Better Anonymous Communications

George Danezis

University of Cambridge
Computer Laboratory
Queens' College

January 2004

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text and summarised in section 1.3.

This dissertation does not exceed the regulation length of 60000[1] words, including tables and footnotes.

[1] using `detex thesis.tex | wc`

Acknowledgements

> *"No man is an island, entire of itself; every man is a piece of the continent, a part of the main."*
>
> *Meditation 17, Devotions Upon Emergent Occasions* — John Donne

I am deeply indebted to my supervisor and teacher Ross Anderson, for allowing me to join his research group, and helping me at key points during the last three years. I hope this document shows that his trust in me was justified. I am also grateful to Robin Walker, my director of studies, for his support during the last six years and for giving me the opportunity to study in Cambridge.

These years of research would have been considerably less productive without the local scientific community, my colleagues in the computer security group of the Computer Laboratory, but also my colleagues from around the world. I would like to specially thank my co-authors, Richard Clayton, Markus Kuhn, Andrei Serjantov, Roger Dingledine, Nick Mathewson and Len Sassaman.

This research would not have been possible without my parents. At every stage of my academic life they acted as role models, provided moral and financial support. The special value they attach to education has marked me for life.

I am grateful to cafes, social centres and autonomous zones around Europe for having me as a guest; it is fitting that some ideas developed in this thesis were first conceived there. Similarly I am indebted to my friends, housemates and companions, that have shared with me everything, at times when none of us had much. Without them, all this would be meaningless.

This research was partly supported by the Cambridge University European Union Trust, Queens' College, and The Cambridge-MIT Institute.

Better Anonymous Communications

George Danezis

Summary

This thesis contributes to the field of anonymous communications over widely deployed communication networks. It describes novel schemes to protect anonymity; it also presents powerful new attacks and new ways of analysing and understanding anonymity properties.

We present Mixminion, a new generation anonymous remailer, and examine its security against all known passive and active cryptographic attacks. We use the secure anonymous replies it provides, to describe a pseudonym server, as an example of the anonymous protocols that mixminion can support. The security of mix systems is then assessed against a compulsion threat model, in which an adversary can request the decryption of material from honest nodes. A new construction, the fs-mix, is presented that makes tracing messages by such an adversary extremely expensive.

Moving beyond the static security of anonymous communication protocols, we define a metric based on information theory that can be used to measure anonymity. The analysis of the pool mix serves as an example of its use. We then create a framework within which we compare the traffic analysis resistance provided by different mix network topologies. A new topology, based on expander graphs, proves to be efficient and secure. The rgb-mix is also presented; this implements a strategy to detect flooding attacks against honest mix nodes and neutralise them by the use of cover traffic.

Finally a set of generic attacks are studied. Statistical disclosure attacks model the whole anonymous system as a black box, and are able to uncover the relationships between long-term correspondents. Stream attacks trace streams of data travelling through anonymizing networks, and uncover the communicating parties very quickly. They both use statistical methods to drastically reduce the anonymity of users. Other minor attacks are described against peer discovery and route reconstruction in anonymous networks, as well as the naïve use of anonymous replies.

Contents

Chapter 1

Introduction

> *"The issues that divide or unite people in society are settled not only in the institutions and practises of politics proper, but also, and less obviously, in tangible arrangements of steel and concrete, wires and transistors, nuts and bolts."*
>
> *Do artefacts have politics?* — Langdon Winner

Conventionally, computer and communications security deals with properties such as confidentiality, integrity and availability. Aside from these a number of techniques were developed to achieve *covertness* of communications and decrease the likelihood that the conversing parties have their communications jammed, intercepted or located. Such techniques often involve a modification of the transport layer such as direct sequence spread-spectrum modulation, frequency hopping or bursty communications.

Usually it is not possible to modify the transmission technology and organisations or individuals are forced to use widely deployed packet-switched networks to communicate. Under these circumstances, technologies providing covertness have to be implemented on top of these networks. The subject of this thesis is the study of how to implement some of these properties, and in particular allow conversing parties to remain *untraceable* or *anonymous* to third parties and to each other.

Anonymous communications can be deployed in high-risk environments, for example in a military context. Such technology could mask the roles of different communicating units, their network location or their position in the chain of command. It would then be much harder for an adversary to perform target selection through signal and communications intelligence.

At the other end of the spectrum, anonymous communications can be used to protect individuals' privacy. They could be used as primitives in security policies for managing patient health records, or on-line support groups. Widespread use of anonymous communications could even be used in e-commerce to mitigate some of the price discrimination tactics described by Odlyzko [Odl03].

Some non-military uses of these technologies still take place in an environment of conflict. Anonymous communications can be used to protect political speech, in the face of censorship or personal threats. Similarly, anonymous communications could be used to improve the resilience of peer-to-peer networks against legal attacks. In both cases powerful adversaries are motivated to trace the participants, and anonymous communications systems can stand in their way.

1.1 Scope and purpose

The scope of the research presented is anonymous communications over widely deployed information networks. The relevance of anonymous communications to common tasks, such as web browsing and transmitting email, will be considered in some depth. Other aspects of anonymity such as location privacy, credentials and elections are outside the scope of this work. Our major contribution consists of methods to measure anonymity, the design of new anonymous communication mechanisms, and a set of generic attacks that can be used to test the limits of anonymity.

As a pre-requisite for our research, a definition of anonymity was proposed. This has since been widely accepted in the anonymity research community [SK03, MNCM03, MNS]. Chapter 2 will present the requirements for anonymous communications, the threat models, our technical definition of anonymity and some applications.

Our contributions have been related to the field of mix systems [Cha81]. Mixes are special nodes in a network that relay messages while hiding the correspondence between their input and their output. Several of them can be chained to relay a message anonymously. These systems provide the most promising compromise between security and efficiency in terms of bandwidth, latency and overheads. A careful explanation of mix systems is presented in chapter 4. This thesis then analyses the major components necessary for mixed anonymous communications: *bitwise unlinkability* which ensures that input and output messages 'look' different, and *dynamic aspects* which make sure that many messages are mixed together at any time.

Our contribution to bitwise unlinkability is Mixminion, presented in chapter 5. It has since been adopted as the official successor of the Mixmaster remailer, the standard for anonymous remailers. In chapter 6 techniques will be presented that allow for forward secure mixing, which makes it extremely expensive for an attacker with compulsion powers to trace back a communication.

Our main contribution to the dynamic aspects of mix systems is an analysis of restricted mix network topologies in chapter 7. Such networks of mix nodes are sparse, but ensure that after a small number of hops all messages are mixed together. Techniques are devised to fine-tune the parameters of these systems to provide maximal anonymity.

Decoy messages, often called *dummy traffic*, can be used in anonymity systems to confuse the adversary. The dummy traffic policy presented in chapter 8 protects the network against active attacks and yet is economical during normal operation. It uses the honest nodes' knowledge about the network to detect and neutralise flooding attacks.

In chapter 9 we present an efficient method of performing intersection attacks, using only trivial vector operations, and analyse its performance. Finally, in chapter 10, we will apply our anonymity metric to delaying anonymous networks, and use pattern matching techniques to trace streams of traffic within an anonymity system. Our attacks help define limits of persistent anonymous communications and the caveats of circuit-based anonymizing systems.

Additionally, novel attacks against the Tarzan peer-to-peer anonymizing system are presented in section 4.2.7 and the *Who am I?* attack against anonymous reply mechanisms is described in chapter 5.

1.2 Schedule of work

As part of this work a series of papers were published [CDK01, SD02, CD02, Dan02, Dan03a, DDM03a, Dan03b, DS03a], some in collaboration with other researchers, in peer reviewed academic conferences and workshops.

- Richard Clayton, George Danezis, and Markus G. Kuhn. Real world patterns of failure in anonymity systems. In Ira S. Moskowitz, editor, *Information Hiding workshop (IH 2001)*, volume 2137 of *LNCS*, pages 230–244, Pittsburgh, PA, USA, January 2001. Springer-Verlag

- Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies workshop (PET 2002)*, volume 2482 of *LNCS*, pages 41–53, San Francisco, CA, USA, 14-15 April 2002. Springer-Verlag

- Richard Clayton and George Danezis. Chaffinch: Confidentiality in the face of legal threats. In Fabien A. P. Petitcolas, editor, *Information Hiding workshop (IH 2002)*, volume 2578 of *LNCS*, pages 70–86, Noordwijkerhout, The Netherlands, 7-9 October 2002. Springer-Verlag

- George Danezis. Forward secure mixes. In Jonsson Fisher-Hubner, editor, *Nordic workshop on Secure IT Systems (Norsec 2002)*, pages 195–207, Karlstad, Sweden, November 2002

- George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 1–17, Dresden, Germany, March 2003. Springer-Verlag

- George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003

- George Danezis. Statistical disclosure attacks. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Security and Privacy in the Age of Un-*

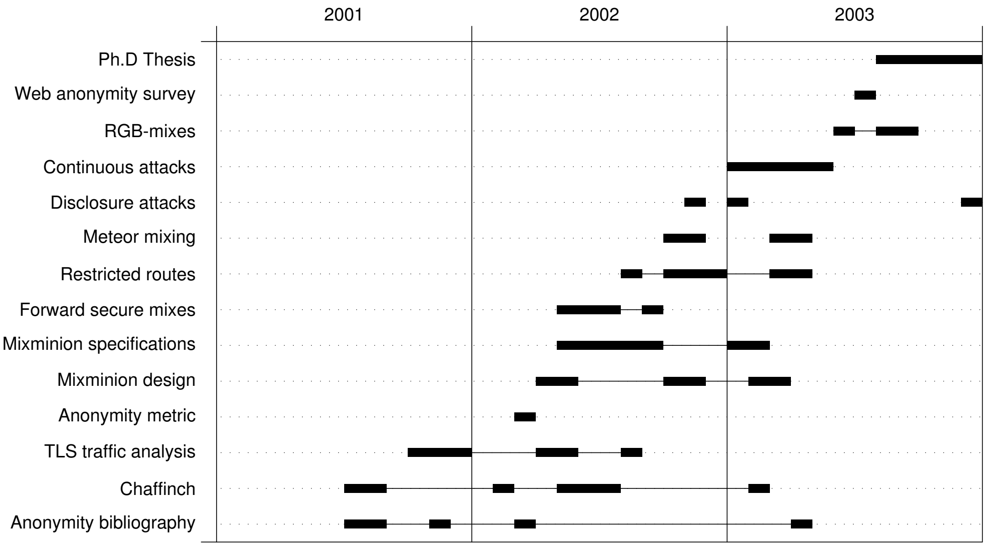


Figure 1.1: Schedule of work extracted from CVS logs

certainty, (SEC2003), pages 421–426, Athens, May 2003. IFIP TC11, Kluwer

- George Danezis and Len Sassaman. Heartbeat traffic to counter $(n-1)$ attacks. In *workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, November 2003

The first year of our research was spent reading background material and getting a solid foundation in the techniques required to design, analyse and build secure systems. During this year a survey paper with practical attacks against anonymous system was published [CDK01] and most of the research for Chaffinch, a communication channel providing plausible deniability, was done [CD02].

In the summer of 2001 all our work was incorporated into a CVS repository. Figure 1.1 shows a calendar of the main projects over time. The thin black lines show when projects were active, and the thick ones show the months in which work was committed.

1.3 Work done in collaboration

The work described in chapter 2 was published in conjunction with Andrei Serjantov. While the idea of using entropy as the metric was mine, using the pool mix to illustrate it was Andrei's. We both derived the anonymity provided by pool mixes independently.

The two attacks against Tarzan presented in section 4.2.7 greatly benefited from collaborating with Richard Clayton, and discussions with Tarzan's 'father' Michael Freedman.

Mixminion described in chapter 5 has been the product of a wide collaboration and intensive work with Roger Dingledine and Nick Mathewson in particular. Resistance to tagging attacks was engineered by myself, as a result of previous schemes being successfully attacked by Roger and Nick. A parallel proposal was also put forward by Bryce "Zooko" Wilcox-O'Hearn.

The work on *rgb-mixes* presented in chapter 8 was done in collaboration with Len Sassaman. He independently came to the conclusion that cover traffic should be destined back to the node that generated it, and was kind enough to correct and present the paper.

Chapter 2

Defining anonymity

> "*The violence of identification is by no means merely conceptual. The scientific method of identitarian thought is the exercise of power-over. Power is exercised over people through their effective identification.*"
>
> *Change the world without taking power* — John Holloway

2.1 Anonymity as a security property

Anonymity allows actors to hide their relation to particular actions and outcomes. Since anonymous communication is the main subject of this work, our objective will be to hide correspondences between senders and the messages they sent, a property we will call *sender anonymity*, or receivers and the messages they receive, namely *recipient anonymity*. It is possible for a channel to offer full bidirectional anonymity to allow anonymous senders to converse with anonymous receivers.

Anonymous communications are studied in the context of computer security because they are taking place in an *adversarial context*. The actor attempts to protect their anonymity vis-a-vis some other parties that try to uncover the hidden links. This information has some *value* for those performing surveillance and would entail some *cost* for the subject if it was revealed.

An example from the commercial world where identification is desirable to a seller of goods is described by Odlyzko [Odl03]. By linking together all the previous purchases of a buyer they can infer how willing they are to pay for particular products and price discriminate by charging as much as possible. In this case the value of performing surveillance can be defined in monetary terms.

Another example, not involving monetary value and cost, is surveillance against a suspected terrorist cell. By uncovering the identities of the participants, investigators can map their social network through drawing 'friendship trees'. As a result, they are able to evaluate the level of threat by measuring the size of the network, they could get warning of an imminent action by analysing the intensity of the traffic and are able to extract information about the command structure by examining the centrality of different participants. In this case, value is extracted by reducing uncertainty and reducing the cost of neutralising the organisation if necessary.

As well as the value and cost of the information extracted there are costs associated with performing surveillance. An extensive network of CCTV cameras is expensive and so are the schemes proposed for blanket traffic data retention. Additionally the cost of analysis and dissemination of intelligence might actually be dominant [Her96].

Similarly, counter-surveillance and anonymity also have costs. These costs are associated with designing, operating, maintaining and assessing these systems. Furthermore, counter-surveillance technologies impose a huge opportunity cost. They inhibit parties under surveillance using conventional, efficient, but usually unprotected technologies. They might even prevent parties from communicating if appropriate countermeasures are not available against the perceived surveillance threats. The element of *perceived threat* is important, since the capabilities of adversaries are often not directly observable. This uncertainty pushes up the cost of counter-surveillance measures, such as anonymous communication technologies. In other words, paranoia by itself entails significant costs.

It is important to note that anonymity of communications is a security property orthogonal to the secrecy of communications. A typical example illustrating this is an anonymous sent letter to a newspaper. In this case the identity of the sender of the letter is not revealed although the content is public. One can use conventional encryption techniques, in addition to anonymous communications, to protect the confidentiality of messages' contents.

2.1.1 Traditional threat model

In the context of anonymous communication protocols, the primary goal of an adversary is to establish a reliable correspondence between the action of sending or receiving a particular message and an actor. A further goal might be to disrupt the system, by making it unreliable. Such attacks fall in the category of denial-of-service, and they are called *selective denial-of-service* if only one user or a particular group is targeted. An attacker might also attempt to mount *reputation* attacks against anonymity systems, in the hope that fewer people will use them. If so then the overall anonymity would be reduced, a particular target would not trust it and would use a less secure means of communicating, or abstain from communicating altogether.

From the computer security point of view, threats to anonymity can be categorised (following the established cryptological tradition) according to the capabilities of the adversary. An adversary Eve is called *passive* if she only observes the data transiting on the communication links. Passive adversaries are called *global* if they can observe all network links. The *global passive adversary* is the main threat model against which mix systems (see chapter 4) are required to be secure and have been thoroughly analysed against.

An attacker Mallory is said to be *active* if he can also insert, delete or modify messages on the network. A combination of these can be used to delay messages in an anonymous communication network, or flood the network with messages.

Besides their interaction with the network links, attackers can have control of a number of *subverted nodes* in the network. The inner workings of those nodes is transparent to the attacker, who can also modify the messages going through them. Usually such attackers are described by the percentage of subverted nodes they control in the system. Which nodes are subverted is not known to other users and identifying them might be a sub-goal of a security protocol.

2.1.2 Compulsion threat model

The traditional threat model presented above has been used in the context of cryptological research for some time, and can indeed express a wide spectrum of threats. On the other hand it suffers some biases from its military origins, that do not allow it to define some very important threats to anonymous communication systems.

Anonymous communication systems are often deployed in environments with a very strong imbalance of power. That makes each of the participants in a network, user or intermediary, individually vulnerable to *compulsion attacks*. These compulsion attacks are usually expensive for all parties and cannot be too wide or too numerous.

A typical example of a compulsion attack would be a court order to keep and hand over activity logs to an attacker. This can be targeted at particular intermediaries or can take the form of a blanket requirement to retain and make accessible certain types of data. Another example of a compulsion attack could be requesting the decryption of a particular ciphertext, or even requesting the secrets necessary to decrypt it. Both these could equally well be performed without legal authority by just using the threat of force.

Parties under compulsion could be asked to perform some particular task, which bears some similarity with subverted nodes discussed previously. For example, this is an issue for electronic election protocols where participants might be coerced into voting in a particular way.

Note that compulsion and coercion cannot be appropriately modelled using the concept of subverted nodes from the traditional threat model. The party under compulsion is fundamentally honest but forced to perform certain operations that have an effect which the adversary can observe either directly or by requesting the information from the node under compulsion. The information or actions that are collected or performed under coercion are not as trustworthy, from the point of view of an adversary, as those performed by a subverted node. The coerced party can lie and deceive in an attempt not to comply. Election protocols are specifically designed to allow voters to freely lie about how they voted, and *receipt-freeness* guarantees that there is no evidence to contradict them.

2.1.3 Assessing real-world capabilities

Both models aim to codify the abilities of our adversaries in such a way that we can use them to assess the security of the technical mechanisms presented in this thesis. An entirely different body of work is necessary to build a "realistic" threat model, namely the set of capabilities that real world opponents have.

National entities are the most powerful adversaries in the real world because of the funding they are able to commit but also because of their legal authority over a jurisdiction. Many national entities could be considered to be global passive adversaries. The interception of communication could happen through signal intelligence gathering networks [Cam99], or through the lawful interception capabilities implemented in network and routing equipment [YM]. Furthermore, some national entities have provisions that allow them to legally request the decryption of material or the handing over of cryptographic keys (such provisions are included in the UK RIP Act, but are not yet active [RIP00]). This could be seen as approximating the compulsion model. Often these powers are restricted to a particular geographical area and their resources are plenty, but still limited.

Large corporations have resources that are comparable to national entities but lack the legal authority to eavesdrop on communications or compel nodes to reveal secrets. On the other hand they are able to run subverted nodes that collect information about users, and launch active attacks. In the case of the music industry fighting what it sees as copyright infringement, nodes have been run within peer-to-peer systems that collected information about files requested and offered, and served files with bad content. The information collected was used to launch a large number of lawsuits [Smi03].

In a military context often the threat model is slightly different from the one presented above, or studied in this work. It is assumed that all participants, senders receivers and relays, are trusted, and their anonymity must be preserved against third parties, that can monitor the network or even modify communications. This property is called *third party anonymity* and systems that provide it *Traffic Analysis Prevention* (TAP) systems.

2.2 Technical definitions and measures

From a technical point of view anonymity has been defined in a variety of ways. Some of the definitions are closely tied to proposed systems, while others are more generic. Technical definitions often try to provide a way to measure anonymity, since it is widely recognised that it can be provided in different degrees.

In Crowds [RR98] a scale of *degrees of anonymity* is provided. The scale ranges from *absolute privacy, beyond suspicion, probable innocence* to *possible innocence, exposed, provably exposed.* While this definition and measure of anonymity is only qualitative, it helps clarify quite a few issues.

An important feature is the fact that there are different categories for *exposed* and *provably exposed.* This should serve as a critique to those designing protocols that claim to be anonymous if they don't include any mechanisms allowing parties to *prove* information about links to the identities of actors. The scale presented makes it clear that not being able to prove something about an identity is very far from providing effective anonymity.

It also draws attention to the different ways in which breaking anonymity could have consequences. While a degree of anonymity guaranteeing *possible innocence* might be sufficient to avoid condemnation in a criminal court, *probable innocence* might still be judged insufficient in a civil court. Furthermore an anonymity degree of *beyond suspicion* would be required in order not to be the subject of further investigations in many cases. Therefore this scale highlights very effectively the existence of a continuum of anonymity that can be provided.

In [Cha88] Chaum presents the *dining cryptographers*(DC) network, a construction that provides perfect (information theoretic) anonymity. A DC network is a multi-party computation amongst a set of participants, some pairs of which share secret keys. It can be shown that as long as the graph representing key sharing is not split by the subverted nodes, an adversary cannot tell which participants sent the message with probability better than uniformly random. Therefore Chaum defines the *anonymity set* of each message to be the set of all participants in the network. Its cardinality is referred to as the anonymity set size, and has been used as a measure of anonymity.

Some attacks relying on subverted nodes can *partition* the anonymity sets in DC networks, and allow an attacker to find out from which of the two partitions a message originated. The worst case is when an attacker manages to partition the sets in such a way that only the actual sender is left in one of them. In these cases the sender anonymity set of the message he sends has cardinality one, and we say that the system does not provide any anonymity.

The notion of anonymity set has been widely accepted and used far beyond the scope in which it was originally introduced. Others have used it to describe properties of mix networks, and other anonymous channels and systems. Mix networks are a composed of special relays, named mixes, that forward messages while hiding the correspondence between their inputs and outputs.

For example the analysis of stop-and-go mixes [KEB98] crucially depends on a particular definition of anonymity sets. In this case the anonymity set is made up of users with a non-zero probability of having a particular role $\mathcal{R}$, such as being senders or receivers of a message. While DC networks give all participants a role with equal probability, this new definition explicitly recognises that different users might be more or less likely to have taken a particular role. Despite this, the cardinality of this set is still used to measure the degree of anonymity provided.

If different participants accounted in the anonymity set are not equally likely to be the senders or receivers, a designer might be tempted to distribute amongst many participants some possibility that they were the senders or receivers while allowing the real sender or receiver to have an abnormally high probability. The cardinality of the anonymity set is in this case a misleading measure of anonymity.

In [PK00] an effort is made to standardise the terminology surrounding anonymity research. Anonymity is defined as *the state of being not identifiable within a set of subjects, the anonymity set.* The 'quality' of an anonymity set is not just measured by its cardinality but also *anonymity is the stronger, the larger the respective set is and the more evenly distributed the sending or* *receiving, respectively, of the subjects within that set is.*

So if different potential senders or receivers can have different probabilities associated with them, anonymity is maximal when the probabilities are equal. This is not captured when the cardinality of the anonymity set is used as a metric for anonymity.

2.3 An information theoretic definition

The limitations of expressing the quality of anonymity provided by a system simply based on qualitative categories, or cardinality of sets, prompted us to formulate an alternative definition. The principal insight behind the metric is that the goal of an attacker is the unique identification of an actor, while at the same time the goal of the defender is to increase the attacker's workload to achieve this. Therefore we chose to define the anonymity provided by a system as *the amount of information the attacker is missing to uniquely identify an actor's link to an action.*

The term *information* is used in a technical sense in the context of Shannon's information theory [Sha48, Sha49]. Therefore we define a probability distribution over all actors α_i, describing the probability they performed a particular action. As one would expect, the sum of these must be one.

$$\sum_i \Pr[\alpha_i] = 1 \tag{2.1}$$

It is clear that this probability distribution will be dependent on the information available to an attacker and therefore intimately linked to the threat model. Therefore anonymity is defined vis-a-vis a particular threat model, and it is indeed a mistake to simply use the metric without reference to the threat model it is defined against.

As soon as the probability distribution above is known, one can calculate the anonymity provided by the system as a measure of uncertainty that the probability distribution represents. In information theoretic terms this is represented by the *entropy* of the discrete probability distribution. Therefore we call the *effective anonymity set size of a system*, the entropy of the probability distribution attributing a role to actors given a threat model. It can be calculated as:

$$\mathcal{A} = \mathcal{E}[\alpha_i] = \sum_i \Pr[\alpha_i] \log_2 \Pr[\alpha_i] \tag{2.2}$$

This metric provides a negative quantity representing the number of bits of information an adversary is missing before they can uniquely identify the target.

A similar metric based on information theory was proposed by Diaz *et al.* [DSCP02]. Instead of directly using the entropy as a measure of anonymity, it is normalised by the maximum amount of anonymity that the system could provide. This has the disadvantage that it is more a measure of fulfilled potential than anonymity. An anonymity size of 1 means that one is as anonymous as possible, even though one might not be anonymous at all.[1]

The non-normalised entropy based metric we propose, intuitively provides an indication of the size of the group within which one is hidden. It is also is a good indication of the effort necessary for an adversary to uniquely identify a sender or receiver.

2.3.1 Application: the pool mix

The information theoretic metric allows us to express subtle variations in anonymity and compare systems. To a limited extent it also allows us to understand properties of larger systems by understanding the anonymity provided by their components.

A typical example where the conventional, possibilistic and set theoretic definitions fail, is measuring the anonymity provided by a *pool mix.* A pool mix works in rounds. In each round it accepts a number of messages N and puts them in the same pool as n messages previously stored. Then N messages are uniformly and randomly selected out of the pool of $N + n$ messages and sent to their respective destinations. The remaining n messages stay in the pool.

[1] A similar metric applied to block ciphers would be to divide the actual entropy of the cipher, given all the attacks known, by the key size, which represents the maximum security potentially provided. Such a metric would indicate how invulnerable the block cipher is to attacks, but would not differentiate between the security of a block cipher with key sizes of 40 bits or 128 bits.

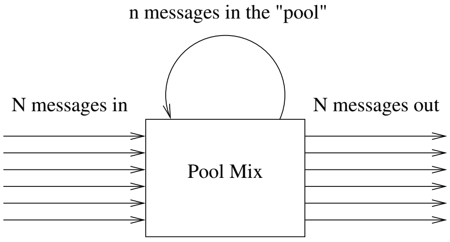


Figure 2.1: The pool mix

The question we will try to address is how, given a message coming out of the mix, we can determine the sender anonymity set for it. It is clear that if we follow the approach based on set cardinality, including all potential senders, the anonymity would grow without limit and would include all the senders that have ever sent a message in the past. Therefore it is crucial to use a metric that takes into account the differences between the probabilities of different senders.

We assume that at each round a different set of N senders each send a message to the pool mix. We also assume that, at the very first round, the pool is filled with cover messages from the mix itself. We are going to calculate the probability associated with *each sender* being the sender of some message coming out at round k. By convention we assume that dummy messages inserted in the message pool at round 0 are sent by the mix itself. Figure 2.2 shows the tree used to derive the probability associated with each user, depending on the round in which they sent their message.

Given a message received at round k, the probability that it was sent by a particular sender at round x (where $x \leq k$) is:

$$\Pr[\text{User at round } x, \text{ where } x > 0] = \frac{1}{N+n}\left(\frac{n}{N+n}\right)^{k-x} \tag{2.3}$$

$$\Pr[\text{Introduced by the Mix at round } x = 0] = \frac{n}{N+n}\left(\frac{n}{N+n}\right)^{k-1} \tag{2.4}$$

One can check that the above probability distribution always sums to one

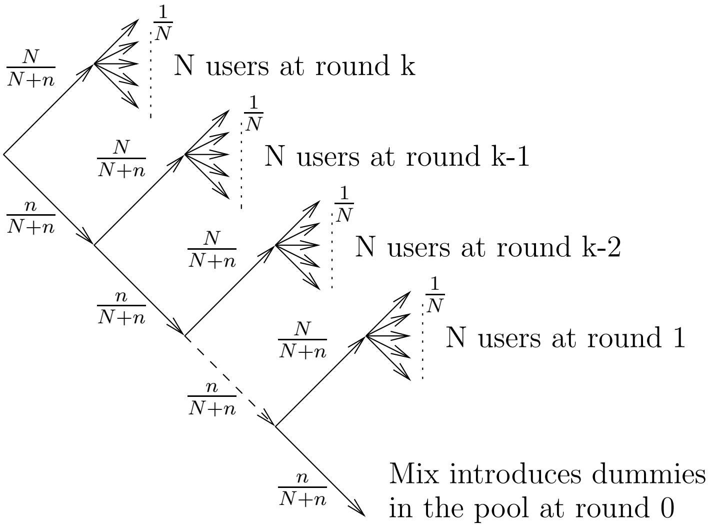


Figure 2.2: Tree representing the pool mix probability distribution

(for clarity we set $p = \frac{n}{N+n}$).

$$\forall k > 0, \left[\sum_{x=1}^{k} N\frac{p}{n}p^{k-x}\right] + p^k \tag{2.5}$$

$$= (1-p)\sum_{j=0}^{k-1} p^j + p^k \tag{2.6}$$

$$= (1-p)\frac{1-p^k}{1-p} + p^k = 1 \tag{2.7}$$

It is important to keep in mind that the increase in anonymity that will be measured is associated with an increase in the average latency of the messages. Using the above probability distribution we can calculate the mean number of rounds a message will be in the pool before being sent out. This represents the latency introduced by the mix. The expected latency μ and its variance σ^2 can be derived to be:

$$\mu = 1 + \frac{n}{N} \tag{2.8}$$

$$\sigma^2 = \frac{n(N+n)^2}{N^3} \tag{2.9}$$

Given this probability distribution we can now calculate the effective sender anonymity set size $\mathcal{A}_k$ of the pool mix:

$$\mathcal{A}_k = \sum_{x=1}^{k} N \times \Pr[\text{round } x] \log \Pr[\text{round } x] + \Pr[\text{round } 0] \log \Pr[\text{round } 0] \tag{2.10}$$

$$= p^k \log p^k + N \sum_{j=0}^{k-1} \frac{p}{n} p^j \log \frac{p}{n} p^j \tag{2.11}$$

$$= p^k \log p^k + \frac{Np}{n} \left[\left(\log \frac{p}{n} \right) \left(\sum_{j=0}^{k-1} p^j \right) + (\log p) \left(\sum_{j=0}^{k-1} j p^j \right) \right] \tag{2.12}$$

$$= p^k \log p^k + \frac{Np}{n} \left[\left(\log \frac{p}{n} \right) \left(\frac{1-p^k}{1-p} \right) + (\log p) \left(\frac{1 - kp^{k-1} + (k-1)p^k}{(1-p)^2} \right) \right] \tag{2.13}$$

We are particularly interested in the value of the effective anonymity set size after a very large number of rounds k therefore we calculate the above as k tends to infinity (note that $\lim_{k\to\infty} p^k = 0$).

$$\lim_{k\to\infty} p^k \log p^k = \lim_{k\to\infty} k p^k \log p = 0 \tag{2.14}$$

Therefore the anonymity of a pool mix after a very large number of rounds tends to:

$$\mathcal{A}_\infty = \lim_{k\to\infty} \mathcal{A}_k = \frac{Np}{n} \left[\left(\log \frac{p}{n} \right) \left(\frac{1}{1-p} \right) + (\log p) \left(\frac{p}{(1-p)^2} \right) \right] \tag{2.15}$$

$$= -(1 + \frac{n}{N}) \log(N+n) + \frac{n}{N} \log n \tag{2.16}$$

Note that in the special case $n = 0$, namely the pool size is zero, the anonymity of the mix becomes $\mathcal{A}_\infty = -\log N$. This is indeed the entropy of the uniform distribution over N elements, and intuitively what we would expect the anonymity of a threshold mix to be. Richard E. Newman has proved the same result using a different method, and which was also used by Serjantov to calculate the anonymity of timed pool mixes [SN03].

If we express the pool size n as a function of the input size N, such that $n = lN$ we can rewrite (2.16), in a more intuitive manner: $\lim_{k\to\infty} \mathcal{A}_k = -\log N + l \log l - (1+l) \log(1+l)$. A pool mix with a pool size equal to its input size (that is $l = 1$) will therefore always provide two additional bits of anonymity, in comparison with a threshold mix of the same input size.

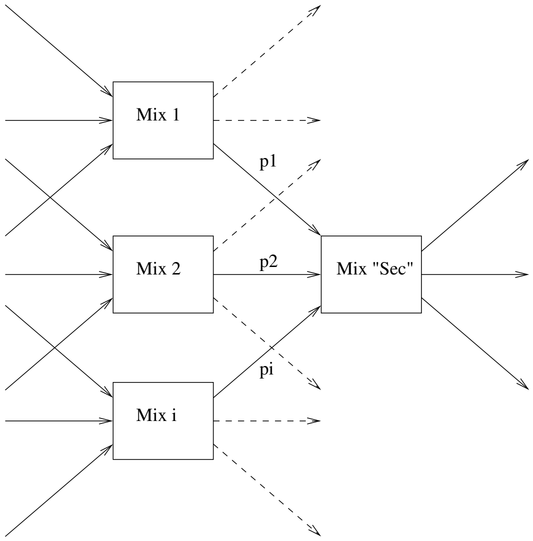


Figure 2.3: Mix composition

2.3.2 Application: composing mixes

Given a particular arrangement of anonymous communication systems it is possible to calculate the overall anonymity provided just by knowing the anonymity provided by the components and how they are connected. As before, the basic assumption is that different inputs originate from different senders, and are delivered to different receivers. Such an assumption is not realistic, but the resulting calculation provides a useful upper bound on the anonymity that can be provided by such a system. Furthermore we do not allow for cycles and feedback in the arrangement of anonymity systems.

Assume there are l mixes each with effective sender anonymity size S_i where $0 < i \leq l$. Each of these mixes sends some messages to a mix we call *sec*. The probability a message received by mix *sec* was sent by mix i is defined as p_i where $0 < i \leq l$ and $\sum_i p_i = 1$. Using our definition it is clear that the sender anonymity provided by mix sec is $\mathcal{A}_{\text{sec}} = \sum_i p_i \log p_i$, namely the entropy of the probability distribution describing where messages have come from.

The effective sender anonymity size can be derived for the whole system. We denote as p_{ij} the input probability distributions of the different elements of mix i, and the function $f(i)$ provides the number of the elements of mix i

$$\mathcal{A}_{\text{total}} = \sum_{i=1}^{l} \sum_{j=1}^{f(i)-1} p_{ij} p_i \log p_{ij} p_i = \sum_{i=1}^{l} p_i \sum_{j=1}^{f(i)-1} p_{ij} \log p_{ij} p_i \tag{2.17}$$

$$= \sum_{i=1}^{l} p_i \left(\sum_{j=1}^{f(i)-1} p_{ij} \log p_{ij} + \sum_{j=1}^{f(i)-1} p_{ij} \log p_i \right) \tag{2.18}$$

$$= \sum_{i=1}^{l} p_i \left(\mathcal{A}_i + \log p_i \right) \tag{2.19}$$

$$\mathcal{A}_{\text{total}} = \sum_{i=1}^{l} p_i \mathcal{A}_i + \mathcal{A}_{\text{sec}} \tag{2.20}$$

Therefore it is possible just by knowing the sender anonymity sizes of the intermediary mixes and the probabilities that messages are sent along particular links, to calculate the overall anonymity provided by the system. As pointed out before, it is not possible to apply this calculation to networks with cycles. This is the subject of the next section.

2.3.3 A framework for analysing mix networks

In chapter 7 we will analyse the anonymity different mix network topologies provide. We consider the effective sender anonymity set size of a message, as the entropy of the probability distribution describing the likelihood particular participants were senders of the message.

We analyse properties of mix networks in which a core set of nodes is dedicated to relaying messages, and can define the points at which messages enter the mix network, and leave the mix network. Since networks of mixes will generally contain loops, it is not possible to apply directly the formulae for composing mixes presented in section 2.3.2 to calculate the anonymity provided by the network. Instead we calculate the probability that input messages match particular output messages directly, approximating the routing of messages as a Markov process (see chapter 7 for details).

The model considered is the following. A message m_e exits the mix network at time t_e from node n_e. The network is made out of N mix nodes, n_1 to n_N. Messages m_{ij} are injected at node n_i at time t_j. The task of an attacker is to link the message m_e with a message m_{ij}. Figure 2.4 summarises this model.

We consider the probability distribution

$$\begin{aligned} p_{ij} &= \Pr[m_e \text{ is } m_{ij}] \\ &= \Pr[m_e \text{ is } m_{ij} | m_e \text{ inserted at } n_i] \times \Pr[m_e \text{ inserted at } n_i] \end{aligned} \tag{2.21}$$

that describes how likely the input messages in the network are to have been message m_e. We can express this as the probability a node n_i was used to inject a message, multiplied by the conditional probability a particular message m_{ij} injected at this node is m_e. The entropy of the probability distribution p_{ij} is the effective sender anonymity set of the message. Because of the strong additive property of entropy we can calculate it as:

$$\begin{aligned} \mathcal{A} &= \mathcal{E}(p_{ij}) \\ &= \mathcal{E}(\Pr[m_e \text{ inserted at } n_i]) \\ &+ \sum_{x=1}^{N} \Pr[m_e \text{ inserted at } n_x] \times \mathcal{E}(\Pr[m_e \text{ is } m_{ij} | m_e \text{ inserted at } n_x]) \end{aligned} \tag{2.22}$$

An attacker might attempt to reduce the anonymity by subjecting the network to traffic analysis to reduce the uncertainty of $\Pr[m_e \text{ inserted at } n_i]$. We shall therefore name $\mathcal{A}_{\text{network}} = \mathcal{E}(\Pr[m_e \text{ inserted at } n_i])$, the anonymity provided by the network. This quantifies how effectively the traffic injected to or ejected from particular nodes is mixed with traffic from other nodes. Given a particular threat model if no technique is available for the attacker to reduce the uncertainty of $\mathcal{A}_{\text{network}}$ beyond her a priori knowledge, we can say that *the network is resistant to traffic analysis with respect to that particular threat model.* Results in [Ser04] show how a global passive adversary can use observations of a mix network to reduce $\mathcal{A}_{\text{network}}$.

The attacker can also try to reduce the anonymity set of the message by reducing the uncertainty of the probability distribution describing the traffic introduced at the mix nodes, $\Pr[m_e \text{ is } m_{ij} | m_e \text{ inserted at } n_x]$. The attacker can do this by using additional information about m_e and m_{ji}, such as the time t_e the message comes out of the network or t_j the time it was injected into the network. She can also control these times by flooding nodes, or stopping messages arriving to the initial nodes. It is worth noting that a

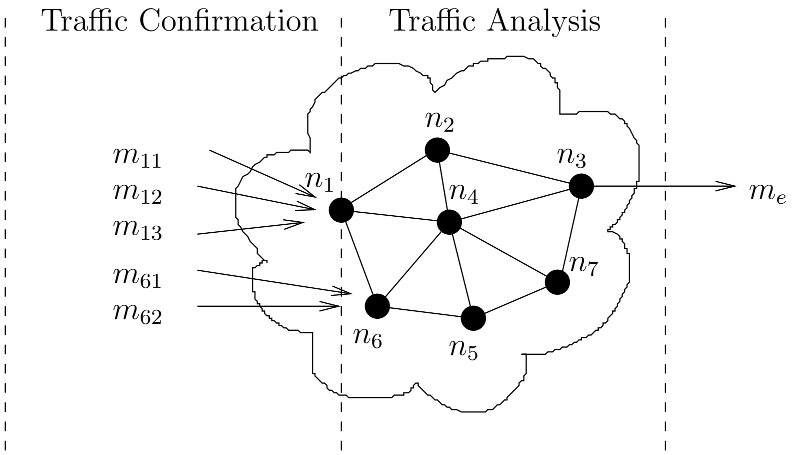


Figure 2.4: Traffic analysis and traffic confirmation of mix networks.

network might protect users from traffic analysis, but still provide inadequate anonymity because of side information leaked by messages as they enter and exit the mix network. Side information is not limited to timestamps, but can also be associated with the protocol or mechanism used, client type, unique identifiers or route length indications observed at the edges of the mix network. Attacks that try to link messages using such side information leaked at the edges of the network, instead of tracing the message through the network, are called *traffic confirmation attacks* [RSG98].

In chapter 7 we study how to protect mix networks from traffic analysis by using selected topologies, long enough paths and adequate volumes of traffic. Protection against traffic confirmation is much more a design specific issue and heavily depends on the efforts of the designers to avoid partitioning attacks and leaking information at the edges of the anonymous communication network.

2.4 Summary

In this chapter we have defined the objectives of anonymous communication, and the threats against it. During the rest of this work we are going to assess the security of the systems proposed against a traditional adversary that can eavesdrop on communication links, performs active attacks and controls a subset of the network nodes. We are also going to take into account the threat of an adversary with compulsion powers, that can ask nodes to decode material, or provide activity logs.

We presented a metric of anonymity, based on Shannon's information theory. This represents how much information an adversary is missing to identify the sender or the receiver of a target message. The pool mix was used as an example to illustrate how the metric can be applied, and a framework was proposed for analysing the anonymity of complete anonymous networks.

