Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems

Jean-François Raymond
Zero-Knowledge Systems, Inc.
jfr@zeroknowledge.com

19th December 2000

Abstract

We present the traffic analysis problem and expose the most important protocols, attacks and design issues. Afterwards, we propose directions for further research.
As we are mostly interested in efficient and practical Internet based protocols, most of the emphasis is placed on mix based constructions. The presentation is informal in that no complex definitions and proofs are presented, the aim being more to give a thorough introduction than to present deep new insights.

1 Introduction

Privacy is becoming a critical issue on the Internet. Polls constantly remind us that users feel that one of the most important barriers to using the Internet is the fear of having their privacy violated. Unfortunately, this isn't unjustified as marketers and national security agencies have been very aggressive in monitoring user activity [1].

Two things can happen as a result of this lack of privacy: either the Internet's popularity diminishes or, as seems more likely, the Internet becomes the most pervasive surveillance system ever. The problem studied in this text isn't a purely theoretic one, in fact some would argue that it is a crucial one to solve if the online world is to continue expanding and *improving*. In any case, from both theoretical and practical perspectives, it certainly deserves to receive much more attention than it has gotten so far.

1.1 Desirable Properties

Our goal is to protect users against *traffic analysis*. That is, we don't want an adversary that can monitor and/or compromise certain parts of the systems to be able to match a message sender with the recipient (sender-recipient matchings).

[1] See http://www.freedom.net and http://www.inf.tu-dresden.de/~hf2/anon for examples.

A related problem is that of *network unobservability* which attempts to hide all communication patterns. (how many, at what time and to whom/from whom messages are sent and received). Notice that network unobservability implies the ineffectiveness of traffic analysis.

Whereas message privacy can be obtained using encryption, it's much harder to protect sender and/or recipient privacy; especially in large open networks. The number of different assumptions and settings is huge which makes it difficult to define and reason about the problem in a rigorous manner.

As with many constructions in cryptography, there are efficiency, practicality/security tradeoffs to be made. For example, if efficiency and practicality weren't issues, we could broadcast messages in order to protect recipient privacy.

Notice that the problem definition isn't entirely trivial. We can't provide ”perfect” privacy since the number of possible senders and recipients is bounded. So, for example, if there are only two parties on the network, an attacker having access to this information can trivially determine who is communicating with whom . . . The best we can hope for is to make all possible sender-recipient matchings look equally likely. That is, the attacker's *view*[2]'s statistical distribution should be independent from the actual sender-recipient matchings. The protocol of subsection 2.2 has this strong property whereas those of subsection 2.3 usually don't. Unfortunately, there are no satisfactory definitions/methods providing a solid framework in which to analyze protocols that fall short of the optimal performance[3] and we usually need to rely on more or less ad-hoc arguments.

1.2 Overview

A concise literature review can be found in section 2. A comprehensive listing of attacks against mix-networks is presented in section 3. Design issues related to large mix based networks are given in section 4. We propose directions for further research in section 5 and conclude in section 6.

2 Literature Review

Before delving more deeply into the problem, we briefly review some privacy protecting mechanisms. Although the link between these and sender-recipient privacy can be tenuous in certain instances, we believe that some of the ideas used in these techniques might be useful.

2.1 Related Problems

- *Secure Multi Party Computations (SMPC)[14] :* A group of users, each having a private input, want to securely compute a function of their private inputs. At the end of the protocol, all users should know only the value of the function. That is, each user will not have gained any information about the other users' private inputs apart from what can be deduced from the function's value.

[2]By *view*, we mean all the information available to the attacker.
[3]i.e. protocols in which traffic analysis can help in obtaining *some* non-trivial information about sender-recipient matchings.

- *Oblivious RAM [22]:* Code privacy can be protected by using a tamper resistant cryptographic processor. The protocol is such than an outside party looking at the memory accesses (reads and writes) can't gain any information about what is being computed and how it is being computed. The code's privacy is protected which could be useful to prevent reverse engineering and software piracy.

- *Private Information Retrieval(PIR) [10, 11, 12] :* A user privately queries one or many disjoint databases. By privately, we mean that the database(s) will not have any information about what element the user has queried.

- *Oblivious Transfers [36]:* This problem has many versions which are equivalent in that one implies the other. We mention a flavor which is related to PIRs and is referred to as 1 out of n oblivious transfer. These protocols have very similar properties as PIRs, the major difference being that the database privacy is also protected: the user doesn't gain any information about the other entries in the database.

- *Steganography [27] :* Steganography is the branch of information privacy that attempts to hide information within publicly observable data (e.g. using digital watermarking [42], subliminal channels [41], etc.).

2.2 Chaum's Dining-Cryptographer Networks (dc-nets)[7, 43]

The goal here is to have one participant anonymously broadcast a message. If the message is aimed at one user, the sender can encrypt the message by, for example, using an asymmetric crypto-system. Since the message is received by all parties, recipient anonymity is trivially maintained.

Let $P = \{P_1, P_2, \dots, P_n\}$ be the set of participants and let $(F, \oplus)$ be a finite Abelian (commutative) group (for example $(\mathbb{Z}_m, +)$) in which all computations will be carried out. The protocol goes as follows:

1. **Initialization:** Each participant securely shares secret keys (chosen at random from F) with *some* other participants. We denote the secret key shared by P_y and P_z by $K_{y,z} (= K_{z,y})$ and define the set G composed of all pairs (P_y, P_z) such that P_y and P_z share a secret key. Notice that if $(P_y, P_z) \in G$ then $(P_z, P_y) \in G$.

