Characterising Anonymity Systems

Joss Wright

This thesis is submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy.

The University of York
Heslington
York
YO10 5DD

Department of Computer Science

November 2007

For Emily

Abstract

Privacy is a value shared by most human societies. The work presented here is inspired by this value and is concerned with methods by which it may be achieved. In a world where we increasingly make use of information systems amenable to surveillance, privacy is no longer an inherent assumption; it has becomes a property that must be explicitly designed.

In this thesis we examine the background and motivation for privacy and how this goal may be achieved by use of systems that provide *anonymity*. We examine the underlying features of such systems, the variety of strategies that may be employed to achieve this aim, and the limitations of these methods.

We employ a definition of anonymity based on various applications of random choice to introduce unpredictability into the sequences of observable events created by the exchange of messages between actors in communicating systems. This leads to a characterisation of anonymity systems according to the fundamental mechanism that they employ to maximise this unpredictability.

The characterisation that we propose leads us to identify four fundamental anonymity strategies, corresponding to known mechanisms that introduce randomness in communicating processes. These strategies form a classification applicable to all anonymity systems, which allows us to consider in isolation the separate strategies for achieving anonymity. Taking this approach we show that each fundamental strategy is individually sufficient to provide anonymity to communicating entities.

We analyse the anonymity strategies identified in the model through a simulation-based approach, and employ an information theoretic quantification to compare the anonymity provided by each type of system. The fundamental strategies are simulated both individually and as part of larger networks, and are compared with respect to the effectiveness of each approach in confusing an observer's ability to link communicating actors. Finally, we demonstrate that combining strategies in a single system can improve anonymity beyond that of individual strategies.

Our results show the relative effectiveness of a range of anonymity systems

at their most fundamental level, and make use of a quantification method that is applicable to any anonymity system based on the communication of messages between actors.

Contents

I Introduction 17

1 Introduction 19
1.1 Privacy . . . 20
1.2 Anonymity . . . 23
1.2.1 Names and Identity . . . 23
1.2.2 Etymology . . . 23
1.2.3 Pseudonymity . . . 26
1.2.4 Applications . . . 27
1.3 Summary . . . 30

2 Basic Concepts 31
2.1 Actors . . . 31
2.2 Structure . . . 32
2.3 Anonymity Definitions . . . 33
2.4 Identifiability . . . 33
2.4.1 Cardinality . . . 34
2.4.2 Certainty . . . 34
2.4.3 Sufficient Anonymity . . . 35
2.5 Characteristics . . . 35
2.5.1 Anonymised Party . . . 35
2.5.2 Connection . . . 36
2.5.3 Distribution of data . . . 36
2.5.4 Participant Relations . . . 37
2.5.5 Revocability . . . 37
2.6 Attackers . . . 38
2.6.1 Internal/External . . . 38
2.6.2 Static/Adaptive . . . 38
2.6.3 Active/Passive . . . 39
2.6.4 Local/Global . . . 39
2.7 Summary . . . 39

3 Anonymity Systems **41**
3.1 Mixes . . . 41
3.1.1 Basic properties . . . 42
3.1.2 Threshold Mixes . . . 43
3.1.3 Timed Mixes . . . 44
3.1.4 Continuous Mixes . . . 44
3.1.5 Mix Networks and Mix Cascades . . . 45
3.1.6 Mix Networks . . . 45
3.1.7 Mix Cascades . . . 46
3.1.8 Hybrid Approaches . . . 46
3.1.9 Synchronous and Asynchronous Batching . . . 47
3.2 Attacks on mix systems . . . 47
3.2.1 Passive Attacks . . . 47
3.2.2 Active Attacks . . . 48
3.2.3 Denial of Service Attacks . . . 49
3.3 Dummy Traffic . . . 49
3.4 Other Technologies . . . 50
3.4.1 Dining Cryptographer Networks . . . 50
3.4.2 Onion Routing . . . 51
3.4.3 Crowds . . . 52
3.5 Deployed Anonymity Systems . . . 53
3.5.1 Remailers . . . 53
3.5.2 Freenet . . . 55
3.5.3 JAP . . . 55
3.5.4 PipeNet . . . 56
3.5.5 Tor . . . 56
3.6 Summary . . . 57

II Simulation **59**

4 Anonymity Mechanisms **61**
4.1 Introduction . . . 61
4.2 Hiding . . . 61
4.3 Spurious Events . . . 62
4.4 Pull Technologies . . . 63
4.5 Internal Randomness . . . 64
4.6 Conclusion . . . 65

5 Analysis Methods **67**
5.1 Introduction . . . 67
5.2 Shannon Entropy . . . 68
5.2.1 Examples . . . 68
5.2.2 Conditional Entropy . . . 70
5.2.3 Experimental Factors . . . 72
5.3 Building the probability distribution . . . 75
5.3.1 Example . . . 75
5.4 Result Significance Testing . . . 79
5.4.1 Null hypothesis testing . . . 79
5.4.2 Statistical tests for homogeneity . . . 80
5.4.3 Details of the Kolmogorov-Smirnov Test . . . 82
5.4.4 Example Application . . . 85
5.5 Summary . . . 88

6 Simulation **91**
6.1 Introduction . . . 91
6.2 Quantification Approach . . . 91
6.3 Purpose . . . 92
6.4 Design . . . 93
6.5 Engine . . . 94
6.5.1 Network Scheduling and Routing . . . 94
6.5.2 Input format . . . 95
6.5.3 Output format . . . 96
6.6 Implementation Details . . . 97
6.6.1 Language choices . . . 100
6.6.2 Environment . . . 100
6.6.3 Sender . . . 101
6.6.4 Receiver . . . 104
6.6.5 Nodes . . . 105
6.7 Processing of Results . . . 110
6.7.1 Choice of pairings . . . 111
6.7.2 Basic Operation . . . 111
6.7.3 Specific considerations for the Drop strategy . . . 112
6.8 Experimental Setup . . . 113
6.9 Summary . . . 114

7 Analysis of Simulation Results **117**
7.1 Introduction . . . 117
7.2 Control Experiments . . . 118
7.2.1 Multiple Hops . . . 119

7.2.2 Entropy of output pairings . . . 123
7.3 Mix . . . 125
7.4 Flood . . . 128
7.5 Hide . . . 131
7.6 Drop . . . 135
7.7 Crowds . . . 136
7.8 Combined Systems . . . 138
7.9 Mix and Flood . . . 139
7.10 Crowds and Flood . . . 139
7.10.1 Enforce hops . . . 141
7.10.2 Isolate Crowds nodes . . . 142
7.10.3 No enforcement of path . . . 142
7.10.4 Results . . . 143
7.11 Crowds and Mix . . . 143
7.12 Summary . . . 145
7.12.1 Multiple Nodes . . . 148

III Conclusions 151

8 Conclusions 153
8.1 Overview . . . 153
8.2 Model . . . 153
8.3 Methods of Providing Anonymity . . . 154
8.4 Simulation . . . 155
8.4.1 Individual Strategies . . . 155
8.4.2 Network Effects . . . 156
8.4.3 Combination of Strategies . . . 156
8.5 Future Work . . . 157
8.5.1 Formal Modelling . . . 157
8.5.2 Data Independence . . . 157
8.5.3 Attacker Models . . . 157
8.5.4 Improved Quantification . . . 158
8.5.5 Detailed Modelling . . . 159
8.6 Summary . . . 159

IV Appendices 161

A Results 163
A.1 Introduction . . . 163

A.2 Single Strategy Networks 163
A.2.1 Mix 163
A.2.2 Flood 164
A.2.3 Hide 164
A.2.4 Drop 164
A.2.5 Four Nodes 165
A.2.6 Four Nodes – All Hops 167
A.2.7 Four Nodes – All Hops 169
A.2.8 Four Nodes – All Hops 170
A.2.9 Four Nodes – All Hops 172
A.2.10 Crowds-style 172
A.3 Mixed Strategy Networks 173
A.3.1 Mix with Flood 173
A.3.2 Crowds with Flood 173
A.3.3 Crowds with Mix 174
A.4 Statistical Significance Tests 174
A.4.1 Single Mix 175
A.4.2 Four Mixes, 1 Hop 175
A.4.3 Four Mixes, 2 Hop 176
A.4.4 Four Mixes, 3 Hop 176
A.4.5 Four Mixes, 4 Hop 177
A.4.6 Single Flood 177
A.4.7 Four Flood, 1 Hop 177
A.4.8 Four Flood, 2 Hop 178
A.4.9 Four Flood, 3 Hop 178
A.4.10 Four Flood, 4 Hop 178
A.4.11 Single Hide 179
A.4.12 Four Hide, 1 Hop 179
A.4.13 Four Hide, 2 Hop 179
A.4.14 Four Hide, 3 Hop 180
A.4.15 Four Hide, 4 Hop 180
A.4.16 Four Crowds 180
A.4.17 Mix / Flood 180
A.4.18 Crowds / Flood 181
A.4.19 Crowds / Mix 181

Bibliography **182**

List of Figures

5.1 Output distances between perfectly mixed messages. 73
5.2 Output pairings in a perfect mixing system (left) and output pairings of all consecutive pairs in a perfect system (right). . . 74
5.3 Example results from a small simulation. 77
5.4 Example calculated distances between messages. 78
5.5 Example tally of output distances for a specific set of input messages. 78
5.6 Example output graph from a set of simulations. 79
5.7 D values large enough to reject the null hypothesis at the given level of significance. n_1, n_2 refer to the number of observations in each sample. Values in this table taken from (Siegel, 1956). 84
5.8 Comparison of result sets from individual experiment runs. Both show a single mix with flushing threshold 16. 86
5.9 Example output of Kolmogorov-Smirnov test implementation . 87
5.10 Comparison of result sets from different experiments. Top: a single mix with threshold 16. Bottom: a single mix with threshold 12. 89
5.11 Results of the Kolmogorov-Smirnov test for experiments with differing parameters. 90

6.1 Example network structure. 93
6.2 Flow of experiment processing. 93
6.3 Simulator Actor Class Hierarchy 95
6.4 EBNF grammar for the simulation engine input file. 96
6.5 Sample simulator input file 97
6.6 Valid actor types in simulation engine 98
6.7 Sample output format from simulation engine with dummy message highlighted. 99
6.8 Example of input and output sequences for a simple simulation run. Target messages are highlighted. 112

6.9 Example of input and output sequences for a simulation run with dropped messages. Target messages are highlighted. . . . 113
6.10 Single-strategy experimental parameters 114
6.11 Combined-strategy experimental parameters 114

7.1 Distribution of spread in output for simple buffers. Number of buffers ranges from a single buffer (top) to four buffers (bottom).120
7.2 Spread in output sequence caused by buffers with multiple hops. Number of buffers in the network ranges from 2 (top) to 4 (bottom). 121
7.3 Frequency of spread in output sequence caused by a four-buffer network. Number of hops per message ranges from 1 hop (top) to 4 hops (bottom). 122
7.4 Entropy of the output distribution for single hop messages as the network size increases from 1 to 4 buffers. 124
7.5 Entropy of the output pairing distribution for a four buffer network as the message paths increase from one to four hops. . 124
7.6 Output distribution for a single MIX node as threshold value (t) varies. 126
7.7 Entropy change for a single MIX node as the threshold varies. 127
7.8 Entropy of MIX node networks as flushing threshold varies. . . 128
7.9 Output of single FLOOD node as percentage of flood messages (f) varies. 129
7.10 Entropy change for a single FLOOD node as the flooding percentage varies. 130
7.11 Output distribution for four FLOOD nodes as the number of message hops (h) varies with flooding rate $f = 20\%$. 131
7.12 Entropy of FLOOD nodes as flooding percentage varies. 132
7.13 Spread of output messages for a single HIDE node as number of hidden senders (h) varies. 133
7.14 Entropy change for a single-node HIDE system as the number of hidden senders varies. 134
7.15 Overlaid entropies of four node HIDE systems with increasing numbers of hidden senders. The base entropy of a buffer network is shown for comparison. 135
7.16 Single DROP node with increasing probability of dropping a message. 136
7.17 Spread of output for four Crowds-style nodes as the percentage probability of messages being delivered rather than forwarded (d) varies. 137

7.18 Entropy of four-node crowds-style network as delivery percentage varies. 138
7.19 Spread of output for networks of FLOOD and MIX nodes as the number of each node type varies. 140
7.20 Entropy of networks of MIX and FLOOD nodes. 141
7.21 Spread of output for networks of Crowds and FLOOD nodes as the number of each node type varies. 144
7.22 Entropy of systems comprised of varying numbers of Crowds and FLOOD nodes. 145
7.23 Spread of messages in systems comprised of differing numbers of Crowds and MIX nodes. 146
7.24 Entropy of systems comprised of varying numbers of Crowds and MIX nodes, compared against entropies of pure MIX and pure Crowds systems. 147

Acknowledgements

I am thankful to so many people for their support in the time that I have worked on this thesis.

I would firstly like to thank deeply Professor Susan Stepney for guiding me through this work. Aside from being a constant inspiration, and from having taught me so much about such a range of topics, she has been everything that I could have hoped for in a supervisor. I will genuinely miss the long conversations that were never restricted to the subject at hand, but always left me knowing what I needed to know.

Secondly, I would like to thank my parents for their support on so many levels. They have always trusted in what I was doing, and have always smiled through incomprehensible explanations of what I was working on before asking when I would be finished. More than I can say, I want them to know that it was worth every minute.