Chapter 3

Primitives and building blocks

"It is a tragicomic fact that our proper upbringing has become an ally of the secret police. We do not know how to lie. "

The unbearable lightness of being. — Milan Kundera

The technical work presented in this thesis makes routine use of an array of cryptological tools. While it is beyond the scope of this dissertation to explain these in detail, it is important to present their capabilities, and limitations. Without going into the theory of cryptology (those who wish can find it in [Sch96, MOV96, Gol01]) we will concisely present the main primitives, and constructions that will be used as basic building blocks in the systems presented in the next chapters.

3.1 Symmetric cryptographic primitives

When describing Mixminion, in chapter 5, we will make extensive use of hash functions, stream ciphers, and block ciphers. These primitives are quite classic in the sense that they predate the invention of public key cryptography. On the other hand we will use them to provide properties that are not usual, such as bitwise unlinkability, instead of secrecy. In this section we will present each of them, and highlight the properties that are important to us.

3.1.1 Hash functions

A cryptographic hash function is a function h that takes an input x of arbitrary size, and outputs a fixed output $h(x)$. Hash functions are assumed to be public, therefore given x anyone can compute $h(x)$. The main three properties that are desirable in a hash function are *pre-image resistance*, *weak collision resistance* and *strong collision resistance.*

Pre-image resistance means that given $h(x)$ it is difficult to extract any bits of x.

Weak collision resistance means that given x it is difficult to find y such that $h(x) = h(y)$.

Strong collision resistance means that it is difficult to find any x and y such that $h(x) = h(y)$.

Pre-image resistance does not protect x if the domain over which it is defined can be exhaustively searched. An attack that computes $h(x)$ for all possible x in order to find a match with $h(y)$ is called a *dictionary attack.*

It can be shown that, if a hash function has an l-bit output, a value y with $h(y)$ that matches a fixed $h(x)$ can be found in $\mathcal{O}(2^l)$. On the other hand finding any two x and y such that $h(x) = h(y)$ is much cheaper because of the birthday paradox, and takes $\mathcal{O}(\sqrt{2^l})$.

The main cryptographic hash function used in this work is the standard SHA-1 [SHA93]. Its output is 160 bits long (20 bytes), and offers good strong collision resistance properties ($\sim 2^{80}$ hashes to break). Being a National Institute of Standards and Technology (NIST) standard it is readily available in most implementations of cryptographic libraries. It is believed to be secure, since no known attacks have been published against it.

Since many protocols make heavy use of hash functions, it is important to use variants of them for operations within the protocols that have a different purpose. This stops some attacks that use one party in a protocol as an oracle that performs operations using hashed secrets. In order to construct a secure variant h_1 of h it is sufficient to prepend a well-known string to the hashed bit-string, $h_1(x) = h(\text{`1'}, x)$. This creates differing hash functions quickly and cheaply.

3.1.2 Pseudo-random functions: stream ciphers

A pseudo-random function, or stream cipher s takes as its input a fixed length key k and generates an infinitely long (in theory) string of bits $s(k)$. Some properties of a stream cipher are:

Key secrecy means that given some of the output of the stream cipher $s(k)$ it is difficult to infer anything about the key k.

Pseudo-randomness means that for any party that does not know the key k, the output $s(k)$ of the cipher, in indistinguishable from a truly random bit-string.

Steam ciphers can be used to provide secrecy by XORing the output of the cipher with plaintext in order to generate the ciphertext. Such a scheme does not protect the integrity of the plaintext, since the adversary can XOR into the ciphertext any bit-string that will be XORed with the plaintext after decryption. Such attacks are called *attacks in depth*.

Stream ciphers could be considered the poor man's one time pad [Bau97, p.144–146]. From a short secret a long pseudo-random stream can be generated. The drawback is that the stream generated from a stream cipher does not provide *plausible deniability*, namely the ability to claim that any plaintext was encrypted. An adversary could compel a user to reveal their keys, and check that the key stream is indeed generated from them. It is impossible to find a key generating any random stream because the key is shorter than the stream generated.

3.1.3 Random permutations: block ciphers

A block cipher takes as an input a secret key and a plaintext bit-string of length l (called the block length). It then outputs another l bit-string called the ciphertext. The mapping between the inputs and outputs of a block cipher is a secret permutation of the l-bit space, for a given key k. For this reason, the operation of a block cipher can be "undone", by a decryption operation that takes the key k and a ciphertext and outputs the original plaintext. The main properties of a block cipher are:

Key secrecy means that given any number of known pairs of plaintext and ciphertext an adversary cannot extract the key k. In other words no operation leaks the key.

Permutation pseudo-randomness means that given any number of pairs of known plaintext and ciphertext it is not possible for the adversary to infer any other mappings of the random permutation.

The block cipher used in this dissertation is the Advanced Encryption Standard (AES) [AES01] as specified by NIST. AES takes keys of 128, 192 or 256 bits in length, and acts on blocks of 128 bits. Being a NIST standard, high quality AES implementations are available in most cryptographic libraries and there are no known plausible attacks.

3.2 Cryptographic constructions

Beyond the primitives themselves, some standard ways of combining them are frequently used.

3.2.1 Block cipher modes

A block cipher can be used to encrypt a single block of text, but encrypting multiple blocks requires special attention. A set of special modes has been designed to perform such operations, each with different properties. The ones that are relevant to our research are briefly presented. We denote as P_i the i^{th} block of plaintext and C_i the block of ciphertext. The block cipher encryption operation is denoted by E_k and decryption by D_k, where k is the key.

Electronic Code Book (ECB) mode just divides the message to be encrypted into blocks, the size of the block length of the cipher, and encrypts them separately.

$$C_i = E_k \left\{P_i\right\} \tag{3.1}$$

Cipher Block Chaining (CBC) mode divides the plaintext into blocks and XORs the encryption of the previous block into the plaintext before it is encrypted. The first block transmitted is a randomly chosen initialisation value (IV), to make sure that repeated messages look different to someone who does not have the key.

$$C_0 = \mathrm{IV} \tag{3.2}$$

$$C_i = E_k \{C_{i-1} \oplus P_i\} \tag{3.3}$$

It is worth noting that while the encryption in CBC is sequential the decryption can be performed in random order. For the same reason errors in the ciphertext do not propagate beyond the block that contains the error itself and the next block. The randomised encryption, yielding different ciphertexts for the same key and plaintext, and the good protection of confidentiality, have made CBC one of the most popular encryption modes.

Counter Mode turns a block cipher into a random function (stream cipher). An initialisation vector is chosen at random and transmitted in clear. Then blocks containing the IV XORed with a counter are encrypted, to generate the stream that will be XORed with the plaintext.

$$C_0 = \mathrm{IV} \tag{3.4}$$

$$C_i = P_i \oplus E_k \{\mathrm{IV} \oplus i\} \tag{3.5}$$

Counter mode does not propagate errors, except if they are in the IV.

The careful reader will note that decryption of the message uses the encryption operation of the block cipher again and the decryption of the block cipher is not needed. Under these circumstances one could be tempted to replace the block cipher with a hash function containing the key and the XOR of the IV and the counter.

3.2.2 Large block ciphers: BEAR

In order to avoid having to choose a block cipher mode, an alternative is to construct a block cipher large enough to encrypt the whole message as one block. This has the advantage that the error propagation, in case the ciphertext is modified, is total and unpredictable. An adversary will not be able to guess, with probability better than random, any of the resulting bits of plaintext.

BEAR [AB96] is a construction proposed by Anderson and Biham that builds such a block cipher from a hash function $h(x)$ and a random function $s(k)$. Their paper uses Luby-Rackoff's [LR88] theorem to prove that such a construction is secure against chosen ciphertexts and plaintext attacks. BEAR is an unbalanced Feistel structure [SK96], which divides the block into the left hand size whose length is the key size of the stream cipher $|sk|$ and a right hand side of arbitrary length $l - |sk|$.

$$(L_{|sk|}, R_{l-|sk|}) = M \tag{3.6}$$

$$L' = L_{|sk|} \oplus h(R_{l-|sk|} \oplus K_1) \tag{3.7}$$

$$R' = R_{l-|sk|} \oplus s(L') \tag{3.8}$$

$$L'' = L' \oplus h(R' \oplus K_2) \tag{3.9}$$

$$\text{return } (L'', R') \tag{3.10}$$

An attack is presented in [Mor] against the key schedule in the original BEAR paper. It can be fixed by making the sub keys K_1, K_2 used equal to the master key K. That way the decryption operation is the same as the encryption operation.

An important feature of BEAR is its all-or-nothing nature. None of the plaintext can be retrieved until all the bits of ciphertext are known. Furthermore, if any ciphertext is modified by an attacker, without knowledge of the key the message will decrypt into a random stream. In many cases even a key-less (or with a fixed globally known key) BEAR transform can be used. For example performing a key-less BEAR transform on a whole message, and then encrypting the first block using a block cipher, or by XORing a random value into it, guarantees the confidentiality of the whole message. Similar schemes can be constructed with asymmetric ciphers. Generally, a rule of thumb when reasoning about BEAR is that every bit of the ciphertext depends on every bit of the plaintext, and vice versa. All bits of the ciphertext and plaintext also depend on all bits of the key when encryption or decryption operations are performed. Modification of any of these will cause an

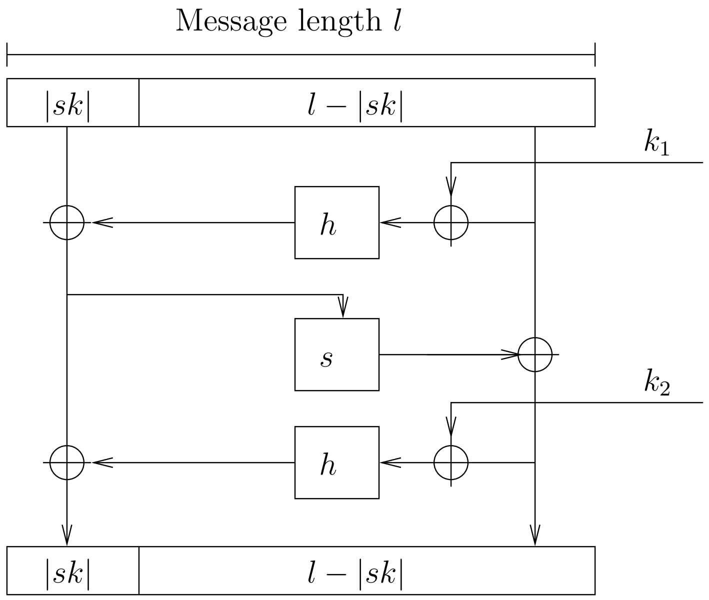


Figure 3.1: The structure of the BEAR block cipher

unpredictable stream to be output. Constructions with similar properties are described by Rivest as all-or-nothing transforms [Riv97] and their use is discussed in [JSY99].

3.2.3 Message authentication codes for integrity

A message authentication code (MAC) is a short bit-string that is derived from a message and a key. The parties with the key can compute the MAC, which also allows them to verify it, while the parties without the key cannot distinguish a valid MAC from a random one. There are many possible construction of MACs. Two of them are presented here:

- A block cipher in CBC mode can be used to compute a MAC. The initialisation vector is set to a globally known value (usually a string of zeros) and only the last block of the CBC ciphertext is kept as the MAC. Using AES would result in a 128 bit MAC.

- A hash function can be used to compute a MAC by appending and

> prepending the key to the text to be checked.
>
> $$\text{HMAC}_k(x) = h(k, x, k) \tag{3.11}$$
>
> It is important to append the key at the end since this prevents an attacker using implementation specificities of SHA-1, to update the message and the MAC, therefore producing a valid MAC without knowing the key [FS03].

While a MAC can be used by any of the parties sharing the key to assure themselves that another party knowing the shared key has authenticated the particular message, they cannot prove this to a third party. In particular if Alice and Bob share a symmetric key, it is not possible for Alice to show a message with a MAC to Charlie, a third party, and convince him that it originated from Bob. Since the key is shared, Alice could equally have constructed the message and the valid MAC. In order to convince third parties of the authenticity of messages, and prove that they originated from a particular principal, some form of digital signature has to be used. Therefore the difference between digital signatures and MACs is that the former provide *non-repudiation.* As argued in Mike Roe's thesis [Roe97], one could consider the complementary property to be *plausible deniability*, which will be discussed in detail in section 3.4.

3.3 Asymmetric cryptographic primitives

All the keyed primitives in the previous section require keys to be shared between Alice and Bob to perform encryption and decryption or authentication and verification. In this section we present asymmetric cryptographic primitives that rely on pairs of private and public keys.

3.3.1 Diffie-Hellman exchange

In [DH76] Whitfield Diffie and Martin Hellman presented for the first time in public, a system that allowed two participants to establish a shared secret, using only publicly available information. The Diffie-Hellman exchange bases its security on the difficulty of the discrete logarithm problem in a finite field. In the case of integers, it is trivial to compute $E = g^x \mod p$ for a prime p, a generator g and a random number x. On the other hand it is computationally very difficult to determine x by computing the logarithm $x = \log_g E \mod p$. In other words, modular exponentiation can be thought as being a one way function with special properties.

In order to perform a Diffie-Hellman exchange Alice and Bob share g and p that are public parameters of the crypto-system. These can be reused, and are globally known. Alice chooses a random string x, her private key, computes $g^x \mod p$ and publishes it as her public key. Bob does the same with the private key y and the public key $g^y \mod p$. Alice and Bob can then exchange their public keys, or simply publish them, and can compute the shared key $g^{xy} \mod p$. This shared key can be used with any of the primitives described in the previous section to protect the confidentiality and integrity of their communications.

Asymmetric, or public key, cryptography, does not totally solve the problem of secure key distribution. It simply transforms it from a confidentiality problem to an integrity problem. Alice must be sure that what she thinks is Bob's public key is indeed Bob's public key, otherwise she could be talking to someone else, such as the eavesdropper. Public key infrastructures [LA99], public key registers [ACL+99], and webs of trust [Zim95] have been proposed to solve this problem.

The Diffie-Hellman crypto-system, and generally systems based on discrete logarithms have quite a nice feature: the private key can be easily generated from public information and a strong pass-phrase. Therefore, there is no need to ever store a private key, since it can simply be regenerated when a user needs to use it. The public key can also be trivially recomputed given the secret pass-phrase and the public parameters of the system.

3.3.2 The El Gamal encryption system

The Diffie-Hellman scheme allows two parties to share a symmetric key. El Gamal developed a system that allows the exchange of encrypted messages [El 85]. The public key $y = g^x \mod p$ is used for encryption and the private key x is used for decryption. If Bob wants to send a message M to Alice he picks a random secret k, and encodes it in the following way.

$$C = (a, b) = \left(g^k \bmod p, y^k M \bmod p\right) \tag{3.12}$$

The first term a can be seen as the "hint" that allows Alice to compute the session key g^{kx} to unblind the second term b using division. El Gamal is a randomised encryption scheme, that produces different ciphertexts for the same plaintext thanks to the different session secrets k used. Its main drawback is that the size of the ciphertext is always the double of the size of the plaintext.

Two important operations that can be done to an El Gamal ciphertext without the knowledge of the key are *blinding* and *re-encryption.* These are used extensively in robust and verifiable mix constructions, described in section 4.2.8. Blinding is performed by raising a ciphertext (a, b) to a power d, resulting in (a^d, b^d). As a result the plaintext M is also raised to this power, and will be M^d. Since p is prime, it is trivial for someone knowing d to recover the plaintext M, but very difficult for anyone else. Re-encryption uses multiplication instead of exponentiations. The factors (g^d, y^d) are multiplied with the ciphertext, resulting in $(g^d a, y^d b)$. The decryption operation will still output the plaintext M. The ability to change the ciphertext in a way that is not predictable is used in many constructions to provide the bitwise unlinkability necessary to build mix systems.

3.3.3 The Rivest-Shamir-Adelman crypto-system

The RSA crypto-system [RSA78] relies on the difficulty of factoring composites of large primes to provide its security. A composite $n = p \times q$ is computed, and made public, while the two primes p and q are kept secret. A value e where $1 < e < (p-1)(q-1)$ is chosen at random, and d such that $d \times e = 1 \bmod (p-1)(q-1)$ is efficiently calculated. The public key is (e, n), while (d, n) is the secret key.

In order to encrypt a message M for the public key (e, n) one simply performs an exponentiation modulo n. The ciphertext is therefore $M^e \mod n$. To decrypt, the message is simply raised to the power of the decryption key,

$M^{ed} \bmod n = M \bmod n$.

Digital signatures can also be implemented. The public verification key is denoted (v, n) while the signature key is (s). The signer raises the message to be signed to the power s and the verifier checks the signature by raising it to the power v. All operations are performed modulo n.

Digital signatures provide integrity properties and non-repudiation properties: if the public key of Bob is well known, Alice can prove to a third party that Bob has signed a message. Often this property is not desirable but for technical reasons other integrity primitives, such as message authentication codes, cannot be used. One can adapt protocols using digital signatures to provide plausible deniability by publishing the private keys as the last step of the protocol.

Chaum invented in [Cha83] a way to provide a valid signature on a unknown plaintext. This can be used to provide unlinkable credentials, special forms of which can be used to construct electronic coins. The setting is simple. Alice wants to make Bob, who controls a key pair $(s, n), (v, n)$, sign a message M, but does not want Bob to be able to link the message M with Alice. Alice can choose a random nonce b and submit to Bob the ciphertext $b^v M \bmod n$. Bob can then perform the normal RSA signature operation and return $(b^v M)^s \bmod n = bM^d \bmod n$. Alice knows b and can therefore get a valid signature $M^d \bmod n$ just by dividing the returned value. This property can be used to build anonymous credential systems or to attack protocols that use “raw” RSA.

3.3.4 Strengthening the primitives

Soon after the introduction of the public key techniques described, it was realised that by themselves they were not as secure as conventional “encryption”. In particular the mathematical structure that they relied upon, could be used to mount adaptive active attacks, blinding and re-encryption, to leak information about the plaintext.

A lot of research has focused on trying to formally define security for encryption, and to try and minimise the potential leakage of information out of public key encryption systems. Semantic security [GM84] means that no useful information can be extracted from the ciphertext about the plaintext. Chosen ciphertext security means that no information about the plaintext can be extracted if modified ciphertexts are submitted to a decryption oracle. This is quite close to the idea of plaintext-aware encryption that detects any modification to the plaintext. Sometimes this property is also called non-malleability [DDN91].

Technically these constructions pre-process the plaintext to give it a particular structure. Luby-Rackoff structures are used to distribute each bit of information across all other bits, so that all or none of the plaintext can be recovered. Hashes are also used to check the integrity of the message. The PKCS#1 [PKC02] standard defines the Optimal Asymmetric Encryption Procedure (OAEP [BR95]) that should be used when encrypting with RSA.

Another issue relevant to anonymity engineering is that encrypting with an asymmetric crypto-system might not leak any information about the plaintext, but could be leaking information about the public key used. Systems that effectively hide the public key used to perform the encryption are called *key private* [BBDP01].

3.4 Plausible deniability

Plausible deniability is the security property which ensures that a principal cannot be linked to some action with an appropriate level of certainty. The level of certainty used is usually *beyond reasonable doubt* in the case of a criminal court, *on the balance of probability* in a civil court. Michael Roe argues that plausible deniability is the complementary property of non-repudiation, the inability to deny that an action has been associated with a principal, or the ability to prove to third parties that an action was linked to a principal [Roe97].

3.4.1 Deniable encryption

Plausibly deniable encryption is the inability to prove that a particular ciphertext decrypts to a particular plaintext. A system that provides perfectly deniable encryption is the one time pad. Since the key is as long as the plaintext and ciphertext, a key can always be constructed to decrypt any ciphertext to any plaintext. Furthermore the key provided will be indistinguishable from noise, and therefore any other possible key that might have been used. It is important to notice that under compulsion the sender and the receiver should release the same fake key, leading to some fake plaintext, otherwise they might be uncovered. This might be difficult to arrange if the ciphertext has not yet reached the receiver.

From the example above it should be clear that the mathematics, and the procedures, surrounding such systems must be carefully designed to yield plausible outcomes. Furthermore what is plausible is not simply the result of the mathematics, but also made plausible by the fact that the human procedures around them are plausible.

A System that does not simply make the content of messages deniable but allow one to deny their presence have been developed by Clayton and Danezis [CD02]. Chaffinch is a system that supports the presence of many streams of data, hidden within each other. It follows closely chaffing and winnowing introduced by Rivest [Riv98, BB00]. .

The principal idea behind Chaffinch is that streams of data are made to look like noise using an all-or-nothing transform, and then are multiplexed together. A cover message that can be revealed under compulsion, is always included to make the act of communication itself plausible. It is impossible for an adversary not knowing the keys to unwrap the other channels and tell if any further messages are present or not. The intended recipient can always untangle the streams by recognising the labels of packets within the stream using different secret keys. An interesting feature of Chaffinch is that the keys are necessary for decoding streams, but any third party can mix streams without directly knowing them.

An equivalent system for storage, instead of transmission, is the steganographic file system, as proposed in [ANS98] and implemented in [MK99]. Again, a user stores files indexed as tuples of file names and keys. An adversary that does not have a key cannot find out if a particular file is present in the file system or not. While the system works very well against an adversary that just gets one snapshot of the file system, it does not protect against an adversary that gets many snapshots at different times, and compares them to find differences. Such an adversary will query the user of the system until all the differences that are due to file insertions are explained.

In [Bea96, CDNO97] some deniable public key encryption schemes are presented. It is also argued that deniable encryption is an important element of *receipt-freeness* that is necessary to build fair election schemes. An attempt to generalise plausible deniability to general multi-party protocols is made in [CDNO97]. These aim to avoid transforming the party's inputs and outputs into unforgeable commitments, which would allow an adversary with coercion powers to check any answers given under compulsion.

3.5 Forward security

Forward security is the property of a system which guarantees that after a certain time, or protocol step, none of the security properties of the current state of the system can be violated. Forward security relies heavily on securely deleting information such as key material.

Ross Anderson points out in [And97] that the name *forward security* is deceptive. He argues that the term would better be suited to describe the property of a system to become secure in the future, after it has been compromised. We shall call this property *self-healing*, and define it as the ability of a system to regain its security properties after an attack, given that the attack has ceased. In practise this property raises the cost of an attacker, and usually forces them to be on-line all the time, recording all the communications as they happen and duplicating all computations.

The simplest way to achieve forward secure confidential communications is to perform *key updating* at regular time intervals. Key updating transforms the secret shared key using a one way function, such as a cryptographic hash function, and then makes sure that the old keys are deleted. Hashing the key using SHA-1 would be sufficient, but another common technique is to encrypt the key of a block cipher under itself and use the resulting ciphertext as the new key. Since the transform is one way, an adversary that captures the current key read any previous communications.

The ephemeral Diffie-Hellman exchange provides forward secure encryption, without the need to previously share any secrets. Alice and Bob simply compute some fresh public keys and exchange them. As soon as they have generated the shared key, they securely delete their respective private keys. The session key can be used to exchange confidential messages, and periodically key updating can be performed as described above. Note that such a scheme is open to man in the middle attacks, unless digital signatures are used to sign the ephemeral public keys exchanged.

We can extend the simple example provided above to also be self-healing. Instead of just hashing the session key to perform the key updating operation, all the plaintext transmitted in the channel is hashed, to generate the new session key. Therefore, even if an attacker gets access to the key, they will have to monitor all communications, and recompute all the keys in order to maintain their ability to eavesdrop on the channel. SSL and TLS can be made to behave like this.

In the same paper Anderson also introduces the concept of forward secure digital signatures. These behave exactly like digital signatures but provide a way of updating the secret signing keys and public verification keys so that signatures in past epochs cannot be forged. A number of such schemes have since been proposed [BM99, AR00, IR01].

A similar idea has been proposed for asymmetric encryption schemes. Forward secure encryption techniques usually make use of identity based schemes, and modify them to provide a succession of public and private keys. Details of such schemes are presented in [CHK03, DKXY02, DFK$^+$03]. The *intrusion-resilience* and *key-insulation* properties discussed in these papers are very similar to *self-healing* described above.

Some work has also been done on using forward secure techniques to detect corruption of signatures [Itk03] or server logs [BY03] after a successful attack that has leaked private information.

3.6 Summary

In this chapter we presented a selection of symmetric and asymmetric cryptographic primitives of particular significance to anonymity systems. The BEAR cipher and RSA OAEP encryption will be used extensively in the design of Mixminion, our anonymous remailer design, presented in chapter 5.

Plausible deniability techniques for message transport, storage and encryption are also reviewed, due to the resemblance of this property to anonymous communications. Furthermore forward secure systems for link encryption and signatures are discussed. In chapter 6 we will present how forward secure mixing can be implemented to make the work of an attacker with compulsion powers more difficult.

Chapter 4

Related work

> "*One technique used in COINTELPRO*[1] *involved sending anonymous letters to spouses intended, in the words of one proposal, to 'produce ill-feeling and possibly a lasting distrust' between husband and wife, so that 'concerns over what to do about it' would distract the target from 'time spent in the plots and plans' of the organisation*"
>
> *Final Report of the Select Committee to Study*
> *Governmental Operations with Respect to Intelligence*
> *Activities of the United States Senate*
> — 94th Congress, 2nd Session, 1976

Research on anonymous communications started in 1981 with Chaum's seminal paper "Untraceable electronic mail, return addresses, and digital pseudonyms" [Cha81]. Since then a body of research has concentrated on building, analysing and attacking anonymous communication systems. In this section we present the state of the art of anonymity systems at the time our research commenced. The principles that are extracted, and the attacks that are described have shaped our view of the field, and marked the design of the protocols presented in the rest of this thesis.

More recent research has been intrinsically linked and shaped by its interaction with our own work and will be presented alongside the actual work

[1] COINTELPRO: FBI counter intelligence programme, operating from 1956 to 1971.

in the relevant later chapters. The attacks against Tarzan presented in section 4.2.7 are original but do not deserve a whole chapter.

4.1 Trusted and semi-trusted relays

We will start presenting anonymous communications by introducing systems that rely on one central trusted node to provide security. They also provide a varying, but usually low, degree of protection against traffic analysis and active attacks.

4.1.1 `anon.penet.fi`

Johan Helsingius started running a trusted mail relay, `anon.penet.fi`, providing anonymous and pseudonymous email accounts in 1993. The technical principle behind the service was a table of correspondences between real email addresses and pseudonymous addresses, kept by the server. Email to a pseudonym would be forwarded to the real user. Email from a pseudonym was stripped of all identifying information and forwarded to the recipient. While users receiving or sending email to a pseudonym would not be able to find out the real email address of their anonymous correspondent, it would be trivial for a local passive attacker or the service itself to uncover the correspondence.

While protecting from a very weak threat model, the service was finally forced to close down through legal attacks. In 1996 the "Church of Spiritual Technology, Religious Technology Centre and New Era Publications International Spa" reported that a user of `anon.penet.fi` sent a message to a newsgroup infringing their copyright. Johan Helsingius, the administrator of `anon.penet.fi`, was asked to reveal the identity of the user concerned. The details of the case, that put enormous strain on the service, can be found in the press releases of the 23 September 1996 [Hel96b, Hel96c] or the information centre set up around this case [New97]. Reputation attacks were also experienced, when unfounded reports appeared in mainstream newspapers about the service being used to disseminate child pornography [Hel96a].

The service finally closed in August 1996 since it could no longer guarantee the anonymity of its users. The closure was quite significant for the privacy and anonymity research community. In the initial judgement the judge had ruled that "a witness cannot refrain from revealing his information in a trial" [Hel96c], even though an appeal was lodged on the grounds of privacy rights protected by the Finish constitution, and the fact that the information might be privileged, as is the case for journalistic material.

The concept that even non-malicious relay operators could be forced under legal or other compulsion, to reveal any information they have access to, provided a new twist to the conventional threat models. Honest relays or trusted nodes could under some circumstances be forced to reveal any information they held concerning the origin or destination of a particular communication. Minimising the information held by trusted parties is therefore not just protecting their users, but also acts to protect the services themselves.

4.1.2 Anonymizer & SafeWeb

Anonymizer[2] is a company set up by Lance Cottrell, also author of the Mixmaster remailer software, that provides anonymous web browsing for subscribed users. The Anonymizer product acts as a web proxy through which all web requests and replies are relayed. The web servers accessed, should therefore not be able to extract any information about the address of the requesting user. Special care is taken to filter out any "active" content, such as javascript or Java applets, that could execute code on the user's machine, and then signal back identifying information.

As for `anon.penet.fi`, the anonymity provided, depends critically on the integrity of the Anonymizer company and its staff. The service is less vulnerable to legal compulsion attacks, since no long-term logs are required to be kept, that could link users with resources accessed. Unlike email, users always initiate web requests, and receive the replies, and all records can be deleted after the request and reply have been processed. Records can be made unavailable to seize after just a few seconds.

SafeWeb is a company that provides a very similar service to Anonymizer.

[2] `http://www.anonymizer.com/`

The two main differences in their initial products, was that SafeWeb allowed the traffic on the link from SafeWeb to the user to be encrypted using SSL [DA99], and "made safe" active content in pages by using special wrapper functions. Unfortunately their system of wrappers did not resist a set of attacks devised by Martin and Schulman [MS02]. Simple javascript attacks turn out to be able to extract identifying information from the users. In the absence of any padding or mixing, a passive attacker observing the service would also be able to trivially link users with pages accessed. The study of this vulnerability has prompted the research presented in [SSW$^+$02].

Censorship Resistance

The threat model that SafeWeb wished to protect against was also very peculiar. The company was partly funded by the United States Central Intelligence Agency (CIA), and attempted to secure funding from The Voice of America and the Internet Broadcasting Bureau in order to "help Chinese and other foreign Web users get information banned in their own company *(sic)*"[3] [Sin01]. This claim explicitly links anonymous communications with the ability to provide censorship resistance properties. The link has since then become popular, and often anonymity properties are seen as a pre-requisite for allowing censorship resistant publishing and access to resources. No meticulous requirements engineering study has even been performed that proves (or disproves) that claim, and no cost benefit analysis has ever been performed to judge if the technical cost of an anonymity system would justify the benefits in the eyes of those interested in protecting themselves against censorship. Furthermore no details were ever provided, besides hearsay claims, about groups using the technology in a hostile environment, and their experiences with it. The latter would be particularly interesting given the known vulnerabilities of the product at the time.

The first paper to make a clear connection between censorship resistant storage and distribution is Anderson's Eternity service [And96]. Serjantov has also done some interesting work on how to use strong anonymous com-

[3] The Freudian slip confusing "country" with "company", and the way this goal can be understood in two opposing ways might be interpreted as quite telling of the nature of the CIA's interest in this product.

munications to provide censorship resistance [Ser02]. The system presented is, for very good technical and security reasons, very expensive in terms of communication costs and latency. Peer-to-peer storage and retrieval systems such as Freenet [CSWH00], FreeHaven [DFM00], and more recently GnuNet [BG03] also claimed to provide or require anonymous communications. Attacks against some anonymity properties provided by GnuNet have been found [Küg03]. Since the design of the three mentioned systems changes frequently it is very difficult to assess the security, or the anonymity they provide at any time. Feamster *et al.* have looked at different aspects of web censorship resistance in [FBH$^+$02, FBW$^+$03] by making use of steganography to send high volumes of data, and what they called "URL hopping" to transmit requests. Finally, aChord [HW02] presents concisely the requirements of a censorship resistant system, and attempts to build one based on a distributed hash table primitive.

4.1.3 Type I "Cypherpunk" remailers

Type I remailers, first developped by Eric Hughes and Hal Finney [Par96], are nodes that relay electronic mail, after stripping all identifying information and decrypting it with their private key. The first code-base was posted to the cypherpunks mailing list, which gave the remailers their nickname. The encryption was performed using the Pretty Good Privacy (PGP) public key encryption functions. The encoding scheme was also designed to be performed manually, using standard text and email editing tools. Many remailers could be chained together, in order for users not to rely on a single remailer to protect their anonymity.

Reply blocks were supported to allow for anonymous reply addresses. The email address of the user would be encrypted using the remailer's public key, and inserted in a special header. If a user wished to reply to an anonymous email, the remailer would decrypt it and forward the contents.

The type I remailers offer better resistance to attacks than the simple `anon.penet.fi` relay. No database that links real user identities with pseudonyms is kept. The addressing information required to reply to messages is included in the messages themselves, in an encrypted form.

The encryption used when the messages are forwarded through the network prevents the most trivial passive attacks based on observing the exact bit patterns of incoming messages and linking them to outgoing ones. However it leaks information, such as the size of the messages. Since PGP, beyond compressing the messages, does not make any further attempts to hide their size, it is trivial to follow a message in the network just by observing its length. The reply blocks provided are also a source of insecurity. They can be used many times and an attacker could encode an arbitrary number of messages in order to mount a statistical attack to find their destination (see chapter 9 for more details on such attacks). The attack can then be repeated for any number of hops.

Despite these drawbacks type I remailers became very popular. This is due to the fact that their reply capabilities allowed the use of Nym Servers.

The anonymity research community is concerned by the inability to phase out type I remailers. Their reply block feature, that is not present in the later type II Mixmaster software, is both essential to build nym servers, but also insecure even against passive adversaries. This has prompted the work presented in chapter 5, which describes a type III remailer, that is extremely resistant to traffic analysis, and provides secure single-use reply blocks.

4.1.4 Crowds

Crowds [RR98] was developed by Reiter and Rubin at the AT&T Laboratories. It aims to provide a privacy preserving way of accessing the web, without web sites being able to recognise which individual's machine is browsing. Each user contacts a central server and receives the list of participants, the "crowd". A user then relays her web requests by passing it to another randomly selected node in the crowd. Upon receiving a request each node tosses a biased coin and decides if it should relay it further through the crowd or send it to the final recipient. Finally, the reply is sent back to the user via the route established as the request was being forwarded through the crowd.

Crowds is one of the first papers to address quantitatively how colluding nodes would affect the anonymity provided by the system. It is clear that after the first dishonest node in the path of a request no further anonymity is provided, since the clear text of all requests and replies is available to intermediate nodes. Therefore, given a certain fraction of colluding attacker nodes it is possible to measure the anonymity that will be provided. Crowds also introduces the concept of *initiator anonymity*: a node that receives a request cannot know if the previous node was the actual requester or was just passing the request along. This property is quite weak and two independent groups have found attacks that identify the originator of requests [WALS02, Shm02]. Despite the difficulty of securing initiator anonymity, a lot of subsequent systems such as achord [HW02] and MorphMix [RP02], try to achieve it.

4.1.5 Nym servers

Nym servers [MK98] store an anonymous reply block, and map it to a pseudonymous email address. When a message is received for this address it is not stored, but immediately forwarded anonymously using the reply block to the owner of the pseudonym. In other words, Nym Servers act as a gateway between the world of conventional email and the world of anonymous remailers. Since they hold no identifying information, and are simply using anonymous reply blocks for routing purposes, they do not require users to trust them in order to safeguard their anonymity. Over the years, special software has been developed to support complex operations such as encoding anonymous mail to go through many remailers, and managing Nym Server accounts.

Nym servers are also associated with pseudonymous communications. Since the pseudonymous identity of a user is relatively persistent it is possible to implement reputation systems, or other abuse prevention measures. For example a nym user might at first only be allowed to send out a small quantity of email messages, that increases over time, as long as abuse reports are not received by the nym server operator. Nym servers and pseudonymous communications offer some hope of combining anonymity and accountability.

At the same time, it is questionable how long the true identity of a pseudonymous user can be hidden. If all messages sent by a user are linked between them by the same pseudonym, one can try to apply author identification techniques to uncover the real identity of the user. Rao *et al* [RR00] in their paper entitled “Can Pseudonymity Really Guarantee Privacy?” show that the frequency of function words[4] in the English language can be used in the long term to identify users. A similar analysis could be performed using the sets of correspondents of each nym, to extract information about the user.

4.2 Mix systems

The type I remailer, presented in section 4.1.3, is the insecure version of a whole body of research that we shall call mix systems and mix networks. This section presents more secure constructions based on these ideas.

4.2.1 Chaum’s original mix

The first, and most influential, paper in the field of anonymous communications was presented by Chaum in 1981 [Cha81]. Chaum introduced the concept of a “mix” node that hides the correspondences between its input messages and its output messages in a cryptographically strong way.

The work was done in the late seventies, when RSA public key encryption was relatively new. For this reason the paper might surprise today’s reader by its use of raw RSA, the direct application of modular exponentiation for encryption and decryption, along with an ad-hoc randomisation scheme. Nonces are appended to the plaintext before encryption in order to make two different encryptions output different ciphertext.

The principal idea is that messages to be anonymized are relayed through a node, called a mix. The mix has a well-known RSA public key, and messages are divided into blocks and encrypted using this key. The first few blocks are conceptually the “header” of the message, and contain the address of the next mix. Upon receiving a message a mix decrypts all the blocks, strips out the first block that contains the address of the next relay in the chain, and appends a block of random bits (the junk) at the end of the message.

[4] Function words are are specific English words used to convey ideas, yet their usage is believed to be independent of the ideas being conveyed. For example: *a, enough, how, if, our, the, . . .*

The length of the junk is chosen to make messages size invariant. The most important property that the decryption and the padding aim to achieve is *bitwise unlinkability.* An observer, or an active attacker, should not be able to find the link between the bit pattern of the encoded messages arriving at the mix and the decoded messages departing from the mix. The usage of the word encoded and decoded instead of encrypted and decrypted serves to highlight that the former operations are only used to achieve unlinkability, and not confidentiality as maybe understood to be the aim of encryption. Indeed, modifying RSA or any other encryption and decryption functions to provide unlinkability against passive or active attackers is a problem studied in depth in chapter 5 in the context of the design of Mixminion.

B. Pfitzmann and A. Pfitzmann [PP90] show that Chaum's scheme does not provide the unlinkability properties necessary. The RSA mathematical structure can be subject to active attacks that leak enough information during decryption to link ciphertexts with their respective plaintexts. Further *tagging attacks* are possible, since the encrypted blocks, using RSA are not in any way dependant on each other, and blocks can be duplicated or simply substituted by known ciphertexts. The output message would then contain two blocks with the same plaintext or a block with a known plaintext respectively. Once again the use of RSA in the context of a hybrid crypto-system, in which only the keys are encrypted using the public key operations, and the body of the message using a symmetric cipher were not very well studied at the time.

A further weakness of Chaum's scheme is its direct application of RSA decryption, which is also used as a signature primitive. An active attacker could substitute a block to be signed in the message and obtain a signature on it. Even if the signature has to have a special form, such as padding, that could be detected, a blinding technique could be used to hide this structure from the mix. It would be unfair to blame this shortcoming on Chaum, since he himself invented RSA blind signatures only a few years later [Cha83].

The second function of a mix is to actually mix together many messages, to make it difficult for an adversary to follow messages through it, on a first-in, first-out basis. Therefore a mix batches a certain number of messages together, decodes them as a batch, reorders them in lexicographic order and then sends them all out. Conceptually, while bitwise unlinkability makes sure that the contents of the messages do not allow them to be traced, mixing makes sure that the timing of the messages does not leak any linking information.

In order to make the task of the attacker even more difficult, dummy messages are proposed. Dummy messages are generated either by the original senders of messages or by mixes themselves. As far as the attacker is concerned, they are indistinguishable in length or content to normal messages, which increases the difficulty in tracing the genuine messages. We will call the actual mixing strategy, namely the batching and the number of dummy messages included in the inputs or outputs, the *dynamic aspects* of mixing.

Chaum notes that relying on just one mix would not be resilient against subverted nodes, so the function of mixing should distributed. Many mixes can be chained to make sure that even if just one of them remains honest some anonymity would be provided. The first way proposed to chain mixes is the *cascade*. Each message goes through all the mixes in the network, in a specific order. The second way proposed to chain mixes is by arranging them in a fully connected *network*, and allowing users to pick arbitrary routes through the network. Berthold, Pfitzmann, and Standtke argue in [BPS00] that mix networks do not offer some properties that cascades offer. They illustrate a number of attacks to show that if only one mix is honest in the network the anonymity of the messages going through it can be compromised. These attacks rely on compromised mixes that exploit the knowledge of their position in the chain; or multiple messages using the same sequence of mixes through the network.

Along with the ability for a sender to send messages anonymously to a receiver, Chaum presents a scheme by which one can receive messages anonymously. A user that wishes to receive anonymous email constructs an *anonymous return address*, using the same encoding as the header of the normal messages. She creates blocks containing a path back to herself, and recursively encrypts the blocks using the keys of the intermediate mixes. The user can then include a return address in the body of a message sent anonymously. The receiver simply includes the return address as the header of his own message and sends it through the network. The message is routed through the network as if it was a normal message.