2. **Message Transmission:** In order to send a message M, P_i broadcasts:

$$M \oplus \sum_{\forall j \text{ s.t. } \{P_i, P_j\} \in G} sign(i-j) \cdot K_{i,j}$$

   Where $sign(x) = 1$ if $x > 1$ and -1 otherwise.

3. **"Noise" Transmission:** All other participants, P_j, broadcast:

$$\sum_{\forall k \text{ s.t. } \{P_j, P_k\} \in G} sign(j-k) \cdot K_{j,k}$$

4. **Computing the Message:** All interested participants can obtain M by adding ($\oplus$) all broadcasted messages. The fact that the sum of all broadcasted message equals M can be seen by noting that all terms except M cancel out because of $sign()$. (i.e. for each term of the form $sign(j-l) \cdot K_{j,l}$ we have $sign(l-j) \cdot K_{l,j} = sign(l-j) \cdot K_{j,l} = -sign(j-l) \cdot K_{j,l}$.)

In order to quantify the scheme's security, we define a graph having n vertices, labelled by $1, 2, \ldots, n$ (each representing a participant), with edges between nodes i and j if and only if P_i and P_j share a secret key. For example, if all participants share a secret key, the graph will be fully connected.

Fact 2.1 *If the graph obtained by removing the vertices corresponding to the participants controlled by an adversary is connected then the protocol protects sender anonymity. (Note that we assume that the broadcasted values are known to the attacker.)*

This analysis is tight in that if the graph isn't connected, an attacker can determine from which of the disconnected parts of the graph the sender is from.

2.2.1 Drawbacks

Unfortunately, the protocol has serious drawbacks:

1. *Secure and reliable broadcast channel:* To protect against active adversaries[4], we need to rely on physical devices because secure and reliable broadcast mechanisms can't be constructed by algorithmic means. This problem can be partially fixed by the technique of Waidner [43] that uses a fail-stop broadcast.

2. *Channel jamming:* If many participants try to send a message at the same time, all is lost as the sum of all broadcasted values will equal the sum of the messages (e.g. $M_1 \oplus M_2 \oplus \ldots \oplus M_j$). An even bigger problem is if a participant acts maliciously and deliberately sends channel jamming messages; this allows him to compute the legitimate message while the other users can't gain any information.

3. *Number of messages:* Every user needs to participate every time a message is broadcasted which is a problem both in terms of efficiency and robustness. This is an unrealistic constraint in large networks.

4. *Shared secret keys:* The number of keys to share could be too large for practical purposes (need a new key for each transmission). Note that pseudo-random numbers can be used as keys to alleviate this problem. The fact that many users need to share secret keys with (possibly many) participants is also a serious problem in terms of practicality.

Despite these problems, dc-nets are useful in many situations (e.g [19]), and, as far as efficiency is concerned, for certain underlying network topologies (e.g. rings), the complexity is acceptable. Also

[4]adversaries capable of adding and removing messages from the communication channels.

note that dc-nets can be used in conjunction with other sender-recipient privacy protecting mechanisms such as mix networks. For example, a certain number of users can transmit information to a mix network using a dc-net. In this setting, even if the mix network security is violated, the attacker can only ascertain that the sender of a given message is one of the parties using the dc-net[5].

2.3 Chaum's Mixes

A mix node is a processor that takes as input a certain number of messages which it modifies and outputs in a random order. The messages are modified and reordered in such a way that it is nearly impossible to correlate a message that ”comes in” with a message that ”goes out”. The mix nodes can be used to prevent traffic analysis in roughly the following manner:

1. The message will be sent through a series of mix nodes (a route), say $i_1, i_2, \ldots, i_d$. The user encrypts the message with node i_d's key, encrypts the result with node i_{d-1}'s key and so on with the remaining keys.

2. The mix nodes receive a certain number of these messages which they decrypt[6], randomly reorder and send to the next nodes in the routes.

Note that each mix node knows only the previous and next node in a received message's route. (The entry and exit node know the source (sender) and destination (recipient) of the message respectively.) Hence, unless the route only goes through a single node, compromising a mix node doesn't trivially enable an attacker to violate sender-recipient privacy.

2.3.1 Different Approaches to Route Selection

The route that a message will follow can be determined in a few ways:

- *Cascade [24, 23]:* The route can be constant, that is, it doesn't change. In this setting, the attacker knows the entry, exit and intermediate nodes. This kind of mix network is usually referred to as “mix-cascade”. Although they are easier to implement and manage, mix-cascades are *much* easier to traffic analyze.

- *Random Order:* The routes can also be chosen at random, that is, the user chooses $i_1, i_2, \ldots, i_d$ uniformly at random. This type of mix network is usually referred to as “mix-net”.

- *Other Methods:* One can think of many other ways of choosing routes, for example: A) part of the route could be fixed B) the route could be chosen at random from a set of pre-determined choices C) the route could be chosen at random subject to some restriction (e.g. mixes not all in the same legal jurisdiction).