I would also like to thank those researchers who have been, and continue to be, both friends and inspirations to me. I would especially like to thank Claudia Diaz, George Danezis and Nick Mathewson for endlessly fascinating conversations and for their highly valued friendship.

Finally, I cannot begin to express my thanks for the love and support that my wife, Emily, has shown me throughout this work. I am constantly amazed by her trust and joy in people, and her belief in the importance of always working for the benefit of others. Underlying everything else, it is this belief that has inspired the work presented here. For that, as for so much else, I can never thank her enough.

Part I

Introduction

Chapter 1

Introduction

"Nec vixit male qui natus moriensque fefellit."
(Not a bad life is that of one who is born and dies in obscurity)[1]

– Horace, *Epistles, I, 17, 10* (c.20 BCE)

To avoid discrimination against, or abuse of, users of communicating systems it is sometimes necessary to prevent their activities from being made public. This may often be achieved by the use of protocols that provide *secrecy* of transaction data, however in a number of important cases it is necessary for the identities of the communicating parties to be obscured.

In certain situations the use of a particular system may be incriminating. Certain sources of information may be censored or viewed unfavourably by observing parties. Other technologies with both legitimate and illegitimate uses, while legal in themselves, may be regarded as suspicious. In such cases, legitimate users may desire privacy to prevent assumptions of guilt by third parties.

It is also the case that users may not wish their participation in certain activities to be made known. In the modern world, public discussions are frequently archived indefinitely and made widely accessible. Users taking part in such discussions may wish to avoid having their participation known in order to avoid associating themselves with particular viewpoints or with an interest in certain topics.

In such cases, a useful and effective method for protection of the user is to sever the link between a user's identity and their observable behaviour in the system—to make the user anonymous.

[1]This quote, often attributed to Cicero, is normally translated as "Nor has he spent his life badly who has passed it in privacy." The more literal translation shown here is our own.

A variety of methods to achieve the goal of anonymity have been proposed and implemented. Public awareness of the growing level of surveillance under which we live is increasing, and approaches aimed at regaining the privacy that has previously been an assumed part of life are beginning to emerge. Despite this, the level of rigorous design and verification work in these systems has been low compared to that of equivalent approaches towards security in general. As a result many deployed systems have been shown to contain serious flaws, and the level of anonymity that they provide is imperfectly understood.

In this thesis we examine anonymity as a property of systems, and explore the methods that can be used to achieve this property, We identify similarities between fundamental approaches to providing anonymity, allowing us to classify and analyse anonymity systems. These analyses provide insight into the design of future systems that support the anonymity, and consequently the privacy, of those individuals that use these systems.

1.1 Privacy

The desire for privacy motivates much research into anonymity systems. Anonymity provides a mechanism by which an individual may render their actions free from observation, and thus protect the privacy of their actions.

The notion of privacy has been discussed at least as far back as Aristotle (384–327 BCE), who defined the separate spheres of public life: 'πολις' (*polis*, city) and private life 'οικος' (*oikos*, home). The importance of these concepts is illustrated by the derivation from them of the English words *politics* and *economics*. Undoubtedly, Aristotle was not the first to elucidate these ideas.

In Britain, privacy has arguably been part of the law since 1361 with the adoption of the Justices of the Peace Act under the reign of Edward III. This act included punishments for "peeping toms" and eavesdroppers, themselves derogatory terms that reflect the already negative view held by society towards those who invade the privacy of others.

Despite this early recognition of personal privacy, privacy as an independent right has been explicitly recognised only in the past 150 years. One of the first extant definitions was made by the United States Supreme Court Justice Louis Brandeis and lawyer Samuel Warren (Brandeis and Warren, 1890), who examined the right to privacy as a natural extension of the individual right to liberty. They stated that "liberty" as a right had initially been enforced with respect to preventing physical assault, but that as newer business models and media coverage started to have significant effects in society, intrusion into private lives for public consumption became of concern to many, and the ideal of liberty was necessarily extended to include unfair intervention into aspects of a person's life that could be embarrassing or dangerous if publicised:

> "Gossip is no longer the resource of the idle and of the vicious, but has become a trade, which is pursued with industry as well as effrontery... The intensity and complexity of life, attendant upon advancing civilisation, have rendered necessary some retreat from the world, and man, under the refining influence of culture, has become more sensitive to publicity, so that solitude and privacy have become more essential to the individual; but modern enterprise and invention have, through invasions upon his privacy, subjected him to mental pain and distress, far greater than could be inflicted by mere bodily injury."
>
> – (Brandeis and Warren, 1890)

In reference to earlier work by a Michigan Supreme Court Justice (Cooley, 1888), Brandeis and Warren define privacy as "the right to be let alone". This concept is still fundamental to almost all legal definitions of personal privacy.

Real interest in privacy, however, appears to have begun only in the second half of the twentieth century. The United Nations Universal Declaration of Human Rights embodies the right to privacy in its twelfth article:

> "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks."
>
> – (United Nations, 1948)

Similarly, the International Covenant on Civil and Political Rights (ICCPR, 1997), a section of the International Bill of Rights adopted to expand on and clarify the Universal Declaration of Human Rights, reiterates the fundamental nature of privacy as a right to which humans are entitled.

Both the European Union and the United States have measures to protect privacy (European Union, 1998; United States Department of Commerce, 2004), however these rights are still emerging and in a state of constant alteration.

In British law, the first explicit electronic privacy legislation was the 1984 Data Protection Act (HMSO, 1984). This act, superseded by the 1998 Act of the same name, regulates the uses to which data processed by automatic equipment may be put. The legislation places strict limitations on individuals who store the personal data of third parties, requiring that any information is subject to a number of “data protection principles”. These principles include the right of the subject to inspect any stored data, and the duty of the holder of the information to provide adequate technical protection for such data. The holder of the data is also obliged to ensure that the data is not used for purposes other than those for which the data was originally collected, unless such a use has been agreed to by the data subject.

The British 1998 Data Protection Act (HMSO, 1998) implements the 1998 European Union Directive on Data Protection (European Union, 1998). This directive, enforced across the European Union’s economic sphere, includes the principle that personal data must not pass to countries outside the European Union that do not also implement adequate data protection measures as defined by the directive. This has caused some difficulties with the passage of data to the United States, which relies on corporate self-regulation (United States Department of Commerce, 2004) rather than legislation to protect privacy, and thus does not meet the requirements of the directive. Various programmes are in place to overcome this barrier between two of the world’s major economic forces, however the insistence of the European Union on strict controls over such a valuable resource as personal data is pushing the world at large towards strong data protection policies.

There have been analyses of privacy that do not support its importance in society. The work of Posner (1981) examines privacy from an economic basis. This view states that personal information should be kept private only if the economic value to society of such information is decreased by its becoming public knowledge. Posner argues that the only personal value in concealing private information is in deceiving or manipulating others for personal gain, and thus is not of economic use to society as a whole. This view proposes corporate privacy as having value, but asserts that personal privacy is not beneficial to a nation’s economy and so should not be protected in law.

Posner’s view of privacy has not achieved wide acceptance, and many societies in the modern world have enacted laws that protect privacy to a greater or lesser extent. The increasing level of information that is transmitted, and thus may be stored, by computers is causing a greater public interest in the right to privacy. In the past, privacy was naturally protected by the difficulty of storing and collating personal data. As comprehensive surveillance and indefinite storage become increasingly feasible, it has become necessary to protect this right in more explicit and effective ways.

1.2 Anonymity

Privacy legislation and regulations that have been enacted around the world are a useful step in preventing the widespread storage and dissemination of personal information. There are, however, always parties that seek to bypass these measures, for a variety of reasons. Rather than regulate the storage and transmission of personal information, advocates of anonymity propose to achieve the same goals by preventing such information ever becoming available. This position provides the motivation for the work presented here.

1.2.1 Names and Identity

Many of the concepts that we examine are in some way concerned with the obfuscation of information that relates to a user's identity. This information can take many forms, but the classic example is the *name*. The name of an individual is intended to be a unique identifier within some group that allows for that individual to be distinguished from the other members of that group. The fact that, in the increasingly large social groups in which we find ourselves, a classical name is rarely unique does not affect the purpose of distinguishing those around us by such labels.

When we discuss the anonymity properties of an individual, we are implicitly assuming definitions of identity. We assume that a user of a system is a unique individual who performs actions that can potentially be traced by another individual. However, there is a dissociation between a user's representation in a system and their "real-world" persona. Multiple users can collaborate to form a single online identity, and a single user may have multiple representations online. The implications of this are not fully understood; the simplifying assumption that a single user is linked to a single representation is almost uniformly made.

While it is possible to work with definitions of a name that rely on our implicit assumptions concerning identity, it should be remembered that there are many possible interpretations of what constitutes an identity.

1.2.2 Etymology

The English word *anonymous* is derived from the classical Greek stem 'ονιμα' (*onyma*, name), combined with the prefix α- (*a*-, the absence or lack of a property). Anonymity may therefore be understood as the state of being nameless or having an absence of identification.

To extend this definition to more common use within the field, consider

> "Anonymity is the state of being not identifiable within a set of subjects"
>
> – (Pfitzmann and Köhntopp, 2000)

This definition succinctly expresses the main purpose of anonymity, subject to questions concerning the term "identifiable".

Anonymity is the fundamental *identity hiding* property, providing total removal of identifying information from its subject. Any identifying information required by individuals can trivially be added into a communication in an otherwise anonymous system. As such, anonymity systems provide the option of limiting identity hiding as much or as little as desired by explicitly revealing identifying information when necessary.

Total anonymity is the focal point for research into identity hiding. To achieve this, anonymity systems are uniformly based on a small set of possible approaches, The *mix*, originally proposed by Chaum (1981), is the most significant of these. The most common approaches to providing anonymity are treated in detail in Chapter 3.

Despite this focus on anonymous systems, total anonymity is very much a two-edged sword. For certain forms of application, such as posting to mailing lists or accessing the world wide web, anonymity can be a highly desirable goal. Other systems, however, suffer greatly if there is no possibility of tracking identities. Sometimes identity needs to be tracked over the course of an extended transaction, but not between transactions. For this reason *pseudonymous communication*, which provides a certain amount of information associated with an identity, is required for a number of practical identity hiding systems.

Sender and Recipient Anonymity

We have already defined anonymity as the property of being nameless, given certain assumptions concerning the meaning of a name. However, this absence of naming is only of use or interest in a communicating system. Communication requires two participants: a sender and a recipient, where either participant may in actuality be a group of individuals. To whom does this anonymity apply?

This question divides our definition of anonymity. We may ensure the anonymity of the sending actor in some communication, but leave the recipient's identity open to the world. Conversely, it is possible for a sender to make available their identity but to ensure that the recipient of their message remains unknown. This provides a very different problem from sender anonymity, and is also less studied. Finally, it may be desirable for both end-points of communication to be hidden from observation. Each of these forms of anonymity has its own set of applications, design problems and potential attacks.

Connection-Based and Message-Based Anonymity

An important distinction in anonymity systems is the form of communication that takes place between the sender and the recipient. Much of the data transmitted across the Internet makes use of stream-based protocols that maintain persistent connections between communicating parties. Despite this, many applications are themselves message-based at a higher level.

Early anonymity literature focused on message-based anonymity, which is an easier problem than that of anonymity within connection-based systems. However there are some applications that cannot be performed effectively using a message-based approach due to the level of latency that these systems typically exhibit. Maintaining anonymity in systems that rely on persistent connections with low tolerances to traffic latency is an open problem.

A number of approaches to connection-based anonymity have been proposed. By far the most popular of these is *onion routing*, originally proposed by Goldschlag et al. (1996). This approach is discussed in greater detail in Section 3.4.2. The main concern with such systems, beyond the anonymity of the users, is to ensure low-latency connections with reasonable bandwidth restrictions. These constraints makes it necessary to use alternative approaches that that reveal a new set of potential attacks.

The Anonymity Set

The traditional quantification of anonymity is the *anonymity set*: the set of all participants who could have performed an action. This approach was originally proposed by Chaum (1981), who made use of the set analyse the anonymity of the *mix* system as described in Section 3.1. This quantification relies on the assumption that the larger the size of the set of participants that could have performed an action, the stronger the anonymity provided by the system.

This quantification is not ideal as it assumes a uniform distribution of probabilities across the set of participants. This assumption is not ideal for a group of heterogeneous users. In response to this, a number of alternative quantification methods have been proposed that seek to deal with both this and other problems inherent in the anonymity set (Serjantov and Danezis, 2002).

Unlinkability

Pfitzmann and Köhntopp (2000), in reasoning about anonymity systems, propose a viewpoint defined by a set of subjects sending messages to a set of recipients. In this setting, the critical concept is an *item of interest*, defined as the sending or receiving of a message.

The desirable property of an anonymity system is that items of interest are *unlinkable* to any identifier in the system, and no identifier in the system can be linked to a specific item of interest. This provides a basic definition of anonymity, although it does not directly lend itself to a quantification of the property.

Bitwise unlinkability (Danezis, 2003) is the concept of cryptographically obscuring the content of messages as they pass through a network node. This obfuscation makes it impossible for an attacker to link an ingoing message to an outgoing message by examining the data. Almost all implemented anonymity systems use this process in combination with some method of overcoming *traffic analysis* in order to achieve practical anonymity.

1.2.3 Pseudonymity

"If liberty means anything at all, it means the right to tell people what they do not want to hear."