The reply scheme proposed has an important feature. It makes replies in the network indistinguishable from normal messages. In order to securely achieve this it is important that both the encoding and the decoding operation provide bitwise unlinkability between inputs and outputs. This is necessary because replies are in fact decoded when processed by the mix. The resulting message, after it has been processed by all the mixes in the chain specified by the return address, is then decoded with the keys distributed to the mixes in the chain. Both the requirement for decryption to be as secure as encryption, and for the final mix to know the encryption keys to recover the message means that raw RSA cannot be used. Therefore a hybrid scheme is proposed that simply encrypts a symmetric key in the header along with the address of the next mix in the chain, that can be used to encrypt or decrypt the message. Since the keys are encoded in the return address by the user, they can be remembered by the creator of the reply block and used to decrypt the messages that are routed using them. Return addresses were also discussed in the Babel system [GT96] and implemented in the cypherpunk type I remailers. Unfortunately other deployed systems like Mixmaster did not support them at all.

Chaum's suggestion that a receipt system should be in place to make sure that each mix processes correctly messages, has become a branch of anonymity research in itself, namely mix systems with verifiable properties. We will give an overview of these systems in section 4.2.8. A system was also proposed to support pseudonymous identities that was partly implemented as the Nym Server described in section 4.1.3.

4.2.2 ISDN mixes, Real Time mixes and Web mixes

In 1991, A. Pfitzmann, B. Pfitzmann and Waidner designed a system to anonymise ISDN telephone conversations [PPW91]. This design could be considered practical, from an engineering point of view, since it meets the requirements and constraints of the ISDN network. Later the design was generalised to provide a framework for real-time, low-latency, mixed communications in [JMP$^+$98]. Finally, many of the design ideas from both ISDN and Real Time mixes were adapted for anonymous web browsing and called Web Mixes [BFK00]. Part of the design has been implemented as a web anonymizing proxy, JAP[5]. All three designs were the product of what could be informally called the Dresden anonymity community (although early research started in Karlsruhe), and the main ideas on which these systems are based are better illustrated by presenting them together.

A major trend in all three papers is the willingness to secure anonymous communication, even in the presence of a very powerful adversary. It is assumed that this adversary would be able to observe all communications on the network (global passive), modify the communications on the links by delaying, injecting or deleting messages and finally control all but one of the mixes. While other designs, such as Mixmaster and Babel (that will be presented next), opted for a free route network topology, ISDN, Real Time and Web mixes always use cascades of mixes, making sure that each message is processed by all mixes in the same order. This removes the need for routing information to be passed along with the messages, and also protects the system from a whole set of intersection attacks presented in [BPS00].

The designs try never to compromise on security, and attempt to be efficient. For this reason, they make use of techniques that provide bitwise unlinkability with very small bandwidth overheads and few asymmetric cryptographic operations. *Hybrid encryption with minimal length* encrypts the header, and as much as possible of the plaintext in the asymmetrically encrypted part of the message. A stream cipher is then used to encrypt the rest of the message. This must be performed for each intermediate mix that relays the message.

Furthermore, it is understood that some protection has to be provided against active tagging attacks on the asymmetrically encrypted header. A block cipher with a globally known key is used to transform the plaintext before any encryption operation. This technique allows the hybrid encryption of naturally long messages with very little overhead. It is interesting to notice that while the header is protected against tagging attacks, by using a known random permutation, there is no discussion about protecting the rest of the message encrypted using the stream cipher. Attacks in depth could be used,

[5] `http://anon.inf.tu-dresden.de/`

by which a partially known part of the message is XORed with some known text, in order to tag the message in a way that is recognisable when the message is decrypted. As we will see Mixmaster protects against this using a hash, while Mixminion makes sure that if modified, the tagged decoded message will contain no useful information for the attacker.

From the point of view of the dynamic aspects of mixing, ISDN, Real Time and Web mixes also introduce some novel techniques. First the route setup messages are separated from the actual data travelling in the network. In ISDN mixes, the signalling channel is used to transmit the onion encoded message that contains the session keys for each intermediary mix. Each mix then recognises the messages belonging to the same stream, and uses the session key to prime the stream cipher and decode the messages. It is important to stress that that both "data" messages and "route setup" messages are mixed with other similar messages. It was recognised that all observable aspects of the system such as route setup and end, have to be mixed.

In order to provide anonymity for both the initiator and the receiver of a call, rendezvous points were defined. An initiator could use an anonymous label attached to an ISDN switch in order to be anonymously connected with the actual receiver. This service is perhaps the circuit equivalent of a Nym server that can be used by message-based systems. It was also recognised that special cases, such as connection establishment, disconnection and busy lines could be used by an active attacker to gain information about the communicating party. Therefore a scheme of *time slice* channels was established to synchronise such events, making them unobservable to an adversary. Call establishment, as well as call ending have to happen at particular times, and are mixed with, hopefully many, other such events. In order to create the illusion that such events happen at particular times, real or cover traffic should be sent by the users' phones through the cascade for the full duration of the time slice. An even more expensive scheme requires users to send cover traffic through the cascade back to themselves all the time. This would make call initiation, call tear-down and even the line status unobservable. While it might be possible to justify such a scheme for ISDN networks where the lines between the local exchange and the users are not shared with any other parties, it is a very expensive strategy to implement over the Internet in the case of Web mixes.

Overall, the importance of this body of work is the careful extension of mixes to a setting of high-volume streams of data. The extension was done with careful consideration for preserving the security features in the original idea, such as the unlinkability of inputs and outputs and mixing all the relevant information. Unfortunately, while the ideas are practical in the context of telecommunication networks, where the mix network is intimately linked with the infrastructure, they are less so for widely deployed modern IP networks. The idea that constant traffic can be present on the lines, and that the anonymity can be guaranteed, but be relatively low, is not practical in such contexts. Onion routing, presented in section 4.2.5, provides a more flexible approach that can be used as an overlay network, but is at the same time open to more attacks.

4.2.3 Babel and Mixmaster

Babel [GT96] and Mixmaster [MCPS03] were designed in the mid-nineties, and the latter has become the most widely deployed remailer. They both follow a message-based approach, namely they support sending single messages, usually email, though a fully connected mix network.

Babel offers sender anonymity, called the "forward path" and receiver anonymity, through replies travelling over the "return path". The forward part is constructed by the sender of an anonymous message by wrapping a message in layers of encryption. The message can also include a return address to be used to route the replies. The system supports bidirectional anonymity by allowing messages to use a forward path, to protect the anonymity of the sender, and for the second half of the journey they are routed by the return address so as to hide the identity of the receiver.

While the security of the forward path is as good as in the secured original mix network proposals, the security of the return path is slightly weaker. The integrity of the message cannot be protected, thereby allowing tagging attacks, since no information in the reply address, which is effectively the only information available to intermediate nodes, can contain the hash of the message body. The reason for this is that the message is only known to the person replying using the return address. This dichotomy will guide the design of Mixminion (see chapter 5), since not protecting the integrity of the message could open a system to trivial tagging attacks. Babel reply addresses and messages can also be used more than once, while messages in the forward path contain a unique identifier and a timestamp that makes detecting and discarding duplicate messages efficient.

Babel also proposes a system of intermix detours. Messages to be mixed could be "repackaged" by intermediary mixes, and sent along a random route through the network. It is worth observing that even the sender of the messages, who knows all the symmetric encryption keys used to encode and decode the message, cannot recognise it in the network when this is done.

Mixmaster has been an evolving system since 1995 [MCPS03]. It is the most widely deployed and used remailer system.

Mixmaster supports only sender anonymity, or in the terminology used by Babel, only the forward path. Messages are made bitwise unlinkable by hybrid RSA and EDE 3DES encryption, while the message size is kept constant by appending random noise at the end of the message. In version two, the integrity of the RSA encrypted header is protected by a hash, making tagging attacks on the header impossible. In version three the noise to be appended is generated using a secret shared between the remailer, and the sender of the message, included in the header. Since the noise is predictable to the sender, it is possible to include in the header a hash of the whole message therefore protecting the integrity of the header and body of the message. This trick makes replies impossible to construct since the body of the message would not be known to the creator of an anonymous address block to compute in the hash.

Beyond the security features, Mixmaster provides quite a few usability features. It allows large messages to be divided in smaller chunks and sent independently through the network. If all the parts end up at a common mix, then reconstruction happens transparently in the network. So large emails can be sent to users without requiring special software. Recognising that building robust remailer networks could be difficult (and indeed the first versions of the Mixmaster server software were notoriously unreliable) it also allowed messages to be sent multiple times, using different paths. It is worth noting that no analysis of the impact of these features on anonymity has ever been performed.

Mixmaster also realises that reputation attacks, by users abusing the remailer network, could discredit the system. For this reason messages are clearly labelled as coming from a remailer and black lists are kept up-to-date with email addresses that do not wish to receive anonymous email. While not filtering out any content, for example not preventing death threats being transmitted, at least these mechanisms are useful to make the network less attractive to email spammers.

4.2.4 Stop-and-go mixes

As we saw above, Babel and Mixmaster implement a traditional mix network model. They also both extend the original idea of mixing batches of messages together to feeding back messages in a pool, in the case of Mixmaster, or to delaying a fraction of messages an additional round, in the case of Babel. Stop-and-Go mixes [KEB98] (sg-mix) present a mixing strategy, that is not based on batches but delays. It aims at minimising the potential for $(n-1)$ attacks, where the attacker inserts a genuine message in a mix along with a flood of his own messages until the mix processes the batch. It is then trivial to observe where the traced message is going.

Each packet to be processed by an sg-mix contains a delay and a time window. The delay is chosen according to an exponential distribution by the original sender, and the time windows can be calculated given all the delays. Each sg-mix receiving a message, checks that it has been received within the time window, delays the message for the specified amount of time, and then forwards it to the next mix or final recipient. If the message was received outside the specified time window it is discarded.

A very important feature of sg-mixes is the mathematical analysis of the anonymity they provide. It is observed that each mix can be modelled as a $M/M/\infty$ queue, and a number of messages waiting inside it follow the Poisson distribution. The delays can therefore be adjusted to provide the necessary anonymity set size.

The time window is used in order to detect and prevent $(n-1)$ attacks. It is observed that an adversary needs to flush the sg-mix of all messages, then let the message to be traced through and observe where it goes. This requires the attacker to hold the target message for a certain time, necessary for the mix to send out all the messages it contains and become empty. The average time that the message needs to be delayed can be estimated, and the appropriate time window can be specified to make such a delayed message be rejected by the mix.

4.2.5 Onion routing

Onion routing [GRS96, RSG98, GRS99, STRL00] is the equivalent of mix networks, but in the context of circuit-based routing. Instead of routing each anonymous packet separately the first message opens a circuit through the network, by labelling a route. Each message having a particular label is then routed on this predetermined path. Finally, a message can be sent to close the path. Often we refer to the information travelling in each of these labelled circuits as an anonymous stream.

The objective of onion routing is to make traffic analysis harder for an adversary. It aims first at protecting the unlinkability of two participants who know each other from third parties, and secondly at protecting the identities of the two communicating parties from each other. Furthermore, onion routing notes that ISDN mixes are not easily implementable over the Internet, and aims to distribute the anonymous network and adapt it to run on top of TCP/IP.

The first message sent through the network is encrypted in layers, that can only be decrypted by a chain of onion routers using their respective private keys. This first message contains key material shared between the original sender and the routers, as well as labels and addressing information about the next node. As with Chaum's mixes, care is taken to provide bitwise unlinkability, so that the path that the first message takes is not trivial to follow just by observing the bit patterns of messages. Loose routing is also proposed, according to which routers relay streams through paths that are not directly specified in the original path opening message. The hope was that such a scheme would increase the anonymity provided.

Data travelling in an established circuit is encrypted using the symmetric keys distributed to the routers. Labels are used to indicate which circuit each packet belongs to. Different labels are used on different links, to ensure bitwise unlinkability, and the labels on the links are encrypted using a secret shared key between pairs of onion routers. This prevents a passive observer from knowing which packets belong to the same anonymous stream, but does not hide this information from a subverted onion router.

Onion routing admits to being susceptible to a range of attacks. It has become clear that in the absence of heavy amounts of cover traffic, patterns of traffic are present that could allow an attacker to follow a stream in the network and identify the communicating parties. Such attacks have been called timing attacks. While they are often cited in the literature [Ray00], details of how they work and how effective they are have only been presented relatively recently in [SS03] and are the subject of chapter 10.

Unlike ISDN mixes, onion routing does not perform mixing on the requests for opening or closing channels. While it might be plausible that enough data would be available to mix properly, it is very unlikely that the anonymity of circuit-setup messages can be maintained. Therefore an attacker could follow such messages and compromise the anonymity of the correspondents. Furthermore very little mixing is done in the system generally, because of the real-time performance that is assumed to be needed. Onion routing aims at providing anonymous web browsing, and therefore would become too slow if proper mixing was to be implemented. Therefore a mixing strategy that is very close to first-in first-out for each stream is implemented. This provides only minimal mixing, and as a result a lot of attacks against onion routing focus on its weak dynamic features.

In order to make deployment easier, it was recognised that some onion routers might wish to only serve particular clients. The concept of *exit policies* was developed to encapsulate this, allowing routers to advertise which section of the network they were configured to serve. Onion routers are also free to peer with only a subset of other routers, with which they maintain long standing connections. The effects that such a sparse network might have on anonymity were first studied as part of this thesis and are presented in chapter 7.

Zero Knowledge, a Canadian company, designed the Freedom network that follows quite closely the architecture of onion routing. The principal architect of the network was Ian Goldberg [Gol00] who published with others a series of technical papers describing the system at various levels of detail [BSG00, BGS01].

4.2.6 Peer-to-peer mix networks

In Chaum's original work it is assumed that if each participant in the mix network also acts as a mix for others, this would improve the overall security of the network. Recent interest in peer-to-peer networking has influenced some researchers to further examine such networks with large, but transient, number of mixes.

Freedman designed *Tarzan* [FM02], a peer-to-peer network in which every node is a mix. A node initiating the transport of a stream through the network would create an encrypted tunnel to another node, and ask that node to connect the stream to another server. By repeating this process a few times it is possible to have an onion encrypted connection, relayed through a sequence of intermediate nodes.

An interesting feature of Tarzan is that the network topology is somewhat restricted. Each node maintains persistent connections with a small set of other nodes, forming a structure called a *mimics*. Then routes of anonymous messages are selected in such a way that they will go through mimics and between mimics in order to avoid links with insufficient traffic. A weakness of the mimics scheme is that the selection of neighbouring nodes is done on the basis of a network identifier or address which, unfortunately, is easy to spoof in real-world networks.

The original Tarzan design only required each node to know a random subset of other nodes in the network. This is clearly desirable due to the very dynamic nature of peer-to-peer networks, and the volatility of nodes. On the other hand Danezis and Clayton found some attacks, that are described at the end of this section, against this strategy in a preliminary version of Tarzan [FSCM02]. To fix this attack the final version of Tarzan requires each node to know all others, which is clearly less practical.

Mark Rennhard [RP02] introduces *MorphMix*, which shares a very similar architecture and threat model with Tarzan. A crucial difference is that the route through the network is not specified by the source but chosen by intermediate nodes, observed by user specified and trusted witnesses. While the attack by Danezis and Clayton does not apply to route selection, variants might apply to the choice of witness nodes.

MorphMix realises that leaving the intermediate nodes to choose the route through the network, might lead to *route capture*, or in other words the first subverted mix on the path choosing only other subverted mixes. For this reason MorphMix includes a *collusion detection* mechanism, that monitors for any cliques in the selection of nodes in the path. This prevents subverted nodes from routinely running attacks on the network but does not provide security in every case.

4.2.7 Attacks against the 'young' Tarzan

Route reconstruction attack The early design of Tarzan [FSCM02] proposes that nodes choose some relays randomly out of the ones they know and establish a route through them. In order to achieve resilience against intermediary nodes leaving the network, or proving to be unreliable, a route reconstruction protocol is provided. The protocol is designed to be low cost and just routes around the failed node. Rebuilding the entire tunnel would be expensive, so the working hops are retained and only the failing connections are replaced.

By the use of active attacks on intermediate nodes, an attacker that can inspect traffic leaving these nodes will be able to exploit the use of this protocol so as to infiltrate the chain and to ultimately compromise the entire path through the Tarzan network. The attacker controls a fraction c of subverted nodes. The initial probability that a chain of length l is totally controlled by an attacker is, for a large network, c^l. However, if the attacker can cause one of the good nodes to fail, then the Tarzan route reconstruction protocol will replace that single node by making another selection from the pool.

The attacker can induce such a failure either by launching a denial of service attack directly across the Internet or by overloading the node by routing large amounts of traffic across the Tarzan network, with all of his traffic going through the node. The latter scheme would be preferable because the attacker should find it simpler to hide.

The attacker can then cause the replacement node to fail in a similar manner until eventually the user selects one of the subverted nodes. The attacker then turns their attention to the next node along the path to try and make the user select a malicious node for that position as well. This attack can be mounted in a linear fashion either from the source towards the destinations it is accessing, or from the destination towards the sources of traffic.

The attack can be made even simpler, once the attacker controls one node in the path. There is no longer a need to mount active denial-of-service attacks, since the malicious node can merely discard traffic packets. Once the source realises that its traffic flow has ceased it will send out "ping" packets to attempt to determine which hop has failed. The attacker discards these as well and the user will conclude that the next hop has failed, and tries to route around it.

The attack is rather faster to mount than might naïvely be expected. The number of nodes disabled, until a subverted node is selected, follows the geometric distribution and has a mean of $m = 1/c$. The attack operation needs to be repeated l times, once until each node along the chain is under the control of the attacker. Therefore, on average, the attacker needs to disable l/c nodes until the path is fully controlled.

Clearly the attack is not possible if the entire path is reconstructed when it fails, but this defeats the original design goal of avoiding excess expense in such circumstances.

Node knowledge profiling A Tarzan user discovers the identity of participating nodes by using a pseudo-random number generator as an index into a Chord ring [SMK$^+$01] that returns the node identities. Tarzan uses this random selection process because it is intended to scale to networks so large that it would not be feasible for every node to have a complete view of the network.

An attacker is able to observe the node selection process, either by seeing the messages from the Chord ring or by inspecting the traffic subsequently exchanged with the nodes that the Chord ring has identified. Clearly if the user solely established the identity of the nodes that were to be used to forward messages then this would compromise the path. Tarzan avoids this trap by arranging to discover a large number of nodes and then use only a small subset for a particular path. Unfortunately, the attacker can use their knowledge of which nodes were known to the user to probabilistically identify traffic as it passes across the Tarzan network.

Let us suppose that there are N nodes in the Tarzan network and the user establishes the identity of a random subset of size k out of them. The attacker can now inspect traffic at any of the k nodes that the user is aware of to determine if any is arriving from a node that is known to the user and also if traffic is departing to another node known to the user. If the node is not both sending and receiving such traffic then the user is not currently using it. If it *is* both sending and receiving such traffic then the user may be using it – and where k is small compared to N it is very likely indeed that it is the user's traffic being observed and not some other participant in the network.

We will calculate the expected number of nodes that could have constructed a path including three observed intermediate nodes (the triplet).

1. We assume, for simplicity, that all participants choose k nodes from the total set of N. Tarzan ensures this selection is uniformly random by requiring users to query a Chord ring.

2. Each node can generate $\binom{k}{3}$ triplets out of the $\binom{N}{3}$ that exist, where $\binom{n}{m}$ is the number of possible ways of selecting m elements from n. Therefore given a random triplet of nodes the probability it is known to a node is $p = \frac{\binom{k}{3}}{\binom{N}{3}}$.

3. For any particular triplet, the number of nodes who could choose it follows a binomial distribution with parameters p and N. Each out of the N nodes has a probability p to know the particular triplet. The

expected number of nodes[6] μ that know the particular triplet is:

$$\mu = N\frac{\binom{k}{3}}{\binom{N}{3}} = \frac{k(k-1)(k-2)}{(N-1)(N-2)} \approx \frac{k^3}{N^2} \tag{4.1}$$

We can now calculate when a node is vulnerable to the attack. If we wish the number of nodes that know a particular triple to be one or less (ie they are, on average, uniquely identifiable) then we need the result of equation (4.1) to be less than or equal to 1.

$$\frac{k^3}{N^2} \leq 1 \Rightarrow k \leq N^{2/3} \tag{4.2}$$

For example, if there are one thousand nodes in the network, then if all nodes learn the identity of 100 or less participants then any triple they use will usually be unique to them.

Of course an attacker may be able to establish that traffic is flowing through a sequence of hops. Generalising equation (4.1) we can see that the average number of nodes that will know about a particular hop sequence of length l is:

$$\frac{N\binom{k}{l}}{\binom{N}{l}} = \frac{k(k-1)(k-2)\dots(k-l+1)}{(N-1)\dots(N-l+1)} \approx \frac{k^l}{N^{l-1}} \tag{4.3}$$

Thus, if the attacker has the ability to observe many nodes they will be able to test all possible combinations of routes against the target node profiles. Even though the combinations of potential routes to trace might seem to grow exponentially, most of the paths can be discarded from the early stages of analysis.

The above results show that the random selection of nodes to provide routing through the network is extremely unwise. A very significant fraction of the nodes must be discovered; ie k must be large enough that the attacks become impractical, although it should be noted that any value short of N will always cause some information to leak. As an alternative, the discovery of participant nodes must be made unobservable.

[6] The variance is $\sigma^2 = N\left(\frac{\binom{k}{3}}{\binom{N}{3}}\right)\left(1 - \frac{\binom{k}{3}}{\binom{N}{3}}\right) \approx \frac{k^2N^3-k^5}{N^5}$

4.2.8 Robust & verifiable mix constructions

Chaum's original mix network design included a system of signed receipts to assure senders that their messages have been properly processed by the network. A whole body of research was inspired by this property and has attempted to create mix systems that are robust against subverted servers denying service, and that could offer a proof of their correct functioning alongside the mixing. Such systems have been closely associated with voting, where universal verifiability of vote delivery and privacy is of great importance.

Most of the proposed schemes use the idea of a mix cascade. For this reason no information is usually communicated between the sender of a message and intermediate mixes in the cascade. It is assumed that routing information is not necessary since mixes process messages in a fixed order. The first scheme to take advantage of this was the *efficient anonymous channel and all/nothing election scheme* proposed by Park, Itoh and Kurosawa [PIK93]. In this system messages are of fixed length, of an El Gamal ciphertext, independently of the number of mixes they go through. Furthermore using a *cut and choose* strategy the scheme is made all-or-nothing, meaning that if any of the ciphertexts is removed then no result at all is output. This property assures that partial results do not affect a re-election. Birgit Pfitzmann found two attacks against this proposal [Pfi94]. The first attack is very similar to [PP90], and makes use of characteristics that are invariant at the different stages of mixing because of the El Gamal crypto-system. An active attack is also found, where the input El Gamal ciphertext is blinded, by being raised to a power, which results in the final output also being raised to this power. This is a chosen ciphertext attack against which a lot of systems will struggle, and eventually fail to eliminate. Birgit Pfitzmann also notes that the threat model assumed is somehow weaker than the one proposed by Chaum. A dishonest sender is capable of disrupting the whole network, which is worse than a single mix, as is the case in Chaum's paper. Birgit did not propose any practical countermeasures to these attacks, since any straight forward fix would compromise some of the interesting features of the systems.

In parallel with Birgit Pfitzmann's work, Sako and Killian proposed a *receipt-free mix-type voting scheme* [KS95]. They attempt to add universal verifiability to [PIK93], which means that all sender will be able to verify that all votes were taken into account, not simply their own. They also highlight that many verifiable mix schemes provide at the end of mixing a receipt, that could be used to sell or coerce one's vote, and attempt to make their system *receipt-free*. They do this by forcing each mix to commit to their inputs and outputs, and prove in zero knowledge that they performed the decryption and shuffle correctly. Unfortunately Michels and Horster [MH96] show that the scheme is not receipt-free if a sender collaborates with a mix, and that the active attacks based on blinding proposed by Birgit Pfitzmann could be used to link inputs to outputs.

In order to avoid disruption of the system if a subset of mixes is subverted Ogata, Kurosawa, Sako and Takatani proposed a *fault tolerant anonymous channel* [OKST97]. This uses a threshold crypto-system to make sure that a majority of mixes can decode messages even if a minority does not collaborate. Two systems are proposed, one based on El Gamal and the other based on the r^{th} residue problem. A zero knowledge proof of correct shuffling is also proposed for the r^{th} residue problem.

In 1998 Abe presented a mix system that provided universal verifiability that was efficient, in the sense that the verification work was independent from the number of mix servers [Abe98]. This scheme shows an attack on [KS95], that uses the side information output for the verification to break the privacy of the system. It then presents a mix system that works in two phases, El Gamal re-encryption and then threshold decryption. The first phase is proved to be correct before the second can proceed, and then a proof of correct decryption is output at the end of the second stage.

The systems that provide universal verifiability based on proofs of permutations, and zero knowledge proofs are computationally very expensive. Jakobsson designs the Practical Mix [Jak98], and tries to reduce the number of expensive operations. In order to prove the correctness of the shuffle, novel techniques called *repetition robustness* and *blinded destructive robustness* are introduced. The network works in two phases: first the ciphertexts are El Gamal blinded, and then the list of inputs is replicated. Each of the replicated lists is decoded by all mixes, which results in lists of blinded plain texts. Then the resulting lists are sorted and compared. If all elements are present in all lists then no mix has tampered with the system and the unblinding and further mixing can proceed. Otherwise the sub-protocol for cheater detection is run. While being very efficient the Practical Mix has not proved to be very secure, as shown by Desmedt and Kurosawa [DK00]. They show that one subverted mix in the practical mix can change ciphertexts, and still not be detected. They then introduce a new mix design, in which verification is performed by subsets of mixes. The subsets are generated in such a way that at least one is guaranteed not to contain any subverted mixes.

In an attempt to further reduce the cost of mixing Jakobsson introduced the Flash Mix [Jak99], that uses re-encryption instead of blinding to keep the number of exponentiations down. As in the practical mix, mixing operates in many phases, and uses *repetition robustness* to detect tampering. Furthermore two dummy messages are included in the input, that are deanonymized after all mixes have committed to their outputs, to make sure that attacks such as [DK00] do not work. An attack against Flash mixing was found in [MK00] and fixed by changing the unblinding protocol.

A breakthrough occurred when Furukawa, Sako [FS01] and Neff [Nef01] proposed efficient general techniques to universally verify the correctness of a shuffle of El Gamal ciphertexts. The first provides proof that the matrix used was a permutation matrix, and the second uses verifiable secret exponent multiplication to gain its efficiency.

Even though the above techniques are more efficient than any other previously known, they are still not efficient enough to scale for elections, with millions of participants. For this reason Golle, Zhong, Bohen, Jakobsson and Juels proposed, optimistic mixing [GZB$^+$02], a mix that works quickly if there is no attack detected, but provides no result if an error occurs. In this case it provides a fall back mechanism for a more robust technique such as [Nef01] to be used. Each mix in the chain outputs a “proof” of permutation, that could be faked by tampering with the ciphertexts. This is detected by making the encryption plaintext-aware. The second decryption, revealing the votes, is only performed if all outputs of mixing are well-formed. A series of attacks were found against this scheme by Wikström [Wik03b, Wik02]. The first two attacks are closely related to [Pfi94] and can break the anonymity of any user. The second attack is related to [DK00] and can break the anonymity of all users and compromise the robustness. Finally attacks based on improperly checking the El Gamal elements are also applicable, and further explored in [Wik03a].

While most designs for robust mix nets use pure El Gamal encryption, some provide solutions for hybrid schemes. Ohkubo and Abe present a hybrid mix without ciphertext expansion [OA00]. Jakobson and Juels [JJ01] also present a scheme that is resistant to any minority coalition of servers. Bodo Möller proves the security properties of a mix packet format in [Möl03].

A hope that universal verifiability can be implemented on generic mix networks, is presented in [JJR02]. In this scheme all mixes commit to their inputs and outputs and then they are required to disclose half of all correspondences. This assures that if a mix is dropping messages it will be quickly detected. Privacy is maintained by pairing mixes, and making sure that the message is still going through enough secret permutations.

4.2.9 Mix building blocks, attacks and analysis

Not all research in the field of mix networks presents complete systems. Many published papers present specific or generic mechanisms, others present attacks and some analyse proposed systems. We will present a selection of each in this order.

Replay attacks were first described by Chaum himself [Cha81]. A passive attacker can trace a message by 'replaying' it in the network. An attacker first records all traffic in the network (or all traffic going in and out of a target mix). Then the message to be traced is reintroduced in the network. Since the decoding process is deterministic the output message will have the same bit-pattern and destination the second time around. Therefore the adversary can determine that, out of the mix batch, the single message that was seen before corresponds to the replayed input message. If the bit-patterns of the messages are not observable, because link encryption is used, the only information leaked by the replay attack is the destination of the message. Therefore the replay attack is reduced to a disclosure attack as described in

Chapter 9. To prevent replay attacks mixes have to remember the messages that they have processed and not process them a second time. The space required to do this can be reduced by using a cryptographically secure hash function. An alternative, which is an active field or research, is to use non deterministic decoding and routing protocols, possibly implemented using the new universal re-encryption primitive [GJJS04].

A series of papers have appeared presenting different mixing strategies. Starting with the proposed metrics for anonymity in [SD02, DSCP02], the work in [SDS02, DS03b, SN03] tries to measure the anonymity provided by different mixing strategies, but also the delay introduced and their susceptibility to $(n-1)$ attacks.

Reputation based schemes have also been used to increase the reliability of mix networks in [DFHM01] and mix cascades in [DS02]. Both these papers present how *statistics pages* compiled in the Mixmaster system using *pingers* [Pal] can be replaced with a more robust system to determine which nodes are reliable and which are not. Users can then choose reliable nodes, or the system can exclude unreliable ones from directories.

A whole body of research concentrates on providing attacks against mix networks. In chapter 5, we will present Mixminion that tries to protect against most of them.

In [BPS00] Berthold, Pfitzmann and Standtke describe a series of attacks, assuming a very powerful adversary that controls all mixes in the network but one:

- The *position* attack allows an adversary to partition the messages coming in and out according to the position of the only honest mix. They assume that all routes have the same length and therefore knowing how many hops each message has gone through and how many hops outgoing messages are going through an attacker can link them.

- The *determining the next mix* attack is a variant of an intersection attack, and is the forefather of the detailed disclosure attacks described in [KAP02].

In [Ray00] Raymond presents a series of what he calls traffic analysis attacks:

- The *brute force attack* is the simplest attack that can be performed against a mix network. The set of all potential recipients is generated by following a message through the network, and adding to the anonymity set all the recipients of the messages that have been mixed together. It is one of the few papers that actually describes how an attacker would perform such tracing and construct the sets, and offers some tips on how to increase the difficulty.

- $(n-1)$ attacks are presented again. An attacker waits until a mix is empty (or stuffs it until it empties), and then sends a single genuine message to it. Then the attacker either waits, or again stuffs the mix until the message under surveillance is output. This leaks the receiver of this single honest message. It is observed that link encryption might help foiling this attack. Other ways are sg-mixes (see subsection 4.2.4), authenticating users, using hashcash [Bac] or re-routing messages to increase anonymity.

- *Timing attacks* try to guess the correlation between inputs to the network and outputs by using the fact that each link introduces a different delay. Therefore, by calculating the delay in each link, a model could be constructed of the probabilities of output messages relating to particular input messages.

- *Communication patterns* can be used to find out when users of pseudonymous systems are on-line and when they are off-line. The patterns of activity might, for example, betray the time zone in which a sender is.

- *Packet counting attacks* take advantage of the fact that systems such as Onion Routing send whole streams along the same path. They correlate ingoing and outgoing streams, and follow them in the network by counting the packets in them. The latest such attack can be found in [SS03].

- *Intersection attacks*, also presented in [BPS00], try to extract the sender of a stream of messages by intersecting the sender anonymity sets of consecutive messages sent by a pseudonym. This attack model is also considered in [KAP02, AKP03] and [WALS03]. The statistical variant of this attack is the statistical disclosure attack presented in chapter 9.

- *User interaction attacks* (or *Sting attacks*) take advantage of the fact that a user will act in a different way if he is the actual sender of a message than if he is not. This attack is not simply social but can target identity management systems that do not implement proper separation between pseudonyms. A variant of this attack, concerning Mixminion single-use reply blocks, is presented in chapter 5.

- *Send n' Seek attacks* are related to the tagging [PP90, Pfi94] attacks, but are slightly more general. They rely on a special message being sent to an anonymous receiver that is then used to track the receiver. The tracking might not involve traffic analysis, but social mechanisms such as breaking down doors and seizing computers until this special message is found.

The work in [BMS01] also refers to an attack invented by Wei Dai. It also discusses the process of anonymity engineering, and how to balance the different aspects of it.

- An attacker wishes to defeat the traffic shaping mechanisms that attempt to hide the real volumes of traffic on an anonymous channel. The attacker creates a route using the link that he wishes to observe, and slowly increases the traffic on it. The router will not know that the stream or streams are all under the control of the attacker, and at some point will signal that the link has reached its maximum capacity. The attacker then subtracts the volume of traffic he was sending from the maximum capacity of the link to estimate the volumes of honest traffic.

In [WALS02] the authors present a set of attacks that can be performed by a group of subverted network nodes. Against mix networks, they calculate the number of routes to be chosen between a sender and a receiver before the full path has been entirely populated by subverted nodes. They also examine the effect that fixed or variable length paths have on this probability. Similar results are found for Crowds and DC-nets. In [WALS03] they extend their analysis to considering a subset of network nodes that simply log all traffic, and provide bounds on how quickly an intersection attack can be performed.

4.3 Other systems

A number of other anonymous communication systems have been proposed through the years. Chaum presents in [Cha88] the dining cryptographers' network, a multi-party computation that allows a set of participants to have perfect (information theoretic) anonymity. The scheme is very secure but impractical, since it requires a few broadcasts for each message sent and is easy for dishonest users to disrupt. A modification of the protocol [WP89] guarantees availability against disrupting nodes.

Other anonymous protocols include Rackoff and Simon [RS93], Sherwood, Bhattacharjee and Srinivasan present $\mathcal{A}5$ [SBS02] while Goel, Robson, Polte and Sirer present Herbivore [GRPS03].

Traffic Analysis Prevention (TAP) systems, attempt to provide third party anonymity (as already introduced in section 2.1.3), given a collaborating set of senders, receivers and relays. Timmerman describes adaptive traffic masking techniques [Tim99], and a security model to achieve *traffic flow confidentiality* [Tim97]. The information theoretic approach to analysing TAP systems is presented by Newman *et al* in [NMSS03]. They study how much protection is offered overall to the traffic going through a TAP system by creating a rigorous mathematical model of traffic analysis, rerouting and cover traffic. This builds on their previous work [VNW94]. The research group at the Texas A&M University, has a long-term interest in traffic analysis prevention of real time traffic [GFX$^+$01]. Similarly Jiang [JVZ00] *et al* present TAP techniques to protect wireless packet radio traffic.

4.4 Summary

We have presented in detail anonymous communications systems based on mixing, both trusted intermediates and more secure constructions. These systems based on the simple idea of a mix, a node that hides the correspondence between its input messages and its outputs, have been adapted to anonymise email traffic, web traffic or telephony.

Novel attacks against an early design of the Tarzan anonymizing network are presented. They take advantage of its attempt to be resilient to node failures and the large size of peer-to-peer networks to attack it. The first one is active and allows all nodes in the path to be captured, while the second one is passive and allows an attacker to identify the node that has setup particular paths through the network.

A series of features and attacks have also been presented, that our anonymous remailer design, mixminion presented in the next chapter, will have to be secure against.

Chapter 5

Mixminion

"Hidden, we are free:
Free to speak, to free ourselves,
Free to hide no more."

First public Mixminion message — Nick Mathewson as 'anonymous'

Mixminion has been designed by myself, Roger Dingledine and Nick Mathewson, who is also the lead developer, to be the successor of the current Mixmaster anonymous remailer infrastructure. While the project has many objectives, the main contribution of this thesis[1] to it is the design of a mix packet format that can effectively:

- be mixed and provide bitwise unlinkability,
- accommodate indistinguishable replies,
- leak minimal information to intermediate mixes,
- be resistant to tagging attacks.

Aspects that are relevant to Mixminion but have been addressed by others are directory servers, end-to-end issues such as efficient information dispersal

[1] This chapter is meant to illustrate my contribution to the design of Mixminion, and does not reflect the current state of the system. Please refer to the detailed and up-to-date specification for this [DDM03b]

algorithms and reliable transmission. It is interesting to note that while *bitwise unlinkability* issues can be considered to be well understood, many of the other aspects are still open problems, and in the process of being specified.

The requirements for Mixminion are presented first and the reasons behind them are explained. Then the Mixminion packet format is described, focusing particularly on the security aspects of the design. We then analyse the security of the mechanisms when Mixminion is under attack. Finally we present some efficiency improvements that can be easily implemented to reduce the size overheads.

5.1 Models and requirements

From the point of view of the properties offered by Mixminion to its users we have identified the following desirable use-cases, that need to be fully supported.

Forward Path. Alice wants to send a message to Bob, without Bob being able to tell that it came from Alice. In order to do this Alice constructs a Mixminion message and sends it to the Mixminion network. The message gets out of the network at some other point and is sent to Bob by conventional email. We call this use case the *forward path*.

Reply Path. Alice wants to send a message to Bob, without Bob being able to tell that it came from Alice. Despite this, Alice wants Bob to be able to reply to her message, again without Bob discovering her identity. Alice therefore constructs a reply block that she includes in a message sent to Bob using the *forward path*. Bob is able to send this reply block to a mix in the network, using conventional mail, along with a message. The message will be routed anonymously back to Alice, who will decode it. We call this use case the *reply path*.

Bidirectional Anonymity. Alice wants to communicate anonymously to someone that she does not know, that she likes to call Beatrice. Through some mechanism (such as a Nym Server) she knows one of Beatrice's

> reply blocks. Alice therefore creates a Mixminion message that is intended to route her message through the network to first hide her identity using the *forward path*, and then uses the *reply path* to reach Beatrice. We call this bidirectional anonymity, and it allows two parties, through some system of anonymous first contact, to maintain an anonymous conversation.

As far as the user is concerned they can perform a set of operations vis-a-vis the anonymous communication system. They can *encode and send a message to a named individual*, they can *construct a reply block* to route back to themselves, or they can *send a message using a reply block*. Furthermore they can combine two of these functions and send an anonymous message to an anonymous recipient using a reply block.

A user-friendly system should try to minimise any exposure to a user that does not fit into the model described above. Any security-sensitive operation that must be performed by the user and does not fit into this "natural" model above, will likely not be performed effectively. Therefore, it should be automated and hidden. For example it might be "natural" to expect the user not to sign the end of an anonymous email with their real name; but asking them to "contact the directory server at least once a day" might not. The only reasonable answer to expect would be "what is a directory server?" This operation should, for the sake of security and usability, be automated and hidden from the user.

5.1.1 The system view

Mixminion follows an architecture that is quite similar to Mixmaster and Babel. It assumes that a set of *mix nodes* or *mix servers* constitute the *mix network*. Each mix node has a *directory entry* associated with it, containing its status, network address, public encryption and verification keys, and capabilities. For the purposes of this work we will assume that there exists an effective way for anyone to get a set of all directory entries using a *directory server*.

Mix clients wish to use some of the anonymity properties offered by the network, and therefore run dedicated software to get information about the network, encode or decode messages. Mix clients who wish to send an anonymous message, or construct a reply block, therefore choose a set of mix nodes that will relay their message. They then proceed to package the message or reply block as described below.

Aside from mix clients, Mixminion assumes that there are also other *network users* who do not benefit from anonymity properties, and therefore do not run dedicated software. They are still able to receive anonymous messages and reply using reply blocks. In fact this capability is implemented by making sure that they require no secrets to reply (since they do not require any security properties) and therefore the operation could be proxied by a suitable mix node using simple email.

The two principal components of Mixminion, as a standard way of offering bitwise unlinkable relayed communication, are the *packet format* and the *inter mix encrypted tunnels*. We shall first present an analysis and design rationale behind the packet format, and the algorithms acting on it. We will present a discussion of the transport between mixes in the context of forward security mechanisms in chapter 6. But before addressing these matters we will present the technical requirements beyond the simple model common to all mix systems.

5.1.2 Requirements

Mixmaster [MCPS03] has been deployed since 1995 and is the most widely established and used strong anonymity system. By strong it is meant that no central authority can compromise the anonymity provided by the system, and that the network is secure against a global passive adversary. Mixminion therefore also aims to provide *anonymity despite a global passive adversary that controls a subset of nodes on the message path and can perform active attacks against honest mix servers.* This means that trivial traffic analysis based on uncovering relationships between the incoming and outgoing messages should be eliminated. In other words, Mixminion should provide effective bitwise unlinkability between inputs to mixes and outputs.

While the original versions of the Mixmaster server have been buggy and unreliable, the software was rewritten by Ulf Möeller, and actively maintained since 2001 by Len Sassaman and others. As a result the reliability of the anonymous transmission has improved drastically. Despite these improvements the design of Mixmaster has not, for a long time, been properly documented making it difficult to assess its security, but also slowing down deployment and compatible implementations of servers and clients. Mixminion aims to create *a very well-understood, conservative and documented standard for anonymous communications.* Mixminion was always intended to be the anonymous transport layer for more complex protocols, such as Free-Haven or anonymous cash, and therefore clear documentation and interfaces would be essential for its wide deployment. For the same reason, Mixminion was never meant to be a proof-of-concept academic system, and therefore it has been engineered to be secure by quite a margin to all known attacks. This approach has lead to some inefficiencies and unnecessary overheads. The designers' feeling has been that these should be eliminated only when there is overwhelming evidence that no security risks are introduced by optimising them out.