In the following, we consider only mix-nets although some comments/attacks will apply to other route selection mechanisms.

[5]Assuming that the dc-net's security hasn't been compromised.

[6]They remove a layer of encryption.

2.3.2 Different Approaches to Flushing Mix Nodes

Many approaches to “flushing” messages can be used:

- *Message threshold:* The mix nodes wait until they receive a certain number of messages before ”releasing” all of them at the same time.
- *Message Pool [13]:* The flushing algorithm for mixmaster [13] has two parameters: the pool size n and the probability of sending p. The nodes wait until they have n messages in their pool at which time, they shuffle the messages and send each one with probability p (e.g. if $p = 1$ the scheme is identical to the message threshold approach). Note that unsent and newly received messages are placed in the pool.
- *Stop and Go [29]:* Kesdogan et al. give an interesting scheme in which messages wait random times at a nodes before being released (note that the waiting period is determined by the sender). In this setting, the attacker has a probability of success: If an empty node (i.e. one not currently processing any message) receives a message and does not receive another one before sending the decrypted message, the attacker can easily “follow” the message – routing the message through this node doesn’t “help”. Perhaps the most interesting contribution of this paper is that a statistical analysis is used to determine the probability of this happening.
- *Other :* There are many other reasonable algorithms for doing this, for example mix nodes could receive a random number of messages and output a constant number of them (using dummy messages to fill the gaps).

2.3.3 Mix Networks in Other Settings

Many researchers have studied how mixes can be used within existing networks such as ISDN, Internet and GSM (see for example [32, 33, 34, 31]).

Although these are very interesting papers, they don’t provide any deep insights about traffic analysis.

2.3.4 Robust Mixes

Very recently, researchers [1, 2, 17, 25, 30]

have invented robust mixes, in which messages are properly delivered to the recipient even if a certain number of mix nodes are *misbehaving.* Unfortunately, the constructions don’t seem practical for most real-world situations since a large number of the mix nodes, a bulletin board and a public key crypto-system are required.

We note that a mix network that works even if a certain number of nodes *are not forwarding the messages* was proposed in [33, 31]. This construction, although it doesn’t have many of the nice properties of robust mixes, can be very attractive as it works with classical mix networks (don’t need bulletin board, public key crypto-system).

2.4 Rackoff and Simon's analysis

In [39], Rackoff and Simon provide a solid theoretic framework in which we can reason about sender-recipient privacy. Using complex arguments about rapidly converging Markov processes, they are able to prove that an attacker can't successfully traffic analyze a specific type of mix network (with constrained user behavior). Unfortunately, the setting is of limited practical interest: it's synchronous, all participants need to send a message, mix-nodes process at most two messages at a time and the routes are constrained. Furthermore, although the constructions are efficient (from a complexity theorist's point of view), they do not seem amenable to real-world implementation. In spite of these shortcomings, their work provides the only known solid theoretic foundation for reasoning about traffic analysis.

2.5 Rubin and Reiter's Crowds

Rubin and Reiter propose an lighter weight alternative to mixes and dc-nets in [37, 38]. Their system can be seen as a P2P (peer-to-peer) relaying network in which all participants forward messages. The messages are forwarded to the final destination with probability p and to some other participants (chosen at random) with probability $1-p$. The authors provide a fairly detailed security analysis but, unfortunately, the system does not protect against very powerful adversaries and so we will not discuss it further.

3 Attacks

The attacks mentioned in this section aren't based on any specific implementation[7], instead, we give attacks that can be mounted on the high level descriptions of the schemes. We assume there are no implementation weaknesses, for example, we assume that messages coming in a mix node can't be correlated with a message going out (by a passive external adversary). Note that securely implementing cryptographic protocols is an extremely difficult task even for protocols that seem very simple like the Diffie-Hellman key exchange [18] and so will not be discussed as it is beyond the scope of this work.

In order to give a list of attacks, it is important to solidly define what assumption are made about the attacker's power. We consider the following attacker properties:

- *Internal-External:* An adversary can compromise communication mediums (external) and mix nodes, recipients and senders (internal).
- *Passive-Active:* An active adversary can arbitrarily modify the computations and messages (adding and deleting) whereas a passive adversary can only listen. For example, an external active adversary can remove and add messages from the wire(s) he controls and a passive internal adversary can easily correlate messages coming in a compromised node with messages going out (but can't modify them).
- *Static-Adaptive:* Static adversaries choose the resources they compromise before the protocol starts and can't change them once the protocol has started. Adaptive adversaries on the other

[7]For an attack on an RSA based implementation of a mix see [35].

hand are allowed to change the resources they control while the protocol is being executed. They can, for example, ”follow” messages.

An adversary can, of course, have any combination of these properties. For example, he could control a mix node in a passive manner and actively control wires. Note that there might be other relevant attacker properties (these are the ones usually considered in theoretical cryptography).

We warn the reader that the immunization mechanisms presented in the following subsections are by no means comprehensive and many details are omitted.

3.1 Brute Force Attack

The brute force attack is very instructive because it can help in determining how much, where and when to use dummy traffic. Dummy messages are messages that are sent through the network in order to complicate the attacker's task. On the Internet, mix network operators sometimes need to pay for each message and so we want to be sure the dummy messages have a good security to cost ratio.