– Eric Arthur Blair (writing as George Orwell)[2]

Closely related to anonymity is the concept of the *pseudonym*. The word "pseudonym", that stems from the Greek 'πσευδος' (*pseudos*, false) again combined with 'ονιμα', refers to the adoption of a false name. Traditional use of pseudonyms was as a method by which authors could publish politically inconvenient material without the threat of retaliation.

Pseudonymity causes users to be associated with a persistent identifier. The purpose of this approach is to allow types of transaction, relying on user history and behaviour, that are not possible using anonymous systems. This is of particular use in systems that rely on networks of trust between users and thus cannot employ single-use approaches to authentication.

As mentioned above, pseudonymity can be achieved through the use of an anonymous infrastructure with suitable user information and history stored within the explicitly transmitted data. If the communication system on which communication relies is inherently anonymous then pseudonymity becomes

[2]from "The Freedom of the Press", Orwell's original unpublished preface to *Animal Farm*, first published in *The Times Literary Supplement*, 15 September 1972.

an easier proposition, as data can be released as chosen by the user without fear of extra information leakage from the system. Care must be taken that the interaction between deliberately released data and otherwise harmless data within the system cannot interact to reveal more than is intended.

Pseudonymity may therefore be seen as a problem that overlays anonymity. An anonymous channel may have some form of persistent user identification added that is kept secret between the sender and recipient. Pseudonymity may be viewed less as a primitive construction and more as a combination of other security properties such as secrecy, anonymity and authentication.

Pseudonymity is often required in censorship-resistant systems. Several systems to achieve this goal have been proposed, such as the Eternity service of Anderson (1996), the "Publius" system of Waldman et al. (2000) and the work of Goldberg (2000); Goldberg and Wagner (1998). Censorship-resistant systems typically aim to provide an online anonymous data store that maintains the availability of published documents in the face of attacks. Such approaches are designed to be resistant against any measure, whether physical or legal, to remove data from the system once it has been published.

1.2.4 Applications

In investigating a property it is generally useful to establish its applications. This is particularly the case for properties such as anonymity that have a clear potential for abuse by malicious parties. It is important to consider the benefits that are made possible through these investigations.

Public Discussions

The purpose of an anonymous system is to allow an individual to perform an action without an observer being able to link the individual to the action. The reason for any individual to perform actions with anonymity is to prevent the consequences of an observer learning the source of the action.

In discussions, an identifiable participant may not wish to express their own views unreservedly. If the topic under discussion is controversial then a user may fear the association of their identity with an unpopular viewpoint. If participants are known to each other they may fear to disagree with each other on particular topics for fear of causing offence, or alternatively may disagree for personal reasons. A level of self-censorship is inherent in many social situations, and this can lead to an unhealthy homogenisation of viewpoints.

More seriously, certain topics may be entirely taboo. In such cases a topic may never be discussed at all due to the fear of being associated with the topic in question. Taboos may be social, in which case a user will fear the censure of their acquaintances; alternatively, a taboo may be legally enforced and thus have more serious ramifications if broken.

A consequence of such taboos is discussed by Templeton (2001). The argument presented there is that societies benefit from“fringe” groups that challenge societal norms and opinions. Over time, certain ideas that were previously unacceptable become accepted as mainstream thought. In an environment where surveillance becomes ubiquitous, fringe groups are stifled by their inability to function outside of the view of society. In such situations, the possibility for individuals to anonymously explore the boundaries of acceptable ideas has the potential to aid in the healthy development of wider society.

Reporting

A more direct application of anonymity than the ability to discuss potentially unpopular ideas is the ability to anonymously report on the actions of powerful entities. Anonymous political reporting as a means of overcoming censorship of authors is well known historically, and is often cited as a motivation for current research into anonymity systems.

Political reporting is a useful example due to the nature of governments as powerful entities with the power to severely restrict or harm individuals. Additionally, the employees of a company may be under similar restrictions and feel unable to report on questionable activities that they observe in the workplace. For such individuals, and for society at large, a mechanism that allows anonymous “whistle-blowers” can be of great benefit.

Voting

Voting is fundamental to many societies in a world where democracy is widely regarded as the ideal form of government. The ability of a person to take part in the governance of their own country is, like the notion of personal privacy, enshrined in the United Nations Universal Declaration of Human Rights (United Nations, 1948).

Voting is critical to a democracy. The collective act of voting decides the nature of government in a country, carrying with it a great deal of power and potential for abuse. In order for a voting system to function, there must be a method of counting votes and verifying that those votes were correctly cast and recorded. In contrast, it is not appropriate for a vote to be directly traced to an individual. If an individual's vote can be traced then that individual can be coerced into voting in a particular way, either by a party offering rewards for voting in the desired fashion or by threatening punishments for voting against them.

Voting therefore presents the interesting problem of requiring some form of linkage that allows a voter to ensure that their vote was correctly recorded, while protecting that user from an outsider discovering their vote. The nature of voting also results in a very powerful attacker inside, or even controlling, the system with the desire to compromise or affect the vote. Balancing these factors is an extremely difficult problem and has been the subject of much research. In particular, Arrow (1950) demonstrates that, under certain reasonable requirements, it is not possible for a voting system to provide all desired properties.

Assessment

A more pragmatic use of anonymity is that of anonymous assessment of work. This application of anonymity is to be found in such areas as the marking of examinations or coursework in academic institutions. In such situations, the user whose work is being assessed must be unknown to the assessor to avoid favouritism or bias. A similar application is to be found in anonymous suggestion boxes used in workplaces and for feedback by employees concerning their superiors.

An interesting feature of these systems is that users may not desire anonymity and may actively seek to compromise themselves. A favoured student or well-known expert may wish to identify themselves, despite the anonymity of the system, in order to gain from their reputation. The challenge of forcibly anonymising a user is very difficult, particularly in the case where the data rather than the traffic flow of the network is under scrutiny.

We do not generally consider the content of messages in the work that follows, concentrating instead on the ability of attackers to identify users from their behaviour. This implicitly suggests that the systems we examine rely on users who do not actively seek to compromise their own anonymity. Despite this, in some cases the inability of a user to prove that they were the originator of a message could be beneficial. Anonymous auctions, in which only a particular party can prove that they were the winning bidder, could make use of such a property. Anonymity system that allow users to compromise their own anonymity at will subject those users to the fear of coercion. If a user is unable to prove to a third-party that they were the initiator of a transaction, it prevents them from being forced to do so.

Data Sanitation

Medical records and census data are interesting in that, while they are certainly concerned with the anonymity of participants, they are not based around the transmission of data. The main aspect of interest to anonymity is the *sanitising* of the data: the removal of identifying information or the formatting of the data in such a way that identification becomes difficult.

The sanitising of data is an interesting problem that does not, however, fit into the transaction model in which we are interested. As such, we do not discuss it further.

1.3 Summary

In this chapter we have examined the historical and philosophical background of anonymity. We have seen justifications for users to desire anonymity and its related properties, as well as briefly detailing concrete examples of applications that are made possible through anonymous communication methods. A belief in the notion of privacy, as well as the beneficial applications of anonymity, provide the motivation for the work that follows.

Chapter 2

Basic Concepts

In working towards the classification of anonymity systems, it is necessary first to define our terminology. We discuss here a number of terms and concepts that relate to anonymity.

2.1 Actors

Here we discuss the terms applied to actors within the system. These are the individuals who perform actions within the system. These encompass the users desiring and providing anonymity as well as the attackers who seek to break the system for their own purposes.

A *user* is an actor within some system who may process and, potentially, create data in the form of messages. Any number of individual users may group together in order to form a communicating party.

In the most common form of system that we examine, a user corresponds to an individual who may send, receive or observe a message. In other systems the user may be an individual selling an item in an online auction, a citizen voting in a national election or an author attempting to publish an article anonymously. Users therefore encompass all those who may take part in a transaction, as well as those who attempt to attack or compromise the system in some way.

One or more individual users that jointly take part in a transaction form a communicating *party*. The defining characteristic of a multi-user party is that all users in the party share access to all data relevant to the transactions in which the party engages. It is important to note that the users in a party do not have full and unrestricted access to all data belonging to other party members, nor do all members of a party necessarily share the same ultimate intent. This concept is discussed further in Section 2.6.1.

Participants in a transaction are those parties that are directly involved in the transaction either as *initiators* or *recipients*. This term does not include users who are merely relaying requests for other users, or attackers who seek to compromise the anonymity of a party.

The party that initially causes an anonymous communication to take place is referred to as the *initiator* of that transaction. In order to clearly distinguish the communicating parties, we treat a transaction that involves anonymity as separate from any preceding setup phase that caused that anonymous data transfer to take place.

For example, a census system that allows subjects to respond anonymously, but does not explicitly have any method to prevent detection of who is sent census forms, would treat the *subject* of the census as the initiating party, not the party that is intending to collect the data. The initiator initiates the anonymity-based element of the transaction, regardless of the ultimate author of the exchange.

As many systems that we consider function under the classic concept of messages being actively sent by the initiator, we often refer to initiators as *senders*.

The *recipient* in a given transaction receives a data transfer from the initiator. In the case that the main source of data is the recipient, for example in a database query, the initiator remains the user that first requested the data. When we consider the standard message sending model, the recipient will generally be referred to as the *receiver*.

Any party that seeks in some way to compromise, weaken or break the anonymity of a party involved in some transaction is referred to as the *attacker*. The nature of compromise depends upon both the attacker's requirements and their capabilities. An attacker is a party in the system, possibly containing multiple colluding users.

The attacker and its capabilities are discussed in greater detail in Section 2.6.

2.2 Structure

Here we define the terms that apply to the structure of the anonymity systems under consideration.

The *system* refers to the combination of all users, the parties they form, the nodes through which messages can pass and the links between them. Each *node* in the system is a discrete point that may be part of the route of a transaction. At this point, messages may be intercepted or modified. A single transaction may encompass any number of intermediary nodes. A node will typically be closely linked to a party, such as a computer in a communication network and the operator of that node. In certain cases, actors or parties themselves may act as nodes.

Links form connections between nodes and between users. Each link, which need not be a physical object, allows for data to pass between nodes and can potentially be compromised by an attacker.

The basic interaction between participants in the systems that we consider is the *transaction.* Each transaction is the exchange of data in some form and may involve any number of parties. Parties take part in a transaction either as intermediaries transferring data, as participants at the end points of the data exchange or as attackers.

2.3 Anonymity Definitions

There are several ways to define what is meant when we refer to the anonymity that may be desired by a user. Possibly the most widely quoted definition in the literature is that of Pfitzmann and Köhntopp (2000):

> "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

This definition reflects the original quantification of an anonymity system: the larger the number of actors that could have performed an action, the better the anonymity provided by the system. Diaz et al. (2002) and Serjantov and Danezis (2002) have expanded on this concept to consider the notion that the strength of the anonymity increases both with the size of the set and the uniformity of the distribution linking actors to observable actions.

Pfitzmann and Köhntopp (2000) explore in greater detail more terms related to anonymity and the systems that provide this property. However, we choose here to examine the concept of identifiability more closely with regards to the requirements of the potential attacker.

2.4 Identifiability

It is important to understand what an attacker considers to be a satisfactory identification of a participant before acting on their knowledge. In many cases we assume a *unique* identifiability property wherein an attacker will only act on reducing the anonymity set to a single actor with absolute certainty. However, there are two factors that the attacker may choose to treat differently: the cardinality of the anonymity set, and the level of certainty with which the identification is made.

2.4.1 Cardinality

An attacker may be content to act once the anonymity set has been reduced to a certain size. For example, in a criminal investigation it may be most effective to reduce the set of suspects to an approachable level before detaining and questioning those few members of the set that remain.

Similarly, a malicious attacker who wishes to perform a denial of service attack on a particular user may have the resources to perform this attack on more than one victim. In this case, once the number of potential victims has been reduced to an acceptable level the attacker may have no qualms in attacking all members of the set in order to ensure that the intended victim is affected.

The concept of protecting users up to a certain cardinality of the anonymity set has been presented in the privacy literature, where it is known as k-anonymity. The notion was first put forward by Samarati and Sweeney (1998) and was further developed by Sweeney (2002). This view of anonymity was originally used as a quantification and model for ensuring privacy in databases.

A system is said to be k-anonymous for a given value of k if a particular user's data is indistinguishable from that of $k - 1$ other users in the system. In the communicating systems that we consider the concern is with a user's traffic patterns and not their stored data, however the applicability of the concept is clear.

2.4.2 Certainty

Similarly to the protection of k-anonymity, we may also consider the attacker who is willing to act while uncertainty remains that their target has been correctly identified. This attacker may act in cases where a more careful attacker would remain indecisive. In the information theoretic quantifications of anonymity, anonymity is represented by a probability distribution across the set of participants that represents their likelihood of being the source of an item of interest. An attacker may choose to consider only those members of the set with a certain probability of being the intended target, given observed data, and thus greatly reduce the time and effort in proving this identity.

In the case that an attacker is looking only for a scapegoat, or is seeking to make an example for other users, that attacker may be content to isolate a set of users with sufficiently high probability of being the source of a message and punish a member of that set at random. The possibility of such an unjust attacker could have implications in the system's strategy for preserving homogeneity amongst users.

2.4.3 Sufficient Anonymity

The concepts discussed in the previous sections suggest that a single quantification of the anonymity provided by a system is not sufficient to judge the safety of users from differing attackers. Although an anonymity system will typically rely on cryptographic functions that may in themselves be treated as unbreakable black box functions, for the system as a whole we can usually only work towards a relatively weak idea of *sufficient* anonymity.