In parallel with Mixmaster, the old Type I Cypherpunk remailers are still running and providing a less secure service (as described in section 4.1.3). These services could not be withdrawn due to the fact that they provide facilities for reply blocks that Mixmaster does not offer. In particular, reply blocks are used by pseudonym servers to support pseudonymous email addresses that are fundamental for interactions with most electronic services. Therefore, a key objective of Mixminion from the start has been to provide *a facility for secure anonymous replies.*

It would be trivial to construct two systems: one for forward messages and one for replies. On the other hand, making the reply messages distinguishable from other kinds of traffic would greatly reduce the anonymity of both replies and forward traffic. At best, assuming that approximately the same volumes of replies are travelling as forward messages, the anonymity set sizes are divided by two. If there is an imbalance in the volume of forward messages and replies, it reduces the anonymity sets of users that might make use of replies. The argument for this relies on intersection attacks that are inherently present in any mix system that supports replies. An attack could be mounted as follows:

1. An adversary generates the effective anonymity set of a forward message that he wishes to trace.
2. An adversary then uses one or many reply blocks contained in the message and computes their effective anonymity sets.
3. The attacker then combines the statistical distributions over the set of potential senders and receivers to find out who was the sender of the original anonymous message.

It is clear that providing smaller effective anonymity sets for the reply path would in this case greatly impact much on the anonymity of the forward path as well. Therefore Mixminion aims to provide *indistinguishable replies*, which means that no only third parties but also intermediary mixes are not able to tell if they are processing a forward path message or a reply.

Other partitioning attacks are also described in [BPS00] that rely on information available to subverted intermediary mixes, such as their position in the mix chain or the total length of the route length. Although the paper addresses a much stricter threat model, where all the mixes on a message path but one are subverted, the decision was taken that *the position of an intermediary mix on the path, and the route length, should be hidden.*

Special software is required to package a message to be anonymized, and to mix messages. On the other hand, it is desirable for receivers of messages not to have to use special software to read Mixminion messages. After all, receivers of sender-anonymous messages do not benefit from any security properties, and limiting the conversation to users with special software would greatly reduce the effective anonymity sets provided by the system. As Back, Möller and Stiglic [BMS01] report,

> “In anonymity systems usability, efficiency, reliability and cost become *security* objectives because they affect the size of the user base which in turn affects the degree of anonymity it is possible to achieve.”

The decision was taken to design a system where *only parties benefiting from anonymity properties are required to use special software.* Others can simply use standard email clients, or other specified transports. Note that while this is trivial for senders and receivers of sender-anonymous messages, it is an engineering challenge to make sure that users of reply blocks do not require any special software to converse with an anonymous party.

A lot of systems, including the original proposal by Chaum, are secure against passive adversaries, but fail against active tagging attacks [PP90, Pfi94]. Early drafts of the Mixminion specifications were susceptible to such attacks, and further design work was needed to reduce them. The decision was taken finally that the *tagging attacks must be totally eliminated* otherwise the system would never provide the assurance required by some highly security-sensitive categories of users.

Finally, it was considered that a lot of its past unreliability was due to Mixmaster's reliance on the email protocol SMTP for message transport. It was decided that *a reliable forward secure protocol should be implemented to communicate messages between mixes.* A possible extension of forward security properties has also prompted the research on forward secure mixing described in chapter 6.

Other requirements are effectively independent from the constraints presented above. There is a need for an integrated directory service, and a system of standard extensions that would allow services to be built on top of Mixminion, or for enhancing Mixminion basic functionality. These are not directly related to the bitwise unlinkability properties of Mixminion and are not therefore described in detail here.

5.1.3 Orthogonal issues

While Mixminion has a set of tight requirements described above, it also aims at being flexible. It has to be understood that Mixminion is a format that provides *bitwise unlinkability* and therefore tries to be agnostic when it comes to the dynamic aspects of mixing. In particular a network operator and designer has the following choices:

Mixing strategy. Mixminion does not impose any restrictions on the mixing, batching or delaying strategy that individual mixes or the network as a whole wishes to implement. An extension mechanism allows clients to pass special information to intermediate mixes, as is needed to implement sg-mixes [KEB98]. It is of course wise not to have a large number of mixing strategies that are available but not supported by everyone, since this would reduce the effective anonymity set by allowing partitioning attacks.

Network Topology. Any network topology can be used to route Mixminion traffic. Although it supports a fully connected network, alternative topologies such as cascades and restricted routes networks can be used. Directory servers are also designed to inform clients about topology. The security of restricted route networks is analysed in chapter 7.

Route Selection. It is not specified how individual clients choose the path their messages will take through the network. Directory servers, statistics servers and reputation metrics could be used to increase reliability, but care has to be taken not to open oneself to intersection attacks.

Dummy Policy. While there is support for both encrypted link padding and dummy messages, Mixminion nodes are not required to implement any particular strategy for generating any kind of cover traffic.

Environment Awareness. We assume that each mix simply knows about the directory servers to which it must periodically upload its information. A node is not explicitly required (or prohibited) from knowing about any other nodes, or the traffic conditions in other parts of the network to perform its mixing. In this respect a node is required to know even less than a client, which needs to have a recent version of the directory of available mixes.

The issues above are either orthogonal to the problems that Mixminion is trying to tackle or were, at the time of the design, active research areas. For example, the security of sparse networks for routing was studied separately, and is discussed in chapter 7. The use of special cover traffic to prevent $(n-1)$ attacks is presented in chapter 8. All the matters above are security sensitive in one way or another, and decisions on which strategy is best should be subject to very serious analysis.

5.2 The anatomy of the Mixminion format

This section presents the Mixminion packet structure and describes how it is created by senders, and processed by intermediary mixes. While this section presents an early design of the packet format, to illustrate the main ideas behind preventing active attacks, an implementor should always refer to the latest design specifications [?].

Each Mixminion packet from its creation to its exit from the Mixminion network is composed of three parts:

- The first header,
- The second header,
- The message body.

Both headers have the same structure, although the second one is encrypted with secrets contained in the first one. The body might be encrypted, in the case of the forward path, or not in case of the return path. Each header contains a number of sub-headers, addressed to different mixes, as we will see. Conceptually each sub-header is a secure communication channel between the creator of the header and a mix on the message path. The sub-header contains the secrets that allow a mix to decode the rest of the first header, the second header and the body of the packet. It also contains the address of the next mix in the chain. One can view a Mixminion packet as being a key distribution protocol, along with a distributed decoding of a message, all in one.

There are also two main algorithms related to a Mixminion packet:

Packet Creation. This algorithm requires a sequence of mix server description, containing the network addresses and the public keys of the mixes, a final address and a body. It will then generate a Mixminion packet suitable for being relayed by the chosen intermediaries. Note that the final address might be a reply block.

Packet Processing. Given a secret key, a public list of hashes of previous message signatures, and a Mixminion message, this algorithm will output the Mixminion packet to be relayed and the address of the next

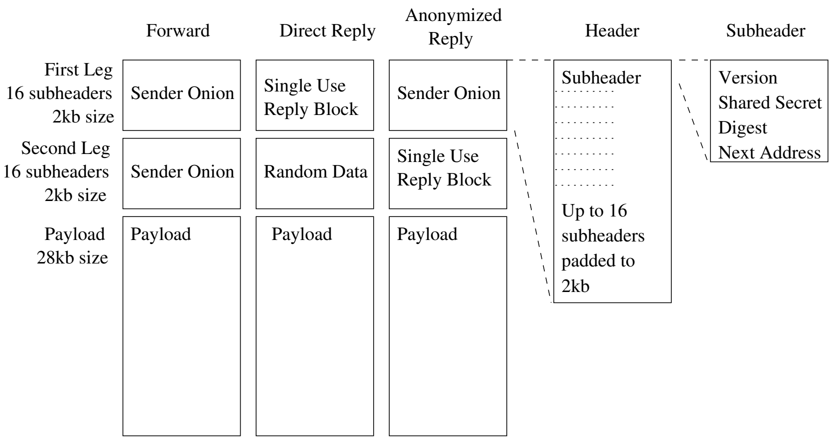


Figure 5.1: The structure of the Mixminion packet format.

mix. If the processing mix is the last one, the procedure will provide a body, and instructions on how to send out the message by conventional email.

We will start by explaining how packets are constructed.

5.2.1 The sub-header structure

A sub-header communicates information to one of the intermediate mixes on the message path. The information fields transmitted are:

```
Version Major: 1 byte
Version Minor: 1 byte
Shared Secret: 16 bytes
Digest:        16 bytes
Flags:         1 byte
Address Size:  2 bytes
Address Type:  2 bytes
Address:       [Address Size] bytes
```

For the purposes of explaining its security properties, we will denote the above sub-header structure as:

$$\mathrm{SH}\,[S, D, F, A] \tag{5.1}$$

S is the secret that will be shared between the creator of the sub-header, and the mix it is destined to. The secret should *never* be used directly. Instead is used as a master secret to derive all the key material for operations on the message. The transformation applied is a secure hash function of the master key concatenated with a type string that is different for each operation.

D is the digest. It is constructed by hashing the contents of the whole header (with the digest field itself set to zero), and the rest of the sub-headers in the same header. By checking this digest a mix ensures that an adversary has not modified the header. This prevents tagging attacks on the header (but not on the rest of the message).

F are the flags, and they indicate what processing is to be applied on the packet.

Finally A is the address of the next mix in the chain or the final address, depending on the flags. It is important to note that the address of the next mix in the chain contains the hash of the signature key of its public key. Therefore an honest mix is able to check that it is passing the packet to the appropriate next mix.

The sub-header is encrypted using the public key of the mix it is destined to. For the purposes of encryption RSA OAEP is used [BR95], with the public encryption key of the mix denoted k_i. We then denote the encrypted sub-header:

$$\mathrm{ESH}_{k_i} = E_{k_i}\left\{\mathrm{SH}\left[S, D, F, A\right]\right\} \tag{5.2}$$

The RSA OAEP padding scheme guarantees that modifications to the ciphertext will be detected. It also guarantees that an adversary cannot submit a modified ciphertext to the mix and gain any information about its contents.

5.2.2 The header structure

Sub-headers cannot be created serially and then aggregated in a header. The creation of the header as a whole, and the individual sub-headers are intrinsically related, and cannot be separated. As we have briefly mentioned above, the digest contained in each sub-header is a hash of the whole header *as it will be seen by the mix node processing the message.* Therefore, while constructing each sub-header, the view that each intermediate mix will have of the full header needs to be computed, to allow the computation of the appropriate hash.

A header can contain up to 16 sub-headers. Given that a 1024bit RSA key is used this gives a size of 2kb for the full header (16×128 Bytes = 2 KBytes). If not all sub-headers are used, the rest of the header must be filled with random noise.

The encoding algorithm for a header is shown in figure 5.2. The inputs to the algorithm are A_i the addresses to which each mix should forward the packets, k_i the public key of each mix in the path, s_i the secrets to be shared, J random noise (junk) to be appended and F the type of header. The type of header relates to the processing that the final hop should do to the packet. As we will see when describing the decoding algorithm it can either pass it to an application (in order to deliver email for example), swap the first and second header after some decoding, or simply forward the message.

The same algorithm is used to encode *single-use reply blocks* (SURB). The address specified, instead of being the receiver, is the address of the sender and creator of the SURB. Furthermore a master key is included in the SURB, that is returned to the sender, and allows them to reconstruct all the intermediate secrets necessary to decode the reply. This allows the SURB decoding to be stateless.

5.2.3 The whole packet

Each Mixminion packet has three parts: two headers, and a body. The second header and the body are encrypted using a derivative of the secrets contained in the first header with BEAR, an all-or-nothing transform. As the packet progresses across the mix network, the second header and body get decrypted by the intermediate nodes.

The second header can either be a header created by the sender, or a reply block. In the first case the body should be encrypted by the secrets contained in the second header; they are known to the sender. In the second case where the second header is a reply block, the body is not encrypted

```
% Requires list of addresses, public and shared keys,
% some Junk and a type.
input: $A_i, k_i, s_i, J, F_n$
    for $i \in (1 \ldots n)$

% Recreates the Junk as seen by each mix.
$J_0 = J$
for $i \in (1 \ldots n)$
    $\text{EncryptionKey}_i = \mathsf{hash}_{\text{H}}(s_i)$
    $\text{JunkKey}_i = \mathsf{hash}_{\text{R}}(s_i)$
    $J_i = J_{i-1}$ append $\mathsf{streamcipher}_{\text{JunkKey}_i}(0^{\text{Size}_i})$
    $J_i = J_i \oplus \mathsf{streamcipher}_{\text{EncryptionKey}_i}(0^{16 \times 128})$
            $[\text{bytes } (16 \times 128 - \mathsf{Len} J_i) \text{ to } (128 \times 16)]$
end for

% Encodes the header a sub-header at a time.
$H_{n+1} = J$
for $i \in (n \ldots 1)$
    $D_i = \mathsf{hash}(\mathsf{streamcipher}_{\text{EncryptionKey}_i}(H_{i+1})$ append $J_i)$
    if $i = n : F = F_n$ else $F = \textit{forward}$
    $H_i = E_{k_i}\{\text{SH}[s_i, D_i, F, A_i]\}$
            append $\mathsf{streamcipher}_{\text{EncryptionKey}_i}(H_{i+1})$
end for

output: $H_1$
```

Figure 5.2: Algorithm for encoding a header

using its secrets since they are not known to the sender.

The second header and the body are encoded in a way that make them interdependent. The encoding scheme is not unlike the Luby-Rackoff structure that provides security against adaptive attacks in block ciphers. The interdependence ensures that if an adversary attempts to change the second header or the body of the message (in order to tag them), both will decode as random noise. Unless an attacker controls all the mixes on the path of a message, he will not be able to extract any information about the message or its final destination.

The full algorithm describing how a whole Mixminion packet is put together is presented in figure 5.3. The procedure takes either two known headers and their key along with a message or one header with known keys, a reply block and a message. The procedure returns a Mixminion packet ready to be sent to the first mix.

5.2.4 Decoding messages

Each mix is only required to be aware of their private key, and a list of previously seen messages. Messages are then received and processed as shown in figure 5.5.

Upon receiving a message the mix attempts to decrypt the first 128 bytes using RSA OAEP and its private key. If the message cannot be decrypted, because the OAEP structure is not valid, it is dropped. Otherwise, the hash contained in the sub-header is used to check the integrity of the full first header. Again if the hash contained in the message is different from the hash of the header the message is dropped.

When the integrity of the first header has been established the keys used to perform different operations are extracted and computed from the sub-header. One of the keys is used with AES in counter mode to generate 128 bytes of noise to be appended at the end of the header. Note that the padding required to keep the length of messages constant is appended at the end of the first header, and not at the end of the message as in Mixmaster.

A variant of the key is used to detect if the packet has been processed before. If the packet has been seen before it is dropped otherwise a hash of

```
% Takes two headers and their associated secrets, and a body.
input: $H_1, s_i, H_2, s'_i, B$
% Takes one headers and its associated secrets,
% a reply block and a body.
        or $H, s_i, H_2 = R, B$

% Phase 1: Hides body with second header's keys.
if $H_2$ not reply: for $i \in (1 \ldots n)$
        $\text{BodyKey}_i = \mathsf{hash}_{\text{B}}(s'_i)$
        $B = \mathsf{bear}_{\text{BodyKey}_i}(B)$

% Phase 2: Makes the second header and the body interdependent.
$\text{HideHeaderKey}_i = \mathsf{hash}_{\text{BH}}(B)$
$H_2 = \mathsf{bear}_{\text{HideHeaderKey}_i}(H')$
$\text{HidePayloadKey}_i = \mathsf{hash}_{\text{HB}}(H_2)$
$B = \mathsf{bear}_{\text{HidePayloadKey}_i}(B)$

% Phase 3: Hides second header and body using first header secrets.
for $i \in (1 \ldots n)$
        $\text{HeaderKey}_i = \mathsf{hash}_{\text{H2}}(s_i)$
        $\text{BodyKey}_i = \mathsf{hash}_{\text{B}}(s_i)$
        $H_2 = \mathsf{bear}_{\text{HeaderKey}_i}(H_2)$
        $B = \mathsf{bear}_{\text{BodyKey}_i}(B)$

% Returns the full Mixminion encoded message.
output: $[H_1, H_2, B]$
```

Figure 5.3: Algorithm for a whole message

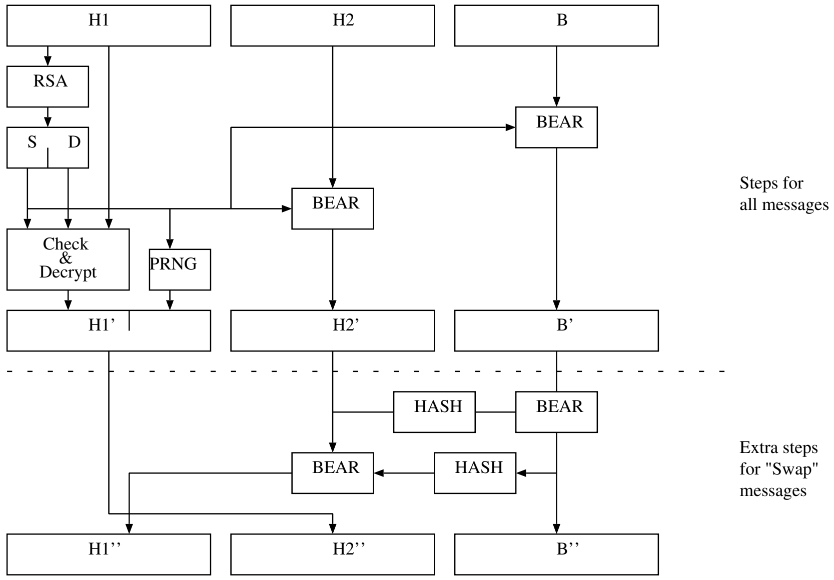


Figure 5.4: The process of decoding a Mixminion packet.

the key is stored to detect replays.

The rest of the first header, beyond the RSA encrypted block, is decrypted using AES in counter mode. Note that the noise is also encrypted as if it was part of the message. An intermediate mix also always decodes the second header and the body of the message using the keys derived from the sub-header. The BEAR cipher is used for this operation.

Beyond the standard decoding specified above, the flags contained in the decrypted sub-header may indicate that the mix should perform the *swap* operation. Conceptually this operation swaps around the first with the second header. More precisely the second header is hashed into a key, and used to decode the body, and the body of the message is in turn used to decode the second header. This ensures that if the body of the message or the second header are modified in any way, they will both result in complete noise. Because of the absolute error propagation of BEAR, if any bit has been changed then an adversary that does not know all the keys involved will not be able to extract any information out of the decoding. Finally the second header and the first header are swapped. Therefore the second header, or the reply block, is providing the routing for the rest of the message's journey.

```
% Requires a message and a private decryption key.
input: $M = [H_1, H_2, B], k_i$

% Decrypts and checks integrity of the first sub-header.
$\mathrm{SH}\,[s_i, D_i, F, A_i] = D_{k_i}\,\{M\,[\textbf{byte } 0 \textbf{ to } 128]\}$
In case of OAEP error drop the packet
Check $D_i \overset{?}{=} \mathsf{hash}(H1\,[\textbf{byte } 128 \textbf{ to } 16 \times 128])$
Check $s_i$ not in database otherwise drop the packet

% Computes keys used to decode all parts.
$\mathrm{EncryptionKey}_i = \mathsf{hash}_{\mathrm{H}}(s_i)$
$\mathrm{HeaderKey}_i = \mathsf{hash}_{\mathrm{H2}}(s_i)$
$\mathrm{BodyKey}_i = \mathsf{hash}_{\mathrm{B}}(s_i)$
$\mathrm{JunkKey}_i = \mathsf{hash}_{\mathrm{R}}(s_i)$
$H_1 = H_1\,[\textbf{byte } 128 \textbf{ to } 16 \times 128]$ append $\mathsf{streamcipher}_{\mathrm{JunkKey}_i}(0^{128})$

% Decodes all parts using the appropriate keys.
$H_1 = \mathsf{streamcipher}_{\mathrm{EncryptionKey}_i}(H_1)$
$H_2 = \mathsf{bear}_{\mathrm{HeaderKey}_i}(H_2)$
$B = \mathsf{bear}_{\mathrm{BodyKey}_i}(B)$

% Performs the swap operation.
if $F = swap$ :
    $\mathrm{HidePayloadKey}_i = \mathsf{hash}_{\mathrm{HB}}(H_2)$
    $B = \mathsf{bear}_{\mathrm{HidePayloadKey}_i}(B)$
    $\mathrm{HideHeaderKey}_i = \mathsf{hash}_{\mathrm{BH}}(B)$
    $H_2 = \mathsf{bear}_{\mathrm{HideHeaderKey}_i}(H_2)$
    Swap $H_1$ and $H_2$

% Gives message to application for processing.
if $F = end$ :
    return $(H1, H2, B)$ to application.

% Sends message to next mix.
else send $(H1, H2, B)$ to $A_i$
```

Figure 5.5: Algorithm for decoding a layer of the packet

5.3 Security analysis of Mixminion

5.3.1 Bitwise unlinkability

In order to provide security against passive observers the Mixminion packet format should guarantee that the message entering a mix *looks* different when it leaves it. The first header of a Mixminion packet will indeed be unlinkable since it has been decoded using AES in counter mode and a private RSA key unknown to the attacker. Note that the quality of the noise appended is not important (it could be zero padding) since it is encrypted with the rest of the first header.

The second header and body of the message are decoded using BEAR with keys unknown to an adversary and cannot therefore be linked to any of the inputs. The swap operation does not add any vulnerability in the sense that the intermediate results (which are not relying on a secret key) are kept secret and only the final form of the packet (after the swap) is output.

5.3.2 Route position and length leakage

A mix knows if it is the last one in the header, because it is required to perform a special operation, such as *swap*, or must forward the message by email. It is very likely that a mix node knows if it is the first one on the path, because the message will arrive from a source that is not an established mix.

Furthermore, the Mixminion packet format does not automatically protect the secrecy of the message transmitted. Therefore an intermediary node might infer that it is the first node on the return path if it is required to transmit a message that is in clear. End-to-end encryption can, of course, be used to avoid this.

Besides the specific cases above, an adversary controlling a mix cannot know the exact position of the mix on the path, or the total length of the path. Note that not even the last node is able to find out the total length of the path, due to the fact that the padding in not included at the end of the message but at the end of the first header. Otherwise the number of trailing blocks of random noise would betray the number of intermediaries.

The total length of the path cannot exceed 16 hops for the first header and

16 hops for the second header or message block. This limit was considered appropriate given that Mixmaster uses much shorter paths. A discussion about the security of choosing paths and path length is provided in chapter 7.

5.3.3 Tagging attacks

Tagging attacks are a family of active attacks performed by modifying mix packets travelling through a sequence of mixes, in order to recognise the packets elsewhere in the network, or when they leave the network. Tagging attacks are greatly enhanced when the attacker controls some subverted nodes in the network, and can therefore control some of the message processing. There are many variants of tagging attacks:

- An attacker tags some part of the message by modifying the ciphertext, and then checks if a message is tagged by using integrity checks (such as a hash or MAC) contained in the message. This attack could be performed by subverted mixes on the path of a message.
- Instead of using integrity checks, an attacker could modify the ciphertext of a plaintext-aware encryption scheme. If the decryption does not yield a well-formed plaintext, the tagging is detected.
- Tagged messages could be detected if the plaintext is at least partially known. This is the case for messages travelling on the forward path since they usually result in a natural language email at the last mix. An attacker could attempt to modify the body to discriminate between well-formed email messages and tagged messages.
- Any of the above attacks can be *permanent* or *temporary*. A tagging attack is temporary if the tagger can remove the tag to allow further processing of the message. This could be the case if a tagging attack relies on an attack in depth against part of a message encrypted using a stream cipher. The attacker would just need to XOR the tag out of the message. An attack is permanent if the tag cannot be removed.

To prevent tagging attacks, Mixmaster attempts to detect any modification of the message. If the message has been modified in any way, it is discarded. Technically this is achieved by making the processing of messages deterministic, by generating the random padding using a shared key, and by communicating a hash of the whole message to each intermediate mix. Resistance to tagging using integrity checks relies on each intermediate mix checking the integrity of the message. It is not sufficient to rely on the last node to check if the message has been tagged, since the last node might be subverted. Therefore the task of protecting against such attacks must be distributed across all mixes in the path.

Mixminion cannot implement this strategy in a straightforward fashion. Because of its support for indistinguishable replies, such a strategy could not be used to detect modifications of the body of the message. The body is legitimately modified to contain the reply to a message. Therefore, while we indeed use this strategy to protect the integrity of the headers that are created by one party, we cannot extend it to protect the integrity of the body of a forward message or a reply. If we used this strategy to protect the integrity of the forward path only (where the message body is known) but not the reply path, this would allow intermediate nodes to infer the nature of the message transported. Making the forward path processing distinguishable from the reply path is contrary to the requirements we have set for Mixminion.

Mixminion follows a different strategy. If the message body is modified, it will be processed correctly but the result will not leak any information about the final destination or the original body of the message. In other words, the tagging attack is not detected early enough to discard the message, but the error introduced is amplified to such a degree that it is impossible to extract any information about the content or destination of the message afterwards. Technically, this is implemented by using large block ciphers to encode the second header and body, allowing maximal error propagation in the event of modification of the ciphertext. The decoding of the second header and body are also made interdependent, so that modifying one would result in both being turned into random noise.

Neither the first nor or the second header of the message can be tagged, because they contain a digest to detect any modifications. Any header modification would result in the packet being dropped. This is similar to the strategy Mixmaster implements, but restricted to the headers.

An attacker can choose to tag the body of the message. This is the reason why the counter-intuitive *swap* operation is performed. If an attacker chooses to tag the message before the *swap* operation is performed then the second header, containing the final destination, and the message body, will turn into unrecoverable noise. If the attacker attempts to tag the message after the swap point, the message has received already enough anonymity for the attack to be fruitless.

It has to be pointed out that Mixminion's resilience to single messages being tagged does not automatically extend to multiple messages known to the attacker to belong to the same stream. In case one of them gets tagged it will turn into random noise, but the information in the other related messages will still be visible. Such attacks are possible against reliable transmission schemes, using forward error-correction codes to ensure the delivery of messages despite messages being dropped by the Mixminion network. Such schemes must be designed with care not to allow attacks.

The careful reader should note the imbalance of security resulting from the routing provided by the first and second header to a forward path message. It is indeed the case that most of the anonymity is provided by the first header, and that the second header could simply contain the final address. In the case of a bidirectional anonymous message, all the sender anonymity is provided by processing the first header and all recipient anonymity by the second.

Tag capacity

It is also instructive to note that not all tagging attacks convey the same amount of information to an attacker. If instead of using BEAR as an encryption primitive, Mixminion used a steam cipher, then an attacker could embed a long stream of information in the packet, that could be later recognised since it would be simply XORed with the plaintext. This would allow the attacker to differentiate between different messages that have been tagged.

Since BEAR is used, any modification results in a completely random decryption, that cannot be correlated in any way with the tag. Therefore one tagged messages could not be distinguished from another, which means that only one tagging attack could be taking place at any time in the whole network to reduce the effective anonymity set of a message to zero.

It has been a matter of intense discussion if such a small amount of information leakage is acceptable or not. Accepting it would greatly reduce the complexity of the Mixminion design, since there would be no need for two headers and a swap point. It would be sufficient to route the anonymous packets using one header, and decoding the body of the message using BEAR at each mix. It anyone tags the contents of the message it is completely destroyed, and is also indistinguishable from any other tagged message.

The decision was taken that no information at all should be leaked, and therefore the swap point was introduced to make sure that all information is destroyed in case a message is tagged.

5.4 Protocols with reply blocks

Mixminion's main novel feature is the provision of secure reply blocks. By secure it is meant that reply blocks provide the same degree of protection against all the above attacks as the messages travelling on the forward path.

A limitation that was imposed on reply blocks, both to make them indistinguishable from forward messages, and to maintain their bitwise unlinkability is that each can only be used once. A list of seen reply blocks is kept, exactly in the same way as for forward messages, and a node will not route a message using the same reply block again. Therefore we have named this feature *single-use reply blocks* or *SURB*. This restriction imposes considerable overheads. Each reply block is 2kb long, and one has to be used for each 28kb message that needs to be sent back as a reply. Therefore protocols need to be modified to minimise the use of reply blocks, and give preference to excess messages in the forward path[2].

It can also be considered insecure to allow a large number of SURBs to be available to untrusted parties. The reason for this relates to the way in which a compulsion attack can be performed on the forward and reply paths.

[2] This was first noted by Peter McIntyre.

In the case of the forward path, a message cannot be traced back from the receiver to the sender. At each intermediary node the message is stripped of the information that could be used to route it back. On the other hand, a reply block can be traced by compelling each intermediary node in turn to perform the decryption until the final recipient is uncovered. This attack is hard for an adversary that does not control all intermediate nodes, but can be performed on SURBs more easily than on the forward path.

For the purposes of discussing protocols with reply blocks, we will first introduce a notation, then illustrate an attack, and then discuss how one could implement a simple Nym Server using the primitives offered by Mixminion.

5.4.1 Protocol notation

The traditional ways of representing protocol steps provide a view of all the participants as an omniscient third party would observe them. Names are included in the protocol steps even in the case where they are not known to the participants, as when describing authentication protocols. It is indeed quite perverse to state that "Alice sends a message X to Bob" when the whole point of the protocol is for Bob to establish that "Alice" is indeed Alice.

In protocols using anonymous communication primitives, it is very important to make a clear distinction between who is actually sending messages, and the knowledge that different participants have of whom they are receiving from or sending messages to.

We shall therefore denote anonymous communication primitives as follows:

$(A) \to B : X$ should denote that Alice sends a message X anonymously to Bob. The message travels using the forward path. Bob is, by convention, not able to observe inside the parenthesis to find that Alice is the actual sender of the message.

$(A)_i$ is used within the body of the messages exchanged to denote a reply block of Alice, that can be used to send a message back to her using the reply path.

$A \rightarrow (B) : X$ denotes Alice replying with message X to Bob using a single-use reply block. Again Alice is not able to see inside the parenthesis, and is not able to link the anonymous recipient with Bob.

$(A) \rightarrow (B) : X$ combines the forward path with the return path and means that Alice anonymously sends a message to Bob.

The protocol notation above assumes that using non-anonymous primitives does not prove who the sender or the recipient is, but provides strong enough evidence to jeopardise anonymity. Therefore, unless an anonymous communication primitive is used, one should assume that the message sent can be linked with the sender or receiver. Note also that one cannot send recipient-anonymous messages without first getting hold of a valid single-use reply block. Therefore an anonymous party can be thought to be giving explicit permission to be contacted back by providing an anonymous reply block.

5.4.2 The *Who Am I?* attack

While the primitives presented above seem quite straightforward a subtle attack is possible if they are simply used as described.

Imagine Alice anonymously runs the following *counting* protocol with many parties:

$$(A) \rightarrow X : \mathrm{id}_x, 0, (A)_{x_0} \tag{5.3}$$

$$X \rightarrow (A) : \mathrm{id}_x, 1 \tag{5.4}$$

$$(A) \rightarrow X : \mathrm{id}_x, 2, (A)_{x_2} \tag{5.5}$$

$$X \rightarrow (A) : \mathrm{id}_x, 3 \tag{5.6}$$

$$(A) \rightarrow X : \ldots \tag{5.7}$$

This protocol introduces some persistent state that, namely the counter that is increased. If a message arrives that does not contain the number expected it is ignored. In order to make the existence of this session explicit, we have introduced the session identifier id_x.

Imagine that Alice runs the simple *counting* protocol with two different parties Bob and Charlie under the respective pseudonymous identities Anna and Amelia respectively. If the two different pseudonyms do not share any data, namely the counters, one would expect them to be unlinkable. By unlinkable we mean that an observer should not be able to tell that they belong to the same principal, namely Alice.

The attack that allows us to link the two pseudonyms proceeds as follows: Anna and Bob are running the counter protocol and have reached the value i, and Bob has Anna's reply block $(A)_{B_i}$. Similarly Amelia and Charlie have reached the number j and Charlie holds $(A)_{C_j}$. Bob and Charlie collude and exchange their reply blocks. Bob then sends to Amelia (not Anna!):

$$B \to (A)_{C_j} : \mathrm{id}_B, i+1 \tag{5.8}$$

$$(A) \to B : \mathrm{id}_B, i+2, (A)_{x_{i+2}} \tag{5.9}$$

If Anna recognises the session and replies to Charlie, although the message was actually sent to a reply address that was provided by Amelia, there is a clear link that is formed. Therefore the two pseudonymous identities have been linked.

In order to prevent this attack, it is important to realise that *in anonymous communication protocols, each principal must cryptographically protect and check their own identity.* This extends the requirement of conventional authentication protocols that authenticate the identity of the other communicating party. Therefore each single-use reply block must provide the genuine recipient in a cryptographically secure way, with the pseudonym to which the message must be addressed. We should therefore denote such a reply block $(A)^{\alpha}_{x}$ where α is the pseudonym provided to the principal A when she receives the message. This pseudonym must be unobservable to anyone else, and its integrity must be cryptographically protected.

The example attack above would then become:

$$B \to (A)^{\text{Amelia}}_{C_j} : \mathrm{id}_B, i+1 \tag{5.10}$$

$$A : \text{Error } j+1 \neq i+1 \text{ and } \mathrm{id}_B \text{ not recognised} \tag{5.11}$$

Alice is explicitly told, by the single-use reply block, that the recipient of this message must be the pseudonym Amelia (that is discussing with Charlie and has state id_C and j). Therefore the message sent by Bob is rejected, and no linking information is received. Note that Bob actually sending the values id_C and $j+1$ would not actually be an attack on the unlinkability of the pseudonyms.

This attack highlights a subtle requirement for single-use reply blocks. What matters, in order to preserve one's pseudonymity, is a cryptographically secure way of knowing who the SURB is addressed to. This is unlike, or maybe complementary, to the requirements of usual authentication protocols that aim to establish the identity of the communicating partner. Note that this should be built in the design of single-use reply blocks, and is not *per se* a protocol failure.

Mixminion prevents such attacks by encoding in the SURB the pseudonym to which the reply block is addressed. This is done by having separate secret keys for each pseudonym to gain access to the information necessary to reconstruct the secrets that decrypt the reply. This unfortunately requires Mixminion to be aware of pseudonyms, although the same service could be provided by allowing a user to encode an arbitrary string with the creation of each pseudonym. This string would be protected against tampering, and revealed when the reply is decoded.

5.4.3 Nym servers

One of the crucial requirements of Mixminion has been to allow users that do not benefit from any anonymity protection to still be able to communicate with users that wish to protect themselves. Furthermore this should be achieved without requiring any special software, aside from a normal email client. This requirement is straightforward in the case where an anonymous sender uses the forward path to send an anonymous mail message. On the other hand, allowing a naïve user to use reply blocks to reply to an anonymous recipient requires manual effort. This manual effort will inevitably lead to mistakes, and will probably provide an unsatisfactory user experience.

Furthermore, Mixminion supports fully bidirectional anonymous channels. Users might wish to use another party's reply block, and anonymously converse with them. Up to now we have never explicitly dealt with the issue of how this initial reply block can be found. One could assume that they are exchanged off-line, but such an assumption is quite unrealistic. It might be acceptable to use an off-line bootstrap mechanism, that for example relies on the exchange of a few bytes representing the fingerprint of a verification key. On the other hand a two kilobyte SURB cannot be exchanged by hand, and other means could leave forensic traces.

In order to solve both problems, special services named *nym servers* provide a gateway between the world of Mixminion and conventional email. Furthermore they allow normal mail users to initiate conversations with parties they did not have any contact with before, and therefore allow the *first contact problem* to be naturally solved.

It is important to realise that nym servers are trusted to reliably perform their tasks, guaranteeing the availability of the service offered, but are not trusted to safeguard the anonymity of the users. In other words a misbehaving or subverted nym server can jeopardise the delivery of messages, but cannot link pseudonyms to users. This is of course assuming the underlying Mixminion transport provides perfect anonymity, and that the aggregate information transmitted, or its content, does not leak enough side information to uniquely identify the pseudonymous user.

A simple protocol to implement a nym server would work in two phases. First the *registration phase* and then the *request phase.* In the registration phase a user registers a pseudonym with the server, along with a verification key V_{Anna}. All the messages coming from the pseudonym will be signed with this key (denoted S_{Anna}). The registration phase is:

$$(A) \to \text{Nym} : \{\text{Anna}, V_{\text{Anna}}, (A)_0^{\text{Anna}}\}_{S_{\text{Anna}}} \tag{5.12}$$

$$\text{Nym} \to (A)_0^{\text{Anna}} : \{\text{Status}\}_{S_{\text{Nym}}} \tag{5.13}$$

The nym server replies with the status of the request. Usual errors might include the request concerning a pseudonym that is already taken. A positive reply assures that the nym server has stored the association between the key and the pseudonym, and that it is ready to accept email for it.

The request phase is then used by Alice to retrieve her email from the nym server. Alice periodically sends a few reply blocks that are used by the nym server to relay the messages back to her.

$$B \underset{\text{SMTP}}{\rightarrow} \texttt{Anna@Nym.net} : M_1 \tag{5.14}$$

$$(A) \rightarrow \text{Nym} : \{\text{Anna}, (A)_1^{\text{Anna}}, \ldots, (A)_i^{\text{Anna}}\}_{S_{\text{Anna}}} \tag{5.15}$$

$$\text{Nym} \rightarrow (A)_1^{\text{Anna}} : \{M_1\}_{S_{\text{Nym}}} \tag{5.16}$$

This simple scheme for a nym server makes some choices that are controversial, and have to be justified. First of all, no single-use reply blocks are intended to be stored on the nym server. There are several reasons for this. As discussed earlier reply blocks are inherently less secure against compulsion attacks, since they provide an adversary with a bit sequence he can request to be decoded. Therefore the nym server holding SURBs would present a greater risk to the anonymity of the users.

On the other hand holding messages does not present the same problems. The content of the messages does not make the nym server liable in most jurisdictions [EU00]. The secrecy of messages can also be safeguarded by encrypting them, which is an orthogonal issue. Therefore, aside from the greater size requirements necessary to store the messages, and the potential additional latency from a security point of view, it is a preferable choice.

Careful examination would also reveal that the authenticity of the messages between the nym server and the anonymous users does not need to be non-repudiable. In other words it is enough to allow the user and nym server to satisfy themselves that messages indeed originate from one another, but never to prove it to third parties. Therefore it is preferable to introduce a repudiable scheme for providing such integrity guarantees. For example a Diffie-Hellman key exchange can take place in the registration round, and provide a secret to be used to compute message authentication codes on subsequent messages. Such a scheme would destroy a great part of the evidential value any seized material might have.

A robust implementation of a nym server would have to make sure that no email is ever deleted until it has received confirmation that the pseudonymous user has received it[3]. A worthwhile observation is that reliable transmission

[3] Such a mechanism has been investigated by Peter McIntyre during his final year undergraduate project.

mechanisms cannot be directly used in this context. There is a clear asymmetry between the nym server and the user, in the sense that the user can always communicate with the nym server, while the server has only a limited number of reply blocks to use. Therefore a reliable transmission layer would need to take into account this asymmetry and preserve the single-use reply blocks available to the server, by requiring, if necessary, the user to retransmit acknowledgements, rather than the nym server retransmitting messages.

Finally it is worth noting that the nym server only requires the ability to receive normal email. As a service it is therefore less prone to abuse than even the regular Mixminion nodes that allow anonymous messages to be sent out to normal email addresses.

5.5 Beyond Mixminion

Mixminion is an evolving system. Early designs concentrated on securing bitwise unlinkability and minimising the potential of active attacks, but further improvements regarding efficiency and reducing the overheads on messages have been achieved or discussed since.

5.5.1 Lowering the overheads of Mixminion

The original 1024 bit OAEP-RSA encryption of each sub-header was replaced by the 2048 bit equivalent. This was done to avoid attacks on the reputation of the Mixminion design, claiming that the key size is smaller than what is conventionally considered secure, and provide security to messages for years in the future. Implementing this change in a straightforward manner would imply that the sub-headers and therefore headers would double in size, if one was to allow the same maximum number of relays to be used. For similar reputation reasons the maximum number of relays could not have been reduced, and therefore alternative strategies had to be employed to reduce the size of the headers.