The idea behind this attack is very simple: follow every possible path the message could have taken (*passive external adversary*). If the mix isn't well designed and the attacker is extremely lucky, he can link sender and recipient. In most cases however, the attacker will be able to construct a list of possible recipients.

We present the attack in a setting in which each mix node waits until it receives t messages before flushing them (i.e. sending all t messages). In addition, we assume that each message goes through exactly d mix nodes. The attack can be carried out in any setting however the analysis then becomes a bit more involved.

1. The attacker first follows a message from a sender to a first mix node.
2. The attacker then follows *every* (t) message that the first node releases. The adversary needs to follow messages going to anywhere between t and 1 different nodes. If all messages are sent to either the same mix node or recipients, the attacker only needs to monitor one node. On the other hand, if all t messages are sent to different nodes, the attacker needs to observe t different mix nodes.
3. The process continues like this until messages reach the dth level nodes. The attacker then need only ”follow” messages leaving the mix network (i.e. going to recipients).

What can the attacker learn from such an attack ? In the worst case, the attacker only needs to follow one path and can match a sender with a recipient. In the best case, the attacker needs to follow t^{d-1} paths through the mix network and t^d messages to the outside world and so can match one sender with t^d possible recipients. Although the worst case is unacceptable, by adding dummy traffic intelligently, we can make the worst case scenario as good as needed. We propose adding dummy messages in the following manner:

- We make sure that each mix node "sprays" its message around adequately. That is, the nodes should ensure that at least t' different mix nodes receive one of its messages (dummy or real).
- We make sure that each mix node sends at least t'' messages outside the mix network every time it sends messages. These should be sent to participating users (or content providers).

The attacker now follows, at the very least, t' different paths through the mix network, and, at the last node, follows messages going to $t' \cdot t''$ distinct recipients (assuming that the final destinations are all different). Hence, the attacker can only match one sender with $t' \cdot t''$ recipients. Note that the probability that an attacker only needs to follow $t' \cdot t''$ is extremely small (if $t \gg t', t''$), and will generally be much larger. Furthermore, if the mix nodes collaborate when choosing who receives dummy messages this bound can easily be increased.

In addition (or in replacement of) to dummy traffic, the users can create routes of random length to fool adversaries. If the routes are arbitrarily large, in order to accurately match one sender with a set of possible recipients, the attacker needs to follow an arbitrarily large number of paths.

The attack can be carried out by passive, static, external adversaries, capable of taping the required wires. (If the attacker can't tap a significant number of wires, the probability of him being able to follow all paths is very low.) Note that if the attacker can't tap a relevant wire, he won't be able to produce a complete potential recipient list since the paths going through the missing wires are lost. A passive, external, adaptive adversary is better suited to this problem as he can "follow" the messages, compromising only the relevant channels (wires).

Since the previous scheme is very simple, it's easy to calculate security/practicality tradeoffs and compare mix-networks with respect to their resistance to brute force attacks. For example it allows us to answer questions like:

- Are busy networks with few nodes more resistant to brute force attacks than quiet networks with many nodes ?
- How helpful is dummy traffic if it's used in a particular manner ?

(Note that if the brute force attack is carried out many times, the techniques of subsection 3.4.3 can be used.)

3.2 The Node Flushing Attack (a.k.a. Spam attack, Flooding attack, n-1 attack)

First mentioned in [8], the flush attack is very effective and can be mounted by an active external adversary. If the nodes wait till they have t messages before "flushing", an attacker can send $t - 1$ messages and easily associate messages leaving the node with those having entered. This can be seen by noting that the adversary will be able to match his inputs with the messages leaving the node.

Dummy traffic can make things a bit more difficult for the attacker since he can't distinguish them from legitimate messages. Unfortunately, if dummy traffic is only used in specific instances, as proposed in subsection 3.1, an attacker can choose his messages so that dummy traffic isn't used.

Another potential solution is to authenticate each message which allows nodes to detect flushing attempts. Unfortunately, this entails authenticating each message and detecting flushing attempts which could be computationally infeasible. We remark that simple-minded implementations of this solution can be broken by message playback attacks. Authentication and privacy protection are two seemingly contradictory requirements however using Chaumian blinding [9] or Brands credentials [5] we can satisfy both requirements.

Stop-and-Go mix nodes (in which a message waits a random amount of time) can *partially* solve this problem. Unfortunately, they only have ”probabilistic” security.

There are some similarities with denial of service attacks [6, 15]. Hence, if t is very large, using hashcach [3] or pricing functions [20] might be effective solutions.

Yet another option is to “re-mix” the messages, that is, the mix nodes use the same mechanism as the user to send the messages to the next nodes – “recursive mixing”. This feature is implemented in mixmaster[13].

Notice that by encrypting the traffic between mix nodes, the attacker looses the ability to easily recognize *his* messages (the partial ($< t - 1$ spams) node flushing attack isn't as effective).

3.3 Timing Attacks

If the different routes that can be taken require different amounts of time, the system could be vulnerable to timing attacks. Precisely, given the set of messages coming in the network and the set of message going out of the network (as well as the arrival, departure times respectively), route timing information might be useful in correlating the messages in the two sets.