Sufficient anonymity is the property that a system's users be homogeneous to the extent that an attacker cannot distinguish any subset with acceptable certainty of containing a specific user such that that subset is small enough to be harmed by the attacker. A sufficiently anonymous system could therefore be informally defined as:

> A system causing the inability of an attacker to prove with acceptable certainty a correlation between a number of transactions and a subset of users such that this subset is small enough to be harmed by the attacker.

It is clear that in this definition there are certain terms requiring further explanation: both the level of acceptable certainty for a user and the ability of an attacker to cause harm are dependent upon the specific requirements and capabilities of the users and the attacker.

2.5 Characteristics

We may informally characterise an anonymity system by a number of factors relating to the nature of communication that take place in the system. These factors are discussed in detail here.

2.5.1 Anonymised Party

It is important to consider the parties that require anonymity in a particular system. It is often the case that the initiator of some transaction requires anonymity, resulting in the common *sender anonymity* seen in the sending of anonymous messages and posting to discussion forums.

In other situations, both the initiator and the recipient of a transaction may require anonymity. This is may be necessary to allow anonymous conversations to take place and in some auction systems. This form of anonymity corresponds to *relationship anonymity* as described by Pfitzmann and Köhntopp (2000).

Finally, it may be the case that the initiator may be identifiable while the recipient remains anonymous. While rare, certain applications such as payments made to anonymous bank accounts can express this anonymity characteristic: *recipient anonymity.*

2.5.2 Connection

An anonymous communication may need to continue over a number of linked transactions. Such transactions must in some way be connected whilst maintaining anonymity for the initiator of the transactions. This is the basis of *pseudonymous* communications, in which a participant is temporarily associated with an identifier that allows them to be tracked across a number of transactions. Pseudonymity is of great use in systems which must offer a level of trust or reputation for users.

Pseudonymity, clearly a different property from anonymity, may be seen as representing the area on a scale between total anonymity and total identifiability. The concept of a "nymity slider" that scales between anonymity and authentication, referred to as "verinymity", is proposed by Goldberg (2000).

A standard anonymised email conversation with anonymous replies forms a connection between participants. More directly, many systems aimed at providing real-time anonymity for purposes such as web-browsing are connection based by their nature.

2.5.3 Distribution of data

In preserving anonymity, it is important to consider the parties that may view the results of the communication. A message posted anonymously to a public discussion is open to the world; a private message is viewable only by the recipient. The visibility can allow attackers to gain extra information about the message and its participants. In particular, analysis of unencrypted text can eventually distinguish between different authors. When a message has appeared in an openly viewable forum, it cannot be denied that a message has been sent. In the case of private messages the existence of dummy traffic passing through a system could conceal such an event.

The distribution of data is linked to the number of participants involved in a transaction, although a private communication may be sent to a number of users. A public transaction is necessarily viewable by any party that chooses to do so. A private communication, whilst perhaps available to a group of recipients, can only be viewed by its intended recipients. There is, of course, nothing to stop a participant from making private data public. Disregarding revocability issues in proving or disproving the origin of data, it is difficult to protect the anonymity of a user against their will.[1]

2.5.4 Participant Relations

Any number of participants can be involved in anonymous communications, and a simple one-to-one relationship is not sufficient to encompass a reasonable model of anonymity. It is important, in examining a system, to define the nature of the communication that can take place.

Most commonly, participant relations are either simple one-to-one or one-to-many multicast transactions. It is also possible, however, that a transaction may involve more complex combinations of participants.

In certain cases, it may be worthwhile to treat a group of participants as a single identity; in other cases a group is necessarily split. Work into the treatment of identity and knowledge in anonymity systems for groups of users is given by Syverson and Stubblebine (1999).

2.5.5 Revocability

In some situations it may be necessary for anonymity to be of limited duration. It may be desirable for anonymised data to reveal the source actor under certain conditions. The nature of such a condition could be the passage of a period of time, the end of an auction or the upholding of an abuse claim. Such systems are also useful in the anonymous marking of students' work, and in academic peer review, as discussed in Section 1.2.4.

One of the most common examples of revocability in anonymity systems is an auction system in which bidders remain anonymous until the final bids have been counted and the winner decided. At this point it becomes necessary to in some way identify the winning bid. A proposal for such a scheme is presented by Stajano and Anderson (1999).

[1] Preventing a user from compromising their own anonymity, while difficult, is not impossible. One method for achieving this is for multiple users to claim to be the source of some data. A well-known example of this is the famous "I am Spartacus" scene from Stanley Kubrick's 1960 film, in which an entire army claim to be Spartacus in order to prevent him from identifying himself. It should be noted that this attempt was unsuccessful.

2.6 Attackers

A detailed study of attacker models in anonymity systems is given by Raymond (2000). With slight extensions to the original definitions, we consider the capabilities of an attacker to be characterised by four criteria. In addition to these capabilities, and in line with our view of anonymity from Section 2.3, we examine here how the requirements and abilities of the attacker interact with the nature of anonymity systems.

2.6.1 Internal/External

The pervasiveness of an attacker limits the type of attack that may be mounted against an anonymity system. Specifically, an *internal* attacker has access to the interior workings of some communication node in the system, where an *external* attacker is limited to data that travels between nodes.

This distinction becomes particularly important in conjunction with the distribution of the transaction. An attacker who has internal access to a private transaction negates the protection gained by secrecy of the data.

The internal attacker often functions as a corrupt node. The attacker may view the passage of data and may gain information concerning routing information that is not possible through observation of the communication medium. The attacker functioning as a corrupt node is typically capable of *active attacks* from that node.

2.6.2 Static/Adaptive

Static attackers are fixed in their location and their responses to data. These attackers are thus unable to alter their behaviour once a transaction is in progress. Simple devices observing a network at fixed locations may not be able to respond to changes in network traffic to exploit observed patterns.

Conversely, the operator of a rogue network node may be able to directly alter aspects of their node in order to take advantage of the passage of a particular message. This allows for a much greater range of potential attacks to be mounted by the adaptive attacker.

The work of Raymond (2000) defines the static attacker in terms of the run of a particular protocol: the static attacker must define its behaviour before the protocol run, whereas the adaptive attacker may choose to alter its strategy once the protocol has started.

2.6.3 Active/Passive

Different attacks on systems require differing behaviour from the attacker. In some cases, passive attacks may be mounted simply by observing transactions between nodes. A *passive* attacker may attempt to compromise a system by making deductions from the naturally occurring flow of data.

Other attacks require that the attacker be able to change the data that passes through the system. The *active* attacker functions by injecting data into transactions or altering the existing data passing between participants. The distinction between active and passive attackers causes great differences in the nature and power of potential attacks.

2.6.4 Local/Global

It is significant in considering attacker models whether the attacker is *global*, and thus present across the entire system; or *local*[2] and so restricted to some subset of links and nodes. The global attacker is a powerful adversary, capable of mounting some of the most effective attacks against users of an anonymity system, as shown by Danezis and Sassaman (2003).

Some forms of global attacker, specifically the external active global attacker, correspond to the standard *Dolev-Yao* attacker considered in security literature. This attacker is considered to be equivalent to the communication medium between actors, and was originally presented by Dolev and Yao (1983).

The pervasiveness of an attacker across a network can vary greatly. Whilst some attacks are possible only for a global attacker, the local attacker who controls particular subsets of nodes or the links between them can still mount weakened forms of global attacks.

2.7 Summary

In this chapter, we have presented an informal overview of generalised anonymity systems. We have explored the nature of the transactions that take place between actors in such systems, and considered the various capabilities of attackers that may seek to compromise the anonymity of individuals. We now examine anonymity systems in more detail.

[2]The term *non-global* is possibly more appropriate, as a local attacker may often be considered to control a limited subset of the entire network without being restricted to a given node.

Chapter 3

Anonymity Systems

In this chapter we will examine the technologies that underlie anonymity systems and their development since the introduction of the *mix* by Chaum (1981). We consider the historical development of anonymity systems, and the attacks that have been mounted against them. This provides us with an insight into the similarities that underpin the existing solutions to providing anonymity and how we may isolate these fundamental features.

We begin by examining the mix, which remains an important and widespread anonymity technology. Next, we explore variations of the mix and its development since Chaum's original proposal. Finally, we examine other technologies that aim to achieve the same goal but that do not derive directly from the mix.

3.1 Mixes

Since proposed by Chaum (1981) the mix has been the focus of much anonymity research, although in recent years there has been increasing focus on real-time technologies such as onion routing as proposed by Goldschlag et al. (1996). There are many variations on the basic mix design that aim to mitigate attacks against the system and to improve on various aspects of security, reliability and performance.

Mix systems were first proposed by Chaum (1981) as an anonymising process for electronic mail. This seminal paper began serious research into Internet-based anonymity systems

3.1.1 Basic properties

A mix node accepts encrypted messages that are then decrypted and stripped of all sender information. This process provides unlinkability between messages, trivially defeating analysis of message flow by removing identifying information.

Messages are stored in the mix until some trigger condition is reached. When this occurs, the mix forwards its internally stored messages in random order to their respective recipients, or to other mixes that repeat the strategy. Mixing defeats *traffic analysis* by removing correlation between the flow of input and output messages. The condition that determines the sending of a message greatly affects the nature of the anonymity provided by the mix and the attacks that can be employed against the users.

Latency is inherent in a mix system due to the delay caused by accumulating a number of messages. This latency makes the system unusable by applications that rely on real-time message passing. As such, mixes are typically used for message-based applications such as email in which delays are acceptable.

Unlinkability

The purpose of encryption in a mix system prevents an attacker from deducing linkages between incoming and outgoing messages based on the message content. Typically, messages are padded to a fixed size with larger messages split into a number of appropriately-sized chunks.

The message sequence from a sender to a recipient through a mix can be expressed as follows:

$$\begin{array}{rcll} \text{Sender} & \longrightarrow & \text{Mix:} & \{\text{Msg, Recipient}\}_{K_{Mix}} \\ \text{Mix} & \longrightarrow & \text{Recipient:} & \text{Msg} \end{array}$$

The above notation is standard in referring to communication protocols to represent the sequence of messages passed between parties. Each line represents a single step of communication; $A \longrightarrow B$ represents that the data in question is sent from party A to party B. Everything following this describes the data that is passed. When considering protocols that involve cryptography we use $\{M\}_K$ to refer to a message M encrypted with some key K.

The behaviour described above can be extended to multiple mixes by use of layers of encryption, each of which corresponds to a mix node in a path. Messages could therefore take the form:

$$\{\{\ \ldots\ \{\text{Msg, Recipient}\}_{K_{Mix_n}},\ \text{Mix}_n\ \ldots\}_{K_{Mix_2}}\}_{K_{Mix_1}}$$

At each step the encrypted content contains the next layer to be decrypted, along with routing information for the next step in the path of the message. This prevents the entire route becoming known to any one intermediary node.

Mixing

For simple buffers, in which messages leave a node in the order that they arrive, it is trivially possible to deduce the relationship between senders, messages and recipients. This is achieved by observing the ordering or timing of messages entering a node and correlating them against the flow of output traffic. A mix is designed to eliminate any correlation between messages entering the node and those that leave. The most common way of achieving this goal is by storing multiple messages and flushing them in a random order.

The trigger conditions that cause mixes to send out their message batches are the subject of much study. Different approaches are vulnerable to different attacks, and the problem is by no means trivial. In the following section we review a number of the common mix designs with reference to their flaws and advantages. An overview of this subject is given by Diaz and Preneel (2004b).

3.1.2 Threshold Mixes

The original mix proposed by Chaum (1981) achieves mixing by storing messages until a certain number of stored messages is reached. The mix then sends out all stored messages in random order.

The technique of storing a number of messages until a given condition is reached before flushing messages is the strategy behind a family of mixes known collectively as *pool* mixes. A contrasting approach is that of *continuous* or *stop-and-go* mixes as proposed by Kesdogan et al. (1998). These strategies are discussed in Section 3.1.4.

An important form of pool mix is the *threshold mix*, that improves anonymity by retaining a certain number of messages at each round. Every message entering the mix therefore has a probability of staying in the mix for an arbitrarily long time. This strategy theoretically increases the size of the anonymity set for that mix node to include all senders that have sent a message following the target message; the set of messages that could contain the desired message is all messages that leave the mix after the desired message entered it.

The fact that a message is increasingly likely to have left the node after a large time period highlights the flaw in using the anonymity set to analyse these mixes directly. That messages may remain in a mix during multiple flushings of the buffer trades an increased latency in message delivery for this increase in confusion regarding the sender of a message.

Simple threshold mixes are open to an $n-1$ *attack*, in which an attacker floods a mix with fake messages that are traceable by the attacker. By creating an overwhelming majority of traceable messages, the attacker is able to isolate a target message at each round. As this single message is the only one that is not owned by the attacker, the correlation between the single unknown input and the single unknown output is trivial. This attack is discussed in Section 3.2.

3.1.3 Timed Mixes

In contrast to the threshold mix, messages in the pool of a timed mix are flushed at a given time interval (Serjantov and Newman, 2003). This approach prevents messages remaining in the mix indefinitely when too few messages are received to reach the threshold value. The timing strategy, assuming total flushing of the pool at each round, also guarantees a lower bound on the latency of message delivery. This approach suffers from the flaw that if only a small number of messages enter the mix in a given time period, the anonymity of the messages is severely compromised.

Most implemented systems that use a pool mix combine a timing and a threshold constraint in order to gain the benefits of both strategies. A notable example is the Mixmaster anonymous remailer network, originally proposed by Möller et al. (2003), discussed in Section 3.5.1.