A technique similar to *hybrid encryption with minimal length* [JMP$^+$98] has been independently invented by Nick Mathewson and used to reduce the header overheads in Mixminion. It uses the spare space inside the RSA encryption of the outer layer, not only to encrypt content destined to the intermediary node, but also part of the information to be relayed. It allows all the space available inside the OAEP envelope to be used, and therefore allows for 16 sub-headers, using 2048 bit RSA, to be encrypted in less than 2 kilobytes. Therefore the overhead of this scheme is comparable to the overhead of the scheme with shorter keys that does not implement this optimisation.

Further reduction of the size of the headers is possible, but requires a careful security analysis. It was an early choice to use OAEP padding to provide plaintext-aware and semantically-secure encryption, thereby protecting the sub-headers from active attacks. The secure and well-understood OAEP padding reduces the size of the encodable plaintext since it includes large nonces (20 bytes) and secure digests (another 20 bytes). The Mixminion sub-header structure replicates some of the functionality of the nonces, and the digests, in order to to secure the integrity of the full headers. Therefore a possible avenue for reducing the overheads could be to design a custom RSA padding scheme, based on the information that can also be used both for keying, namely the master secret and the digest that will be used to check the integrity of the full message. Such a scheme could immediately reduce each sub-header by 40 bytes, but cannot be treated as secure until it has received due consideration by the cryptology community.

There are also compelling arguments for reducing the size of all the SHA-1 hashes to 10 bytes (80 bits). No attack against Mixminion packets has been exposed that relies on breaking the strong collision-resistance properties of the digests present in the sub-headers. The only party that could benefit from such attacks would be the creator of the packet whose anonymity is protected. Therefore, unless the party to be protected consents, the attacker cannot mount any attack based on birthday properties. Such a modification would be extremely controversial, and would open Mixminion to reputation attacks. Therefore the full 20 bytes of SHA-1 are included in each sub-header.

5.5.2 Simplifying the swap operation

As mentioned before, the first header of the Mixminion packet, offers most of the anonymity. The swap operation is only there to make sure that the message and the routing information are destroyed in case of tagging attacks, and the second leg, using the second header, can be very short. A more efficient construction would therefore make the second header extremely short.

A further refinement would be to construct such a cryptographic transform that if the message is modified in any way the final address and the content are destroyed. To do this a convention can be used: the data to be destroyed should be at the end of the packet, and a mode of operation could be used that propagates errors forward. Therefore if the packet is modified at any point the sensitive data will be unreadable.

An appropriate, although inefficient, mode of operation, using a block cipher with encryption operation E and a hash function H, could be:

$$C_i = E_{K \oplus H(C_1 P_1 \ldots C_{i-1} P_{i-1})}(P_i) \tag{5.17}$$

It is clear that this mode of operation will propagate any errors in a particular ciphertext to the rest of the decryption since the subsequent keys are all dependant on this ciphertext. We will denote encryption using this mode of operation and secret key K as $\{\cdot\}_K$.

A packet travelling through mixes $A_0, A_1, \ldots, A_2$, with payload M and final address A could then be constructed using the new mode of operation and RSA OAEP. We denote RSA OAEP encryption using the public key A_x as $\{\cdot\}_{A_x}$.

$$\{A_1, K_1\}_{A_0}\{\{A_2, K_2\}_{A_1}\{\{-, K_3\}_{A_2}\{\mathrm{BEAR}(A, M)\}_{K_3}\}_{K_2}\}_{K_1}, J \tag{5.18}$$

This construction is extremely similar to traditional mix network packets, except that it uses the new mode of operation and instead of communicating to mix A_3 the final address in the RSA encrypted header it includes it in the body of the packet. Furthermore the BEAR all-or-nothing transform is used on the final address and the content of the packet. This makes sure that if any part of it is modified no information can be extracted any more.

The decoding of the packets is very simple. Each node decrypts the header using its RSA private key, and extracts the shared key K_x and the final address A_x. It then appends a block of junk J_x, as long as the RSA encrypted block, at the end of the packet and decodes the whole message using the shared key K. The result is sent to the next mix.

Indistinguishable replies can also be used. A reply block addressed back to a user S would look like:

$$\{A_1, K_1\}_{A_0}\{\{A_2, K_2\}_{A_1}\{\{S, K_3\}_{A_2}\{\{K_{\mathrm{master}}\}_S\}_{K_3}\}_{K_2}\}_{K_1} \tag{5.19}$$

The key K_{master} is returned to the receiver S and can be used to reconstruct all the intermediate keys $K_1, \ldots, K_4$ to decode the message appended to the reply block.

The security of the scheme presented relies on the fact that all sensitive information will be destroyed if an adversary modifies the message. If the modification takes place on one of the headers, then the final address and the content will be completely destroyed. If the content is modified, the BEAR operation will ensure that again the content and final address are destroyed. If the tagging takes place on the junk, the adversary will not be able to observe any difference since it is random noise anyway.

5.6 Summary

We have presented Mixminion, an anonymous remailer providing facilities for both sender-anonymous messages and anonymous replies. We have analysed its security properties and it should be resistant to passive attackers, subverted intermediaries, and active attackers. The technical breakthrough is the use of "fragile" primitives to render any information contained in the message useless in case of a tagging attack.

The single-use reply block facilities provided can be used to build a variety of protocols. The simple *Who Am I?* attack demonstrates that such protocols are subtle and require careful study. A nym server is provided as an example of such a protocol.

Chapter 6

Forward secure mixing

> *"Behind our black mask, behind our armed voice, behind our unnameable name, behind the 'us' that you see, behind this we are you."*
>
> *Speech, July 1996* — Major Insurgent Ana Maria

In the previous chapter we examined Mixminion, a state-of-the-art anonymous remailer, that can be used to anonymise message-based communications. We have presented a security analysis of Mixminion, and in particular its resistance to passive attacks, active tagging attacks, and resilience to a set of subverted mix nodes. These threats are very much in line with the traditional cryptological threat model, as presented in detail in section 2.1.1, but the analysis does not fully address the compulsion threat model. Here we will assume that an adversary is capable of requesting information from honest mix nodes and clients.

In this chapter, we will describe how an adversary can use compulsion powers to trace anonymous communications through the network. We will also define the notion of *forward secure anonymity.* This is similar to forward security in the sense that unconditional anonymity is provided after a certain point in time. We present a construction that can be used to implement forward anonymous properties in remailers such as Mixminion. A security analysis is also provided to assess how the cost of tracing messages is increased as a result of the techniques proposed.

6.1 How to trace a mixed message

The object of our study is to determine how an opponent could use the compulsion powers (described in section 2.1.2), in order to trace a sender-anonymous communication or to find out the originator of a reply block. In particular we will assume that the adversary can ask mixes to decode messages presented to them, and provide the resulting message and its destination, along with any other information embedded in it.

Compulsion powers are usually used in addition to more traditional powers, such as passive logging of messages. In the absence of any content interception of mixed messages through the network, it is extremely difficult to *trace back* a forward path communication. Assuming that the mix network provides some anonymity the recipient of the message does not hold any bit-string that they could ask any intermediate mix node to decrypt and trace back. All the routing information has been stripped and deleted by the time the message arrives at its final destination.

In order to trace back a communication, an attacker is required to work forward starting at an interception point, by requiring mixes to decrypt the material intercepted, hoping that the communication traced ends up being the one that had to be traced. This process means that the opponent will need to acquire exponentially many decryptions in the number of hops to trace a message. A good anonymizing protocol would force this effort to be as large as to require all messages present in the mix network to be decrypted.

However an opponent interested in *tracing forward* a communication initiated by a particular user, only needs to put this specific user under surveillance and request all intermediate nodes that relayed the messages to reveal the destinations of the messages. Therefore in the case of directed surveillance against a particular user, interception needs only to take place around that user, and then intermediate nodes can be requested to decrypt the ciphertexts until the ultimate destination is revealed.

If near-ubiquitous content surveillance is a reality, the above procedure becomes much more efficient. It is only necessary for the opponent to require the last hop to decrypt all communications that were intercepted until the message to be traced is decrypted. Then the procedure is repeated on the previous hops recursively. Different mixing strategies [SDS02] may make such an attack slightly harder, but it should in general be linear in the number of mixes used by the communication (although all content has to be intercepted).

Where a single-use reply block has to be traced back to its destination, much more information is available to the attacker. Reply blocks contain all the routing information needed to deliver the message, encrypted under the private keys of intermediate mixes. Therefore the attacker can simply request the reply block to be decrypted iteratively which requires a number of consecutive requests equal to the number of mixes. It has to be noted that single-use reply blocks are inherently easier to trace, and the process does not require any interception to be reliably performed[1].

This has a considerable impact on the forward channels if they contain single-use reply blocks, for the purpose of allowing the recipient to reply. It is therefore intrinsically safer not to include a reply block in very sensitive sender-anonymous communication. It is also wise, to design protocols with the compulsion threat model in mind, to minimise the time validity and volume of single-use reply blocks available to intermediaries that could be subject to attacks. The nym server presented in section 5.4.3, is a good example of such a design.

To summarise, it takes a number of requests equal to the number of hops to trace the recipient of a reply block, or the recipient of a forward path communication. On the other hand, in the absence of blanket content interception, it takes an exponential number of decryption requests to confirm the sender of a given message received. If blanket content interception is in place the cost of confirming the sender in the latter case is also proportional to the number of hops times the number of messages mixed together in each batch.

[1] This was first stated by Roger Dingledine during one of our discussions.

6.2 Defining forward anonymity

We have presented, in section 3.5, forward security as the property that guarantees that a successful attack will not jeopardise any past sensitive information beyond a certain point in time. In the context of a secure channel, for example, key compromise should not reveal any information about any previous communication.

In the context of anonymous communications, we want to make the decoding of a transited message impossible by honest mixes after a certain amount of time or after a certain event. We shall call this property *forward secure anonymity* and a mix that implements it a *forward secure mix* (*fs-mix*).

In the case of mixes, the event that will trigger this property is the message being processed. In other words we will aim for a mix not to be able to decode any message a second time. As we will see the scheme proposed does not provide perfect forward secure anonymity, but raises the cost of an attack considerably by requiring the adversary to perform a large number of requests for decryption.

6.3 The *fs-mix*

We can achieve forward secure anonymity by introducing secret state into the mix nodes. This is not a radically new requirement, since most techniques implementing replay prevention already expect mixes to keep a database of the hashes of the packets, or other data, that have been processed in the past. The difference is that the new state we require has to be kept secret, since it will be used to derive keying information.

In traditional mix systems [MCPS03, GT96, DDM03a] the address of the next mix ($A_{M_{n+1}}$) and the session key used to decrypt the payload to be sent (K_{msg}) are included in the asymmetrically encrypted header, under the public key of the mix (Pk_n).

$$M_{n-1} \rightarrow M_n : \{K_{\text{msg}}, A_{M_{n+1}}\}_{Pk_n}, \{\text{msg}\}_{K_{\text{msg}}}$$

Traditional mix systems, such as Mixmaster, store a packet ID that is used along with some integrity checking to avoid messages being processed more than once by the node [MCPS03]. In the case of Mixminion a special public hash N_{replay} of the symmetric key K, otherwise used to decrypt parts of the message, is kept in order to avoid replays of the same message. If another message's secret hashes to the same value, it is dropped.

$$N_{replay} := H_1(K_{\text{msg}})$$

We propose keeping a second secret hash of the symmetric secret (K_i) in a table indexed by a public value N_i:

$$K_i := H_2(K_{\text{msg}}) \text{ and } N_i := H_3(K_i)$$

When a message goes though the mixing network it leaves behind it a sequence of keys and their indexes on each of the nodes it went past. Future packets can therefore use these keys, in conjunction with secrets they carry, in order to decrypt both their addressing information and their payload. So a node would first decrypt the headers of a message using its private decryption keys and then read the index of the key to be used N_j, and retrieve the appropriate secret K_j. It would then compute a secret shared key K_{final} based on both the key K_j and the secret K_{msg} contained in the decrypted header:

$$K_{\text{final}} := H_4(K_{\text{msg}}, K_j), K_l := H_2(K_{\text{final}}) \text{ and } N_l := H_3(K_l)$$

$$M_{n-1} \rightarrow M_n : \{S_{\text{msg}}, N_j\}_{Pk_n}, \{A_{M_{n+1}}, \text{Message}\}_{K_{\text{final}}}$$

In order to make it impossible for an attacker to force the decryption of this message or to gain any information by accessing the list of secrets K some key updating operations need to be performed, to replace the old keys by new values:

$$K_l := H_5(K_{\text{final}}) \text{ and } N_l := H_3(K_l)$$

As soon as the old values (N_j, K_j) are replaced by the newly generated (N_l, K_l) they must be permanently deleted. It is wise to perform this operation before the decoded message has been sent out into the network.

To summarise, the H_1 hash function is used to extract a public tag of the message to avoid replays and is always applied to the secret contained in the

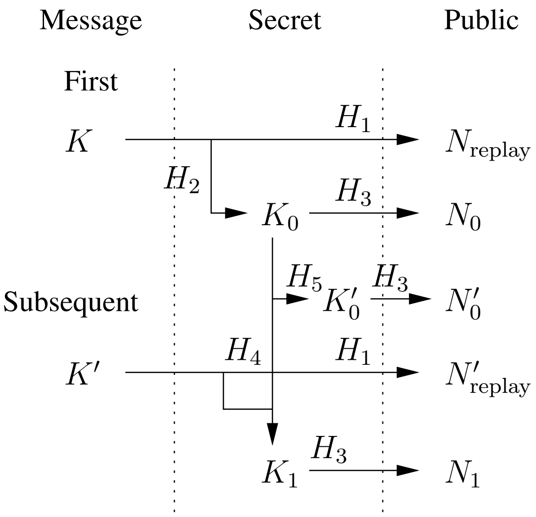


Figure 6.1: Cryptographic key derivation in an *fs-mix*.

message. The function H_2 is applied to the final symmetric secret in order to generate keys that can be used in future communications, and the function H_3 is applied to these keys to generate their public indexes. The function H_4 is used on the secrets stored in the mix and the packet to create final shared secrets while the function H_5 is used to update secrets. All functions H_x are secure hash functions, in particular they have to be pre-image resistant, to avoid giving any information about the key K_i associated with an index $N_i := H_3(K_i)$ or the relation between new and old keys $K_j := H_5(K_{\text{final}})$.

There must be a special index N_0 indicating that the message does not depend on any existing key present on the server, and the message should be processed in the traditional way. This way clients and mixes can bootstrap the communication and relay messages even if no previous secrets are shared and stored. If messages refer to a key index that does not exist they should be dropped.

6.3.1 The cost of an *fs-mix*

The properties provided by *fs-mixes*, as described above, are not achieved for free. The next section will explore the security advantages of this new scheme but first we will discuss the additional expenses both in terms of storage and processing.

A traditional mix only needs to maintain a public record of the ID of all the messages it has ever processed under its current private key. Therefore the storage required after processing n messages is $\mathcal{O}(n)$. For each new message processed a lookup is performed on the public record. Assuming that the lookup table is implemented using a hash table one can expect a cost of $\mathcal{O}(\log n)$ to perform each lookup.

An fs-mix stores more state than a traditional mix. It will need to store m pairs of (N, K) values. Unlike n, the number of messages processed by the current private key of the server, m is proportional to the number of messages that have ever been processed by the mix. As above the cost of finding a particular element should be proportional to $\mathcal{O}(\log m)$. In addition to the lookups as many as four hash operation might need to be performed per message.

In order to minimise the state needed by an fs-mix, entries in the index and key table can be made to expire, either automatically or as requested by the sender. More details about this scheme will be presented in section 6.5. Care must be taken not to reveal through the granularity of the timestamps or dates the link between messages and keys stored.

6.4 Security analysis

We shall argue that the modifications presented do not make the mix any weaker, when it comes to bitwise unlinkability, while providing additional security features. It is clear that if the keys K_i contained in every mix were public, an adversary would be exactly in the same position as in a traditional mix architecture. He would have to compel the mix to perform decryption using its private key or surrender it, in order to trace any material that is in his possession. Therefore in that case the new scheme is equivalent to the traditional one.

6.4.1 Protection against compulsion

An attacker that tries to trace back a message that has already gone past an fs-mix cannot decrypt it unless all the messages upon which this communication's key depends were also intercepted and decrypted. Such an exercise would require all traffic to all the mixes to be logged and all of it to be decrypted in order of arrival for the attack to work, since the attacker does not have any a priori knowledge of the message dependencies. The above is true for both sender-anonymous communications and single-use reply blocks.

If keys K are seized, the first messages referring to them can be traced. The attacker then needs to intercept all subsequent messages in order to update his knowledge of keys to maintain his decryption capabilities. For each of these messages there must be a decryption request made to the mix (unless the private keys are seized).

Since it is impossible to decrypt not only the address also but the whole message, there is no way an opponent could try to use the body of the message in order to find out its destination. Therefore even a single honest mix in a chain that implements a *forward secure anonymous* channel, that has genuinely deleted the used keys, is sufficient in order to provide forward security for the whole communication.

The cost of attacking the system is not much more expensive if the attacker performs an active attack, and extracts the message from the network before it is processed. He can then proceed to show the message to all the intermediate nodes, requesting the usage of the appropriate stored secrets. In particular this could be done for single-use reply blocks, since they are often readily available. Therefore the lesser security of distributing reply blocks, in the face of compulsion attacks remains.

6.4.2 Traffic analysis

Although the cryptographic security of fs-mix messages is stronger than in the traditional mixes, there is a new requirement imposed on the selection of routes that packets need to travel through. If there is a need for state to be present on the mix from previous packets, that means that the same mixes are chosen repetitively in a non-random manner. That selection process might provide enough information for the adversary to be able to link messages between them.

Since the number of nodes applying the proposed scheme does not need to be very large, and provided that there is a small number of mixes, the traffic analysis of the route should be hard. This is the case because a particular node has a high probability of being present in two routes anyway. In the alternative scenario of a large peer-to-peer mix scheme, the security of such protocols against traffic analysis has to be re-evaluated.

Additionally, the intermediate mixes are aware of the correlation between messages, since the only party that knows the keys stored is the party that has sent the previous messages. They are also aware of two other nodes on the path that are seeded by the same messages (the one preceding them and the one after them).

It is worth noting that the messages used to distribute keys are identical to otherwise normal messages, as far as any outside observer or compromised mix is concerned. This is an essential feature, that makes the traffic selection task of extracting messages that contain key material extremely hard for the adversary. In particular the key trail left behind normal messages could be used as key material for further messages.

6.4.3 Robustness

If the anonymity of a network cannot be broken, an attacker might choose to degrade its performance, hoping that it will put off people from using it. Therefore a large body of work concentrates on proving the correctness of mix networks (described in section 4.2.8). Unfortunately, the requirement upon fs-mix nodes to store additional state makes the transport of messages more fragile and the nodes more prone to denial-of-service attacks.

If messages can get dropped at random, due to traffic congestion or network failures, making future messages reliant on past ones makes the network even less reliable overall. One could ensure the proper delivery of special key distribution messages, before making further messages dependent on them. For example a client could send messages back to herself and check for delivery before using the secrets distributed to route further messages. Unfortunately there are security implications, namely the risk of having the key distribution message intercepted and traced, as well as all further messages linked. On the other hand this strategy ties in very well with *rgb-mix* techniques described in chapter 8, to avoid $(n-1)$ attacks.

6.5 Additional features

In addition to the mechanisms described above, having facilities to make messages dependent on keys on the nodes could be used to implement *interdependent reply blocks* and *path burning messages*.

Without modifications to the mechanisms described above it is possible to make many reply blocks depend on each other, in such a way that once one of then has been used the other ones cannot be traced back. We put in the path of all the reply blocks a common mix, that has been given a particular secret by a previous message. All SURBs are constructed so that they can only be decrypted by a single secret entry on the shared mix node. As soon as the first of the messages routed using one of the interdependent SURBs is decoded, the single secret entry is updated and the other SURBs get dropped. Furthermore, it is impossible to trace any of them back.

Such a mechanism could be very useful if for purposes of resilience many single-use reply blocks have been communicated to a third party in order to carry back a single message. If all these SURBs share a reliable mix and use the same key, the first one going through the mix will destroy all information that could be used by an adversary to trace the others.

Much in the same way, it is possible to make reply blocks valid for only a limited amount of time. After a determined time period a message is constructed that uses the same intermediate secrets as the SURBs, and is fired into the mix network. This message updates the keys, and any subsequent messages depending on them will get dropped. The same techniques can be used to make reply blocks valid only after a particular time period by only providing the necessary keys at some future time.

By adding a timestamp to each of the keys one can make sure that the intermediate nodes automatically delete the keying material. This would provide the same functionality as the Path Burning Messages without requiring the principals who wish to maintain their anonymity to actively delete keys. An expiry time could be specified by the user along with the key. This has the disadvantage that it could be used for traffic analysis if any logic is used to calculate this expiry date, that takes into account the current time, or other information local to the user.

6.6 Other forward security mechanisms

Some other solutions are available to make the cost of compulsion attacks higher or prohibitive for an adversary.

6.6.1 Forward secure link encryption

A cheap way to render link level surveillance and recording of content fruitless for an opponent is to use a forward secure encrypted channel between mix nodes. Technically this involves encrypted channels established using key exchanges with ephemeral public keys, signed with long-term signing keys. The ephemeral keys are discarded immediately after a session key has been established. After each message is processed the session key is updated using some hash function, some exchanged fresh nonces and possibly the message content. The old keys and the nonces are systematically deleted after the update has taken place. This makes it impossible for nodes to decrypt past messages that are presented to them, since the keys used have been deleted. It is essential that fresh nonces are used during the key updating. This forces an adversary that at some point in time seizes the keys used by the mixes to indefinitely observe the channel to keep his knowledge of the current keys up to date. Mixminion uses TLS [DA99] with a signed ephemeral Diffie-Hellman key exchange and key updating as a forward secure link between mixes.

This technique renders interception at the link level useless, assuming that mixes have good ways of authenticating each other. If only a small number of mixes operate, this should not be a practical problem. The task of authenticating mixes could also be performed by the creator of the anonymous message, by including the hash of their verification keys in the anonymous routing data, as it is the case in Mixminion. An honest node should check that this hash matches the verification key during the key exchange protocol before establishing the encrypted channel. It also makes it difficult to perform active attacks, as described in [SDS02], such as inserting, delaying, deleting or modifying messages on the network links, that can be used to flood nodes to decrease the amount of anonymity they provide .

Messages, in a form suitable for compelling mixes into decrypting them, can still be intercepted by malicious nodes and presented to the next honest mix in the chain. In order to detect malicious nodes the name of the previous mix could be contained in the headers of messages. In that way it is impossible to hide which node performed the interception or the previous decryption.

6.6.2 Tamper-proof secure hardware

In order to minimise the risk of being compelled to surrender key material or decrypt intercepted material, one could implement the core functions of the mix inside a tamper-proof hardware box. The sensitive functions, including the decryption of messages and the replay prevention, would need to be performed by the secure co-processor in order to avoid compelled decryption. Assuming that the cryptographic co-processor is secure it should be extremely difficult to extract the secret key material inside it. This construction protects against compulsion to reveal keys, but does not protect against subverted mixes, that want to trace the correspondence between inputs and outputs in the cryptographic module. In addition to the decryption and replay prevention, the secret permutation needs to be performed inside the cryptographic module in order to protect against such attacks. Furthermore, mechanisms must be in place that allow third parties to assure themselves that the trusted hardware and software is indeed being used, and has not been modified. This is a likely area of future research.

6.7 Summary

We have described the impact that an adversary with compulsion powers might have on a mix network, and described the effort required to trace messages and single-use reply blocks. Then we presented the fs-mix, that increases the cost of such an adversary by making messages interdependent on each other. Therefore all messages need to be intercepted and decrypted for an adversary to be sure that any target message can be traced. The resulting system is more secure, but also more fragile.

Mixminion implements a partial solution to the problem of compelled decryption by using forward secure link encryption.

Chapter 7

Sparse mix networks

> *"Those targets like terrorism and weapons transport are used as a cover for the traditional areas of spying, which are political, diplomatic, economic and military."*
>
> *European Parliament ECHELON Committee*
> — Nicky Hager

In this chapter we present and discuss some proposals for the topology that mix networks might assume. We look at a fully connected graph, and a mix cascade. We then discuss the advantages and disadvantages of a restricted network topology, a sparse constant-degree graph, and analyse it using existing work on expander graphs. Finally we compare the anonymity and other properties provided by this new topology against more traditional topologies.

We will prove that such restricted networks scale well in the number of mix nodes. The route length necessary to provide maximal anonymity grows only logarithmically in the number of nodes in the network and the total amount of genuine traffic required to protect the network against traffic analysis and intersection attacks grows linearly with the number of mix nodes.

Although the research presented was inspired by uncertainties concerning how path selection and topology restrictions might affect Mixminion (see chapter 5) the results presented concern a much wider variety of mix networks. In particular the calculation of traffic required to perform traffic analysis on a stream of traffic taking the same route applies directly to onion routing and other circuit-based anonymity mechanisms.

7.1 Previous work

Some work has already been done on the topology of mix networks. The network topology influences how clients choose the path their messages take through the network.

In [Cha81] Chaum introduces mix networks as a collection of nodes that relay messages between each other from their original senders to their final recipients. Throughout the paper there is an implicit assumption that all nodes can communicate with all other nodes, therefore forming a fully connected network. Clients choose the path their messages take by selecting a sequence of nodes at random from the set of all existing mix nodes. Mixmaster [MCPS03] which follows Chaum's original proposals quite closely, also allows clients to choose any route through the network, using reliability statistics [Moc] to select nodes. Other proposals for route selection use reputation metrics [DFHM01] to quantify how reliable mixes in a network are.

While the fully connected nature of the mix networks seemed to improve the anonymity provided, Berthold *et al* [BPS00] found that they can be vulnerable against very powerful adversaries, such as those who control all the mix nodes except one. In particular, if only one of the mixes in the network is honest, the anonymity of the messages going through it will most likely be compromised. Attackers can perform intersection attacks, while the probability that all nodes in a short path are compromised is very high. Additionally if two or more messages follow the same route, attacks are trivial to perform.

As a solution a *cascade* [JMP$^+$98, PPW91, BPS00] of mixes is proposed. Users of the network are not free to choose which route to take, but are all forced to route their messages though a predefined sequence of mixes. Further work has been done to improve the reliability of networks of cascades against subverted nodes using reputation [DS02]. The obvious drawbacks of cascades are the small anonymity sets they provide in the general case due to the fact that they do not scale well to handle heavy load, and high-latency. Cascades are also vulnerable to denial-of-service attacks, since disabling one node in the cascade will stop the functioning of the whole system. Some solutions are proposed to solve the problem that active attacks could pose, but require user authentication to work properly [BPS00].

The freedom network [BSG00] only allows restricted routes to be used, for performance reasons but without any published analysis about what repercussions on anonymity such restrictions on the network might have. In [BPS00] Berthold *et al* briefly introduces the possibility of having mix networks that are sparse, but then as we will see, focuses on describing the benefits of mix cascades.

7.2 Mix networks and expander graphs

We propose a mix network with a network topology based upon sparse constant-degree graphs, where users may only choose routes following this topology. Furthermore, each link out of a node should be followed by messages according to a predefined probability distribution. Therefore, selecting a path in the network can be approximated as a random walk on the corresponding weighted graph. We will show that this network provides some of the advantages of cascades, while being more scalable. We will provide theoretical anonymity bounds using the metric presented in section 2.3, and define under which conditions the network provides anonymity close to the theoretical limit. Minimum traffic bounds to prevent the network being vulnerable to traffic analysis and intersection attacks are also calculated.

The topology that we propose for the mix network is based on expander graphs. Expanders are a special family of graphs with the following properties: a D-regular graph $\mathcal{G}$ is a $(\mathcal{K}, \mathcal{A})$-expander if for every subset $\mathcal{S}$ of vertexes of $\mathcal{G}$, if $|\mathcal{S}| \leq \mathcal{K}$, then $|N(\mathcal{S})| > \mathcal{A}|\mathcal{S}|$ where $|\mathcal{S}|$ is the number of vertexes in $\mathcal{S}$ and $|N(\mathcal{S})|$ is the number of nodes sharing an edge (neighbouring) with a vertex in $\mathcal{S}$. In plain language it means that any random subset of nodes will have "many" different neighbouring nodes. In practise expanders have a relatively small and constant-degree D in relation to the number of edges of the graph, and a large expansion factor A, that is proportional to the number of “neighbours”. A good introduction to expander graphs and their applications can be found in [LW03].

A relevant result is that most bipartite graphs with degree of at least three provide good expansion properties. A topology based on a random bipartite graph, with each mix node having three fixed neighbours will be an expander [Pin73] with high probability. Therefore, such networks can be constructed by brute force, or by using the surveyed or proposed methods in [RVW00]. The families of expanders with explicit constructions presented in [RVW00] have a constant, but large, degree and also an arbitrary large number of nodes, which makes them practical for large networks.

The first question that comes to mind is quantifying the anonymity that such networks provide in comparison to fully connected networks. In a fully connected network a message coming out of the network has a probability of originating from a particular node that is proportional to the input load of that node. As we will see, the position of a message after a random walk though the expander graph will converge toward the same probability after a number of steps proportional to $\mathcal{O}(\log N)$ where N is the number of nodes in the network [Gil93]. This represents the a priori knowledge of an adversary that only knows the topology of the graph, but no details about the actual traffic going through it.

The intersection attacks presented in [BPS00] rely on the fact that messages using the same sequence of nodes will only occur in a fully connected network with a relatively small probability. Since networks based on small constant-degree graphs only provide a limited choice of routes for messages to take, nodes can wait so that enough messages are accumulated before sending them, to make sure that all links have traffic on them. Because there is only a linear number of routes to fill with traffic, only order $\mathcal{O}(DN)$ messages are required where N is the number of nodes and D the degree of the graph. This strategy is more efficient than filling all the $\mathcal{O}(N^2)$ links in a fully connected graph, since adding more nodes only requires the total traffic to increase linearly in order to maintain the network's resistance to traffic analysis.

7.3 The anonymity of expander topologies

In analysing the anonymity provided by networks with restricted routes we will limit ourselves to considering traffic analysis resistance since it depends heavily on the topology, while traffic confirmation attacks depend on the particular mix batching and flushing strategy individual nodes use (see section 2.3.3 for a definition of these attacks). Having defined in this section a way of quantifying the anonymity provided by the network, we will study the route length necessary to achieve maximal anonymity in expander graph based mix networks and the volumes of traffic necessary to avoid traffic analysis attacks.

In a fully connected mix network it is intuitive that a message that comes out of a mix node, after a number of hops, could have originated from any node in the network with a probability proportional to the input load of the mix. Since users chose their initial nodes at random, or taking into account reliability statistics [Moc], we can say that the probability the messages originated from an initial node is equal to the probability a client has chosen this node as an entry point to the network. The same probability distribution is often used to determine the intermediate and final nodes of the anonymous path. This observation allows us to compute the traffic analysis resistance of the network $\mathcal{A}_{\text{network}}$ for fully connected networks, using the probability distribution describing the selection of the entry node.

For a graph that is not fully connected, we need to calculate the probability that a message present in a node after a number of mixing steps has originated from a particular initial node. This requires us to trace the message backwards in the network. If the graph is not directed, the likelihood a message was injected at a particular node is equal to the probability a random walk starting at the final node finishes on a that node after a certain number of hops.

Therefore, we consider the network as a graph and the act of selecting a path through it as a random walk, and we model the route selection procedure as a Markov process. In practise, some anonymous route selection algorithms exclude nodes from being present on the path of a message twice, which violates the memoryless property of the Markov process. Despite this a Markov process is still a good approximation to the route selection process.

After an infinite number of steps the probability a message is present on a particular node should be equal to the stationary probability distribution π of the Markov process. Therefore the maximum anonymity provided by the network will be equal to its entropy, $\mathcal{A}_{\text{network}} = \mathcal{E}[\pi]$

For reasons of efficiency we need to find how quickly the probability distribution $q^{(t)}$ describing where a message is after a number of random steps t, converges to the stationary probability π of the Markov process. A smaller t would allow us to minimise the number of hops messages need to be relayed, therefore reducing the latency and increasing the reliability of the network.

Motwani and Raghavan [MR95] provide a theoretical bound on how quickly a random walk on a graph converges to the stationary distribution. If π_i is the stationary distribution of a random walk on a graph G and $q^{(t)}$ the probability distribution after t number of steps starting from any initial distribution $q^{(0)}$. We define $\Delta(t)$ as the relative point-wise distance as follows:

$$\Delta(t) = \max_i \frac{|q_i^{(t)} - \pi_i|}{\pi_i} \tag{7.1}$$

They go on to show that this distance after a number of random steps t is bounded by the number of nodes in the network N and the second eigenvalue λ_2 of the transition matrix corresponding to the graph of the network:

$$\Delta(t) \leq \frac{\sqrt{N}(\lambda_2)^t}{\min_i \pi_i} \tag{7.2}$$

$$\Rightarrow \log \Delta(t) + \log \min_i \pi_i - \frac{1}{2} \log N \leq t \log \lambda_2 \tag{7.3}$$

$$\Rightarrow t \leq \frac{\log \Delta(t) + \log \min_i \pi_i - \frac{1}{2} \log N}{\log \lambda_2} \sim \ \mathcal{O}(\log N) \tag{7.4}$$

The distance decreases exponentially as the number of steps t, for which the message travels through the networks, increases linearly. The quick rate of convergence of the probability distribution is also dependent on the second eigenvalue being small. An extensive body of research has concentrated on linking the value of the second eigenvalue to expansion properties of graphs, to show that good expanders exhibit small second eigenvalues (see [MR95] for details). There is a fundamental limit of how quickly a network can mix that depends on the degree D of the graph:

$$1 > \lambda_2 \geq \frac{2\sqrt{D-1}}{D} \tag{7.5}$$

The results above assure us that a mix network with a topology corresponding to a good expander graph would mix well, in a number of steps logarithmic in its size, $\mathcal{O}(\log N)$. This means that in this small number of steps a message will leave the network at a node selected with probability approaching the probability after an infinite number of steps, namely the stationary probability distribution π.

In fact the methods described above can be used to calculate the theoretical probability that message that comes out at a mode n_e of the network has been injected at another node n_i. In theory the a priori knowledge of the attacker, concerning where a message was injected, corresponds to the probability distributions after the random walk on the graph representing the network. It also depends on the way that initial nodes are being chosen by clients, using reliability statistics, or other information. As the number of intermediaries grows, this distribution converges towards being the same for all initial choices of entry nodes. Therefore as the number of hops grows, a network based on expander graphs offers *uniform anonymity*, which means that the anonymity provided by the network is the same regardless of the node used to inject a message.

In the next section we will study how much traffic is needed to make the network resistant to traffic analysis, in other words an actual observation of it running will not give the attacker much additional information beyond the theoretical calculations presented above.

7.3.1 Protection against intersection attacks

An advantage of mix cascades, as argued in [BPS00], is that they are not susceptible to intersection attacks. Such attacks use the fact that many messages are sent using the same path to perform traffic analysis and follow the messages through the network. The authors note that, if every time a message is sent by the user under surveillance, the set of possible destinations of every mix is intersected with the set of possible destinations of previous messages, then the actual path of the message will become apparent. This is due to the very small probability the same, even partial, route is used by different messages. Since in mix cascades all messages use the same sequence of intermediary nodes, such an attack does not yield any results. Of course traffic confirmation is always possible, by observing all the edges of the network, and finding correlations between initial senders and final recipients. Such attacks will always be possible if the network does not provide full unobservability [PK00], or other specific countermeasures.

In a mix network with restricted routes and small degree, such as one based on expander graphs described in the previous section, the potential for intersection attacks, can be greatly reduced. This is done by making sure that all the links from a node to its neighbours are used in a flushing cycle. This is possible in practice since the number of these links in the network is relatively small, and does not grow proportionally to $\mathcal{O}(N^2)$ as for fully connected networks. Making sure that all links are used is sufficient to foil the simplest intersection attacks that use the intersection of sets of potential senders to trace messages [KAP02]. Traffic analysis is still possible if the probability a message has used a link is skewed. Therefore we need enough genuine traffic to be mixed together for the observable load on the network links to be proportional to the theoretical probability distribution described by the transition matrix representing the topology graph.

Using a threshold mix as an example we will calculate how much traffic is needed for no link from a node to be left empty. We assume that clients select the next hop from a particular node i using a probability distribution p_n, where n is the number of N_i neighbouring nodes. Then the probability that the link to a node is left empty in a batch of b messages is:

$$\Pr[\exists i.N_i \text{ empty}] < \Pr[N_1 \text{ empty}] + \ldots + \Pr[N_n \text{ empty}] \tag{7.6}$$

$$\Pr[\exists i.N_i \text{ empty}] < \sum_{\forall N_i} (1 - p_i)^b \tag{7.7}$$

As the size of the batch of messages to be processed grows, the probability that a link is empty decreases exponentially, making simple intersection attacks infeasible. It is important to note that the same effect can be achieved by adding dummy traffic on the links that are not used. Again the amount of dummy traffic in the network will only grow linearly with the number of nodes in the network.

In order to avoid the attacker gaining more information than the theoretical anonymity, which is the entropy of the stationary probability distribution on the nodes $\mathcal{E}[\pi_i]$, the actual volume of messages on the links should be proportional to the matrix describing the network topology. As described above, each node receives a number of messages b, some of which will be output on a particular link i according to a binomial distribution, with probability p_i. We can require the number of messages that are actually transmitted not to diverge on a particular round or time period by more than a small percentage f from the average mean. We approximate the binomial distribution describing the volume of traffic on a link i, by the normal distribution with mean $\mu = bp_i$ and variance $\sigma^2 = bp_i(1 - p_i)$. We then know that the normal distribution will not diverge from its mean by more than three standard deviations 99% of the time. We can require this deviation to be less than a fraction of the mean.

$$(1 - f)\mu = \mu - 3\sqrt{\sigma^2} \tag{7.8}$$

$$\Rightarrow (1 - f)bp_i = bp_i - 3\sqrt{bp_i(1 - p_i)} \tag{7.9}$$

$$\Rightarrow f\sqrt{b}p_i - 3\sqrt{p_i(1 - p_i)} = 0 \tag{7.10}$$

$$\Rightarrow b = \frac{9}{f^2}\left(\frac{1 - p_i}{p_i}\right) \tag{7.11}$$

This result can then be used in conjunction with $p_{\min}$, the probability associated with the link that is least likely to be used in the network or mix, to derive how much genuine traffic would be necessary in a node to protect against traffic analysis.

Another way of calculating the appropriate threshold value for a threshold mix would be to calculate the number of rounds necessary to perform an intersection attack against a stream of k messages. The techniques used do this are related to the statistical disclosures attacks described in chapter 9.

The attacker performs a hypothesis test on each of the links, with H_0 representing the hypothesis that the stream of messages under surveillance are output on the link under observation, and H_1 representing the hypothesis the messages are output on another link. In case H_0 is true the volume of messages on the observed link follows a probability distribution $Y_0 = k + X_{b-1}$ otherwise it follows a probability distribution $Y_1 = X_{b-1}$, where b is the threshold of the mix, k the number of mixing rounds, and p_i the probability the link is used by messages. X_{b-1} is the random variable following the binomial distribution with probability p_i after $b-1$ trials. The mean μ and standard deviation σ^2 of these distributions are:

$$\mu_{Y_0} = k + k(b-1)p_i \qquad \sigma^2_{Y_0} = k(b-1)p_i(1-p_i) \tag{7.12}$$

$$\mu_{Y_1} = k(b-1)p_i \qquad \sigma^2_{Y_1} = k(b-1)p_i(1-p_i) \tag{7.13}$$

In order to be able to accept or reject hypothesis H_0 we will require the observed volume of traffic to be within a distance of a few standard deviations σ_{Y_0} from the mean μ_{Y_0}, while also at a minimum distance of a few standard deviations σ_{Y_1} from μ_{Y_1} to avoid false positives. The number of standard deviations l depends on the degree of confidence required. The minimum number of mixing rounds k that need to be observed by an attacker to confirm of reject the hypothesis can therefore be calculated by:

$$\mu_{Y_0} - l\sigma_{Y_0} > \mu_{Y_1} + l\sigma_{Y_1} \tag{7.14}$$

$$k > 4l^2 \frac{p_i}{1-p_i}(b-1) \tag{7.15}$$

For values of $l = 1$ we get a confidence of 68%, for $l = 2$, 95% and for $l = 3$, 99%. The above formula is true both for general mix networks and for mix networks with restricted routes. We can require the value of rounds k to be greater than one $k > 1$, with $l = 0.6745$ for the attacker to have only a confidence of 50% in order to frustrate traffic analysis of messages that are not part of a stream that follows the same route.

7.3.2 Subverted nodes

An important factor that has to be taken into account when judging an anonymous network, is how robust it is to subverted nodes. In particular one has to assess the likelihood that all the nodes that have been selected to be on the path of a message are subverted nodes. For the topology presented this amounts to determining the probability $p_{l/c}$ that l nodes selected by a random walk on the expander graph, might include $c \leq l$ subverted nodes. Gillman provides an upper bound for this probability [Gil93], that is dependent on the expansion properties of the graph, and the "probability mass" of the subverted nodes.