For example, suppose there are two routes, one taking 2 second and the other 4 seconds and assume that the two messages coming in the network arrive at 0:00 and 0:01 and that the two messages leave the network at 0:03 and 0:04. The attacker doesn't need to carry out expensive computations in order to correlate the messages coming in with the messages going out . . .

Remark also that an attacker having access to just one of the communicating parties might be able to infer which route is taken by simply computing the *round trip time*. That is, calculating the time it takes to receive a reply. This attack is interesting in that even if one of the parties uses “constant link padding[8]” the attack is still effective.

The attack motivates the use of mix nodes that wait variable amounts of time before flushing messages. We remark that randomly increasing the latency doesn't completely solve the problem since an attacker might be able to rule out some routes (e.g. if a message exits the mix network faster than the minimum time needed to go through some routes then these routes can be ruled out). Hence, the minimum time needed to go through each route should be the same. (It's not clear if this can be directly used in real-world situations since some routes could be very slow – because of mix node processing speed, speed of the communication wires, number of mix nodes in the route, etc.). This kind of attack is mentioned in [28, 40].

[8]The flow of messages between the participant and the first node is constant.

3.4 Contextual Attacks

These are the most dangerous attacks and, unfortunately, they are very difficult to model in a rigorous manner. The problem is that real-world users don't behave like those in the idealized model. We remark that this class of attack is particularly effective for real-time interactive communications.

3.4.1 Communication Pattern Attacks

By simply looking at the communication patterns (when users send and receive), one can find out a lot of useful information. Communicating participants normally don't "talk" at the same time, that is, when one party is sending, the other is usually silent. The longer an attacker can observe this type of communication synchronization, the less likely it's just an uncorrelated random pattern.

This attack can be mounted by a passive adversary that can monitor entry and exit mix nodes. Law enforcement officials might be quite successful mounting this kind of attack as they often have a-priori information: they usually have a hunch that two parties are communicating and just want to confirm their suspicion.

3.4.2 Packet Counting Attacks

These types of attacks are similar to the other contextual attacks in that they exploit the fact that some communications are easy to distinguish from others. If a participant sends a non-standard (i.e. unusual) number of messages, a passive external attacker can spot these messages coming out of the mix-network. In fact, unless all users send the same number of messages, this type of attack allows the adversary to gain non-trivial information.

A partial solution is to have parties only send standard numbers of messages but this isn't a viable option in many settings.

The packet counting and communication pattern attacks can be combined to get a "message frequency" attack (this might require more precise timing information).

Communication pattern, packet counting and message frequency attacks are sometimes referred to as traffic shaping attacks and are usually dealt with by imposing rigid structure[9] on user communications[4]. Notice that protocols achieving "network unobservability" are immune to these attacks.

3.4.3 Intersection Attack

An attacker having information about what users are active at any given time can, through repeated observations, determine what users communicate with each other. This attack is based on the observation that users typically communicate with a relatively small number of parties. For example, the typical user usually queries the same web sites in different sessions (his queries aren't random). By performing an operation similar to an intersection on the sets of active users at different times it is probable that the attacker can gain interesting information. The intersection attack is a well known open problem and seems extremely difficult to solve in an efficient manner.

[9] It's not clear whether this is a viable option for large Internet based systems.

3.5 Denial of Service Attacks

By rendering some mix nodes in-operational, an active adversary might obtain some information about the routes used by certain users. It seems highly probable that users that have their routes "destroyed" will behave differently than parties that haven't. Network unobservability or "client challenges[10]" (e.g. [3, 20, 26]) might be required to properly solve this problem for real-time interactive communications.

3.6 Active Attacks Exploiting User Reactions

Active adversaries might be able to gain non-trivial sender-recipient matching information by exploiting the fact that user behavior depends on the message received. A variation on the following attack can even be used against dc-nets that don't use a secure broadcast channel (see [43]).

1. The adversary first intercepts a message M just before it enters the mix net.
2. M is then sent to a set of possible recipients. The parties not expecting to receive this message message will probably react in a different manner than a party expecting it.

The attacker can use this to get some sender-recipient matching information. Note that if the nodes in the route authenticate the messages the attack is prevented.

3.7 The "Sting" Attack

If one of the party involved in a dialog is corrupt, he might be able to, in a sense, "encode" information in his messages (see [28]). For example, government agencies might set up a fake "bomb making instruction web sites" and try to find out who accesses it. Many methods for identifying a user querying the web page come to mind: varying the reply latency, sending messages of a specific length, etc.

In some situations, it might be even easier to compromise user privacy. For example, if the sting web site gives fake information pertaining to financial fraud, the user might (non-anonymously) act upon this information at which point he can be arrested.

3.8 The "Send n' Seek" Attack

This attack is, in a sense, the opposite of the "sting" attack of subsection 3.7. Instead of having the recipient try to find the sender's identity, it's the sender that attempts to uncover the recipient's identity. This attack is particularly dangerous against non-interactive processes. For example, privacy protecting e-mail systems (see for example [21]) can be attacked by sending an easily identifiable number of messages and trying to identify these messages at suspect destinations (e.g. POP boxes). Notice that the terms sender and recipient are used very loosely here; the sender refers to the party initiating the connection.

[10]Prevention mechanisms for denial of service attacks.