3.1.4 Continuous Mixes

Kesdogan et al. (1998) present the concept of a *stop-and-go*, or *continuous*, mix. In this approach, each sender selects a delay from a given distribution and uses this to define a per-message delay. The mix delays the message for the specified time period before forwarding it.

This approach allows flexibility in the latency of messages. Users can balance strength of anonymity against latency. This gives the continuous mix potential use in applications that require tighter time constraints than is possible to achieve with other forms of mix. In addition, a user has a predictable latency for messages and so can potentially detect interference in the network from an attacker who is delaying messages.

3.1.5 Mix Networks and Mix Cascades

It is not desirable to rely on a single mix node for anonymity. Pfitzmann and Waidner (1985) show that the possibility of traffic analysis is much higher for a single location on a network. In addition, a lone mix node is a single point of failure. The possibility of the mix being compromised by an attacker or a dishonest mix operator is too high to risk the total loss of anonymity that these situations would cause.

Messages are therefore typically routed through several mixes. With appropriate use of encryption, this ensures the anonymity of a message provided that at least one mix on the route is not compromised.

The routing strategy for a message through such networks is important. *Mix networks*, as presented by Rennhard and Plattner (2003) use a free route strategy in which users select their own routes through a network; *mix cascades*, presented by Dingledine and Syverson (2002a), restrict the path a message may take through the network. There is debate as to which strategy is more effective and robust; Berthold et al. (2000) argue the case for using a mix-cascade over the more common mix-network. The relative advantages and disadvantages of both approaches are discussed here.

3.1.6 Mix Networks

Mix networks use a free-route approach to the network path. The user chooses a path from the available mixes for each message sent, which provides a level of flexibility in the network. A user may choose to route their message through mix networks that they trust, or that have behaved reliably in the past. Free-route networks scale well as the number of network nodes increase. The free-route approach also provides fault-tolerance, as nodes that cease to function can be routed around by the system.

The free-route approach does, however, suffer from a number of potential active and passive attacks that can compromise anonymity. These are described in detail by Berthold et al. (2000), and typically rely on an attacker that controls the majority of mixes in a network.

It is possible for an attacker to partition messages going through an honest node based on the number of other nodes through which they have already passed, this information being available to a global attacker observing network traffic through compromised nodes. With successive rounds, this partitioning can reduce the anonymity set for a message to only those messages that have passed along the same route.

Danezis (2003) notes that in a free-route mix network with a large percentage of compromised nodes, the chances of a choosing an entirely compromised path become high. In such a compromised path, anonymity is completely lost.

3.1.7 Mix Cascades

Pfitzmann and Waidner (1985) present the concept of *mix cascades*, in which a number of fixed paths through the network are used. This ensures that more messages take a particular path than in a free routing mix network. This increases the homogeneity of traffic flow and so mitigates the partitioning attack.

In free route mix networks, each node is a member of a large set of potential routes. There may therefore exist a significant difference between the maximum and minimum observed traffic through that node. At some times a node may be overwhelmed by the traffic passing through it, while at other times it may handle only a tiny amount of traffic. In mix cascades, all traffic is routed consistently through the fixed paths. By doing so, the traffic is made as uniform as possible, helping to prevent exploitable patterns in message flow.

Mix cascades suffer from a number of problems. In free route networks, an attacker must compromise a large number of nodes in order to successfully mount many known attacks; in a mix cascade only the nodes on a particular route need be compromised. An attacker can therefore focus their resources more effectively. The flexibility and fault-tolerance seen in free route networks is also lost to an extent in mix cascades. Cascades are also subject to denial of service attacks by shutting down a single node in a route. The nature of the mix cascade networks causes such an action to destroy a significant portion of the network.

3.1.8 Hybrid Approaches

Neither mix cascades nor networks offer the perfect solution to routing within a network of mixes. As such, a combination of the two approaches may be appropriate for some situations. Berthold et al. (2000) enumerate a number of proposed advantages to free route networks and show that each can, to some extent, be achieved within a cascade.

Many of the techniques to improve the behaviour of cascade networks move away from the strict single-route towards a more flexible routing system. Danezis (2003) proposes to provide a number of cascade routes through the system, with the user selecting one of these. Berthold et al. (2000) allow each node in a path to nominate a number of potential next steps with the user being able to select from these options.

Another hybrid approach by Danezis (2003) employs a network based on sparse, constant degree graphs known as *expander graphs*. This network topology, and the use of probabilistic techniques to select the next node in the message path, provide a system that holds many desirable characteristics of both free route mix nets and mix cascades.

3.1.9 Synchronous and Asynchronous Batching

Dingledine et al. (2004b) suggest that the perceived difference between mix networks and cascades is largely due to the synchronicity of batching rather than to the network routing. The authors present an analysis of a free route, a cascade, and a hybrid stratified topology that all process messages synchronously. In these systems, each mix forwards messages at the end of a given *batch period*, and queues any messages that enter the network during such a period. Such a network, where synchronous batching is the defining characteristic of the network, was first proposed by Dingledine et al. (2001).

Dingledine et al. (2004b) argue that the use of a synchronous batching strategy in free route networks provides stronger anonymity overall than other approaches, and that the traditional problems with free routes may also be avoided to some extent.

3.2 Attacks on mix systems

Attacks on anonymity systems typically seek to weaken the anonymity of participants in the network, or to render the network unusable so that anonymous messages cannot pass. The majority of these attacks rely on *traffic analysis*: the observation of the flow of messages in an attempt to learn sender or recipient identities (Back et al., 2001). These attacks may arise purely from passive approaches, or may be employed by an attacker injecting their own traffic in an attempt to reveal underlying patterns in the flow of data. Here we provide a high level review of the more well-known attacks against mix-based systems. A comprehensive survey of such attacks is given by Raymond (2000).

3.2.1 Passive Attacks

There are two major threat models considered for mix networks. The first of these is the global passive attacker who has the ability to observe all packets on the network, but not to inject, capture, delay or otherwise interfere with traffic. All attacks possible for this attacker therefore rely on observing information about senders and recipients that leaks from the unaltered behaviour of the network.

The *intersection attack* (Berthold and Langos, 2002) relies on the fact that user traffic is not random and can often have predictable behaviour within certain time periods. In the case of direct conversations, pairs of users may send messages back and forth for a short time period. By observing this behaviour and performing successive set intersections on active users at particular times, it is possible over time to restrict the set of possible actors involved in given transactions.

Similar approaches include the well-known *disclosure attack* of Kesdogan et al. (2002) and *hitting set attack* of Kesdogan and Pimenidis (2004). These attacks again rely on finding intersections of anonymity sets associated with a particular sender over time in order to identify that sender's communication partners. As both the disclosure attack and the hitting set attack require the solution of an NP-complete problem in order to uniquely identify a user, there is consequently a statistical variant of each that sacrifices accuracy for efficiency.

Other attacks available to the global passive attacker follow a similar form, referred to by Raymond (2000) as *contextual attacks*. Any feature that distinguishes a particular message from background traffic can be used to infer information about the sender or receiver. Uniformity is key to defeating the global passive attacker.

3.2.2 Active Attacks

A different common attacker is the local[1] *active* attacker. Such an attacker has the ability to inject messages into the network, to compromise a number of nodes, to delay messages passing through the network, and to alter existing messages.

A comprehensive overview of active attacks on mixes is given by Serjantov et al. (2002). Here we review the more common approaches.

The majority of attacks that an active attacker may attempt against a mix are known as *blending attacks*. Blending attacks, most notably the $n-1$ attack, allow a sufficiently powerful attacker to identify the receiver of a message with absolute certainty for basic pool mixes, and probabilistically for some more advanced mix designs. Blending attacks generally take the form of *trickle* or *flood* attacks.

[1]The *local attacker* refers to an attacker who does not have total coverage of the network, however is not necessarily limited to a single machine.

Trickle attacks rely on the ability of an attacker to delay or delete messages approaching a mix. An attacker waits until a mix has flushed its contents then prevents all, or almost all, messages except a target from entering the mix in the next round. This attack is less effective with more complex mixes that use dummy traffic and advanced pool-flushing algorithms.

Flooding attacks also wait until the pool of a mix is emptied. After allowing a single target message through, an attacker floods the node with enough messages to cause the pool to be flushed. As the attacker can recognise their own messages, the target message is easily traced.

The $n-1$ attack is particularly resistant to attempts to reduce its effect. The use of complex pool flushing algorithms only degrade this attack to the level of tracing messages with a certain probability of incorrect identification. The attack relies on a very powerful attacker model, as the $n-1$ attack must be performed on all mixes within a network to be fully effective.

3.2.3 Denial of Service Attacks

Another category of attacks against mix systems or networks does not seek to reduce or compromise the anonymity of users, but rather to prevent the functioning of the mix networks. These denial of service attacks are of great concern in mix cascades that use known chains of mixes. Destroying the functionality of a single mix node removes an entire route from the cascade. A determined attacker could have a serious effect on a network by attacking a small percentage of nodes in the network.

3.3 Dummy Traffic

In any proposed mix design there may be fewer messages passing through a system than required for sufficient anonymity, which relies on a base level of interactions occurring in a system. Low levels of traffic make most attacks on the anonymity of users far simpler. *Dummy traffic* (Berthold and Langos, 2002) can be inserted into the network to confound traffic analysis by destroying observable patterns of genuine traffic flow.

Dummy traffic may be generated by network nodes themselves, or alternatively be directly introduced by end users. Some systems, such as PipeNet as presented by Dai (1996), pad traffic with a constant flow of dummy messages in order not to reveal any features of genuine traffic flow. As we see in later chapters, dummy traffic is an interesting method of gaining anonymity in its own right without needing to employ a mix.

Flooding the network to capacity with dummy traffic overlaying genuine messages is often considered infeasible due to the bandwidth requirements that this would cause. Systems that employ dummy traffic, such as that of Möller et al. (2003), use a lower level of dummy traffic generated by the nodes themselves. This traffic is created according to a probability distribution that aims to mimic real traffic, although the ideal probability distribution for such a task remains an open problem.

There are a number of design decisions in creating dummy traffic. A node may generate traffic itself, or may pass traffic generated by initiator and recipient actors. In mix systems, dummy traffic may be inserted into the message pool, thus potentially affecting the trigger condition for flushing the mix, or added to the traffic as it leaves the mix. There are also choices as to whether the dummy traffic should adapt as the message flow increases, or remain constant. These issues are discussed in detail by Diaz and Preneel (2004a), Diaz and Preneel (2004b), and Möller et al. (2003). In all cases, the desired behaviour of dummy messages is to be indistinguishable from the normal traffic in the network.

3.4 Other Technologies

Despite the focus on mixes within the anonymity community, there are other approaches that seek to achieve anonymous communication. Many of these approaches suffer from flaws either in their scalability or in the level of anonymity that they provide. For this reason most are not the subject of intensive ongoing research, although most have utility in specific application domains.

3.4.1 Dining Cryptographer Networks

In 1988, seven years after the mix (Chaum, 1981), Chaum proposed another scheme for anonymous message sending. The *dining cryptographer* network provides unconditional sender and recipient untraceability within a network, given assumptions of cryptographic security between parties (Chaum, 1988).

The dining cryptographer network was originally described by considering three cryptographers dining in a restaurant around a circular table. At the end of the meal, the cryptographers are informed that the bill has been paid by some anonymous party. The cryptographers desire to know whether one of the dining parties or an outsider is the benefactor.

To determine the answer to this question while maintaining the anonymity of the buyer, the cryptographers each flip a coin that is seen by both themselves and their right-hand neighbour. Each states the parity of the two coin flips that they have seen: whether they were the same or different. If a cryptographer was the party that paid for the meal they report an inverted result for the parity of their observed coin flips. By following this protocol the cryptographers can determine if one of the diners is lying. In the case that a cryptographer has lied, the reported number of coin flips displaying the same result will be even. If all parties had told the truth, the reported number of flips will be odd. The desired information can be deduced without revealing the identity of the buyer.

This approach can be generalised to a reliable broadcast network in which participants share cryptographic keys. The results of the hypothetical coin flip are broadcast one bit at a time into the network. If a participant wishes to make a message public, they flip the status of their broadcast bit for those bits that correspond to the high bits of their message. For each round a default parity of the broadcast bits corresponds to a 0 bit in some message, and an inverse parity of all broadcast bits corresponds to a 1 bit in the message.

Chaum (1988) proves that, under the assumption of a reliable broadcast channel and collusion of no more than $n-2$ participants in the network, this approach provides unconditionally secure sender and recipient anonymity.

Dining cryptographer networks are impractical for large-scale systems. The requirement for a reliable broadcast channel between all participants causes the protocol to be fragile against active attackers who can flip bits and thus potentially gain information. Waidner and Pfitzmann (1990) present some approaches to resolving this problem, however there is no entirely satisfactory solution to date.

In addition to being fragile, dining cryptographer networks are extremely inefficient. Messages are passed at the rate of one bit per round on the network, and rely on a large network of shared keys. Approaches to solving this problem are presented by Chaum (1988), however the logistics of achieving these on a large-scale network are far too complex for widespread implementations to be considered.

Despite the inefficiencies of dining cryptographer networks, they do provide an effective method to unlink actions and actors. For small-scale systems, such as board room voting, an implementation of the dining cryptographer network could find use.

3.4.2 Onion Routing

Onion routing aims to overcome the high latency inherent in mix systems. The design was proposed by Goldschlag et al. (1996) as a method for hiding routing information in applications that demand almost real-time network connections.