If the matrix representing the graph of the mix network has a second eigenvalue λ_2 then define $\epsilon = 1 - \lambda_2$. Assume that the set C of nodes is subverted. Then define π_c as the probability mass represented by this corrupt set, $\pi_c = \sum_{i \in C} \pi_i$ where π is the stationary probability distribution of the random walk on the graph. After a number of steps l the probability that a walk has only been performed on subverted nodes is:

$$\Pr[t_c = l] \leq \left(1 + \frac{(1 - \pi_c)\epsilon}{10}\right) e^{-l \frac{(1-\pi_c)^2 \epsilon}{20}} \tag{7.16}$$

The probability that a path is totally controlled by subverted nodes therefore depends on the amount of traffic processed by the subverted nodes, and the mixing properties of the graph, but decreases exponentially as the route length increases. Assuming a particular threat model, the route length can be increased until that threat is very improbable. In practice the constant factors of the bound are too large to lead to values of the route length that are practical in real systems. Therefore, despite the initially encouraging results, for even modest π_c other methods might have to be used to determine and minimise the probability that the full route is composed of subverted nodes.

7.4 Comparing topologies

We have studied in the previous sections some properties of sparse mix networks, namely the route length and the batch size or volume of traffic necessary to provide nearly maximal anonymity. Next we shall compare these properties with previously introduced topologies.

Sparse networks based on expander graphs scale well by providing maximal network anonymity for a route length l proportional to $\mathcal{O}(\log N)$. Furthermore, they can be made resistant to traffic analysis and intersection attacks using a constant volume of traffic per node, depending on the degree

D of the network. Using (7.15) we observe that if the route selection algorithm is uniform, then the batch size b of nodes can be $b < \frac{1}{4l^2}k(D-1)+1$ which is independent of the number of nodes in the network.

7.4.1 Mix cascades

Given our definitions, it is clear that a mix cascade is resistant to traffic analysis, since observing the network traffic does not provide an attacker with more information than she originally had about the correspondence of input and output nodes. This is the case because there is no uncertainty about the node where all messages were inserted, since there is only one. The fact that $\mathcal{A}_{\text{network}} = 0$ does not mean that the network does not provide any anonymity to messages, but simply that all the anonymity provided to the messages originates conceptually form the single $\mathcal{E}[\Pr[m_e \text{ is } m_{ij} | m_e \text{ inserted at } n_x]]$ component of (2.22).

This absolute protection against traffic analysis comes at a very high cost. The anonymity provided is reduced to the volume of messages that can be processed by the node with least throughput in the cascade. The latency of the messages is also large, since each message has to be processed by all nodes in the cascade.

Despite the inefficiencies presented above, mix cascades are a valuable design. They are resistant to very powerful adversaries that control all nodes but one. They also highlight the advantages of implementing topologies that can be analysed, in order to understand their anonymity properties.

7.4.2 Mix networks

General mix networks are distinct from sparse, constant-degree, mix networks because anonymous messages are allowed to follow arbitrary routes through them. This sometime is misinterpreted as meaning that the traffic matrix corresponding to the mix network is fully connected. Indeed an attacker who has no additional knowledge of the network, beyond the way routes are selected, has no other way of attributing probabilities linking output messages to input nodes other than by using a random walk on this fully connected graph, for a number of steps corresponding to the route length.

On the other hand, an attacker that can observe the traffic in the network can get much better results. If we assume that the number of nodes is larger than the threshold of the mixes then some links remain unused in each mix round. Furthermore, even if the threshold is comparable to the number of mixes, the volume of messages sent will give the attacker a different probability distribution from the theoretical one described by the route selection distribution. Therefore an attacker can use additional information, extracted from these observations to trace messages more effectively (for an example see [Ser04]).

We will denote the graph used for the route selection through the network as G. This graph has N nodes, the number of mixes, that are all connected with each other by weighted edges. The weights correspond to the probability that a node is selected as the next mix in the path, and can be uniform if the selection is performed at random, or it can be based on reliability statistics or reputation metrics. Given a column vector v describing where a message was injected, the probability P a message comes out of the network at a particular node after l steps, can be calculated to be $P_l = G^l v$ This is the a priori information that an attacker has about the correspondence between input and output nodes, even before any traffic analysis has been performed.

As the attacker observes the network, for round i it can deduce a matrix G_i with the mixes as the vertexes, and the traffic load that was observed between them during round i as the weights on the edges. It is worth observing that G_i is closely related to G in the sense that the selection of routes for any round is performed using G, but is sparse if the threshold of the mixes is lower than the number of nodes. In fact, the weights on the edges follow the same probability distribution as for G, but are going to be different, subject to the variance of the multinomial distribution and the threshold of the mixes. An adversary that observes the actual traffic patterns in the networks will therefore be able to have more accurate information about where the messages injected are going, by calculating the probability distribution $P'_l = G_l \dots G_2 G_1 v$.

The relation of G_i the graph of the traffic observed at round i with the graph G used to route messages, is crucial in understanding the anonymity that generic mix networks provide. The smaller the difference between G_i and G the more resistant the network will be to traffic analysis. In order for G_i to be close to G there needs to be enough genuine or cover traffic to make the mean load on all the links proportional to the probabilities of the route selection, as described for sparse topologies in section 4.2. In general one would expect $\lim_{l\to\infty} \mathcal{E}[P'_l] = \lim_{l\to\infty} \mathcal{E}[P_l]$, but also $\forall l, \mathcal{E}[P_l] \leq \mathcal{E}[P'_l]$, where $\mathcal{E}[\cdot]$ denotes the entropy of a distribution.

For G_i to be a good approximation of G it is necessary each round to fill all links with traffic volumes proportional to the values on the edges of G. This requires the volumes of traffic handled by the network to be proportional to $\mathcal{O}(N^2)$ as the number of nodes N in the network grows. The batch size that needs to be handled by each node therefore grows proportionally in the size of the network $b < \frac{k}{4l^2}(N-1)+1$, as described by (7.15). The increased batch size also has repercussions on the latency of messages that travel through the network, since mixes will wait for more messages before they operate.

Mix networks that do not use a deterministic threshold mixing strategy, where the first batch of messages to go in are also the first batch of messages to go out, can also be analysed in a similar fashion by redefining G_i. It would then need to represent the probability distributions leading to the effective anonymity sets of messages instead of the volumes of traffic in the network.

7.5 An example network

In order to illustrate how all the results presented on mix networks based on expander graphs fit together, we will present an example network and analyse it. We will proceed to calculate the route length necessary for it to provide uniform anonymity, the amount of real traffic that should be present in each node for it to be resistant to traffic analysis and intersection attacks.

7.5.1 Selecting a good topology

We aim to create a network with $N = 400$ mix nodes, each with $D = 40$ neighbours. The neighbour of a mix both sends and receives messages from the mix and therefore we can represent this network as an undirected graph. Furthermore, we will assume that senders will choose their path across the network using a random walk on the graph, with equal weights on all the edges. Therefore the probability that a messages follows a particular link, given that it has already reached a node is equal to $p_n = p_{\min} = \frac{1}{40}$. We expect such a graph to have $N_l = 16 \cdot 10^3$ links instead of $N^2 = 16 \cdot 10^4$ that a fully connected graph would have. Therefore it is sparse in the sense that only one in ten links are used.

Using a brute force algorithm, we created a number of networks and compute their second eigenvalue until a network with good expansion properties is found. After testing less than ten candidates we find a graph with a second eigenvalue $\lambda_2 = 0.3171$, which is close to the theoretical limit of $\lambda_2 > 0.3122$ given by equation (7.5).

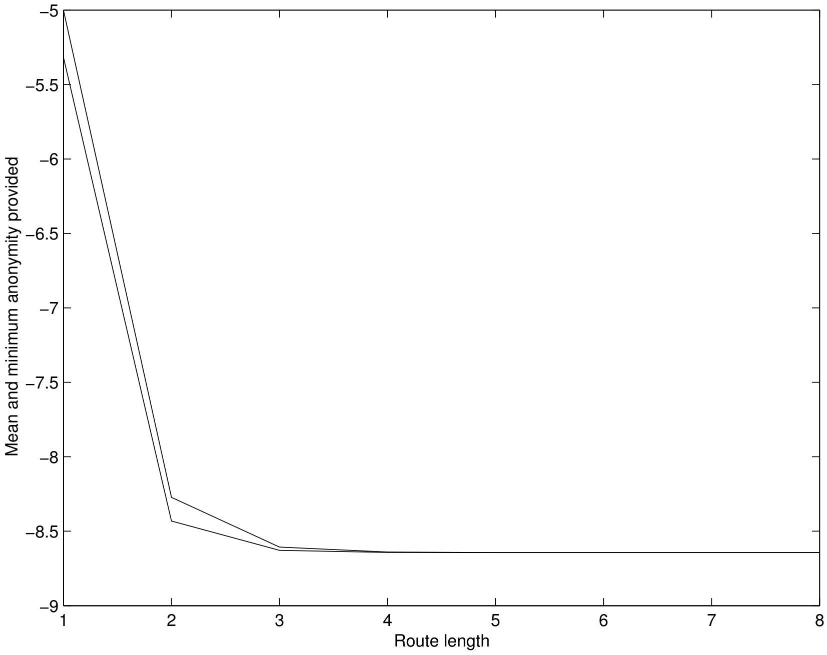
7.5.2 Mixing speed

Using the formula (7.2) we know that the network will provide nearly uniform anonymity after a number of mixing steps proportional to $\log N$. From the graph we know that the $\min_i \pi_i = \frac{1}{400}$ since the stationary distribution is uniform, and therefore the resistance to traffic analysis should be equal to $\mathcal{A} = -\log_2 N = -8.6438$.

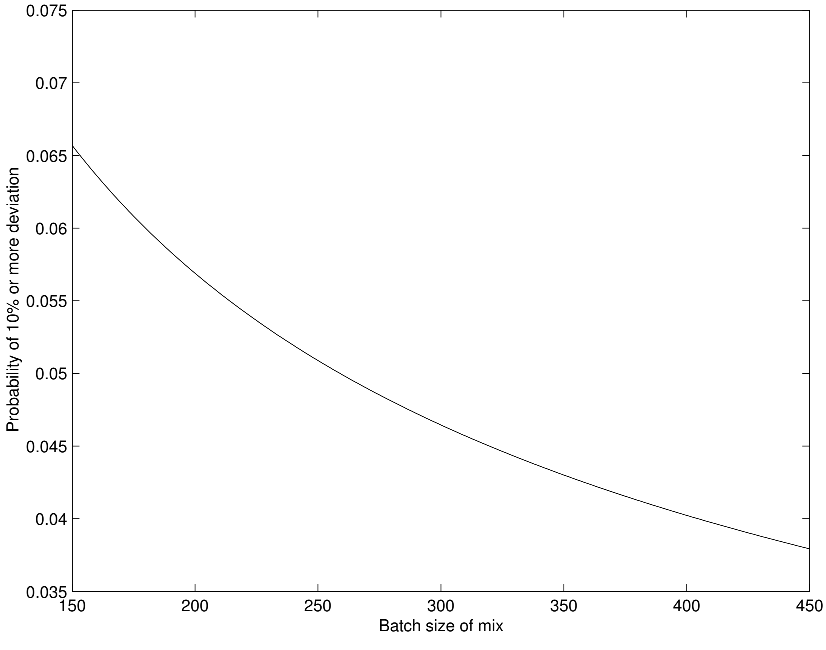
In theory the relative point-wise distance $\Delta(t)$ between the observed $q^{(t)}$ distribution after t steps and the stationary distribution π_i should converge following $\Delta(t) \leq n\sqrt{n}\lambda_2^t$. This allows us to calculate that the safe route length is around six. In practise much tighter bounds can be computed by directly calculating using G^t the distributions $q^{(t)}$ from the available graph G. It is therefore observed that after four steps of the random walk the entropy of $q^{(t)}$ is equal to the theoretical entropy calculated above. Figure 7.1(a) illustrates this by showing how the mean entropy provided to messages entering on any node compares with the minimum entropy that is offered by the network. Their convergence indicates that after four steps the network provides uniform and also maximum anonymity.

7.5.3 Resisting intersection and traffic analysis attacks

In order to avoid the simplest forms of intersection attack, all the network links need to be used for every round. The probability a network link is not used is described by equation (7.7). For this particular network all $p_i = \frac{1}{40}$ where p_i is the probability a link is followed. The probability that any link is left empty for threshold mix with batch size $b = 300$ is $\Pr[\exists i. N_i \text{ empty}] < 2\%$. Therefore for batches larger than 300 messages the probability a link is left empty is very small.



(a) Mean and lowest entropy after a random walk



(b) Probability the distribution deviates more than 10%

Figure 7.1: Characteristics of the example topology

In order to protect against more sophisticated traffic analysis attacks taking into account statistical differences in the observed distributions from the graph G, we need to calculate the probability this deviation is large (for example larger than 10% as shown in figure b). In practice with a batch size of $b = 300$ as specified above, the attacker would need to observe $k > 4\frac{1}{40-1}(300 - 1) = 30$ messages in a stream in order to have a confidence of 68% that a particular link was used.

7.6 Summary

We have presented different topologies that mix networks can assume, and analysed the traffic analysis resistance that each offers. We have calculated the number of hops necessary to provide adequate mixing, and the volume of traffic necessary to make the network invulnerable to traffic analysis.

It turns out that sparse networks based on expander graphs, can have low degree and still mix traffic quite quickly. A further advantage is that they require less genuine traffic to be present in the network than fully connected graphs, while scaling much better than mix cascades.

Chapter 8

Red-green-black mixes

> "*One FBI internal memorandum describing steps to be taken against the Black Panther Party, included a plan to release false police films 'indicating electronic surveillance where none exists'. Another spoke of the need 'to get the point across that there is an FBI agent behind every mail box'.*"
>
> *The technology of political control* — C. Ackroyd, K. Margolis, J. Rosenhead, T. Shallice

In this chapter we describe a technique that mix nodes can employ to detect whether they are under an active attack. The most powerful active attack is the $(n-1)$ attack, performed by flooding a node with attacker messages alongside a single target message to be traced. The attacker can recognise her messages as they emerge and therefore link the sender with the receiver of the target message. This attack is active in the sense that it critically depends on the adversary's ability to inject fake messages in order to flood the honest node and delete or delay other genuine messages except the one under surveillance. Clearly it also requires the ability to observe arbitrary network links.

The ability to detect and prevent such attacks relies upon individual mixes being aware of their network environment and their state of connectivity with it by sending anonymous messages through the network back to themselves. We call these *heartbeat* messages, or "red" traffic. When a mix is under attack it cannot directly detect how much of the traffic it receives is genuine "black" traffic and how much is the attackers' flooding traffic. Therefore the mix tries to estimate the amount of flooding traffic from the rate of heartbeat messages and injects dummy "green" traffic in order to artificially increase the anonymity provided to honest messages.

The key intuition for understanding the properties of rgb-mixes is that the different colours of the traffic can only be observed by the mix itself. To all other mixes or attackers, traffic exiting the mix looks the same. In order to perform the $(n-1)$ attack, an attacker would need to delete or delay selected messages while simultaneously allowing the mix to receive heartbeat messages. While an attacker flooding the mix will be able to distinguish her black messages from other messages exiting the mix, the attacker is prevented from filtering out genuine traffic from heartbeat messages. Thus, the number of heartbeat messages received can be used by the mix to estimate the number of honest messages present in the mix's input.

8.1 Related work

Active attacks, and in particular the $(n-1)$ attack, were known in different communities working on anonymous communications [GT96] for a long time. In their survey of mixing strategies Serjantov *et al.* [SDS02] assess the effectiveness of different mixing strategies against a number of active attacks. They calculate the number of rounds that it would take an attacker to be successful, and find that some mix strategies are more expensive to attack than others. On the other hand no mixing strategy provides an absolute defence since they can all be attacked in a finite amount of time or rounds. The $(n-1)$ attack (applicable primarily to threshold mixes) is generalised for other mixing strategies and called a blending attack. It is a simultaneous trickle attack, namely stopping genuine messages, and a flooding attack, that fills the mix with the attacker's messages.

In designing sg-mixes to resist $(n-1)$ attacks Kesdogan *et al.* [KEB98] followed a different approach. They observe that the ability to realistically perform the $(n-1)$ attack relies on delaying rather than deleting messages. Therefore if messages follow a tight schedule in the network, and are dropped if they are late, an attacker would have to destroy traffic and ultimately the network would become aware of the attack. Furthermore, only a fraction of the traffic could be attacked at any time. In order to provide real-time guarantees, they use a continuous mixing strategy based on delaying messages according to an exponential distribution. Messages contain timestamps and are delayed for as long as requested by the original sender. If a message misses its deadlines it is dismissed.

Mixmaster [MCPS03], the only widely deployed mix network, uses dummy traffic to counter $(n-1)$ attacks. A random number of dummy messages are included in the message pool every time an message arrives from the network. This is an effective, but quite expensive, strategy since dummy messages are sent even during normal operation.

Other mix designs, such as Mixminion [DDM03a], use link encryption that makes it difficult for an attacker to recognise even her own messages in the network. This can be effective, particularly if it is combined with each mix peering only with a small set of others. However it cannot provide an absolute protection against flooding since the attacker controls the path through which her messages are routed. Designs that disallow or restrict source routing could be one way to defend mix networks from flooding attacks.

8.2 Design principles, assumptions and constraints

Using the analysis of Serjantov *et al.* one can calculate how much time, or how many messages, should be injected into a mix until an adversary can trace a message. While this can make an attack expensive, and will delay the overall functioning of the network, it does not guarantee that an attack will not succeed. On the other hand, we will aim to completely eliminate the potential for $(n-1)$ or blending attacks.

Kesdogan *et al.* guarantee that most messages delayed will be dropped, but do not guarantee that single messages will not be traced. Again it would be easy to notice that such an attack is taking place (since messages are dropped) but no algorithmic way of doing this is included in the mix strategy.

Therefore one of our aims will be to specify a way for the mix to detect such an attack, and a strategy to counter it.

In designing rgb-mixes to resist active attacks, we will assume that the mixes have some knowledge of their environment, in particular the addresses, keys and capabilities of the other mixes in the network. This assumption is not unrealistic since clients require this information to send or receive anonymous messages, and directory server infrastructures are deployed to provide them [DDM03a]. We also require the rgb-mix to be included in the list of active mixes in the directory listing and clients or other mixes to use it to relay traffic.

Furthermore we will assume that the network, through which the “red”, “green” or “black” messages travel, provides some anonymity against the attacker. In most mix networks this means that the network is either not fully under the control of the adversary, or that a large fraction of the mix nodes are honest. The key requirement is for the network to make indistinguishable to the attacker the *colour* of the traffic, which could be, as we will see, red, green or black.

While recognising that introducing dummy traffic into the network increases its cost, we do so for two purposes: first as signalling, to assess the state of connectivity with the rest of the network in the form of red traffic; and secondly in order to increase the anonymity sets while the mix is under attack, in the form of green traffic. It is a requirement that the amount of green traffic should be minimal (or even zero) if the mix is not under attack. On the other hand it increases when the mix is under attack in order to reduce latency, or to bootstrap the functioning of a network of rgb-mixes.

8.3 Red-green-black mixes

An rgb-mix receives a certain number of *black* messages per round. These are genuine messages to be anonymized, or could be the product of a flooding attack mounted against the mix. The mix needs to estimate how many of these black messages are genuine in order to guarantee some quality of anonymity. Unfortunately, because of the nested encryption, and the absence of identifying information in the packets, the mix cannot do this by simple inspection.

In order to get an estimate of the number of genuine messages, a mix uses the same property that makes it unable to distinguish genuine from flooding traffic: namely that mixed traffic is not separable by a third party. With each output batch it includes a fraction of *red* messages, which are indistinguishable from other anonymous messages but are anonymously addressed back to itself. These messages are mixed with the outputs of the mix and are impossible to distinguish from other genuine black messages (notice that an attacker can distinguish them from flooding traffic). After a certain number of rounds we expect the same fraction of red messages to come back to the mix. These messages can be distinguished by the mix since they were created by itself. This should be done in order to calculate their fraction in comparison with the black traffic received.

If the fraction of red messages received in a round is smaller than expected, subject to statistical fluctuations, this could mean one of three things. Firstly, the mix could be under a blending attack, meaning that the genuine traffic is being blocked and only the attacker's messages are let through. Since the attacker cannot distinguish red messages from the genuine traffic it cannot selectively allow some of them through. Therefore it has to block them, and the fraction of red messages will drop depending on the severity of the attack. A second reason why the fraction of red messages could be small or zero is the fact that the mix has only recently started its operation and the red messages sent did not have enough time to loop back. A third is that traffic load is changing.

When the fraction of red messages drops, a possible strategy would be to stop the operation of the mix until enough red messages are received, or forever if the attack persists. Unfortunately this transforms the blending attack to a denial-of-service attack on the mix. Furthermore if all the mixes implement this strategy it would be very difficult for the network to start its operation: all the nodes would block since their respective heartbeat red messages would not have yet arrived. This creates deadlock.

Instead dummy messages are introduced in order to guarantee the quality of the anonymity provided by the mix. A certain number of *green* messages are generated when necessary and injected in the output of the mix. These

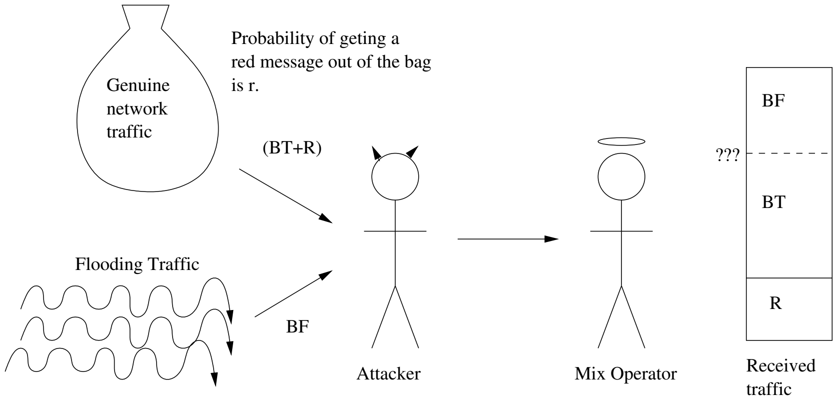


Figure 8.1: Model of the attacker and rgb-mix

messages are multiple hop dummy messages that will be dismissed at the end of their journeys. Since an adversary is not able to distinguish them from other genuine black or red traffic these messages increase the anonymity set of the genuine messages trickled through by the attacker.

The objective we have set for the functioning of the mix is to reduce the amount of dummy traffic during normal operation, namely when the mix is not under flooding attack. The key to achieve this is to estimate the number of genuine black messages in the input, and only include green traffic if this is below a threshold.

8.4 The security of rgb-mixes

The key to understanding the security of rgb-mixes is the realisation that an attacker is not able to distinguish between red, green and black messages. Allowing through the red messages but not any genuine black message is difficult, so can only be done at random. On the other hand the rgb-mix cannot distinguish the genuine black messages from the attacker's flooding messages, but can estimate their numbers using the calculated frequency of the red messages received during a mix round or time interval.

A number of messages $R + B$ is received by the mix during a period or round, with R being the number of red messages and B the number of black messages received. Out of the black messages some might be genuine traffic BT but some might be flooding traffic BF, with $B = BT + BF$. The probability of the adversary choosing a red message along with any genuine traffic chosen is equal to the fraction r of red messages included in the output of the mix. This assumes that the overall genuine traffic volumes do not change significantly.

An attacker will try to substitute genuine black traffic with flooding traffic that she can identify, thereby reducing the anonymity of the remaining message(s). If the substitution is done naïvely then no red messages will be received by the mix, which will use green cover traffic to maintain the sizes of the anonymity sets. Therefore an attacker will try to allow through some red messages. Since the attacker is "colour blind", she can only choose messages at random, according to the fraction injected in the network, until a certain number of red messages are present in the input batch.

The rgb-mix needs to answer the following question: Given that R red messages are received, how many genuine traffic messages BT are likely the have been allowed through? The number of genuine messages that an attacker needs to choose for R red messages are present, if for each message the probability of being red is a fraction r, can be described by a negative binomial distribution.

$$\Pr[BT = x] = \binom{R + x - 1}{R - 1} r^{R-1}(1 - r)^{x} \tag{8.1}$$

We can also calculate for a number R of red messages the expected number E of genuine black messages, and its variance V. Detailed derivation of these can be found in [Bha72].

$$E[BT] = \frac{R(1 - r)}{r} \tag{8.2}$$

$$V[BT] = \frac{R(1 - r)}{r^2} \tag{8.3}$$

The calculation above takes into account that the attacker is able to observe the event where the mix receives a certain number of red messages. While an adversary is not able to tell that the message just input into the mix is red, she could still be able to observe some side effect, such as the mixing of a batch. This provides the mix designer with the flexibility to implement mixing strategies conditional upon the number of heartbeat messages received.

Let $(R + B)$ be the number of messages received in a batch and r the fraction of red messages sent by batch. Given that $(R + B)r$ red messages are expected during each round, this would provide a standard anonymity set size of on average $\frac{((R+B)r(1-r)}{r} \approx B$. This number should be made large enough to provide adequate anonymity set sizes for the messages. If the number of red messages received is smaller, then a number of green messages G' needs to be generated and output by the mix to make up for the potential loss of anonymity, where:

$$G' = \underbrace{\frac{((R+B)r)(1-r)}{r}}_{\substack{\text{Expected genuine black}\\ \text{traffic given total volume}\\ \text{received}}} - \underbrace{\frac{R(1-r)}{r}}_{\substack{\text{Expected genuine black}\\ \text{traffic given number of red}\\ \text{received}}} = \underbrace{\frac{((R+B)r-R)(1-r)}{r}}_{\substack{\text{Difference is the number of green}\\ \text{dummies to be injected to}\\ \text{compensate}}} \tag{8.4}$$

$$R' = (R + B)r \tag{8.5}$$

Therefore if the mix is functioning properly and is not under flooding attack, it only outputs a minimal number of green, cover traffic, messages. When it is under attack it maintains the anonymity provided by outputting greater amounts of green cover traffic.

If the attacker cannot observe the number of red messages in the stream reaching a threshold (such as a mixing batch being processed), a slightly different model can be used to estimate the number of genuine traffic messages BT. The probability a certain number of messages BT are present in the batch, given that there is a number of red messages R and the probability a message addressed to the mix is red is r can be described as follows.

$$\Pr[R|N, r] = \binom{N}{R}(1-r)^{(N-R)} \tag{8.6}$$

$$\Rightarrow \Pr[N|R, r] = \frac{\binom{N}{R}r^R(1-r)^{(N-R)}}{\sum_N \binom{N}{R}(1-r)^{(N-R)}} \quad \text{note that } N = BT + R \tag{8.7}$$

$$\Rightarrow \Pr[BT = x|R, r] = \frac{\binom{x+R}{R}(1-r)^x}{\sum_{0\le x\le B}\binom{x+R}{R}(1-r)^x} \tag{8.8}$$

A similar procedure to the first model can then be followed to estimate the deviation of the received genuine traffic from what would be expected if the number of red messages were indeed $(B + R)r$.

8.5 A cautionary note

The security of rgb-mixes is calculated for the average case, namely the expectations are taken into account to calculate the amount of green traffic to be injected. This expected value will only be attained when the batch sizes are large enough, in comparison with the probability r a message received is red. Furthermore, the analysis above is only accurate when the network traffic received by the attacker can be approximated by the red-black traffic "bag", as shown in figure 8.1. This means that the attacker taking a message from the network has a certain probability r of choosing a red message. In practise this is only an approximation since there is only a limited number of red messages, that will eventually run out if the experiment is repeated enough times. A more accurate model can be derived from the hypergeometric distribution.

Another critical assumption on which the models presented above are based is that the levels of genuine traffic do not change very much in time. Indeed there is no way a mix can tell the difference between an active attack and a genuine spike in traffic load. The traffic loads of the previous mixing rounds, or times, are therefore used to calculate the probability a red message is chosen by the attacker.

Another weak point of the method described above is that the attacker might try to influence r, the probability a message from the network is red. In order to avoid this the number of red messages injected in the network should be calculated based on a longer history of traffic load, not just the previous round of mixing. This way an attacker will have to attack for a very long time before getting any results.

The worst case presents itself when the mix does not receive any genuine traffic at all from the network through which the red messages are relayed. This means that the adversary will know which messages are red, and will be able to trivially perform flooding attacks, without being detected. The operational conditions under which this attack could be performed are a bit unusual. The mix under attack would have to not be included in the directory servers' lists, and therefore others not using it in order to relay traffic. Why would then the attacker try to attack it, since there is only minimal traffic on it? One reason could be that the attacker has lured a victim into sending a message through this particular mix. Again other methods of attack could be easier, such as forcing a victim to use a completely compromised node, instead of an "attackable" mix.

Finally it is worth noting that the green traffic offers some degree of protection against traffic analysis of the network, namely the traffic of a message node by node as it travels. It does not on the other hand offer any end-to-end protection against traffic confirmation. The green messages are simply discarded by mix nodes a few hops away, and modifying them to be sent to actual users is still a not very well-understood problem.

8.6 Summary

The most dangerous active attack against mixing is the $(n-1)$ attack, by which the adversary injects only one genuine message into a mix along with a flood of his own messages. We devise a strategy that allows mixes to detect that such an attack is taking place. They send heartbeat messages back to themselves, and use the rate at which these messages are received to estimate the amount of genuine traffic they receive.

In case an adversary is delaying traffic to perform the $(n-1)$ attack the rate at which the heartbeat messages are received is reduced, and the attack is detected. In this case special cover traffic messages are generated to maintain the quality of anonymity provided. This technique is very efficient, since it only uses large volumes of cover traffic when under attack.

Chapter 9

Statistical disclosure attacks

> *"[...] privately our intelligence officers were helping the US and Australia spy on East Timor people at a time where that intelligence was being funnelled through to the Indonesian military. "*
>
> *European Parliament ECHELON Committee*
> — Nicky Hager

A family of well-known attacks against mix systems are intersection attacks [BPS00]. These rely on the fact that a sequence of messages use the same route through the network, or are ultimately destined to the same receiver to perform traffic analysis. The set of potential receivers is computed for each message in the sequence. The intersection of these sets will eventually only contain the actual receiver of the stream.

Kesdogan *et al.* present an interesting variant of this attack in [KAP02] called the *disclosure attack*, where it is applied to a whole anonymity system. They assume that a particular user, Alice, sends messages only to a restricted set of recipients. They then note that it is possible by observing the recipient anonymity sets attributed to her messages to extract information about the ultimate recipients. The attack is generalised by viewing mix networks, or other anonymity systems, as abstract mixes, since the attack does not rely upon any particular properties of mixing other than the unlinkability it provides.

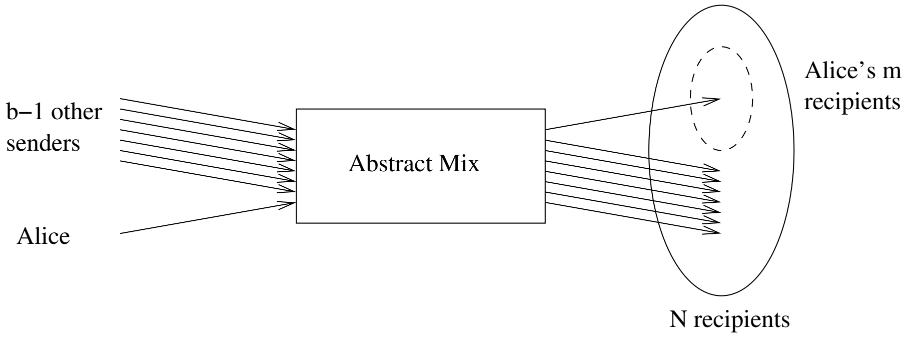


Figure 9.1: A single round of mixing in the abstract model

In this chapter we are going to briefly describe the disclosure attack as originally presented. A more efficient attack, the statistical disclosure attack, will then be presented. It requires less computational effort by the attacker and yields the same results. An analysis of the applicability and efficiency of the statistical disclosure attack, and a discussion of its relevance to other systems beyond the formal model is included.

9.1 The disclosure attack revisited

The formal model on which the disclosure attack is based is quite simple. A single mix is used by b participants each round, one of them always being Alice, while the other $(b-1)$ are chosen randomly out of a total number of $N-1$ possible participants. The threshold of the mix is b so it fires after each of the round's participants has contributed one message. Alice chooses the recipient of her message to be a random member of a fixed set of m recipients. Each of the other participants sends a message to a recipient chosen uniformly at random out of N potential recipients. We assume that the other senders and Alice choose the recipients of their messages independently from each other. Figure 9.1 illustrates a round of communication. The attacker observes $R_1, \ldots, R_t$ the recipient anonymity sets corresponding to t messages sent out by Alice during t different rounds of mixing. The attacker then tries to establish which out of all potential recipients, each of Alice's messages was sent to.

The original attack as proposed by Kesdogan *et al.* [KAP02] first tries to identify mutually disjoint sets of recipients from the sequence of recipient anonymity sets corresponding to Alice's messages. This operation is the main bottleneck for the attacker since it takes a time that is exponential in the number of messages to be analysed. The underlying method used by Kesdogan *et al* is equivalent to solving the Constraints Satisfaction Problem which is well known to be NP-complete. An analysis of the cost of performing the attack, and how quickly results are expected, can be found in [AKP03].

The second phase of the algorithm proposed intersects the disjoint sets found with anonymity sets of messages. When this intersection generates a set of only one element it is assumed that it is a correspondent of Alice.

9.2 The statistical disclosure attack

We wish to use the same model as above to show that a statistical attack is possible that yields the set of potential recipients of Alice's messages. In turn this set can be used to find the recipients of particular messages sent out by Alice.

We define as $\vec{v}$, the vector with N elements corresponding to each potential recipient of a messages in the system. We also set the values corresponding to the m recipients that might receive messages by Alice to $\frac{1}{m}$ and the others to zero, therefore requiring $|\vec{v}| = 1$. Observe that $\vec{v}$ is the probability distribution that is used by Alice to choose the recipient of her message for each round of the abstract mixing as described in the formal model above.

We also define $\vec{u}$ to be equal to the uniform distribution over all potential recipients N. Therefore all elements of $\vec{u}$ are set to be equal to $\frac{1}{N}$ with $|\vec{u}| = 1$. This vector represents the probability distribution used by all other senders to select their recipients' for each round of mixing.

The information provided to the attacker is a sequence of vectors $\vec{o_1}, \ldots, \vec{o_t}$ representing the recipient anonymity sets observed corresponding to the t messages sent by Alice. Each of $\vec{o_i}$ is the probability distribution assigning potential recipients to Alice's message during round i. The adversary will therefore try to use this information in order to infer $\vec{v}$ that, as described above, is closely linked to the set of recipients that Alice communicates with.

The principal observation underlying the statistical disclosure attack is that for a large enough set of observations t it holds true that (by using the Law of Large Numbers):

$$\bar{O} = \frac{\sum_{i=1}^{t} \vec{o_i}}{t} = \frac{\vec{v} + (b-1)\vec{u}}{b} \tag{9.1}$$

It is therefore possible, just from the knowledge of the observations $\vec{o_1}, \ldots, \vec{o_t}$, the batch size b of the mix and the model $\vec{u}$ of other senders to calculate $\vec{v}$, the set of recipients of Alice:

$$\vec{v} = b\frac{\sum_{i=1}^{t} \vec{o_i}}{t} - (b-1)\vec{u} \tag{9.2}$$

When the vector $\vec{v}$ is reconstructed by the adversary it can then be used to give an indication on the particular communications partners of Alice in a round k. The attacker simply multiplies each element of the $\vec{v}$ vector with each element of the observation $\vec{o_k}$ of round k, and normalises the resulting vector.

$$\vec{r_k} = \frac{\vec{v} \cdot \vec{o_k}}{|\vec{v} \cdot \vec{o_k}|} \tag{9.3}$$

The elements with highest probability out of $\vec{r_k}$ are the most likely recipients of Alice's message k.

The statistical disclosure attack therefore allows an attacker to identify all possible recipients m of Alice's messages and even further to establish the precise recipients of particular messages in the formal model, with an arbitrary degree of confidence that, as we will see, depends on the number of observations t.

9.2.1 Applicability and efficiency

The main drawback of the original disclosure attack was its reliance on solving an NP-complete problem. The statistical disclosure attack only relies on collecting observations and performing trivial operations on vectors, and therefore is computationally cheap and scales very well. Therefore we foresee the collection of observations, and the calculation of anonymity sets corresponding to messages, to be the main computational bottlenecks of an attacker.

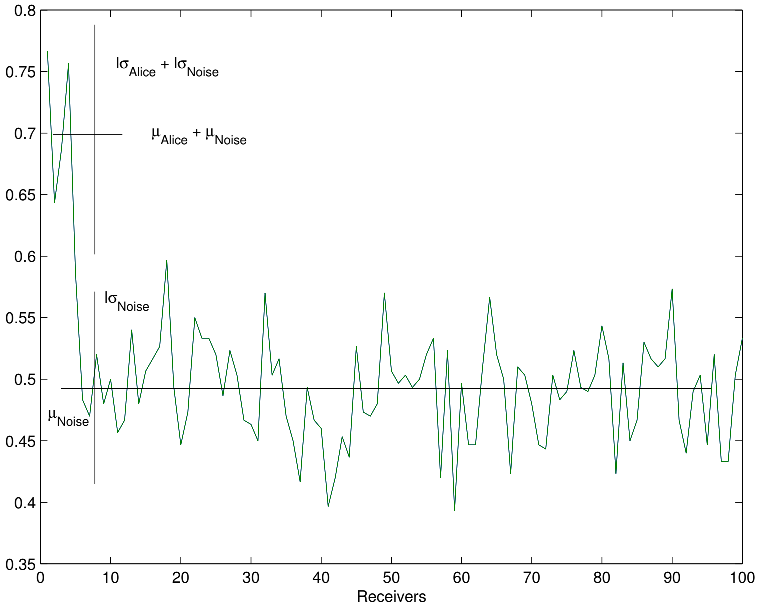


Figure 9.2: Means and variance of received messages for Alice's and other's recipients

It is important to establish the limits of the statistical disclosure attack and calculate the number of observations that are necessary in order to reliably perform it. We observe that extracting the vector $\vec{v}$ is a typical signal detection problem. The problem therefore is to differentiate the signal of Alice from the noise introduced by the other senders. In this case the signal strength of Alice is $\frac{1}{m}t$ versus the noise strength of the other senders that is equivalent to $\frac{b-1}{N}t$. Note that the signal, namely the volume of messages sent by Alice, is added to the noise produced by the other senders. For the signal to noise ratio to be larger than one we require:

$$\frac{\text{Alice's Signal}}{\text{Noise Strength}} = \frac{\frac{1}{m}t + \frac{1-b}{N}t}{\frac{1-b}{N}t} > 1 \Rightarrow \text{true} \qquad (9.4)$$

The above bound on m provides the necessary condition for a mix system following the formal model to be susceptible to the statistical disclosure attack. It is interesting that the disclosure attack, according to [KAP02] is only applicable when $m < \frac{N}{b-1}$.

It is important to calculate how many observations t are necessary to reliably retrieve $\vec{v}$. This depends on the variance of the signal $\vec{v}$ and the noise $(b-1)\vec{u}$.

The observations in $\bar{O}$ corresponding to Alice's recipients have a mean $\mu_{\text{Alice}} = \frac{1}{m}t$ and a corresponding variance of $\sigma^2_{\text{Alice}} = \frac{m-1}{m^2}t$ while the noise has a mean of $\mu_{\text{Noise}} = \frac{1}{N}(b-1)t$ and a variance of $\sigma^2_{\text{Noise}} = \frac{N-1}{N^2}(b-1)t$. We will need a number of observations t large enough for the mean of the signal to be larger than the sum of the standard deviations, multiplied by an appropriate factor to provide us with a satisfactory confidence interval.

$$\mu_{\text{Alice}} + \mu_{\text{Noise}} - l\sqrt{\sigma^2_{\text{Alice}} + \sigma^2_{\text{Noise}}} > \mu_{\text{Noise}} + l\sqrt{\sigma^2_{\text{Noise}}} \tag{9.5}$$

$$\Rightarrow \mu_{\text{Alice}} > l\sqrt{\sigma^2_{\text{Noise}}} + l\sqrt{\sigma^2_{\text{Alice}} + \sigma^2_{\text{Noise}}} \tag{9.6}$$

$$\Rightarrow \frac{1}{m}t > l\left(\sqrt{\frac{N-1}{N^2}(b-1)t} + \sqrt{\frac{N-1}{N^2}(b-1)t + \frac{m-1}{m^2}t}\right) \tag{9.7}$$

$$\Rightarrow t > \left[ml\left(\sqrt{\frac{N-1}{N^2}(b-1)} + \sqrt{\frac{N-1}{N^2}(b-1) + \frac{m-1}{m^2}}\right)\right]^2 \tag{9.8}$$

With $l = 2$ we have a 95% confidence of correct classification, when determining if a recipient is associated with Alice or not, while $l = 3$ increases the confidence to 99%.