3.9 Attacks Based on the Message's Distinguishing Features

If the user's (unencrypted) messages have distinguishing characteristics, the recipient (and maybe the last node, depending on whether the message is encrypted) might be able to link the message with one or many individuals. For example, analyzing writing styles probably reveals non-trivial information about the sender.

3.10 Message Delaying

The attacker can withhold messages until he can obtain enough resources (i.e. wires, nodes, etc.) or until the network becomes easier to monitor or to see if the possible recipients receive other messages, etc. In view of this attack, it makes sense to have the mix nodes verify authenticated timing information (the authenticated timing information could be inserted in the message by the sender or the nodes).

3.11 Message Tagging [16]

An active internal adversary that has control of the first and last node in a message route, can tag (i.e. slightly modify) messages at the first node in such a way that the exit node can spot them. Since the entry node knows the sender and the exit node the recipient, the system is broken.

A solution to this problem is to make it difficult to tag messages. The techniques that can be used to do this depend on the implementation and so are not discussed here.

A slight variant of this attack can be mounted by an active external adversary if the messages don't have a rigid structure. Remark that this attack has many similarities with subliminal channels [41]; this observation forms the basis of some of the following variations:

- **Shadow Messages:** If an adversary sends messages that follow the same path as the message being followed, it can easily transmit some information to the output. For example, the attacker can just replay the message in such a way that it can spot them leaving the mix network (e.g. varying the message frequency).
- **Message Delaying:** The attacker can delay messages to obtain some information. These delays can presumably detected.
- **Broadcast:** An attacker can broadcast messages notifying his accomplices that a particular message has entered the network. This isn't a particularly powerful attack but it could be virtually impossible to detect.

The message tagging based attacks motivate using extremely rigid message structure and authenticating timing information (in order to prevent message delays and message playbacks).

3.12 Partial or Probabilistic Attacks

Most of the preceding attacks can be carried out partially, that is, the attacker can obtain partial or probabilistic information. For example, he could deduce information of the form:

1. With probability p, A is communicating with B or A is communicating with one of the users in a group.
2. A is not communicating with B,C and D.

These attacks haven't been thoroughly addressed so far and seem very promising, especially when carried out a large number of times.

3.13 Approaches to Modeling Attacks

The previous attacks have assumed that the adversary controlled all the required resources (wires, nodes). When only considering static adversaries, it might make sense to calculate the probability that the required resources are controlled. This approach is especially relevant when the adversary just wants to obtain *a* sender-recipient matching.

Unfortunately, assuming static adversaries doesn't seem helpful for making design decisions (i.e. how much dummy traffic); it might however help us in determining if there is a reasonable threat that the system can be broken.

4 Mix Network Design Issues

In this section, we present issues related to mix-network design.

4.1 Anonymity versus Pseudonymity

Probably the most important design issue is that of anonymity versus pseudonymity. Note that by pseudonymous, we mean that some node(s) knows the user's pseudonym (it can't link a pseudonym with a real-world identity). Another option is to have the user be anonymous in the mix network but be pseudonymous in its dealings with other users (half-pseudonymity). Half-pseudonymity won't be discussed in any detail because its properties are similar to the pseudonymity ones. Here are the most important advantages of both anonymity and pseudonymity

- **Anonymity**
  1. Provides better security since if a pseudonym (nym) is linked with a user, all future uses of the nym can be linked to the user.
- **Pseudonymity**

1. We get the best of both worlds: privacy protection and accountability (and openness). Since pseudonyms (nyms) have a persistent nature, long term relationships and trust can be cultivated. (half pseudonymity also)
2. Pseudonym based business models (for mix node operators) are more attractive than anonymity based ones.
3. Abuse control is easier to deal with when nyms are used.
4. Authentication (verifying that someone has the right to use the network) is easier: either Brands credentials [5] or Chaumian blinding [9] needs to be used [11] when using anonymity.
5. Allows non-interactive processes (e.g. e-mail).

4.2 Packet Sizes

In many situations, using different message sizes yield substantial performance improvements. For example TCP/IP connections require on average one small control packet for every two (large) data packet. It might be inefficient for small messages to be padded or large packets split up in order to get a message of the correct size. As usual in cryptography, we have a security/performance tradeoff: Using more than one message size gives better performance but worse security. We strongly suspect however that there are techniques which improve the security properties of the multiple packet size option (e.g. randomly expanding small messages.).

4.3 Dummy Messages

Dummy traffic is often used in an unstructured manner and so might not be as effective as it could be, we note the following observations:

1. If a node sends its message to less than t' nodes we suggest sending dummy messages in such a way that t' nodes receive messages. The larger t', the harder it is to mount the brute search attacks.
2. Each node should send messages to at least t'' destinations outside the mix network (dummy messages should be used to fill the gaps). The larger t'', the harder it is to mount the brute search attack. Furthermore, this technique also seems to complicate attacks in which the adversary monitors the exit nodes.
3. In order to randomize the user's communication patterns, we should seriously consider having the user send dummy traffic to the entry node. The challenge here is to have good security and minimize the amount of dummy messages used (see [4]).
4. Dummy messages could also be used to reduce the amount of time messages stay at a given node. It seems that waiting for c messages to enter a mix node before sending b ($b > c$) has similar security properties as waiting to receive b messages before releasing them. This trick could be used to reduce the time messages wait at nodes.