In onion routing, a data payload is wrapped with successive layers of encryption corresponding to nodes in a chain of communicating servers. The resulting packet, with multiple layers of encryption, conceptually resembles an onion. Encryption is performed by the first node of a routing network, which acts as a proxy server for future traffic. Each node along the route peels off a layer of encryption, and forwards the resulting payload to the next node in the chain. To defeat analyses of the decreasing size of each packet, nodes pad outgoing data to a constant size.

Pure onion routing differs from a typical mix system, as described in Section 3.1, in that an onion routing node does not necessarily perform explicit mixing operations; onions are directly forwarded to the next node.

Onion routing functions via *virtual circuits* that are set up by initiators before any content data is transmitted. Each node stores the predecessor and next nodes in a circuit along with decryption keys for data. The initiator of the connection encrypts its data with the successive encryption keys of the routers in the chain and sends the data. Each onion routing node decrypts the outermost layer and forwards the data to the next node in the circuit. As each node is only aware of the next node in the circuit it is impossible for an attacker to determine the entire path without compromising either the entire circuit or the end-point nodes.

Onion routing is the subject of much ongoing research (Syverson et al., 2000; Clayton, 2003; Dingledine et al., 2004a) as a method to achieve relatively strong anonymity without the latency overheads of mix systems. The reference implementation of an onion routing system is the Tor project, described in detail in Section 3.5.5.

3.4.3 Crowds

Reiter and Rubin (1998) proposed the Crowds system as a method for providing anonymous web-browsing capabilities. The system relies on a network of nodes, referred to as *jondos*, that forward web requests for other participants; each actor functions as a node for other actors.

The crowds system builds persistent virtual circuits through a network of nodes for each initiator by re-routing requests randomly. The aim of this random path selection is to obscure the location of the original request. When a user attempts to connect to a recipient through the network, a circuit is initiated by selecting a random node. With a certain probability, that node either forwards the request to another node, or fetches the website itself and returns the information to the original requesting node. A circuit identifier is maintained between these nodes, allowing the same path to be reused when the initiator requests new data. The path is periodically rebuilt as a measure against long-term statistical attacks.

The crowds system is designed to overcome the threat model of multiple corrupt nodes, and is not resistant against the global passive attacker. The global passive attacker is able to observe the flow of message requests and so trivially trace the initiator, thus breaking the anonymity of the system.

Shmatikov (forthcoming) shows that corrupted nodes can collude to increase the likelihood that a given member of the crowd originated a request, however the system is designed to provide a reasonably high level of protection against this *predecessor* attack. The predecessor attack, which is identified in the original paper by Reiter and Rubin (1998), is analysed in greater detail by Wright et al. (2002) who also apply the attack to a variety of other anonymity systems.

3.5 Deployed Anonymity Systems

Chaum (1981) sparked serious research into online anonymity. Since then a number of anonymity systems have been implemented and actively deployed. The most famous historical anonymity services, and the leading contemporary approaches, are reviewed here.

3.5.1 Remailers

The earliest deployed systems that provided anonymity to their users with the techniques that we have discussed were *remailers*. These systems were a development from real world postal remailers used to forward letters anonymously, and applied the concept to electronic mail.

Type-0 (`anon.penet.fi`)

The original electronic anonymous remailer system was a single machine run by Johan Helsingius in Finland. This machine operated a simple remailing policy in which users could send the server an email containing an extra header specifying the ultimate destination of the message. The server stripped technical identifying information from the email before forwarding the email to its destination. A simple reply service was also provided by which a user could set up a pseudonym on the server. All email to the pseudonym would be forwarded to the owner's real email address.

This system was tremendously popular, however the single server was open to both legal and technological attacks. Notably, the nature of the server caused the administrator to be in possession of the identities of all users.

`anon.penet.fi` was eventually shut down due to a combination of factors resulting from legal attacks, and excessive load on the server and its operator. A user of the service began anonymously distributing material from the secret teachings of the Church of Scientology. The church responded by suing the user. In the course of the legal case, the logs and data stored on `anon.penet.fi` were subpoenaed. This, in combination with allegations of the distribution of child pornography through the server, caused its eventual closure in 1996 after three years of providing anonymous service.

Type-I (Cypherpunks)

Members of the online cryptography advocacy mailing list Cypherpunks developed and deployed the *Type-I* or *Cypherpunks* remailer network. This network used multiple servers and encryption for all messages. By encrypting emails with the keys of a number of servers, messages could be passed through a series of nodes before reaching the final destination. This removed the critical flaw of reliance on a single server with access to message content that eventually caused the demise of the original Type-0 remailer.

The Cypherpunks remailer system also made use of *reply blocks* to allow anonymous replies to emails. A reply block is a series of routing instructions that allow the message to be delivered to a pseudonym. This data is included in anonymous messages when sent, allowing the recipient to reply without knowing the identity of the sender.

Type-II (Mixmaster)

Despite the lessons learnt from earlier work, the Type-I remailer network was open to many attacks on the anonymity of users. In 1994, in an attempt to rectify these problems, Cottrell developed the Type-II *Mixmaster* remailer (Mixmaster; Möller et al., 2003).

Mixmaster was an improvement over the older Type-I remailer. Firstly, it enforced fixed-length messages, with shorter messages being padded and larger messages being broken up into multiple messages. This defeated trivial traffic analysis based on message size. Secondly, Type-II remailer network nodes each contain a message pool to increase the difficulty of performing traffic analysis. This made Mixmaster the first widespread implementation of the mix design of Chaum (1981).

Type-III (Mixminion)

Mixminion is a remailer network jointly designed by members of the academic and remailer communities, and improved on the Type-II remailer by including a secure reply capability (Danezis et al., 2003) as well as other improvements in mixing strategy and security for users.

Many of the features used in Type-II remailers are employed by Mixminion to transport messages, however Mixminion added improved encryption for links between mixes and frequent rotation of authentication keys for each mix in order to prevent known attacks against earlier systems.

Mixminion includes a secure reply capability for anonymous messages through *single-use reply blocks.* In this scheme, a message path is split into two halves, the first of which is encoded into a reply block included with the message. When sent, a message travels along its intended path normally. When an anonymous reply is formed, the path encoded in the reply block is used to determine the second half of path along which the reply travels. The message therefore returns to the original sender along part of the original route, with the first half of the return route being randomly chosen by the mix. By treating replies no differently from standard messages when they are passed through the network, this scheme allows forward and reply messages to be indistinguishable.

3.5.2 Freenet

Clarke et al. (2000) presented an anonymous document publication and distribution service that uses encrypted data storage, geographical distribution, and anonymous communication between nodes. Anonymity within the network is provided both to the authors and to the readers of documents.

The original Freenet protocol is most similar to the Crowds system of Reiter and Rubin (1998), with requests randomly redirected between nodes before reaching their destination. As such, the anonymity that the system provides is vulnerable to many of the attacks that mixes and onion routing seek to prevent.

3.5.3 JAP

JAP (2004) is an anonymising proxy server developed and hosted at the Technical University of Dresden, with certain nodes on the network hosted by other universities. The JAP software uses a mix cascade approach to provide anonymity, however the application of the network is geared towards low-latency requirements such as web browsing.

JAP gained some notoriety in 2003 when the German Federal Bureau of Criminal Investigation successfully ordered the JAP service to record data concerning connections for future analysis. This ruling was later overturned by the District Court in Frankfurt (JAP Ruling, 2004). The legal situation brought about by JAP highlighted the non-technical issues that surround anonymity and privacy technologies.

3.5.4 PipeNet

PipeNet was proposed by Dai (1996) as a real-time anonymity system comparable to onion routing. Although the system design was presented only in a rudimentary form, PipeNet is notable for its strong reliance on dummy traffic. PipeNet, similar to onion routing, maintains virtual circuits between an initiator and a responder. These circuits form persistent encrypted connections that spans several intermediary nodes.

The PipeNet design enforces that each node on an established virtual circuit should receive one packet of data tagged with that circuit's identifier per timeslice. Packets are then randomly reordered in the style of a standard mix and sent out. In the case that a node does not receive a full quota of packets from each circuit it will queue those packets that it has received until the next timeslice.

The purpose of this strategy is to ensure that each circuit handled by a node produces a uniform stream of traffic. Such a stream is therefore hardened against timing or traffic analysis that can used to compromise other real-time anonymity systems such as onion routing.

The PipeNet design was never implemented, although a similar approach was employed by an early version of Zero Knowledge Systems' "Freedom Systems" architecture (Boucher et al., 2000). This design made use of traffic padding between links rather than the end-to-end padding of PipeNet. The use of traffic padding was subsequently dropped from the architecture entirely, due to its bandwidth requirements.

3.5.5 Tor

Tor (Dingledine et al., 2004a) is an implementation of an onion routing network designed to allow anonymous access to services that require low-latency connections.

Tor presents a scalable implementation of onion routing that uses standard proxy protocols to anonymously forward existing services. The combination of low-latency overheads of the network with the integration of the system into a standard proxy format and a simple user interface has allowed

Tor to be more widely adopted than many previous anonymity services, which have traditionally been weighed down with complex requirements for users.

Tor can provide anonymity not only for the user of the system, but also can obscure the location of services such as web sites offered through the network. This is achieved through the ability of a server to extend a circuit into the network with no fixed client on the other end. Circuits in onion routing are discussed in Section 3.4.2. A client using Tor can connect to this extended service and relay requests through to the server without knowing the server's address. This provides censorship-resistance capabilities.

3.6 Summary

In this chapter we have examined a variety of anonymity systems. Many of these are variations on a mix-based architecture, however we have also described systems based on alternative approaches to anonymising users. These systems, and their respective advantages and disadvantages, form the basis of our later modelling work.

We now examine approaches towards the classification and simulation of anonymity systems and their properties, and explore how these relate to the anonymity technologies described in this chapter.

Part II

Simulation

Chapter 4

Anonymity Mechanisms

4.1 Introduction

Based upon the survey of the previous chapter, we now discuss four underlying mechanisms for providing anonymity in communicating systems. For the purposes of this work, we consider a system to provide anonymity if it prevents, to a greater or lesser extent, observers from determining the linkage between the senders and recipients of messages passing through that system.

For illustration we consider systems in the form of a traditional computer network, comprising a number of hosts connected by channels across which messages can be passed. To anonymise such systems, the desired effect is to introduce confusion as to the ordering of messages passing across the network, when considered by an observer viewing all traffic passing across the network channels. More simply, an observer with global access to network traffic must not be able to link incoming messages to outgoing messages by observing the order in which they are sent and received.

The mechanisms that we present here have been inspired by four fundamental mechanisms for introducing nondeterminism into processes in Hoare's Communicating Sequential Processes (CSP) process calculus (Hoare, 1985). In this work we focus on a probabilistic and simulation-based interpretation inspired by these mechanisms, rather than a formal mathematical abstraction.

We will now describe each of the four mechanisms that we have identified.

4.2 Hiding

To confuse observers regarding the relationship between messages passing across the network, some messages may be *hidden*. As such, from the viewpoint of an observer, such messages happen *eagerly* and cannot be blocked or delayed by the environment.

As observers are unaware of the details of hidden messages, it becomes impossible to deduce directly the relation of input messages to output messages, although the flow of observable messages may, in some circumstances, allow an observer to deduce some information concerning the *occurrence* of hidden messages.

Hiding achieves anonymity through the restricting of observations possible by an attacker. This is most commonly expressed as a non-global attack model that hides certain actors from an observer's view. By restricting the attacker to an incomplete knowledge of the system, confusion is introduced as to the source of messages.

Existing anonymity systems do not typically rely on the hiding of messages as their basic strategy to achieve anonymity; indeed, many systems are designed to provide anonymity even when an attacker can view all network traffic. In almost all cases, however, hiding is assumed as a secondary method to increase the confusion introduced into the system.

The most notable current example of a system that relies inherently on a non-global attacker is the second-generation onion-routing project: Tor (Dingledine et al., 2004a). In order to achieve the low-latency traffic required for interactive usage such as web-browsing, Tor leaves itself open to traffic confirmation attacks that allow an observer to link senders and recipients by observing the arrival rates of packets passing between them. Despite this possibility, the scale and decentralised nature of the Internet prevents such an attack from being feasible for all but the most powerful adversaries, and the Tor system can claim to offer reasonably strong anonymity in the general case.

4.3 Spurious Events

As a contrast to hiding, that may be considered as removing messages, we can also *obscure* messages to introduce confusion as to their source.

Spurious messages can be generated from a set of potential messages and injected into the system. Any message from that set is thus obscured: it is impossible to tell whether an observed message was a genuine message from some user of the system, or a spurious message introduced in order to confuse observers.

As spurious messages are still visible to an attacker, their behaviour remains observable and potentially alterable. Spurious messages, from an attacker's point of view, are subject to all the delays or retransmissions of genuine messages. A more powerful attacker with the capability to delay, modify or delete messages can interact just as easily with spurious messages.

This strategy is typically referred to as *dummy traffic*. Any messages observed by an attacker may be the result of the passage of genuine messages, or may be the result of fake messages generated either by intermediary nodes or by initiators. Without considering real-world implications of bandwidth, spurious events are an extremely powerful strategy for confusing observers.

Of particular interest is that spurious events prevent an observer from deducing that a genuine event *has ever occurred*. In all other approaches towards providing anonymity, the observation of a message provides an attacker with the basic information that a message was sent. If any observed message may be spurious, however, the attacker cannot even deduce this.