9.3 Statistical attacks against a pool mix

So far the statistical disclosure attack has been applied to the exact model that is described in the previous disclosure attack research [KAP02]. As we will see, one of the main advantages of the statistical disclosure attack presented is that it can be generalised and used against other anonymous communication network models. In particular [KAP02] assumes that the anonymous network can be abstracted as a large threshold mix, where batches of messages (of size b) are anonymized together and sent out to their respective recipients. We will illustrate how the statistical disclosure attack can be generalised to anonymous communication mechanisms that can be modelled as pool mixes, or in other words where some messages are fed forward to the next mixing rounds.

We are going to modify the model introduced in [KAP02] to model a pool mix. It is worth noting that the threshold mix is a special example of a pool mix, with no messages feeding forward to the next mixing round. Figure 9.3 illustrates the model used for the attack.

The anonymous communication system that we are going to attack works in rounds from 1 to K. In each round k a number b of messages are put into the mix from the previous round. We call these messages the pool. A number B of messages are input from the senders of this particular round. Out of the $B + b$ messages input in the mix a random subset of size B is output, and sent to their respective receivers, during a round k. The remaining b messages stay in the pool for the next round.

One of the senders, Alice, is singled out to be the victim of the attack. Each time that she has to send a message she selects a recipient randomly out of a probability distribution described by the vector $\vec{v}$ over all possible N receivers in the system. Alice does not send in each round (as was the case in the model described in the previous section) but only sends at rounds described by the function $s(k)$. Depending on whether it is a round when Alice sends or not, $B - 1$ or B other senders respectively, send a message. They choose the recipient of their messages, each independently, according to a probability distribution described by the vector $\vec{u}$, over all possible recipients N. The initial b messages present in the pool at round 1 are also destined to recipients chosen independently according to the same probability distribution $\vec{u}$.

9.3.1 Approximating the model

We are going to define a series of approximations. These approximations distance the generalised statistical disclosure attack from other exact attacks, but allow the adversary to make very quick calculations and to decrease the anonymity of Alice's set of recipients.

We will first model the input distribution $\vec{i_k}$ of recipient of messages of each round k as being a combination of the distributions $\vec{u}$ and $\vec{v}$. Depending on whether Alice sends a message or not the component $\vec{v}$ will be present.

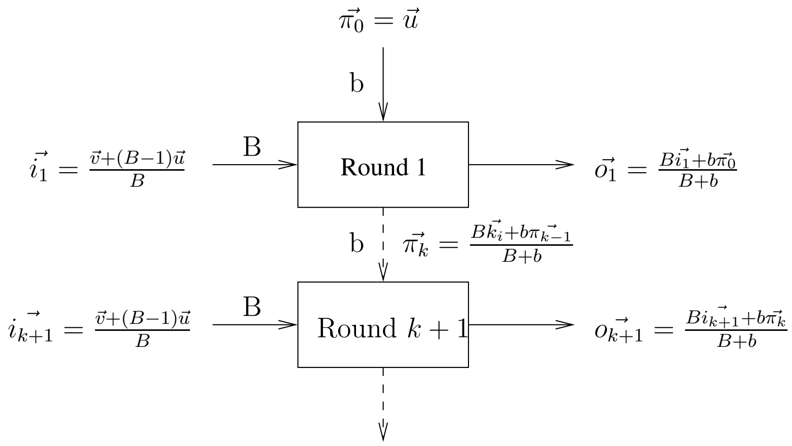

$\vec{\pi_0} = \vec{u}$
b
$\vec{i_1} = \frac{\vec{v}+(B-1)\vec{u}}{B}$
B
Round 1
$\vec{o_1} = \frac{B\vec{i_1}+b\vec{\pi_0}}{B+b}$
b
$\vec{\pi_k} = \frac{B\vec{k_i}+b\vec{\pi_{k-1}}}{B+b}$
$\vec{i_{k+1}} = \frac{\vec{v}+(B-1)\vec{u}}{B}$
B
Round $k+1$
$\vec{o_{k+1}} = \frac{B\vec{i_{k+1}}+b\vec{\pi_k}}{B+b}$

Figure 9.3: The pool mix model and the probability distributions defined.

$$\vec{i_k} = \begin{cases} \frac{\vec{v}+(B-1)\vec{u}}{B} & \text{if } s(k) = 1 \\ \vec{u} & \text{if } s(k) = 0 \end{cases} \tag{9.9}$$

$\vec{i_k}$ is a vector modelling the distribution of messages expected after a very large number of rounds with the input characteristic of input round k. Depending on whether Alice is sending at round k, ($s(k)$ being equal to one), the appropriate distribution is used to model this input.

At the same time we model the output of each round k, and name it $\vec{o_k}$. This output is the function of the input distribution at the particular round k and the distribution of recipients that is forwarded to the present round via the pool. We call the distribution of recipients that are in the pool $\vec{\pi}_{k-1}$. The output distribution of each round can then be modelled as

$$\vec{o_k} = \frac{B\vec{i_k} + b\vec{\pi}_{k-1}}{B+b} \tag{9.10}$$

By definition $\vec{\pi_0} = \vec{u}$ and for all other rounds the distribution that represents the pool has no reason to be different from the distribution that represents the output of the round. Therefore $\vec{\pi}_k = \vec{o}_k$.

The attacker is able to observe the function $s(k)$ describing the rounds at which Alice is sending messages to the anonymous communication channel.

The adversary is also able to observe for each round the list O_k of receivers, to whom messages were addressed.

The generalised statistical disclosure attack relies on some approximations:

- The set of receivers O_k at round k, can be modelled as if they were each independently drawn samples from the distribution $\vec{o_k}$ as modelled above.

- The outputs of the rounds are independent from each other, and can be modelled as samples from the distribution $\vec{o_k}$.

Using the samples O_k we will try to infer the distributions $\vec{o_k}$ and in turn infer the distribution $\vec{v}$ of Alice's recipients.

One can solve equation (9.10) for a given function $s(k)$ and calculate $\vec{o_k}$ for all rounds k. Each distribution $\vec{o_k}$ is a mixture of $\vec{u}$, the other senders' recipients, and $\vec{v}$ Alice's recipients. The coefficient x_k can be used to express their relative weights.

$$\vec{o_k} = x_k\vec{v} + (1 - x_k)\vec{u} \tag{9.11}$$

By combining Equations (9.9) and (9.10) one can calculate x_k as:

$$x_k = \sum_{i \leq k} s(i) \left(\frac{b}{B+b}\right)^{(i-1)} \frac{B}{B+b}\frac{1}{B} \tag{9.12}$$

This x_k expresses the relative contribution of the vector $\vec{v}$, or in other words Alice's communication, to each output in O_k observed during round k. When seen as a decision tree, each output contained in O_k has a probability $(1 - x_k)$ of being unrelated to Alice's set of recipients, but of being drawn instead from another participant's distribution $\vec{u}$.

9.3.2 Estimating $\vec{v}$

The aim of the attack is to estimate the vector $\vec{v}$ that Alice uses to choose the recipients of her messages. Without loss of generality we will select a particular recipient Bob, and estimate the probability v_{Bob} Alice selects him as the recipient.

We can calculate the probability of Bob being the recipient of Alice for each sample we observe in O_k. We denote the event of Bob receiving message i in the observation O_k as $O_{ki} \rightarrow$ Bob. Given our approximations we consider that the particular message O_{ki} was the outcome of sampling $\vec{o_k}$ and therefore by using equation (9.11) we can calculate the probabilities.

$$\Pr[O_{ki} \rightarrow \text{Bob}|v_{\text{Bob}}, u_{\text{Bob}}, x_k] = (x_k v_{\text{Bob}} + (1 - x_k)u_{\text{Bob}}) \tag{9.13}$$

$$\Pr[\neg O_{ki} \rightarrow \text{Bob}|v_{\text{Bob}}, u_{\text{Bob}}, x_k] = 1 - (x_k v_{\text{Bob}} + (1 - x_k)u_{\text{Bob}}) \tag{9.14}$$

As expected, Bob being the recipient of the message is dependent on the probability Alice sends a message v_{Bob} (that is Bob's share of $\vec{v}$), the probability others have sent a message u_{Bob} (which is Bob's share of $\vec{u}$) and the relative contributions of Alice and the other's to the round k, whose output we examine.

Now applying Bayes' theorem to Equations (9.13) and (9.14) we estimate p.

$$\begin{aligned}&\Pr[v_{\text{Bob}}|O_{ki} \rightarrow \text{Bob}, u_{\text{Bob}}, x_k] = \\ &\qquad \frac{\Pr[O_{ki} \rightarrow \text{Bob}|v_{\text{Bob}}, u_{\text{Bob}}, x_k] \Pr[v_{\text{Bob}}|u_{\text{Bob}}, x_k]}{\int_0^1 \Pr[O_{ki} \rightarrow \text{Bob}|v_{\text{Bob}}, u_{\text{Bob}}, x_k] \Pr[v_{\text{Bob}}|u_{\text{Bob}}, x_k] \mathrm{d}v_{\text{Bob}}} \\ &\qquad \sim (x_k v_{\text{Bob}} + (1 - x_k)u_{\text{Bob}}) \Pr[\text{Prior } v_{\text{Bob}}]\end{aligned}$$

$$\begin{aligned}&\Pr[v_{\text{Bob}}|\neg O_{ki} \rightarrow \text{Bob}, u_{\text{Bob}}, x_k] = \\ &\qquad \frac{\Pr[\neg O_{ki} \rightarrow \text{Bob}|v_{\text{Bob}}, u_{\text{Bob}}, x_k] \Pr[v_{\text{Bob}}|u_{\text{Bob}}, x_k]}{\int_0^1 \Pr[\neg O_{ki} \rightarrow \text{Bob}|v_{\text{Bob}}, u_{\text{Bob}}, x_k] \Pr[v_{\text{Bob}}|u_{\text{Bob}}, x_k] \mathrm{d}v_{\text{Bob}}} \\ &\qquad \sim (1 - (x_k v_{\text{Bob}} + (1 - x_k)u_{\text{Bob}})) \Pr[\text{Prior } v_{\text{Bob}}]\end{aligned}$$

Note that we choose to ignore the normalising factor for the moment since we are simply interested in the relative probabilities of the different values of v_{Bob}. The Pr[Prior v_{Bob}] term encapsulates our knowledge about v_{Bob} before the observation, and we can use it to update our knowledge of v_{Bob}. We will therefore consider whether each message observed has been received or not by Bob and estimate v_{Bob} considering in each step the estimate of v_{Bob} given the previous data as the a priori distribution[1]. This technique allows us to

[1] Since we are calculating relative probabilities we can discard the *a priori* since it is the uniform distribution over $[0, 1]$

estimate the probability distribution describing v_{Bob} given we observed R_k messages sent to Bob in each round k respectively.

$$\begin{aligned}&\Pr[v_{\text{Bob}}|(x_1, R_1)\dots(x_l, R_l), u_{\text{Bob}}]\\ &\quad\sim \prod_k (x_k v_{\text{Bob}} + (1-x_k)u_{\text{Bob}})^{R_k}(1-(x_k v_{\text{Bob}} + (1-x_k)u_{\text{Bob}}))^{(B-R_k)}\end{aligned}$$

The calculation above can be performed for each receiver in the system to estimate the likelihood it is one of Alice's receivers. The resulting probability distributions can be used as an indication of who Alice is communicating with, and their standard deviations can be used to express the certainty that this calculation provides.

9.4 Evaluation of the attack

Figure 9.4 shows the set of probability distributions for 50 receivers. In this case we take the probability distribution $\vec{u}$ to be uniform over all receivers and Alice to be choosing randomly between the first two receivers and sending messages for a thousand consecutive rounds (the mix characteristics in this case were $B = 10, b = 0$, namely it was a threshold mix). Figure 9.5 shows the same data for a pool mix with characteristics $B = 30, b = 15$. Note that the receivers 1 and 2 are Alice's and their respective v_1 and v_2 have different characteristics from the other receivers.

The same information can be more easily visualised if we take the average of all the distributions of receivers that do not belong to Alice, and compare them with the receivers of Alice. Figures 9.6(a) and 9.6(b) show the distributions of Alice's receivers and the averaged distributions of other receivers. The curves can be used to calculate the false positive rates, namely the probability a receiver has been attributed to Alice but is actually not in Alice's set, and false negative, namely a receiver wrongly being excluded from Alice's set of receivers.

It is unfortunate that we do not yet have analytic representations for the means and variances of the distribution describing v_{receiver}. Such representations would allow us to calculate the number of rounds for which Alice can

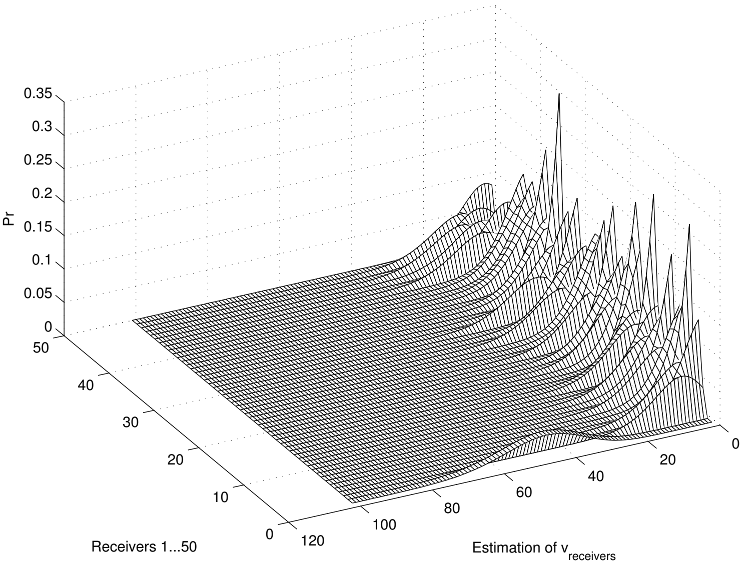


Figure 9.4: Comparing the distributions of v_{receiver} for $B = 10, b = 0$

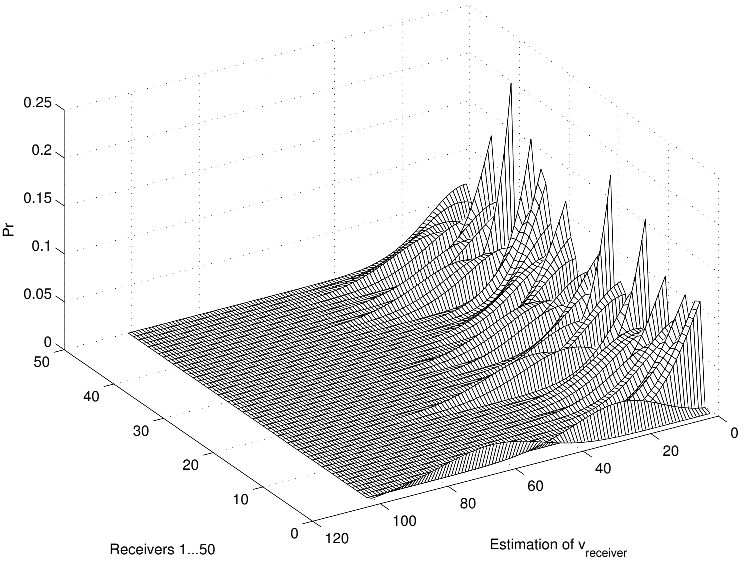
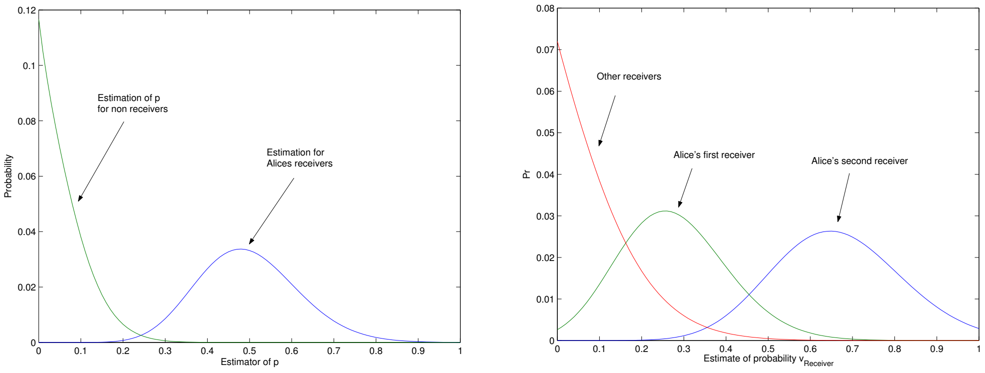


Figure 9.5: Comparing the distributions of $v_{\text{receiver}}$for $B = 30, b = 15$



(a) Comparing the distributions of v1 and others. B=10, b=0

(b) Comparing the distributions of v1, v2 and others. B=30,b=15

send messages, given a particular set of mix characteristics, without being detected with any significant degree of certainty. The attack presented allows an attacker to understand where they stand, and how much certainty the attack has lead to, by numerically calculating them. On the other hand the network designer must simulate the behaviour of the network for particular characteristics to get some confidence that it does not leak information.

9.5 Summary

In this section we have presented a family of powerful intersection attacks that can be applied to the whole anonymity system to find the recipients of particular senders. The first attack models the anonymity system as a threshold mix, and a detailed analysis of its applicability and efficiency is provided. The second attack analyses the anonymity system as a pool mix which should give more realistic results. Bayesian methods are used to extract the receivers, that have been validated experimentally, but no analysis has yet been possible. Both attacks are based on a set of carefully selected approximations that make them extremely efficient and very effective.

Chapter 10

Continuous stream analysis

> *"Intelligence agencies pursued a 'vacuum cleaner' approach to intelligence collection – drawing in all available information about groups and individuals, including their lawful political activity and details of their personal lives."*
>
> *Final Report of the Select Committee to Study*
> *Governmental Operations with Respect to Intelligence*
> *Activities of the United States Senate*
> — 94th Congress, 2nd Session, 1976

Round-based mix strategies, such as the pool mixes presented in section 2.3, provide well-understood anonymity properties. The same is not true for mixes that operate in continuous-time, by individually delaying messages, instead of batching then and ejecting them in rounds. Example of these include timed mixes but also the sg-mix construction presented by Kesdogan *et al.* [KEB98]. Its inventors present an analysis of its anonymity, but this cannot easily be generalised to other mix strategies. Furthermore, the definitions of anonymity used in Kesdogan's paper are different from the newer ones presented here which have, in our view, certain advantages.

We will therefore present a new framework for analysing the anonymity provided by mix strategies, functioning in continuous-time, that individually delay messages according to a predetermined strategy. In order to make the analysis easier, we assume that the rate of arrival of messages to the mixes is Poisson distributed. Using the work presented here, many different mix strategies can be analysed but we choose to illustrate our work with an analysis of an exponential mix (sg-mix), both because it is relatively simple (but not trivial as the simple timed mix) and because it has been extensively mentioned in the literature. Furthermore, a section is devoted to showing that the exponential mix has, given some latency constraints, the optimal mixing strategy.

We then proceed to show a powerful attack that, given enough packets, can break the anonymity provided by connection-based mix networks functioning in continuous-time. The attack is based on detecting an input traffic pattern, at the outputs of the mixes or network, using signal detection and estimation techniques. A detailed description is given on how to perform this attack, and confidence intervals are provided to assess how reliable are the results provided. The attack can be used effectively against many proposed anonymous communications systems such as Onion Routing [RSG98], TARZAN [FM02] or MorphMix [RP02].

10.1 The delay characteristic of a mix

The main aim of a mix node, as introduced by Chaum in [Cha81], is to hide the correspondence between its inputs and outputs. First it makes its inputs and outputs bitwise unlinkable, which means that a third party cannot link them by observing their bit patterns without knowledge of the cryptographic keys used to perform the transform. Second it blurs the timing correlations between inputs and outputs by batching, introducing appropriate random delays, or reordering the messages. Continuous-time mixes employ a family of mixing strategies that simply delay each message independently of the others.

We can say that a particular mix strategy is characterised by its *delay characteristic*. This is a function $f(\beta|\alpha)$ that calculates the probability a message injected in the mix at time α leaves the mix at time β, where $\alpha \leq \beta$. Since $f(\beta|\alpha)$ is a conditional probability distribution, it is normalised.

$$\forall \alpha. \int_{\alpha}^{+\infty} f(\beta|\alpha)\, d\beta = 1\,. \tag{10.1}$$

The *inverse delay characteristic*, $f'(\alpha|\beta)$, of the same mix strategy is a probability distribution that describes the likelihood a message being ejected at time β was injected at time α. Again because it is a conditional probability distribution it is normalised.

$$\forall\beta.\ \int_{-\infty}^{\beta} f'(\alpha|\beta)\, d\alpha = 1\,. \tag{10.2}$$

It is obvious that the two characteristics are related. Indeed the second f' can be calculated using Bayes theorem from f. Some knowledge of the probability of arrivals at particular times is necessary to perform this conversion. In order to simplify the analysis we will consider that arrivals are Poisson distributed with a rate λ_α. We rely on the fact that in a Poisson process, the probability of an arrival is independent from other arrivals and the particular time α.

$$\begin{align} f'(\alpha|\beta) &= \frac{f(\beta|\alpha)\mathrm{Pr}[\text{Arrival at }\ \alpha]}{\int_{-\infty}^{\beta} f(\beta|\alpha)\mathrm{Pr}[\text{Arrival at }\ \alpha]\, d\alpha} \tag{10.3}\\ &= \frac{f(\beta|\alpha)}{\int_{-\infty}^{\beta} f(\beta|\alpha)\, d\alpha} \tag{10.4}\end{align}$$

Therefore, given the delay characteristics and some assumptions about the traffic in the network we can calculate the inverse delay characteristic. As we will see, these will allow us to measure the effective sender and receiver anonymity sets for this mix strategy.

10.1.1 Effective sender anonymity sets

We will use the metric introduced in section 2.3 to calculate the sender anonymity that a particular mix strategy provides. This metric is based on creating a random variable that describes the possible senders of a departing message and calculating the entropy of its underlying probability distribution. The value of the entropy is then a measure of the anonymity provided, and can be interpreted as the amount of information an attacker is missing to deterministically link the messages to a sender.

We assume that in a time interval $(\beta - T, \beta)$, K messages arrive at the mix, where K is distributed according to a Poisson distribution with parameter

$\lambda_\alpha T$. These messages arrive at times $X_{1\dots K}$ each distributed according to a uniform distribution $U(t)$ over the time interval of length T.

Given that the inverse delay characteristic of the mix is $f'(\alpha|\beta)$ the anonymity provided by the mix can be calculated.

$$\mathcal{A} = \sum_{i=1}^{K} \frac{f'(X_i|\beta)}{\sum_{j=1}^{K} f'(X_j|\beta)} \log \frac{f'(X_i|\beta)}{\sum_{j=1}^{K} f'(X_j|\beta)} \tag{10.5}$$

$$= \frac{1}{\sum_{j=1}^{K} f'(X_j|\beta)} \left(\sum_{i=1}^{K} f'(X_i|\beta) \log f'(X_i|\beta) \right) - \log \sum_{j=1}^{K} f'(X_j|\beta) \tag{10.6}$$

From the Law of Large Numbers we know that the sums will respectively converge.

$$\sum_{j=1}^{K} f'(X_j|\beta) \to \frac{K}{T} \tag{10.7}$$

$$\sum_{i=1}^{K} f'(X_i|\beta) \log f'(X_i|\beta) \to \frac{K}{T} \int_{\beta-T}^{\beta} f'(t|\beta) \log f'(t|\beta) dt \tag{10.8}$$

Thus the fraction K/T converges to λ_α, which is the rate of arrival of messages to the mix and the integral above reduces to the entropy of the inverse delay characteristic function $\mathcal{E}[f'(\alpha|\beta)]$. Therefore the sender anonymity of a continuous mix with this delay characteristic f' and a rate of arrival λ_α can be expressed.

$$\mathcal{A} \to \mathcal{E}[f'(\alpha|\beta)] - \log \lambda_\alpha \tag{10.9}$$

Putting this into words, the effective sender anonymity set size of the mixing strategy will converge towards the relative entropy of the inverse delay characteristic, as defined by Shannon [Sha48], minus the logarithm of the rate at which messages are received. Similarly the recipient anonymity set size can be calculated using the same techniques and the delay characteristic of the mix strategy.

10.1.2 The exponential mix

In order to illustrate how to quantify the anonymity provided by a continuous mixing strategy we will present an analysis of the exponential mix. The exponential mix has been presented as a mixing strategy by Kesdogan *et al.* [KEB98], but in his design additional features are implemented to avoid $(n-1)$ attacks [GT96, SDS02].

The exponential mix can be abstracted as an $M/M/\infty$ queue. We assume, as required from the calculations above, the arrival rates of messages to be Poisson distributed with rate λ_α. Each of the messages that arrives at the Mix is delayed according by a random variable that follows the exponential distribution with parameter μ. Therefore the delay characteristic of the exponential mix is

$$f(\beta|\alpha) = \mu e^{-\mu(\beta-\alpha)} \,. \tag{10.10}$$

From equation (10.4) we can calculate the inverse delay characteristic, and see that is equal to the delay characteristic, due to the nature of the exponential distribution.

$$\begin{aligned} f'(\alpha|\beta) &= \frac{f(\beta|\alpha)}{\int_{-\infty}^{\beta} f(\beta|\alpha)\, d\alpha} && (10.11)\\ &= f(\beta|\alpha) && (10.12)\\ &= \mu e^{-\mu(\beta-\alpha)} && (10.13) \end{aligned}$$

Using the inverse delay characteristic, and (10.9) we can now calculate the expected sender anonymity set size. As defined previously, $\mathcal{E}[\cdot]$ is the entropy function.

$$\begin{aligned} \mathcal{E}[\Pr[\alpha]] &\to \mathcal{E}[f'(\alpha|\beta)] - \log \lambda_\alpha && (10.14)\\ &= \int_{-\infty}^{\beta} \mu e^{-\mu(\beta-\alpha)} \log \mu e^{-\mu(\beta-\alpha)}\, d\alpha - \log \lambda_\alpha && (10.15)\\ &= -\log \frac{\lambda_\alpha e}{\mu} && (10.16) \end{aligned}$$

To check the above result a simulation was run for some values of λ_α and μ, and the results were compared with the metric predictions in equation (10.16). The inverse delay characteristic was used to calculate the probability assigned to a number of messages arriving at a mix. The number of messages was Poisson distributed according to λ_α, and their time of arrival was chosen uniformly. Their delay was a random variable distributed according to the

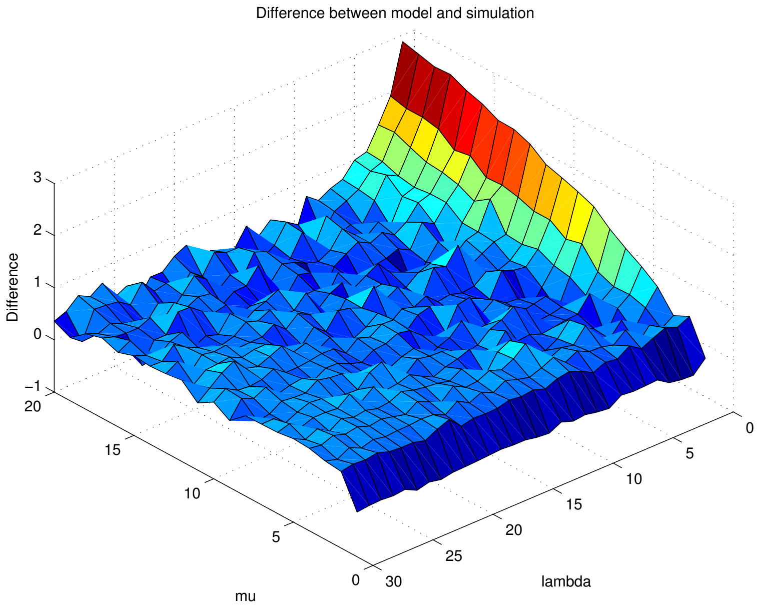
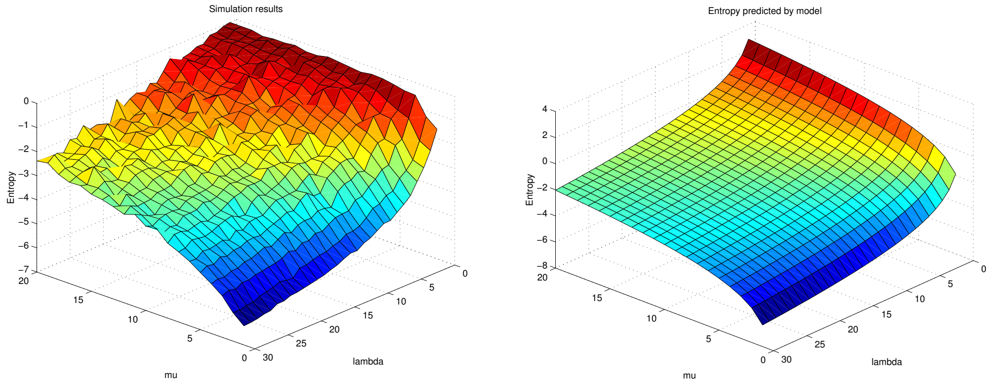


Figure 10.1: Absolute difference between metric and simulation



(a) Exponential mix simulation results

(b) Metric predictions for exponential mix exponential distribution with rate μ. The results are presented in figure 10.1.

It is worth noting that the main divergence of the simulated results from the predicted results, is in the region where the metric predicts positive values for the entropy. This is intuitively impossible and indeed is the largest error from the actual simulation results. The conditions for which the model, that the equation (10.16) describes, should not be considered accurate is described by:

$$-\log \frac{\lambda_\alpha e}{\mu} > 0 \tag{10.17}$$

$$\mu > \lambda_\alpha e \tag{10.18}$$

It is clear that an $M/M/\infty$ queue with a departure rate μ larger than the arrival rate λ_α would not provide much anonymity most of the time. The average time a message would spend in the mix is $\frac{1}{\mu}$ while the average time between message arrivals is $\frac{1}{\lambda_\alpha}$, which is larger. Therefore the mix would behave most of the time as a first-in first-out queue.

10.1.3 The timed mix

A simpler example that also illustrates the method introduced to calculate the anonymity of a mix strategy, is the timed mix. A timed mix gathers messages for a period of time t and sends them all in a batch at the end of this period. Its inverse delay characteristic is easier to express than the delay characteristic and is equal to:

$$f'(\alpha|\beta) = \begin{cases} \frac{1}{t} & \text{if}\, \alpha > \beta - t \\ 0 & \text{if}\, \alpha \leq \beta - t \end{cases} \tag{10.19}$$

The effective sender anonymity set size can then be calculated using equation (10.9).

$$H \rightarrow \int_{-\infty}^{\beta} f'(\alpha|\beta) \log f'(\alpha|\beta)\, d\alpha - \log \lambda_\alpha \tag{10.20}$$

$$= \int_{\beta-t}^{\beta} \frac{1}{t} \log \frac{1}{t}\, d\alpha - \log \lambda_\alpha \tag{10.21}$$

$$= -\log \lambda_\alpha t \tag{10.22}$$

Since positive values mean that no anonymity at all is provided, we require the above to be negative for the system to provide any mixing.

$$-\log \lambda_\alpha t < 0 \tag{10.23}$$

$$t > \frac{1}{\lambda_\alpha} \tag{10.24}$$

10.1.4 The latency of a mix strategy

The delay characteristic of a mix can also be used to calculate the latency introduced by a mix strategy, and its variance. This can be done trivially since the latency of the mix strategy is the expectation $E[\cdot]$ of the delay characteristic function $f(\beta|\alpha)$.

$$E[f(\beta|\alpha)] = \int_\alpha^{+\infty} (\beta - \alpha)\, f(\beta|\alpha)\, d\beta \tag{10.25}$$

Similarly the variance $V[\cdot]$ of the delay can be calculated using the expectation:

$$V[f(\beta|\alpha)] = \int_\alpha^{+\infty} (E[f(\beta|\alpha)] - (\beta - \alpha))^2\, f(\beta|\alpha)\, d\beta \tag{10.26}$$

For the exponential mix presented in the previous section the mean delay is $\frac{1}{\mu}$ and the variance is $\frac{1}{\mu^2}$.

10.1.5 Optimal mixing strategies

So far, we have described how to measure a continuous-time mix's anonymity, latency and variance, given its delay strategy. The next natural problem is finding a mix strategy that maximises entropy, and therefore anonymity.

Without any constraints the uniform constant distribution f in the interval $[0, +\infty)$ is optimal. This function would clearly be impractical for mixing since the latency of messages would be unpredictable. Furthermore the length of the queues in the mixes would tend to infinity, which makes such a mix strategy impossible to implement.

We need to find a distribution f with a particular mean a, which represents the average latency of the mix. Furthermore because of causality, namely a packet only being able to leave the mix after it arrived, the function f can only occupy half the line, namely the interval $[0, +\infty)$. We need to prove that the optimal probability distribution f is the exponential probability distribution. This result was first proved by Shannon [Sha48] using techniques from the calculus of variations [Wei74]. We want to minimise:

$$E[f(x)] = -\int_0^{-\infty} f(x) \log f(x) dx \tag{10.27}$$

Subject to the constraints:

$$a = \int_0^{-\infty} x f(x) dx \tag{10.28}$$

$$\int_0^{-\infty} f(x) dx = 1 \tag{10.29}$$

Then by the calculus of variations we must solve:

$$\frac{\partial(-f(x)\log f(x) + \lambda x f(x) + \mu f(x))}{\partial f} = 0 \tag{10.30}$$

$$\Rightarrow\ -1 - \log f(x) + \lambda x + \mu = 0 \tag{10.31}$$

$$\Rightarrow\ f(x) = e^{\lambda x + \mu - 1} \tag{10.32}$$

By using the constraints, the resulting function is:

$$f(x) = \frac{1}{a} e^{-\frac{1}{a}x} \tag{10.33}$$

This is exactly the exponential mix as analysed in section 10.1.2.

10.2 Traffic analysis of continuous mixes

In the previous sections we have considered the anonymity of single packets mixed using a continuous-time mixing strategy. Continuous-time mixes can approximate circuit-based systems, since they implement minimal mixing trying to provide real-time communications. In such systems a number of packets, all belonging to the same stream, are routed through the same route in the network.

The Onion Routing project [STRL00] first drew the community's attention to the need for traffic padding to protect against fine-grained traffic analysis. Since then many publications have discussed traffic analysis and possible defences against it [BMS01, Ray00]. Others refer to the same problem in the context of intersection attacks [BPS00, BL02, KAP02] and present padding as a potential protection.

Some previous work has drawn attention to the vulnerabilities of anonymous systems to "timing" attacks [RP02], but only Kesdogan *et al.* present a concrete attack [KAP02]. We will now present a very general way of performing traffic analysis on streams of packets travelling through the same route in a continuous-time mix network. We will show that after a certain number of messages, that can be calculated, the communication can be traced with high confidence.

10.2.1 Concrete traffic analysis techniques

We denote as $f(t)$ the function that describes the traffic on one of the input links of a continuous mix with delay characteristic $d(x)$. We assume that all messages described by $f(t)$ belong to the same stream, and will therefore be ejected on the same output link.

An attacker might acquire the knowledge that a series of messages belong to the same stream in a number of ways. The simplest one is by observing unpadded links at the edges of the mix network. A subverted node will also be able to link messages to streams. We will assume that there are two output links. The attacker's aim is to determine on which output link the stream is redirected.

On the first link we observe messages coming out at times $X_{1\cdots n}$ and on the second link messages come out at times $Y_{1\cdots m}$ in the time interval $[0, T]$. H_0 represents the hypothesis the input stream $f(t)$ is interleaved in the first channel described by the observations X_i, and H_1 that is in the second corresponding with Y_i.

In order to detect the streams we will make some approximations.We will create two model probability distributions C_X and C_Y and will assume that all messages in the output channels are independent samples out of one of these distributions. The difference between C_X and C_Y is due to our attempt to model the noise in the two output channels. We will also consider that all the other messages are uniformly distributed in the interval $[0, T]$ according to the distribution $U(t) = u$.

When H_0 is true the stream under observation is interleaved in the observations X_i. We will model each of them as following probability distribution:

$$C_X(t) = \frac{\lambda_f (d * f)(t) + (\lambda_X - \lambda_f)U(t)}{\lambda_X} \tag{10.34}$$

In the above probability distribution $(d * f)$ is the convolution of the input signal with the delay characteristic of the mix. The probability a message delayed by $d(x)$ is output at time t given an input stream of messages described by $f(t)$ is described by this convolution.

$$(d * f)(t) = \int d(x) f(t - x) dx \tag{10.35}$$

Furthermore λ_f is the rate of messages in the input signal, while λ_X is the rate of the output channel. Finally $U(t) = u$ is the uniform distribution in the interval $[0, T]$.

Similarly if hypothesis H_1 is true, the signal is interleaved in the observations Y_i that follow the distribution:

$$C_Y(t) = \frac{\lambda_f (d * f)(t) + (\lambda_Y - \lambda_f)U(t)}{\lambda_Y} \tag{10.36}$$

In order to decide which of the two hypothesis is valid, H_0 or H_1, we can calculate the likelihood ratio of the two alternative hypothesis.

$$\frac{\mathcal{L}(H_0 | X_i, Y_j)}{\mathcal{L}(H_1 | X_i, Y_j)} = \frac{\prod_{i=1}^{n} C_X(X_i) \prod_{j=1}^{m} u}{\prod_{i=1}^{n} u \prod_{j=1}^{m} C_Y(Y_j)} > 1 \tag{10.37}$$

We choose to accept hypothesis H_0 if the condition (10.37) is true, and hypothesis H_1 otherwise. Section 10.2.3 will show how we calculate our degree of confidence when making this choice.

10.2.2 Observations

Figure 10.2 shows six diagrams illustrating the traffic analysis attack. The first column represents, from top to bottom, the signal that we inject in a mix and the two output channels, one of which contains the delayed signal. The

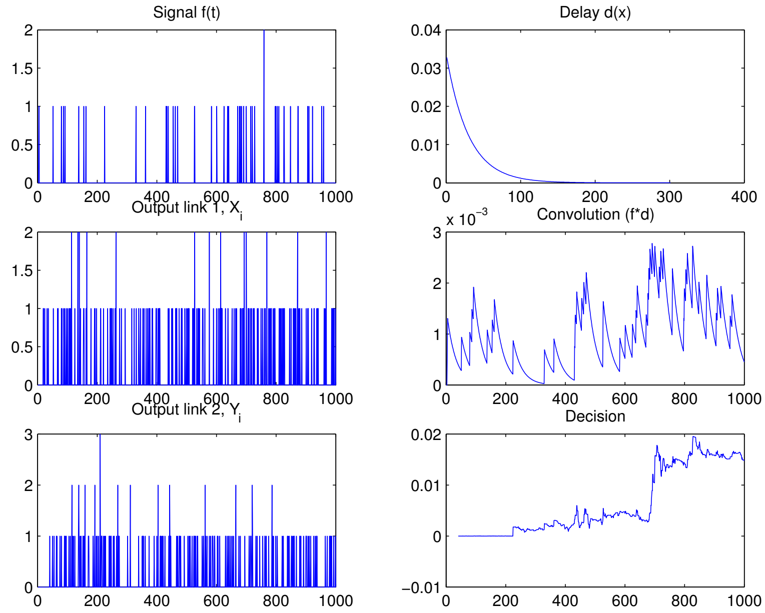


Figure 10.2: Final and intermediate results of traffic analysis.

right hand side column represents the delay characteristic of the network, an exponential distribution in this case (sg-mix), the “model” that is created by convolving the input signal with the delay characteristic and, at the bottom, the log-likelihood ratio.

The “noise” in the above experiments is assumed to be a Poisson process. Noise is added both to the channel that contains the stream under surveillance (in this case link 1, X_i) and the other link (Y_i). The rate of the signal $f(t)$ in the traffic analysis graphs shown above is 50 messages, while the noise added in X_i has a rate of 150 messages. The second link contains random padding with a rate of 200 messages (Y_i). The delay characteristic $d(x)$ chosen to illustrate the traffic analysis technique is exponential with a departure rate of 30. The graphs therefore illustrate the traffic analysis of an sg-mix node. The decision graph presents the logarithm of the likelihood ratio $\log \frac{\mathcal{L}(H_0|X_i,Y_j)}{\mathcal{L}(H_1|X_i,Y_j)}$, as an attacker would compute it at each point in the simulation time.

10.2.3 Performance of the traffic analysis attack

There are two question that need to be answered concerning the traffic analysis attack presented. First the conditions under which it is at all possible must be established. Second the number of observations necessary to get reliable results has to be calculated.