[11] both of these techniques are patented.

4.4 Routing

For large Internet based systems especially, having the user choose the nodes in his route randomly doesn't seem like a viable option because:

1. The nodes and users must "know[12]" each other node which might be impractical.
2. Some servers are far from each other and it doesn't make sense from a performance view point to have, for example, a route consisting of nodes in Australia, Canada, South Africa and China.
3. Nodes should be "socially" independent. Ideally, the nodes in a route should belong to different organizations and be located in different legal jurisdiction. The whole idea behind using more than one node is that none of them have enough information to determine sender-recipient matchings. Hence, if all nodes in a route belong to the same organization we might as well just use a single node. The motivation for having nodes in different legal jurisdiction is that more than one subpoena needs to be obtained to legally compromise nodes.

Creating good network topologies and route finding algorithms with respect to security and efficiency doesn't seem entirely trivial.

Note also that in order to limit the number of public key operations executed, some systems (e.g. [21]) use static routes that allows mix nodes to associate each message with a connection identifier which makes some of the attacks mentioned previously a lot easier to carry out.

4.5 Node Flushing Algorithm

As seen in subsection 2.3.2, there are many different approaches to flushing nodes. Again, there is a security/practicality tradeoff: the longer messages *can* stay in mix-nodes the better the security (in most settings).

4.6 Query Servers and Privacy Protection

In many situations, the user needs to retrieve some information from a query server, for example network configuration information, pseudonym public keys, etc. These queries shouldn't erode privacy: the query servers shouldn't obtain non-trivial information about sender-recipient matchings. The obvious approach to this problem is to have the user download the entire databases (the answer to every possible query) but unfortunately, the amount of data to transfer might be too large. We suspect that private information retrieval protocols [10, 11] might be very useful in these situations. This design issue illustrates a fundamental security principle:

A system is only as secure as its weakest link.

[12]e.g. know the IP address, Port number and status.

5 Directions for Further Research

Probably the most important direction for further research in this field is that of attacks. As it seems unlikely that we can obtain Rackoff-Simon [39] type bounds for real-world implementations, it's a good idea to find and rigorously analyze as many attacks as possible and either :

- Try to immunize our protocols against these attacks.
- Detect when the attack is mountable and take the appropriate measures.

The new and clever attacks that will be effective against mix networks will probably be empirical in nature. Mathematical analysis of mix network traffic seems like the most promising avenue for mounting attacks. Perhaps ideas from the field of pattern recognition and measure theory could be used . . .

We now give a listing of some other relevant problems (in no particular order):

1. Most of the attacks mentioned are aimed; what about a more general class of attack in which an attacker doesn't require a particular sender-recipient matching but would settle for an arbitrary one ?
2. The best we can hope for is that the attacker's view be independent from the sender-recipient matchings. Is it possible to obtain weaker results in which the view is slightly biased? Such a result would allow us to determine how much information the attacker needs to gather in order to get a "convincing" sender recipient linking (instead of relying on ad-hoc arguments).
3. Another possible avenue of research is formalizing the effectiveness of a given adversary in breaking the protocol. That is working with a more precise adversary descriptions; Instead of active, static and internal adversaries, we could have adversaries taping two specific communication channels, having total control of a particular mix node, etc. It's not clear how this would help us in designing good protocols, however it might be useful when certain parts of the network are thought to be compromised.
4. It's not clear at all what real-world adversary can do. Can they tap wires and compromise nodes at will ? It would be very instructive to know what can be done, the level of sophistication required and the computational resources needed (memory, CPU cycles, network access, etc.).
5. It would be extremely useful to determine when the mix-network is vulnerable or, more generally, what security mix-networks provide in different situations.
6. All issues mentioned in section 4 need to be thoroughly analyzed.
7. Caching popular content would improve security and it's not clear what the best way to go about doing this is.
8. Perhaps the exit nodes can perform some computations for the users. For example, the TCP control messages could be handled by the exit mix node (i.e. the control messages would not be handled by the user).

9. Heaps of security and efficiency problems arise when incorporating privacy protecting mechanisms within existing protocols (e.g. http, telnet, etc.).

10. A detailed specification (e.g. within IETF) could be devised to help mix-network designers. This would protect mix-network operators from known attacks and give attackers a precise model to "study" (thus helping us improve the specification . . .)

6 Conclusion

We have given an introduction to the traffic-analysis problem by presenting the most important constructions, attacks, design issues and direction for further research. It is hoped that research addressing some of the problems exposed in this work will allow us to stop using terms such as : "seems", "probably", "I suspect" in our discussions about traffic analysis.

Acknowledgements

We would like to thank Adam Back, Adam Shostack, Anton Stiglic, Frédéric Légaré, Michael Freedman and Ulf Moller for excellent feedback, interesting discussions and for having mentioned many of the attacks presented in this paper. We also wish to thank the anonymous referees for their very helpful comments.

References

[1] M Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 437–447, Helsinki, Finland, 31 May– 4 June 1998. Springer-Verlag.

[2] M Abe. Mix-network on permutation networks. In *Advances in cryptology — ASIACRYPT'99*, volume 1716, pages 258–273. Springer-Verlag, 1999.

[3] Adam Back. Hashcash. http://www.cypherspace.org/˜adam/hashcash/, march 1997.