The problems of a system that implements such a strategy are easily seen. Systems that constantly send messages without genuine content are unlikely to be popular with system operators or other network users. In some cases, however, an implementation of this type of system may not always be considered infeasible as bandwidth capabilities increase. Additionally, there are subtle factors that may allow observers to distinguish traffic even in the presence of high volumes of dummy traffic that make this strategy far from a panacea.

The unreasonable costs of introducing a high level of spurious events have prevented the development of systems that rely solely on this strategy to gain anonymity. The dining cryptographer network presented by Chaum (1988) may be considered as using a form of this strategy, albeit combined with other factors, but is rarely deployed in real systems. Due to the inherent impracticality of dining cryptographer networks we have chosen not to focus on applying our model to this case.

4.4 Pull Technologies

Many of the strategies employed by existing anonymity systems, and many of the attacks and defences, assume that messages are “pushed” through the network. This typical model relies on messages that travel through a network with each node actively forwarding traffic according to timing rules and flushing strategies. In contrast to this, a system may simply make messages available in such a way that potential recipients can “pull” data out of the network as they require.

The confusion in this approach is achieved *externally*. It is the environment, consisting of potential recipients and other nodes, that determine the ordering of output messages, and not the node that currently holds the message. In real-world systems this describes a variation on the standard mix, but one that works according to a "pull" strategy – messages are pooled in the mix until they are requested by another node. The behaviour of these other nodes, and receivers, is what causes the sequence of messages to be re-ordered, not the internal behaviour of the node itself.

The use of this strategy in providing anonymity is not widely seen in real-world systems. Some forms of private information retrieval and message-board systems, where users collect messages rather than receiving them automatically, may be considered as employing this strategy.

The potential advantage of such systems is that the responsibility for introducing confusion into the sequence of messages does not rest solely on the action of a single node; it is a property that emerges from the interactions of a large number of users. This has the effect of decreasing the reliance on any single node, whilst also potentially increasing the randomness in the system by relying to a greater extent on the actions of unpredictable users.

4.5 Internal Randomness

Internal random, or pseudorandom, choice represents the most commonly fielded strategy for achieving anonymity in existing systems. Chaum's mix and its many variants are systems in which messages are stored by a network node until some criterion is reached, whereupon the internally stored messages are sent out in random order. An attacker is, theoretically, unable to determine the exact correspondence between individual input and output messages.

It is also possible for this internal randomness to be applied to different aspects of the system than the ordering of messages. A system may choose not to re-order messages, but to decide whether or not to forward at all. This can be considered the opposite of spurious events—causing confusion by dropping messages rather than injecting them. Of course, this example creates a system that is largely impractical due to the functional requirement that messages entering the system should, at some point, be delivered to their final destination.

Despite the clear limitations to such a scheme, there is some potential for this idea in applications that do not have a requirement for perfect delivery. Certain networked applications allow for a number of messages to be dropped provided that the overall stream of data is of sufficient quality. Similarly, spurious events, being superfluous to the "genuine" flow of data, may also be dropped from those systems where they are employed. While it is likely that an infeasibly large number of messages would need to be dropped in order to gain any useful effect, this approach should not be entirely ignored.

Another system that can be considered as relying on internal choice, is the node-level routing behaviour of the Crowds system of Reiter and Rubin (1998). As an alternative to dropping messages this strategy makes a decision either to forward each message immediately to its ultimate recipient, or to another node in the system. The Crowds system therefore cannot function as a single node, relying instead on the interaction between multiple nodes passing messages.

The Crowds system causes messages to be randomly routed through a number of intermediary nodes before finally being delivered to the recipient. This random routing is designed to prevent an observer from tracing a single message. The system relies on a non-global attacker view, or the "hiding" strategy, as each individual node still functions as a deterministic buffer that cannot individually reorder messages.

4.6 Conclusion

The four strategies outlined above broadly cover the range of possible approaches that may be taken to anonymise users in communicating systems. Some real-world systems may focus on a single strategy that neatly fits into the classification presented here; others may combine strategies in order to exploit advantages or alleviate disadvantages of the individual approaches. In the next chapter we develop a method by which we can effectively quantify the anonymity provided by differing strategies in order to compare their relative effectiveness in anonymising users.

Chapter 5

Analysis Methods

5.1 Introduction

The previous chapter described the nature of anonymous communication systems according to a classification based on the fundamental strategy that they employ to confuse an observer as to the flow of messages. In this chapter we develop this idea further into a form suitable for quantitative analysis via simulation and detail the analyses that we wish to perform on the behaviour of these systems. This provides us with the required form for the simulation experiments.

Our view of anonymity declares that for a system to provide anonymity for some sequence of messages, it must randomly produce one of a number of possible output sequences for that input sequence. The purpose of these multiple output sequences is to confuse an attacker as to the possible location of a given input message in the output sequence.

In order to quantify this randomness, and hence the anonymity of the system, we are concerned not simply with the number of possible output sequences, but with the level of confusion as to the possible location of any given message in the output. As has been demonstrated by Diaz et al. (2002) and Serjantov and Danezis (2002), the size of the set of participants that could have sent a message fails to provide an accurate quantification in a number of important cases. By examining the likelihood of a given message coming from a given potential sender, a more useful characterisation of the level of anonymity can be reached.

This can be demonstrated by considering a buffer that may produce, with equal probability, either of two permutations of some input sequence; and a second system that produces one of six given permutations for an input sequence, but with a probability of 0.9 of preserving the input ordering. It can be shown that the second system is likely to be less effective in confusing an observer than the first, despite there being a larger set of possible outputs. The details of these calculations are shown in Section 5.2.1.

To measure the confusion in such systems, we turn to information theory as presented by Shannon (1948). The information theoretic entropy of the output sequence of some system with respect to the location of messages in the input sequence provides a quantification of the level of uncertainty that an attacker has as to the location of an observed input message in the output sequence.

The nature of the specific choice of tests, and how they are simulated and analysed, are discussed in greater detail below.

5.2 Shannon Entropy

Shannon's work on information theory is concerned with the quantification of the amount of information in systems and has found widespread use in a variety of fields, but was originally developed to calculate the capacity of communication channels and the limits of compression. The key concept in information theory, *entropy*, provides a measure of the amount of uncertainty associated with a discrete random variable, representing the average amount of information required to describe the behaviour of that variable. Information entropy is traditionally measured in binary *bits*.

The Shannon entropy, H, of a discrete random variable X is:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{5.1}$$

where $p(x)$ is the probability that $x \in X$ occurs.

5.2.1 Examples

The classic illustration of information theoretic entropy is its use to describe the average information required to report the winner of a horse race. The following example is based on that found in Cover and Thomas (1991):

In the first case, consider a horse race in which eight equally matched horses compete. In such a case, each horse has an equal probability of winning. The entropy in the system, according to Equation 5.1, can be calculated:

$$\begin{aligned}
H(X) &= -\sum_{x \in X} p(x) \log_2 p(x) \\
&= -\sum_{i=1}^{8} p(win_i) \log_2 p(win_i) \\
&= -\sum_{i=1}^{8} \frac{1}{8} \log_2 \frac{1}{8} \\
&= -\sum_{i=1}^{8} \frac{1}{8} \times -3 \\
&= 3 \text{ bits}
\end{aligned}$$

As might be expected for a set of 8 uniform outcomes, we require 3 bits of information to describe the system.

In the case that probabilities are non-uniform, fewer bits are required to represent the outcome of the system. For a horse race with winning probabilities $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})$, the entropy may be calculated as follows:

$$\begin{aligned}
H(X) &= -\sum_{x \in X} p(x) \log_2 p(x) \\
&= -\sum_{i=1}^{8} p(win_i) \log_2 p(win_i) \\
&= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{1}{16} \log_2 \frac{1}{16} - 4\frac{1}{64} \log_2 \frac{1}{64} \\
&= 2 \text{ bits}
\end{aligned}$$

Message Reordering Examples

Equation 5.1 may be used to verify the example of message reordering systems from Section 5.1: The first system presented there offers two possible orderings of output with equal probability. We therefore calculate $H(X_1)$ in the following way:

$$\begin{aligned}
H(X_1) &= -(0.5 \ \log_2 0.5 + 0.5 \ \log_2 0.5) \\
&= 1 \text{ bit}
\end{aligned}$$

The amount of information required to describe the output of the first system is 1 bit.

The entropy of the second system, presenting six possible orderings, can be calculated as follows:

$$\begin{aligned} H(X_2) &= -(0.9 \ \log_2 0.9 + 5 \times (0.02 \ \log_2 0.02)) \\ &\approx 0.7 \text{ bits} \end{aligned}$$

On average, this system requires roughly 0.7 bits of information to describe its output.

In the simulation experiments, we examine the distance between each pair of messages in the output sequence that were consecutive in the input sequence. This investigates the spreading effect on such message pairs by the system, regardless of their overall position. Over a large number of simulation runs, we build a distribution of the output distances for all such message pairs in an experiment. Shannon entropy provides a method for quantifying the amount of uncertainty that we observe in these distributions.

The specific focus on consecutive pairs of input messages is useful in that it represents not only an underlying feature of the system under investigation, but also represents a very simple *tagging attack* that is applicable across a wide variety of systems. A minimally active attacker can gain information from a system by injecting a message into the system directly after an interesting target message. By observing the output position of this message, the attacker hopes to gain some information as to the possible location of the target.

5.2.2 Conditional Entropy

If we are interested in two potentially related random variables we may consider their *conditional entropy*. Specifically, we may calculate the conditional entropy of one of the variables *given* the second variable. This quantity provides a measure of the amount of information that is revealed about the second variable given a knowledge of the first.

As an illustrative example of conditional entropy, consider a bag containing two red balls and a blue ball. If a ball is randomly drawn from the bag, the probability that the ball is blue is one-third and the probability that the ball is red is two-thirds. This is case of standard entropy. From Equation 5.1 above, we can calculate the information entropy of the system to be:

$$\begin{aligned} H(X) &= -\left(\frac{1}{3} \ \log_2 \frac{1}{3} + \frac{2}{3} \ \log_2 \frac{2}{3}\right) \\ &\approx 0.92 \text{ bits} \end{aligned}$$

On average, we will require 0.92 bits of information to describe the outcome of this system.

If the experiment is repeated after replacing the ball that was drawn, the probability of drawing a red or blue ball remains the same. If, however, the ball is not replaced then we gain information regarding the possible outcome of the second draw.

If we drew a blue ball in the first experiment, the only possible outcome of a second draw is a red ball. If the first ball was red, there is now an equal chance of drawing either a red or a blue ball; the outcome of the second experiment is *conditional* on the outcome of the first experiment.

The conditional entropy of two discrete random variables is:

$$H(Y \mid X) = -\sum_{x\in X}\sum_{y\in Y} p(y,x)\ \log_2 p(y \mid x) \tag{5.2}$$

For the case the above experiment without replacement of balls, the conditional entropy of the second drawing given the first is therefore:

$$\begin{aligned} H(Y \mid X) &= -\big(p(b)\big(p(r \mid b)\log_2 p(r \mid b)\big) + \\ &\quad p(r)\big(p(r \mid r)\log_2 p(r \mid r) + p(b \mid r)\log_2 p(b \mid r)\big)\big) \\ &= -\left(0 + \frac{2}{3} \times \left(\frac{1}{2}\ log_2\frac{1}{2} + \frac{1}{2}\ log_2\frac{1}{2}\right)\right) \\ &= \frac{2}{3}\ \text{bits} \end{aligned}$$

If we know the outcome of the first experiment, the average number of bits required to describe the output of the second experiment will be $\frac{2}{3}$. Note that the first term in the above calculation reduces to 0. This corresponds to the case that the first experiment resulted in a blue ball. In this case the output of the second experiment *must* be a red ball, and all information about the state of the system is known. The remaining case, where the first ball drawn was red, results in a system in which there is an equal choice of the two remaining options.

To relate this to a simulation approach, we may wish to describe the distance between a pair of messages in the output sequence *given* that we know the distance between those messages in the input sequence. This factor would be of interest to an attacker seeking to gain statistical knowledge of the system for the purpose of relating output messages to input messages.

5.2.3 Experimental Factors

The simulation experiments that we perform are designed to demonstrate the behaviour of various anonymity systems. Due to the rate at which the number of possible permutations of output increase, it is not feasible for us directly to calculate the distribution of output produced for a given input sequence. We therefore select a representative feature of the systems that allows us to explore the level of randomness produced by varying factors.

We use information entropy as a measure of the confusion that would be experienced by an attacker in attempting to predict the behaviour of the output ordering given knowledge of the input ordering of the entire system. Again, it is not feasible to calculate this property directly for systems large enough to be of any interest. We therefore choose to examine the behaviour of pairs of messages that enter the system consecutively. We are interested in the distance between those two messages on leaving the system, and the variation in those distances as each simulation is repeated multiple times.

This method is approachable computationally, and has the advantage of representing a simple attack possible against any anonymity system that fits into the abstract model of Chapter 4. In a real-world system, considering consecutive pairs of input messages is equivalent to an attacker "tagging" a message by injecting their own message into the input sequence directly following some target message.

We represent the amount of confusion concerning the possible distances in output between consecutive input messages by calculating the entropy of the distribution as estimated over a large number of simulation runs. As the input distance between the messages is always one, and thus is not a random factor, the conditional entropy of the output distances given the input distances is equivalent to the simple entropy of the output distances[1].