By simple mathematical manipulations with logarithms, we can derive that the likelihood ratio test, applied to select the most appropriate hypothesis can be expressed using sums of random variables:

$$\mathcal{L}_{H_0/H_1} = \frac{\mathcal{L}(H_0|X_i, Y_j)}{\mathcal{L}(H_1|X_i, Y_j)} = \frac{\prod_{i=1}^{n} C_X(X_i) \prod_{j=1}^{m} u}{\prod_{i=1}^{n} u \prod_{j=1}^{m} C_Y(Y_j)} > 1 \tag{10.38}$$

$$\Rightarrow \sum_{i=1}^{n} \log C_X(X_i) + m \log u > n \log u + \sum_{j=1}^{m} \log C_Y(Y_j) \tag{10.39}$$

$$\Rightarrow \log \mathcal{L}_{H_0/H_1} = \sum_{i=1}^{n} \log C_X(X_i) - \sum_{j=1}^{m} \log C_Y(Y_j) + (m-n) \log u > 0 \tag{10.40}$$

The expression above is the rule by which we choose the hypothesis to accept. The condition for which the attack is possible is that the decision rule above must equal to zero. This could be the case if both C_X and C_Y were the uniform distribution.

But even through the inequality might hold it does not give us any measure of confidence in the result. We will therefore attempt to find bounds within which we are confident that the decision is correct.

Note that the two sums will converge to the expectations $nE\left[\log C_X(X)|X_i \sim X\right]$ and $mE\left[\log C_Y(Y)|Y_j \sim Y\right]$. The notation $X_i \sim X$ means that the samples X_i are sampled from the distribution X, and the samples Y_j from the distribution Y. The two distributions X and Y are different according to the two hypothesis accepted. In case H_0 then $X_i \sim C_X, Y_j \sim U$. Alternatively if H_1 is true then $X_i \sim U$ and $Y_j \sim C_Y$.

Without losing generality we will demonstrate when to accept hypothesis H_0. The derivations are the same in the other case.

In case the hypothesis H_0 is correct, $E\left[\log C_X(X)|H_0 : X_i \sim C_X\right]$ converges to the entropy of the probability distribution $C_X(t)$, denoted $\mathcal{E}[C_X(t)]$, since the probabilities assigned to each value of the random variable $\log C_X(X)$ follow the distribution C_X.

$$E\left[\log C_X(X)|H_0 : X_i \sim C_X\right] = \int_0^T C_X(t)\log C_X(t)dt = \mathcal{E}[C_X(t)] \quad (10.41)$$

On the other hand $E\left[\log C_Y(Y)|H_0 : Y_j \sim U\right]$ converges to the expectation of C_Y namely $E\left[\log C_Y(t)\right]$.

$$E\left[\log C_Y(Y)|H_0 : Y_j \sim U\right] = \int_0^T \frac{1}{T}\log C_Y(t)dt = E\left[\log C_Y(t)\right] \quad (10.42)$$

Therefore in case we accept hypothesis H_0 the expected value of the decision rule $\log \mathcal{L}_{H_0/H_1}$ (10.40) is μ_{H_0}:

$$\begin{aligned}\mu_{H_0} = E\left[\sum_{i=1}^{n}\log C_X(X_i) - \sum_{j=1}^{m}\log C_Y(Y_j) + (m-n)\log u|H_0\right] = \\ nE\left[\log C_X(X)|H_0\right] - mE\left[\log C_Y(Y)|H_0\right] + (m-n)\log u = \\ n\mathcal{E}[C_X(t)] - mE\left[\log C_Y(t)\right] + (m-n)\log u \quad (10.43)\end{aligned}$$

The variance can be calculated using the above observations:

$$V\left[\log C_X(X)|H_0\right] = \int_0^T C_X(t)(\log C_X(t) - \mathcal{E}\left[C_X(X)\right])^2 dt \quad (10.44)$$

$$V\left[\log C_Y(Y)|H_0\right] = \frac{1}{T}\int_0^T (\log C_Y(t) - E\left[\log C_Y(Y)\right])^2 dt \quad (10.45)$$

Using these we will calculate the variance $\sigma^2_{H_0}$ of the decision rule $\log \mathcal{L}_{H_0/H_1}$ (10.40) is:

$$\begin{aligned}\sigma^2_{H_0} = V\left[\sum_{i=1}^{n}\log C_X(X_i) - \sum_{j=1}^{m}\log C_Y(Y_j) + (m-n)\log u|H_0\right] = \\ = nV[\log C_X(X)|H_0] + mV\left[\log C_Y(Y)|H_0\right] \quad (10.46)\end{aligned}$$

Using Chebyshev's inequality we can derive that in order to accept hypothesis H_0 with confidence p it must hold that:

$$p = \Pr\left[\left|\log \mathcal{L}_{H_0/H_1} - \mu_{H_0}\right| \geq \mu_{H_0}\right] \leq \frac{\sigma^2_{H_0}}{\mu^2_{H_0}} \Rightarrow \quad (10.47)$$

$$p \leq \frac{\sigma^2_{H_0}}{\mu^2_{H_0}} \quad (10.48)$$

An equivalent test can be derived to assess our confidence when accepting hypothesis H_1.

10.3 Further considerations and future work

The work presented in this chapter measures the average anonymity provided by a mix strategy. One of the important assumptions is that the expected number of messages is received in any time interval t, namely $\lambda_\alpha t$. The actual number of messages received in any interval may vary according to the Poisson distribution. Should a mix be flooded by the attacker's messages the rate needs to be adjusted to the level of genuine traffic. The rgb-mix strategy could be a way of calculating this.

Mix strategies that take into account the number of messages queueing or that adapt their parameters according to the rate of arrival of messages have not been explicitly studied. The metric proposed should still be usable with them, although their delay characteristic function may be dependant of additional factors such as the rate of arrival of messages λ_α. We expect the functions depending on the delay characteristic, such as the mean and variance of the latency, to still be usable.

10.3.1 Traffic analysis of streams

Much more work needs to be done on how far the traffic analysis attack presented on stream-based anonymity systems can be exploited. Techniques from transient signal detection, as surveyed in [WW00], can be used as a theoretical foundation for such traffic analysis.

The attack assumes that an adversary can observe a "naked" stream somewhere in the network, in order to build a model later used for detection. This assumption might be invalidated if cover traffic is used on all links, but variants of the attack might still work. Some preliminary results suggest that good models can be created despite this.

The attack can be performed by a passive adversary, without any knowledge of the relationships between packets on the attacked links. When an attacker knows the relationship between packets in the same stream, as a subverted node would, it is much easier to perform the statistical tests since the cover traffic can in fact be discarded.

Furthermore the attacks are passive, in the sense that the attacker does not modify in any way the characteristics of the traffic. An active attacker would modulate the input traffic in order to maximise the chances of detecting it. Techniques used might be to introduce periodicity, allowing for periodic averaging for noise cancellation, injecting patterns of traffic[1] special designed to be easily detected. Unless the anonymity system takes special steps beyond delaying the traffic to destroy such structure, traffic streams will quickly be traceable.

10.4 Summary

The information theoretic anonymity metric is adapted to describe the properties of mixes that simply delay individual packets. We discovered that the optimal delaying strategy is the exponential mix, for which we calculate the anonymity and latency.

A very powerfully attack is then presented that traces streams of messages following the same path through a delaying mix network. We present the conditions under which it is possible, and derive expressions that an adversary can use to assess his confidence. This attack is applicable to systems that provide real-time anonymous communications and leaves us very sceptical about the possibility of secure and efficient such constructions.

[1] Markus Kuhn was the first to observe this.

Chapter 11

Conclusions and future work

> *"Whoever thinks his problem can be solved using cryptography, doesn't understand his problem and doesn't understand cryptography."*
>
> — Roger Needham or Butler Lampson

During our three years of work in the field of anonymous communications we have discovered a series of attacks, proposed new systems and provided a framework for understanding anonymity.

Mixminion, presented in chapter 5, is recognised as the state-of-the-art anonymous remailer. At the time of writing the alpha reference implementation[1] runs on twenty six machines, and is fully usable. Due to the secure reply block facilities and other features, the remailer community has agreed that it will gradually phase out the older Cypherpunk and Mixmaster remailers. Mixminion shows that the cryptographic security of mix systems is now well understood, and secure systems can be constructed to leak very little information.

While Mixminion provides very good security against passive or active attackers and subverted nodes, operators are still liable to be compelled to decode messages. For this reason we explore in detail how messages can be traced in a network, and design the *fs-mix* to provide forward secure anonymity (chapter 6).

[1] Implemented and maintained by Nick Mathewson.

The static security aspects of anonymous communications are well understood, and time has come to build systems using them. Section 5.4 presents the design of a secure pseudonym server, to act as a gateway between the world of anonymous communications and regular email. It also presents the *Who am I?* attack reminding us that such protocols must be implemented with care, not to jeopardise the anonymity of users.

On the other hand the dynamic aspects of mixing and anonymous communications have not been previously studied in such depth. We contributed to this field by providing a technical definition for anonymity described in section 2.3. This definition, based on information theory, brings anonymity measures in line with traditional security measures. It also allows systems that bear little other resemblance to be compared and partial attacks to be expressed[2].

The anonymity metric is used to assess how mixes can be arranged in fully connected or sparse networks to relay traffic (chapter 7). For the first time a full analysis of the impact of these topologies on anonymity is provided. A new topology based on expander graphs proves to be quite effective: messages need to only travel for a few hops and less traffic is needed to protect the network against traffic analysis.

Although the work on network topologies examines how much anonymity is provided against a passive adversary it fails to address active attackers. In order to prevent such attackers flooding honest nodes to perform traffic analysis, we have engineered the *rgb-mix*, in chapter 8. Such a mix uses active countermeasures to detect if it is under attack, and if necessary uses cover traffic to maintain the anonymity it provides.

The analysis of network topologies and the active countermeasures against flooding attacks ensure it is difficult to perform traffic analysis, and therefore extract any information from observing the working of mix networks. On the other hand they do not protect against an attacker that draws inferences by observing all inputs and outputs to the whole network. Such attacks are

[2] This by itself is important for an academic field to grow, since partial attacks can be the subject of publications. Imagine how backwards the field of block cipher security would be if the insecurity introduced by differential and linear cryptanalysis could not be quantified.

presented and analysed in chapter 9. These statistical attacks will, in the long run, always uncover the user's persistent patterns of communication.

Even more devastating attacks have been devised against anonymizing networks that hardly mix, and simply delay and interleave streams. Chapter 10 presents an anonymity analysis of such mix strategies, and uses it to calculate the strategy providing maximal anonymity. It then presents an attack, based on statistical pattern recognition, that traces streams of traffic through the network.

When possible, and inspired by our anonymity metric, we have attempted to provide a full analysis of both the defences and attacks proposed. As a result we have laid engineering foundations for a whole section of the field of traffic analysis that used to be considered a black art.

A lot of work [SK03, MNCM03, MNS] has been inspired by the anonymity metric proposed in section 2.3, which is quite expressive. What is still questionable is how it can be used as a tool to understand complex anonymizing networks by measuring the anonymity offered by their components. The composition rules are a first step in this direction.

Mixminion offers the core of an anonymizing channel for email traffic, but a lot of its peripheral infrastructure is still not well defined. Open questions include the impact of crooked directory servers, reliable transmission protocols that are not prone to traffic analysis, and implementing anonymous services at higher layers. The conservatism of the design also imposes some overheads that could be simplified, as discussed at the end of chapter 5. A formal analysis of the bitwise unlinkability properties of the packet format, might allow us to simplify it further. More research is also needed to make sure that Mixminion is not prone to denial-of-service attacks.

Given that compulsion threats are very realistic, much more research should be done on forward security and plausible deniability properties. It is an embarrassing realisation that most anonymous communication protocols are not receipt-free, and cannot therefore be used for electronic elections. Receipt-freeness and plausible deniability properties could also be sought in future mix network architectures in order to protect the mix node operators from compulsion attacks.

The work on mixing using sparse networks, presented in chapter 7, needs to be extended for the case of networks with highly dynamic membership. This could answer some questions about the anonymity provided by peer-to-peer mix networks, such as Tarzan. Generally solutions to other problems, such as peer discovery and route reconstruction attacks, presented in section 4.2.7, have to be found before the peer-to-peer paradigm can effectively be used for mixing.

Taking active steps to avoid flooding attacks, as the rgb-mix does, and making nodes aware of their environment, is a new field that may well lead to interesting new systems and attacks. The active monitoring for colluding nodes performed by MorphMix [RP02] is a step in this direction. More research is required on how traffic padding can be used in a well-understood manner, rather than to confuse everyone (including the designers) about the security of the system.

Both statistical disclosure attacks and the attack presented against continuous-time streams illustrate the difficulty of properly anonymising persistent communications between two parties. A lot more research is required on the attacks themselves. In particular they are based on some approximations that, while validated experimentally, are not fully understood. Protecting again such attacks is then the next challenge.

Anonymous communications, aside from their military applications, were once presented as the straightforward solution to protecting one's privacy in an increasingly surveilled world. This thesis illustrates the dangers of unconditionally adopting such an attitude.

We have shown that low-volume, high-latency email anonymous communications can be secured by a system such as Mixminion for a finite number of message exchanges. The packet formats are cryptographically robust and can even be modified to resist compulsion attacks. On the other hand, the statistical disclosure attacks clearly show that the anonymity of long-term conversations will eventually be compromised. Therefore anonymous communications can only offer tactical, and not strategic, protection against a global passive adversary.

On the other hand, high-volume, low-latency communications, as required for web browsing, seem immediately susceptible to attack. The statistical disclosure attacks and the attacks against continuous-time stream-based mix systems leaves us very sceptical about the possibility of securing them. Furthermore approaches such as peer-to-peer systems, contrary to popular belief, seem to offer a lesser degree of protection. They are susceptible to peer discovery attacks as with Tarzan, and they offer weaker protection against traffic analysis because their topologies spread the traffic too thinly across large numbers of links.

Our results might at first appear very negative. It is worth keeping in mind that the threat models we used are harsh, and that adversaries will need huge resources in order to mount such attacks. A new line of research should assess how these techniques protect individuals and organisations against weaker threat models.

These conclusions might also be disturbing because a lot of protocols performing other security tasks, such as censorship resistance, have traditionally assumed anonymous channels. Peer-to-peer protocols in turn assumed that large amounts of data can be transported anonymously to do file sharing. Unless some unforeseen breakthrough is made, such channels seem unlikely to materialise in the near future.

The assumption of the existence of secure anonymous channels has for a long time encouraged protocol and network designers to explore particular avenues. Maybe these negative conclusions should be a new beginning. It is time for all of us to go back and perform some solid requirements engineering and design work to find out if the same properties can be offered using weaker, purpose specific anonymity, or even no anonymous channels at all.

Bibliography

[AB96] Ross Anderson and Eli Biham. Two practical and provably secure block ciphers: BEAR and LION. In D. Gollmann, editor, *International workshop on Fast Software Encryption*, volume 1039 of *LNCS*, pages 113–120, Cambridge, UK, 1996. Springer-Verlag.

[Abe98] Masayuki Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In K. Nyberg, editor, *Advances in Cryptology (Eurocrypt'98)*, volume 1403 of *LNCS*, pages 437–447, Helsinki, Finland, 31 May–4 June 1998. Springer-Verlag.

[ACL+99] Ross Anderson, Bruno Crispo, Jong-Hyeok Lee, Charalampos Manifavas, Vaclav Matyas, and Fabien Petitcolas. *The Global Internet Trust Register*. MIT press, 1999. ISBN: 0262511053.

[AES01] Advanced Encryption Standard, FIPS-197. National Institute of Standards and Technology, November 2001.

[AKP03] Dakshi Agrawal, Dogan Kesdogan, and Stefan Penz. Probabilistic treatment of mixes to hamper traffic analysis. In *IEEE Symposium on Security and Privacy*, pages 16–27, Berkeley, CA, USA, May 2003. IEEE Computer Society.

[And96] Ross Anderson. The eternity service. In *1st International Conference on the Theory and Applications of Cryptology (Pragocrypt '96)*, pages 242–252, Prague, Czech Republic, September/October 1996. Czech Technical University Publishing House.

[And97] Ross Anderson. Two remarks on public-key cryptology. Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`, 1997. Invited Lecture, ACM-CCS '97.

[ANS98] Ross Anderson, Roger Needham, and Adi Shamir. The steganographic file system. In David Aucsmith, editor, *Information Hiding (IH'98)*, volume 1525 of *LNCS*, pages 73–82, Portland, Oregon, USA, 15-17 April 1998. Springer-Verlag.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology (Asiacrypt 2000)*, volume 1976 of *LNCS*, pages 116–129, Kyoto, Japan, December 2000. Springer-Verlag.

[Bac] Adam Back. Hashcash — a denial of service counter-measure. `http://www.hashcash.org/`.

[Bau97] Friedrich Ludwig Bauer. *Decrypted secrets.* Springer-Verlag, 1997. ASIN: 3540604189.

[BB00] Mihir Bellare and Alexandra Boldyreva. The security of Chaffing and Winnowing. In T. Okamoto, editor, *Advances in Cryptology (Asiacrypt 2000)*, page 517, Kyoto, Japan, January 2000. Springer-Verlag.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology (Asiacrypt 2001)*, volume 2248 of *LNCS*, pages 566–582, Gold Coast, Australia, 9-13 December 2001. Springer-Verlag.

[Bea96] D. R. Beaver. Plausible deniability. In *1st International Conference on the Theory and Applications of Cryptology (Pragocrypt '96)*, pages 272–288, Prague, Czech Republic, September/October 1996. Czech Technical University Publishing House.

[BFK00] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 115–129. Springer-Verlag, July 2000.

[BG03] Krista Bennett and Christian Grothoff. GAP – practical anonymous networking. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 141–160. Springer-Verlag, March 2003.

[BGS01] Adam Back, Ian Goldberg, and Adam Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[Bha72] U. Narayan Bhat. *Elements of applied stochastic processes.* John Wiley & Sons, Inc., 1972.

[BL02] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies workshop (PET 2002)*, volume 2482 of *LNCS*, pages 110–128. Springer-Verlag, 2002.

[BM99] Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology (Crypto'99)*, volume 1666 of *LNCS*, pages 431–448, Berlin Germany, 15-19 August 1999. Springer-Verlag.

[BMS01] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Information Hiding workshop (IH 2001)*, volume 2137 of *LNCS*, pages 245–257. Springer-Verlag, April 2001.

[BPS00] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 30–45. Springer-Verlag, July 2000.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption — How to encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology (Eurocrypt '94)*, volume 950 of *LNCS*, pages 92–111. Springer-Verlag, 1995.

[BSG00] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[BY03] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In *Topics in Cryptology — CT-RSA 2003*, volume 2612 of *LNCS*, pages 1–18, San Francisco, CA, USA, 13-17 April 2003. Springer-Verlag.

[Cam99] Duncan Campbell. Development of surveillance technology and risk of abuse of economic information. Technical report, Scientific and Techonological Option Assessment unit (STOA), European Parliament, April 1999.

[CD02] Richard Clayton and George Danezis. Chaffinch: Confidentiality in the face of legal threats. In Fabien A. P. Petitcolas, editor, *Information Hiding workshop (IH 2002)*, volume 2578 of *LNCS*, pages 70–86, Noordwijkerhout, The Netherlands, 7-9 October 2002. Springer-Verlag.

[CDK01] Richard Clayton, George Danezis, and Markus G. Kuhn. Real world patterns of failure in anonymity systems. In Ira S. Moskowitz, editor, *Information Hiding workshop (IH 2001)*, volume 2137 of *LNCS*, pages 230–244, Pittsburgh, PA, USA, January 2001. Springer-Verlag.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In B. S. Kaliski, editor, *In Advances in*

Cryptology (Crypto'97), volume 1294 of *LNCS*, pages 90–104, Santa Barbara, CA, USA, 17-21 August 1997. Springer-Verlag.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[Cha83] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology (Crypto'82)*, pages 199–203, New York and London, 1983. Plenum Press.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology (Eurocrypt 2003)*, volume 2656 of *LNCS*, pages 255–271, Warsaw, Poland, 4-8 May 2003. Springer-Verlag.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, number 2009 in LNCS, pages 46–66, Berkeley, CA, USA, July 2000. Springer-Verlag.

[DA99] T. Dierks and C. Allen. Rfc 2246: The tls protocol version 1.0. `http://www.ietf.org/rfc/rfc2246.txt`, January 1999.

[Dan02] George Danezis. Forward secure mixes. In Jonsson Fisher-Hubner, editor, *Nordic workshop on Secure IT Systems (Norsec 2002)*, pages 195–207, Karlstad, Sweden, November 2002.

[Dan03a] George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 1–17, Dresden, Germany, March 2003. Springer-Verlag.

[Dan03b] George Danezis. Statistical disclosure attacks. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[DDM03a] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003.

[DDM03b] George Danezis, Roger Dingledine, and Nick Mathewson. Type III (Mixminion) mix protocol specifications. Technical report, The mixminion project `http://mixminion.net`, 2003.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. pages 542–552, 1991.

[DFHM01] Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. In Ira S. Moskowitz, editor, *Information Hiding workshop (IH 2001)*, volume 2137 of *LNCS*, pages 126–141. Springer-Verlag, 25-27 April 2001.

[DFK$^+$03] Yevgeniy Dodis, Matt Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. Intrusion-resilient public-key encryption. In M. Joye, editor, *Topics in Cryptology - CT-RSA 2003*, volume 2612 of *LNCS*, pages 19–32, San Francisco, CA, USA, 13-17 April 2003. Springer-Verlag.

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 67–95, Berkeley, CA, USA, July 2000. Springer-Verlag.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DK00] Yvo Desmedt and Kaoru Kurosawa. How to break a practical mix and design a new one. In Bart Preneel, editor, *Advances in Cryptology (Eurocrypt 2000)*, volume 1807 of *LNCS*, pages 557–572, Bruges, Belgium, 14-18 May 2000. Springer-Verlag.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *Advances in Cryptology (Eurocrypt 2002)*, volume 2332 of *LNCS*, pages 65–82, Amsterdam, The Netherlands, 28 April–2 May 2002. Springer-Verlag.

[DS02] Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, *Financial Cryptography (FC '02)*, volume 2357 of *LNCS*. Springer-Verlag, March 2002.

[DS03a] George Danezis and Len Sassaman. Heartbeat traffic to counter $(n-1)$ attacks. In *workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, November 2003.

[DS03b] Claudia Diaz and Andrei Serjantov. Generalising mixes. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 18–31, Dresden, Germany, March 2003. Springer-Verlag.

[DSCP02] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies workshop (PET 2002)*, volume 2482 of *LNCS*, pages 54–68, San Francisco, CA, USA, 14-15 April 2002. Springer-Verlag.

[El 85] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.

[EU00] Legal aspects of information society services, in particular electronic commerce, in the internal market. Directive 2000/31/EC of the European Parliament and of the Council. Official Journal of the European Communities, July 2000.

[FBH+02] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In Dan Boneh, editor, *USENIX Security Symposium*, pages 247–262, San Francisco, CA, 5-9 August 2002.

[FBW+03] Nick Feamster, Magdalena Balazinska, Winston Wang, Hari Balakrishnan, and David Karger. Thwarting web censorship with untrusted messenger discovery. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 125–140, Dresden, Germany, March 2003. Springer-Verlag.

[FM02] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206, Washington, DC, November 2002. ACM.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *Advances in Cryptology (Crypto 2001)*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, CA, USA, 19-23 August 2001. Springer-Verlag.

[FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. John Wiley & Sons, 2003. ISBN: 0471223573,.

[FSCM02] Michael J. Freedman, Emil Sit, Josh Cates, and Robert Morris. Introducing tarzan, a peer-to-peer anonymizing network layer. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *International workshop on Peer-to-Peer Systems (IPTPS)*, volume 2429 of *LNCS*, pages 121–129, Cambridge, MA, March 2002. Springer-Verlag.

[GFX+01] Yong Guan, Xinwen Fu, Dong Xuan, P. U. Shenoy, Riccardo Bettati, and Wei Zhao. Netcamo: camouflaging network traffic for qos-guaranteed mission critical applications. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A 31(4):253–265, 2001.

[Gil93] David Gillman. A chernoff bound for random walks on expander graphs. In *34th Annual Symposium on Foundations of Computer Science*, pages 680–691, Palo Alto, California, 3-5 November 1993. IEEE.

[GJJS04] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's track*, San Francisco, USA, February 2004.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[Gol00] Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, December 2000.

[Gol01] Oded Goldreich. *Foundations of cryptography*. Cambridge University press, 2001. ISBN: 0521791723.

[GRPS03] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In Ross J. Anderson, editor, *Information Hiding*, volume 1174 of *LNCS*, pages 137–150, Cambridge, U.K., 1996. Springer-Verlag.

[GRS99] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[GT96] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium — NDSS '96*, pages 2–16, San Diego, California, February 1996. IEEE.

[GZB$^+$02] Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In Yuliang Zheng, editor, *Advances in Cryptology (Asiacrypt 2002)*, volume 2501 of *LNCS*, pages 451–465, Queenstown, New Zealand, 1-5 December 2002. Springer-Verlag.

[Hel96a] Johan Helsingius. Johan helsingius closes his internet remailer. `http://www.penet.fi/press-english.html`, August 1996.

[Hel96b] Johan Helsingius. Johan helsingius gets injunction in scientology case privacy protection of anonymous messages still unclear. `http://www.penet.fi/injunc.html`, September 1996.

[Hel96c] Johan Helsingius. Temporary injunction in the anonymous remailer case. `http://www.penet.fi/injuncl.html`, September 1996.

[Her96] Michael Herman. *Intelligence Power in Peace and War.* Cambridge University Press, November 1996. ISBN: 0521566363.

[HW02] Steven Hazel and Brandon Wiley. Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *Peer-to-Peer Systems, First International workshop, IPTPS 2002*, volume 2429 of *LNCS*, Cambridge, MA, USA, 7-8 March 2002. Springer-Verlag.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *Advances in Cryptology (Crypto 2001)*, volume 2139 of *LNCS*, pages 332–354, Santa Barbara, California, USA, 19-23 August 2001. Springer-Verlag.

[Itk03] Gene Itkis. Cryptographic tamper evidence. In *ACM conference on Computer and communication security (CCS 2003)*, pages 355–364. ACM Press, 2003.

[Jak98] Markus Jakobsson. A practical mix. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *LNCS*, pages 448–461, Espoo, Finland, 31 May – 4 June 1998. Springer-Verlag.

[Jak99] Markus Jakobsson. Flash Mixing. In *Principles of Distributed Computing - PODC '99*. ACM Press, 1999.

[JJ01] Markus Jakobsson and Ari Juels. An optimally robust hybrid mix network. In *Principles of Distributed Computing (PODC 2001)*, pages 284–292, Newport, Rhode Island, USA, 26-29 August 2001. ACM.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In Dan Boneh, editor, *USENIX Security Symposium*, pages 339–353, San Francisco, CA, USA, 5-9 August 2002. USENIX.

[JMP$^+$98] Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4):495–509, 1998.

[JSY99] Markus Jakobsson, Julien P. Stern, and Moti Yung. Scramble all, encrypt small. In Lars R. Knudsen, editor, *Fast Software Encryption (FSE '99)*, volume 1636 of *LNCS*, pages 95–111, Rome, Italy, 24-26 March 1999. Springer-Verlag.

[JVZ00] Shu Jiang, Nitin H. Vaidya, and Wei Zhao. Routing in packet radio networks to prevent traffic analsyis. In *IEEE Information Assurance and Security Workshop*, June 2000.

[KAP02] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien A. P. Petitcolas, editor, *Information Hiding workshop (IH 2002)*, volume 2578 of *LNCS*, pages 53–69, Noordwijkerhout, The Netherlands, 7-9 October 2002. Springer-Verlag.

[KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-Go MIXes: Providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding workshop (IH 1998)*, volume 1525 of *LNCS*, pages 83–98, Portland, Oregon, USA, 14-17 April 1998. Springer-Verlag.

[KS95] Joe Kilian and Kazue Sako. Receipt-free MIX-type voting scheme — a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology (Eurocrypt 1995)*, volume 921 of *LNCS*, pages 393–403, Saint-Malo, France, 21-25 May 1995. Springer-Verlag.

[Küg03] Dennis Kügler. An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 161–176, Dresden, Germany, March 2003. Springer-Verlag.

[LA99] Steve Lloyd and Carlisle Adams. *Understanding the Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations.* Que, 1999. ISBN: 157870166X.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[LW03] Nati Linial and Avi Wigderson. Expander graphs and their applications. Collection of Lecture Notes `http://www.math.ias.edu/~avi/TALKS/expander_course.pdf`, January 2003.

[MCPS03] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003.

[MH96] Markus Michels and Patrick Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme.

In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology (Asiacrypt '96)*, volume 1163 of *LNCS*, pages 125–132, Kyongju, Korea, 3-7 November 1996. Springer-Verlag.

[MK98] David Mazières and M. Frans Kaashoek. The Design, Implementation and Operation of an Email Pseudonym Server. In *ACM Conference on Computer and Communications Security (CCS'98)*, pages 27–36, San Francisco, CA, USA, 3-5 November 1998. ACM Press.

[MK99] Andrew D. McDonald and Markus G. Kuhn. StegFS: A steganographic file system for linux. In Andreas Pfitzmann, editor, *Information Hiding (IH '99)*, volume 1768 of *LNCS*, pages 462–477, Dresden, Germany, 29 September – 1 October 1999. Springer-Verlag.

[MK00] Masashi Mitomo and Kaoru Kurosawa. Attack for flash MIX. In Tatsuaki Okamoto, editor, *Advances in Cryptology (Asiacrypt 2000)*, volume 1976 of *LNCS*, pages 192–204, Kyoto, Japan, 3-7 December 2000. Springer-Verlag.

[MNCM03] Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In *workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.

[MNS] Ira S. Moskowitz, Richard E. Newman, and Paul F. Syverson. Quasi-anonymous channels. In *Communication, Network, and Information Security (CNIS 2003)*, New York, USA, 10-12 December.

[Moc] Christian Mock. Mixmaster Statistics (Austria).
`http://www.tahina.priv.at/~cm/stats/mlist2.html`.

[Möl03] Bodo Möller. Provably secure public-key encryption for length-preserving chaumian mixes. In Marc Joye, editor, *Topics in Cryptology CT-RSA 2003*, volume 2612 of *LNCS*, pages 244–262, San Francisco, CA, USA, 13-17 April 2003. Springer-Verlag.

[Mor] Pat Morin. A critique of BEAR and LION. Manuscript, `citeseer.nj.nec.com/124166.html`.

[MOV96] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996. ISBN: 0-8493-8523-7.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge Univerisity Press, 1995. ISBN: 0521474655.

[MS02] David Martin and Andrew Schulman. Deanonymizing users of the safeweb anonymizing service. Technical Report 2002-003, Boston University Computer Science Department, February 2002.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Pierangela Samarati, editor, *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 116–125. ACM Press, November 2001.

[New97] Ron Newman. The Church of Scientology vs. `anon.penet.fi`, March 1997. `http://www.xs4all.nl/~kspaink/cos/rnewman/anon/penet.html`.

[NMSS03] Richard E. Newman, Ira S. Moskowitz, Paul Syverson, and Andrei Serjantov. Metrics for traffic analysis prevention. In *Privacy Enhancing Technologies Workshop*, Dresden, Gernamy, March 2003.

[OA00] Miyako Ohkubo and Masayuki Abe. A Length-Invariant Hybrid MIX. In Tatsuaki Okamoto, editor, *Advances in Cryptology (Asiacrypt 2000)*, volume 1976 of *LNCS*, pages 178–191, Kyoto, Japan, 3-7 December 2000. Springer-Verlag.

[Odl03] Andrew M. Odlyzko. Privacy, economics, and price discrimination on the internet. In N. Sadeh, editor, *International Conference on Electronic Commerce (ICEC 2003)*, pages 355–366. ACM Press, 2003.

[OKST97] Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault tolerant anonymous channel. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communication Security, First International Conference (ICICS'97)*, volume 1334 of *LNCS*, pages 440–444, Beijing, China, 11-14 November 1997. Springer-Verlag.

[Pal] Peter Palfrader. Echolot: a pinger for anonymous remailers. `http://www.palfrader.org/echolot/`.

[Par96] Sameer Parekh. Prospects for remailers: where is anonymity heading on the internet? *First Monday*, 1(2), August 5 1996. On-line journal `http://www.firstmonday.dk/issues/issue2/remailers/`.

[Pfi94] Birgit Pfitzmann. Breaking efficient anonymous channel. In Alfredo De Santis, editor, *Advances in Cryptology (Eurocrypt '94)*, volume 950 of *LNCS*, pages 332–340, Perugia, Italy, 9-12 May 1994. Springer-Verlag.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. volume 765 of *LNCS*, pages 248–259, Lofthus, Norway, 1993. Springer-Verlag.

[Pin73] M. S. Pinsker. On the complexity of a concentrator. In *Proceedings of the 7th International Teletraffic Conference*, pages 318/1–318/4, Stockholm, 1973.

[PK00] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.14, July 2000.

[PKC02] PKCS #1 v2.1: RSA Cryptography Standard. RSA Security Inc., June 2002.

[PP90] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct RSA-implementation of MIXes. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology (Eurocrypt '89)*, volume 434 of *LNCS*, pages 373–381, Houthalen, Belgium, 10-13 April 1990. Springer-Verlag.

[PPW91] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In Wolfgang Effelsberg, Hans Werner Meuer, and Günter Müller, editors, *GI/ITG Conference on Communication in Distributed Systems*, volume 267 of *Informatik-Fachberichte*, pages 451–463. Springer-Verlag, February 1991.

[Ray00] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 10–29. Springer-Verlag, July 2000.

[RIP00] Regulation of Investigatory Powers Act 2000. The Stationery Office Limited, 2000. ISBN 0-10-542300-9.

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *Fast Software Encryption (FSE '97)*, volume 1267 of *LNCS*, pages 210–218, Haifa, Israel, 20-22 January 1997. Springer-Verlag.

[Riv98] Ronald L. Rivest. Chaffing and winnowing: Confidentiality without encryption. CryptoBytes (RSA Laboratories), 4(1):12–17, Summer 1998.

[Roe97] Michael Roe. *Cryptography and Evidence.* PhD thesis, University of Cambridge, Computer Laboratory, 1997.

[RP02] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[RR98] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[RR00] Josyula R. Rao and Pankaj Rohatgi. Can pseudonymity really guarantee privacy? In *Proceedings of the 9th USENIX Security Symposium*, pages 85–96. USENIX, August 2000.

[RS93] Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *ACM Symposium on Theory of Computing (STOC'93)*, pages 672–681. ACM, 1993.

[RSA78] Ron L. Rivest, A. Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[RSG98] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[RVW00] Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Symposium on Foundations of Computer Science (FOCS 2000)*, pages 3–13, Redondo Beach, California, USA, 12-14 November 2000. IEEE Computer Society.

[SBS02] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy*, page 58, Berkeley, California, USA, 12-15 May 2002. IEEE Computer Society.

[Sch96] Bruce Schneier. *Applied Cryptography.* Wiley, 1996. ISBN 0-471-11709-9.

[SD02] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and

Paul Syverson, editors, *Privacy Enhancing Technologies workshop (PET 2002)*, volume 2482 of *LNCS*, pages 41–53, San Francisco, CA, USA, 14-15 April 2002. Springer-Verlag.

[SDS02] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien A. P. Petitcolas, editor, *Information Hiding workshop (IH 2002)*, volume 2578 of *LNCS*, pages 36–52, Noordwijkerhout, The Netherlands, 7-9 October 2002. Springer-Verlag.

[Ser02] Andrei Serjantov. Anonymizing censorship resistant systems. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *International Peer-To-Peer Systems workshop (IPTPS 2002)*, volume 2429 of *LNCS*, pages 111–120, Cambridge, MA, USA, 7-8 March 2002. Springer-Verlag.

[Ser04] Andrei Serjantov. *On the anonymity of anonymity systems.* PhD thesis, University of Cambridge, 2004.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[SHA93] FIPS PUB 180-1: Secure Hash Standard (SHS), 1993. National Institute of Standards and Technology.

[Shm02] Vitaly Shmatikov. Probabilistic analysis of anonymity. In *Computer Security Foundations workshop (CSFW-15 2002)*, pages 119–128, Cape Breton, Nova Scotia, Canada, 24-26 June 2002. IEEE Computer Society.

[Sin01] Michael Singer. CIA Funded SafeWeb Shuts Down. `http://siliconvalley.internet.com/news/article.php/3531_926921`, November 20 2001.

[SK96] Bruce Schneier and John Kelsey. Unbalanced feistel networks and block cipher design. In Dieter Gollmann, editor, *Fast Software Encryption, Third International workshop*, volume 1039 of *LNCS*, pages 121–144, Cambridge, UK, 21-23 February 1996. Springer-Verlag.

[SK03] Sandra Steinbrecher and Stefan Köpsell. Modelling unlinkability. In Roger Dingledine, editor, *Privacy Enhancing Technologies workshop (PET 2003)*, volume 2760 of *LNCS*, pages 32–47, Dresden, Germany, March 2003. Springer-Verlag.

[Smi03] Tony Smith. European RIAA-style anti-file swap lawsuits 'inevitable'. The Register `http://www.theregister.co.uk/content/6/34547.html`, December 2003.

[SMK$^+$01] Ion Stoica, Robert Morris, David Karger, M. Francs Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Conference on applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM 2001)*, pages 149–160, San Diego, CA, USA, 27-31 August 2001. ACM Press.

[SN03] Andrei Serjantov and Richard E. Newman. On the anonymity of timed pool mixes. In *workshop on Privacy and Anonymity Issues in Networked and Distributed Systems*, pages 427–434, Athens, Greece, May 2003. Kluwer.

[SS03] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *European Symposium on Research in Computer Security (ESORICS 2003)*, Gjovik, Norway, 13-15 October 2003.

[SSW$^+$02] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkat Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy*, pages 19–30, Berkeley, California, USA, 12-15 May 2002. IEEE Computer Society.

[STRL00] Paul F. Syverson, Gene Tsudik, Michael G. Reed, and Carl E. Landwehr. Towards an analysis of onion routing security. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 96–114, Berkeley, CA, USA, 25-26 July 2000. Springer-Verlag.

[Tim97] Brenda Timmerman. A security model for dynamic adaptive traffic masking. In *New Security Paradigms Workshop*, pages 107–115, Langdale, Cumbria, UK, 1997. ACM.

[Tim99] Brenda Timmerman. Secure dynamic adaptive traffic masking. In *New Security Paradigms Workshop*, pages 13–24, Ontario, Canada, September 1999. ACM.

[VNW94] B. R. Venkatraman and Richard E. Newman-Wolfe. Performance analysis of a method for high level prevention of traffic analysis using measurements from a campus network. In *Proceeding of the IEEE/ACM Tenth Annual Computer Security Applications Conference*, pages 288–297, Orlando, FL, December 5-9 1994. IEEE CS Press.

[WALS02] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Network and Distributed Security Symposium (NDSS '02)*, San Diego, California, 6-8 February 2002.

[WALS03] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *IEEE Symposium on Security and Privacy*, page 28, Berkeley, CA, USA, 11-14 May 2003. IEEE Computer Society.

[Wei74] Robert Weinstock. *Calculus of variations*. Dover publications, 1974. ISBN: 0486630692.

[Wik02] Douglas Wikström. How to break, fix, and optimize "optimistic mix for exit-polls". Technical Report T2002-24, Swedish Institute of Computer Science, SICS, Box 1263, SE-164 29 Kista, SWEDEN, 2002.

[Wik03a] Douglas Wikström. Elements in $Z_p^* \backslash G_q$ are dangerous. Technical Report T2003-05, Swedish Institute of Computer Science, SICS, Box 1263, SE-164 29 Kista, SWEDEN, 2003.

[Wik03b] Douglas Wikström. Four practical attacks for "optimistic mixing for exit-polls". Technical Report T2003-04, Swedish Institute of Computer Science, SICS, Box 1263, SE-164 29 Kista, SWEDEN, 2003.

[WP89] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco — underconditional sender and recipient untraceability with computationally secure serviceability. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology (Eurocrypt '89)*, volume 434 of *LNCS*, page 690, Houthalen, Belgium, 10-13 April 1989. Springer-Verlag.

[WW00] Zhen Wang and Peter Willett. A performance study of some transient detectors. *IEEE transactions on signal processing*, 48(9):2682–2685, September 2000.

[YM] John Young and Erich M. On obtaining "lawful interception" documents. `http://www.quintessenz.org/etsi`.

[Zim95] Philip Zimmermann. *PGP Source Code and Internals.* The MIT Press, 1995. ISBN 0-262-24039-4.

Production notes

This thesis was written in LaTeX and its bibliography was compiled using BibTeX. The source files have been stored in the Concurrent Versions Control (CVS) system, which allows for a full history of the write-up process to be extracted. The `make` utility was used to run the compilation process and to manage the production of this document. Over time "debugging" functionality was added to the scrips, using `grep` and `perl`, to check for repeated words and inconsistent hyphens as well as inconsistencies in the bibliography. You are holding in your hands thesis version 1.67 (bibiography version 1.26).