner: *ISDN-MIXes – Untraceable Communication with Very Small Bandwidth Overhead*, Proc. Kommunikation in verteilten Systemen, IFB 267, Springer Verlag, 1991.

[PfPW01] A. Pfitzmann, B. Pfitzmann, M. Waidner et. al.: http://www.semper.org/sirene/, 2001

[RaSi93] C. Rackoff, D. R. Simon: *Cryptographic defense against traffic analysis*, In Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing, pp 672-681, May 1993.

[Raym01] J. F. Raymond: *Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems*, International Workshop on Design Issues in Anonymity and Unobservability, Berkley, 2009 LNCS, Springer-Verlag, 2001.

[ReSG98] M. G. Reed, P. F. Syverson, D. M. Goldschlag: *Anonymous Connections and Onion Routing*, IEEE Journal on Special Areas in Communications, 16(4):482-494, May 1998.

[ReRu98] M. K. Reiter, A. D. Rubin: *Crowds: Anonymity for Web Transactions*, ACM Transactions on Information and System Security, volume 1, pages 66-92, 1998.

[Ross72] S. M. Ross: *Introduction to Probability Models*, New York, Academic Press.

[RiSA78] R. L. Rivest, A. Shamir, L. Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystem*, Comm. ACM, Feb. 1978, Vol. 21, No. 2.

[Shan49] C. E. Shannon: *Communication Theory of Secrecy Systems*; The Bell System Technical Journal, Vol. 28, No. 4, Oktober 1949.

[Schn95] B. Schneier: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, 1995, ISBN: 0471117099

[Stin95] D. R. Stinson: *Cryptography Theory and Practice*, CRC Press, 1995.

[SySt99] P. F. Syverson, S. G. Stubblebine: *Group Principals and the Formalization of Anonymity*, FM'99, Vol. I, LNCS 1708, pp. 814-833, 1999

[Tane96] A. S. Tanenbaum: *Computer Networks*, Prentice-Hall, Englewood Cliffs, NJ., 1996.

[Waid89] M. Waidner: *Unconditional sender and recipient untraceability in spite of active arracks*, Eurocrypt '89, LNCS 434, Springer-Verlag, 1989.

[WaPf89] M. Waidner, B. Pfitzmann: *The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability*, Universität Karlsruhe 1989.

[Zero01] Zero-Knowledge-Systems, Inc.: *The Freedom Network Architecture*, http://www.freedom.net/ (2001)