[4] Oliver Berthold, Hannes Federrath, and Marit Kohntopp. Project anonymity and unobservability in the internet. Presented at CFP 2000.

[5] Stefan A. Brands. Restrictive blinding of secret-key certificates. Technical Report CS-R9509, CWI - Centrum voor Wiskunde en Informatica, February 28, 1995.

[6] CERT. Advisory ca-96.21: Tcp syn flooding and ip spoofing attacks, 24 September 1996.

[7] D Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[8] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the A.C.M.*, 24(2):84–88, February 1981.

[9] David Chaum. Blind signatures for untraceable payments. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. CRYPTO 82*, pages 199–203, New York, 1983. Plenum Press.

[10] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th IEEE Conference on the Foundations of Computer Science*, pages 41–50. IEEE Computer Society Press, 1995.

[11] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998.

[12] David A. Cooper and Kenneth P. Birman. Preserving privacy in a network of mobile computers. In *1995 IEEE Symposium on Research in Security and Privacy*, pages 26–38. IEEE Computer Society Press, 1995. http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR85-1490.

[13] Lance Cottrell. Mixmaster. http://www.obscura.com/˜loki/.

[14] Ronald Cramer. Introduction to secure computation. In *Lectures on data security : modern cryptology in theory and practice*, volume 1561 of *Lecture Notes in Computer Science*, pages 16–62. Springer, 1999.

[15] Daemon9. Project neptune. Phrack Magazine, 48(7): File 13 of 18, 8 November 1996. Available at www.fc.net/phrack/files/p48/p48-13.html.

[16] Wei Dai. private communication, 1999.

[17] Yvo Desmedt and Kaoru Kurosawa. How to break a practical mix and design a new one. To be presented at Eurocrypt 2000.

[18] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[19] Shlomi Dolev and Rafail Ostrovsky. Efficient anonymous multicast and reception. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT ' 97*, Lecture Notes in Computer Science, pages 395–409. Springer-Verlag, Berlin Germany, 1997.

[20] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer-Verlag, 1993, 16–20 August 1992.

[21] Ian Goldberg and Adam Shostack. Freedom network whitepapers.

[22] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[23] C. Gulcu and G. Tsudik. Mixing E-mail with BABEL. In *Symposium on Network and Distributed Systems Security (NDSS '96)*, San Diego, California, February 1996. Internet Society. http://www.zurich.ibm.com/ cgu/publications/gt95.ps.gz.

[24] Ceki Gulcu. The anonymous E-mail conversation. Master's thesis, Eurecom Institute, 229 route des Cretes, F-06904 Sophia-Antipolis, France, June 1995.

[25] Jakobsson. A practical mix. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 448–. Springer-Verlag, 1998.

[26] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In S. Kent, editor, *NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, 2000.

[27] D. Kahn. *The Codebreakers*. Macmillan Publishing Company, 1967.

[28] John Kelsey. private communication, 1999.

[29] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go mixes providing probabilistic security in an open system. In David Aucsmith, editor, *Information Hiding: Second International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 83–98. Springer-Verlag, Berlin, Germany, 1998.

[30] W Ogata, K Kurosawa, K Sako, and K Takatani. Fault tolerant anonymous channel. In *Information and Communications Security — First International Conference*, volume 1334 of *Lecture Notes in Computer Science*, pages 440–444, Beijing, China, 11–14 November 1997. Springer-Verlag.

[31] A Pfitzmann and M Waidner. Networks without user observability – design options. In *Advances in Cryptology – Eurocrypt '85*, volume 219 of *Lecture Notes in Computer Science*. Spinger-Verlag, 1985.

[32] Andreas Pfitzmann. A switched/broadcast ISDN to decrease user observability. 1984 International Zurich Seminar on Digital Communications, Applications of Source Coding, Channel Coding and Secrecy Coding, March 6-8, 1984, Zurich, Switzerland, Swiss Federal Institute of Technology, Proceedings IEEE Catalog no. 84CH1998-4, 183-190, 6–8 March 1984.

[33] Andreas Pfitzmann. How to implement ISDNs without user observability–some remarks. Technical report, Institut für Informatik, University of Karlsruhe, Institut für Informatik, University of Karlsruhe, 1985.

[34] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *GI/ITG Conference: Communication in Distributed Systems*, pages 451–463. Springer-Verlag, Heidelberg 1991, February 1991.

[35] B Pfitzmann and A Pfitzmann. How to break the direct rsa-implementation of mixes. In *Advances in Cryptology – Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[36] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[37] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security, volume 1, pages 66–92, 1998.

[38] Michael K. Reiter and Aviel D. Rubin. Anonymous Web transactions with crowds. Communications of the ACM, volume 42, number 2, pages 32–48, 1999.

[39] Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 672–681, San Diego, California, 16–18 May 1993.

[40] M G Reed, P F Syverson, and D M Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Special Areas in Communications*, 16(4):482–494, May 1998.

[41] G. J. Simmons. The history of subliminal channels. *IEEE Journal on Selected Areas in Communications*, 16(4):452–462, May 1998.

[42] L. F. Turner. Digital data security system, 1989. Patent IPN WO 89/08915.

[43] M Waidner. Unconditional sender and recipient untraceability in spite of active attacks. In *Advances in Cryptology – Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.