An ideally anonymous system, according to this quantification, demonstrates a uniform distribution of probabilities for the possible output distances between messages. For each pair of messages that enter the system, there should be an equal probability of any possible output distance. Intu-

[1]If we consider equation 5.2 with a single possibility for the result of the first variable:

$$\begin{aligned} H(Y \mid X) &= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) log_2 p(y \mid x) \\ &= -1 \times \sum_{y \in Y} p(y \mid x) log_2 p(y \mid x) \\ &= -\sum_{y \in Y} p(y) log_2 p(y) \end{aligned}$$

This is equivalent to equation 5.1.

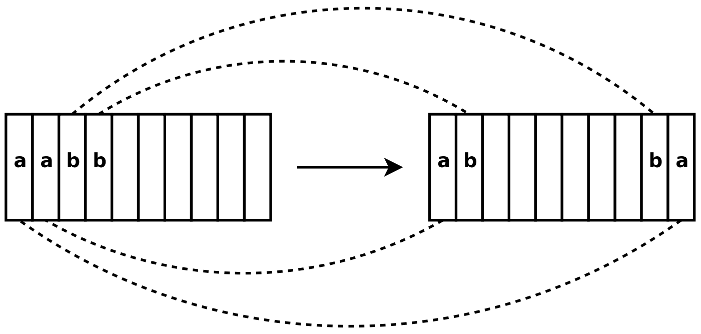


Figure 5.1: Output distances between perfectly mixed messages.

itively, this is the behaviour that we would expect from a MIX node with a sufficiently large threshold—every message has the possibility of exiting the system at any point in the output sequence.

In our simulations we consider not individual pairs of consecutive messages but *every* consecutive pair of messages in a run, and thus the output distance between each pair is linked to the output of each other pair. If we consider a system that accepts ten messages which are then randomly reordered in the output, it is clear that a distance of nine messages between input and output can occur in only one case. Once a pair has occupied this possibility, the remainder of messages are constrained to a lower distance. This is demonstrated in Figure 5.1. If the pair of messages marked **a** are spread by the maximum distance, the pair marked **b** cannot possibly be equally far apart.

The spread of results produced by our simulation and analysis therefore represent the combination of probabilities for *all* messages in the system. When considered as a complete system, we expect the spread of observed output distances to fit as closely as possible with the possible number of output distances available. In short, we expect the system to *maximise* the entropy of output distances by fitting as closely as possible with a uniform spread of outputs.

The relation between individual output possibilities and the combination of all output possibilities is shown in Figure 5.2. Note the gap in the centre of each histogram showing the impossibility of two messages occurring at the same position in the output sequence.

A number of approaches to quantifying the level of randomness in the output sequence were considered in preliminary experiments.

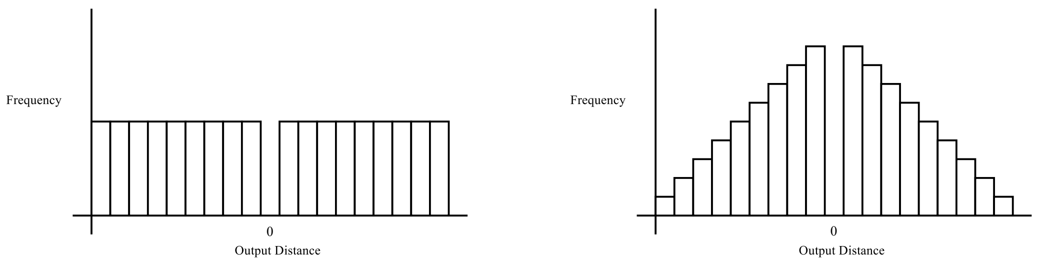


Figure 5.2: Output pairings in a perfect mixing system (left) and output pairings of all consecutive pairs in a perfect system (right).

Random pairing

To explore the entire range of potential input and output, one possible approach is to select random pairs from the input compared against their respective distances in the output sequence. By selecting initially random pairings, however, no information is gained from the input sequence; if message distances from the input sequence are drawn with equal probability then the distances in the output will typically also display only the spread of possible distances. Such selection reveals little more than the possible spread of distances in the output sequence without revealing any underlying structure of the likelihoods of such outcomes with respect to the input distances. We therefore reject random selection of pairings as a useful method to reveal the behaviour of the systems from our model.

Structured Pairing

A more structured approach is the selection of a number of particular positions in the input sequence. The comparison of output distance to input distance can then be made for all combinations of any two messages that fall at these positions. This approach suffers from arbitrarily enforcing the selection of pairs which have no special significance to the nature of the system, and artificially selects for pairings of particular distances. In the absence of any compelling reason why this selection of pairings should be chosen, the approach was discarded.

Non-consecutive Pairing

As a natural extension of the consecutive pairing approach, it would be possible to select pairings that are separated by a given number of messages and to process all such pairings. There is nothing intrinsically wrong with this approach, although the usefulness of ever increasing input distances seems small when considered in light of the computational requirements of such an exhaustive approach.

Consecutive Triples

A second possible extension from consecutive pairings would be to examine consecutive triples rather than pairs. The relative spread of triples in the different systems could potentially reveal more information concerning the spread of possible output messages. Taking this idea to its logical extreme is, of course, to examine the entire set of possible unorderings of the input sequence. The computational requirements for such an approach, of calculating the distribution over all possibilities, are clearly infeasible. With these considerations in mind, consecutive pairings appear the most viable compromise between computational constraints and information revealed. Examining the distribution of triples of messages would, however, be a simple possibility for future work based on this approach.

5.3 Building the probability distribution

Our simulation experiments are capable of giving only an approximation to the genuine probability distribution of output message pairings in the system. As such, we use the simulation results to build an approximate set of probabilities that may then be used to calculate an approximation to the entropy of the real system.

Our method of estimation is simple: each experiment is repeated a large number of times and the distance between consecutive message inputs is tallied. The tally of results is then averaged and normalised to result in an approximation of the probability distribution of output pairings for that system. To counteract potential effects of a single set of initial starting conditions, this procedure is repeated for a number of randomly generated messages taking random paths through the network. The mean average and standard deviation of these results are used in the final comparison of system effectiveness.

5.3.1 Example

We show here a simple example of experimental results being used to calculate the probability distribution as described above. The details of the simulator and its input and output are shown in detail in the next chapter.

Raw Results

A set of results from the simulation experiments is given in Figure 5.3. The preamble shows the setup parameters of the simulation. In the case shown, a single sender is created with a sequence of four messages. Each message travels over a single hop before being delivered to a receiver. A single mix node is created to pass messages. Finally, the list of actors in the system is reported, along with the type of each actor. Those actors marked "`SEND`" create messages, those marked "`PASS`" function as intermediary nodes and those marked "`RECEIVE`" are valid destinations for messages.

Below the preamble, details of three runs are shown. The events occurring in each run are reported in a time-ordered sequence. Each event takes the form `( <owner>.<id>, <sender>, <receiver> )` , where `<sender>` and `<receiver>` refer to the sending and receiving actors for that event, not for the message. The event `(0.02, 1, 2)` therefore refers to the third message[2] originally sent by actor `0` being passed from actor `1` to actor `2`.

These raw results are processed to form a tally of distances between messages in the output. This is achieved by identifying the sequence of input events, in which the sender of the event is of the type `SEND`; and the sequence of output events, taken as those events where the receiver is of type `RECEIVE`.

Each consecutive pair of messages in the input sequence are then identified in the output sequence. The distance between these messages in the output sequence, as measured by their location in the output sequence, is calculated. The distance between each pair of consecutive messages is calculated for the entire run of simulations.

This forms a set of initially processed results as represented in Figure 5.4. I_1, I_2, I_d refer respectively to location of the first and second messages in the pair being considered in the input sequence and the distance between them. As these messages are consecutive, I_d will always be 1. O_1, O_2, O_d give the corresponding locations in the output sequence of messages and their distance: $O_2 - O_1$.

Finally, the results shown in Figure 5.4 are aggregated to create a tally of the number of observed output distances over all runs of the simulation. This produces a set of results such as that seen in Figure 5.5. This provides an estimation of the output distribution for that set of runs.

The above procedure is repeated for a number of fresh simulations with different initial conditions, varying the paths taken by messages and the input ordering. The purpose of this is to reduce any effects that may be caused by particular message behaviours. For the experiments shown in Chapter 7, 30 sets of initial conditions were used in each.

[2]Message, and actor, identifiers are indexed starting from zero.

```
Creating 1 multiple sender(s). Messages: 4. senders: 1, hops: 1
Creating 1 threshold mix(es). Pool Size: 16, tock_limit: 16
Creating 1 receiver(s).
Actor 1: SEND
Actor 2: PASS
Actor 3: RECEIVE

Run 1 of 3.

(0.00, 0, 1)
(0.01, 0, 1)
(0.02, 0, 1)
(0.03, 0, 1)
(0.02, 1, 2)
(0.01, 1, 2)
(0.03, 1, 2)
(0.00, 1, 2)

Run 2 of 3.

(0.00, 0, 1)
(0.01, 0, 1)
(0.02, 0, 1)
(0.03, 0, 1)
(0.00, 1, 2)
(0.02, 1, 2)
(0.03, 1, 2)
(0.01, 1, 2)

Run 3 of 3.

(0.00, 0, 1)
(0.01, 0, 1)
(0.02, 0, 1)
(0.03, 0, 1)
(0.01, 1, 2)
(0.02, 1, 2)
(0.03, 1, 2)
(0.00, 1, 2)
```

Figure 5.3: Example results from a small simulation.

Run	I_1	I_2	I_d	O_1	O_2	O_d
1	0	1	1	3	1	-2
1	1	2	1	1	0	-1
1	2	3	1	0	2	2
2	0	1	1	0	3	3
2	1	2	1	3	1	-2
2	2	3	1	1	2	1
3	0	1	1	3	0	-3
3	1	2	1	0	1	1
3	2	3	1	1	2	1

Figure 5.4: Example calculated distances between messages.

Message distance	Number observed
-3	1
-2	2
-1	1
0	0
1	3
2	1
3	1

Figure 5.5: Example tally of output distances for a specific set of input messages.

The results of these multiple simulations are then combined by calculating the mean and standard deviation for each observed output distance, resulting in an approximate output distribution for the system with the standard deviation providing error bounds. This distribution is then used to produce both histograms showing the distribution of output, and to calculate the information theoretic entropy of that system.

The data is in an appropriate form to produce a histogram, an example of which is shown in Figure 5.6[3].

[3]This figure shows the results of a 4-node mixing system with a threshold of 16 messages processing 64 messages. The simulation was run 1024 times for each of 30 initial starting conditions.

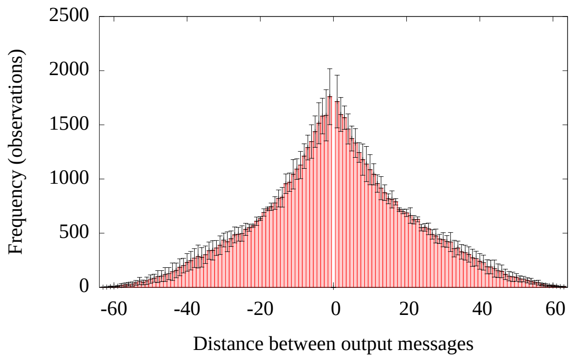


Figure 5.6: Example output graph from a set of simulations.

5.4 Result Significance Testing

The simulation experiments investigate the level of randomness introduced into the sequence of messages by a variety of anonymity strategies. It is desirable to apply statistical testing to ensure that the results of the simulation experiments are a true representation of the underlying structure of the systems, and not the result of random effects.

A feature of the systems under simulation is that the distribution of the spread of output messages is not known. Standard parametric statistical tests rely on a known parametrised distribution, most typically the normal distribution. As we know little about the distribution of results beyond the fact that it is not normal, we must apply *non-parametric* testing to the results of the simulations in order to gain confidence in the results.

5.4.1 Null hypothesis testing

The standard approach in statistical testing is to formulate a *null hypothesis* to be refuted by the observed data in support of an *alternative hypothesis.* The null hypothesis is presumed to be true until rejected at an acceptable, predefined level of confidence.

The null hypothesis presents a default assumption for the behaviour of a system. Suppose that a comparison is made between two populations with respect to some feature of interest. As an example, we may wish to compare the number of encrypted emails sent by samples of two populations, one a sample of computer scientists and the other being a sample of philosophers. The null hypothesis would be that both samples were drawn from the same distribution; to which we would then apply an appropriately chosen statistical test for homogeneity at a given confidence level.

The results of such a test allow us either to reject the null hypothesis at the given level of confidence, or *not to reject* the null hypothesis at the given level of confidence. The applied test cannot cause us to *accept* the null hypothesis, nor can we take the lack of rejection as supportive of a prior belief in the validity of the null hypothesis.

The confidence level of a statistical test represents the probability that the results of the test may be incorrect. At a confidence level of 90%, there is a 10% chance that the rejection of the null hypothesis was caused by the result of a statistical fluctuation in the data. Typically, a test for the homogeneity of two distributions relies on observing a sufficiently large deviation between the two distributions. At higher and higher levels of confidence, the size of the deviation required in order for the test to reject the null hypothesis becomes increasingly large; rejecting the null hypothesis therefore becomes increasingly difficult.

5.4.2 Statistical tests for homogeneity

We are interested in comparing the results of our simulation runs against each other in order to test whether the variation in the systems is reflected in the output distribution of message pairing. There are a number of common statistical tests used to compare two such data sets. The tests that were considered for application to the results of the simulation runs are briefly discussed here. A useful overview of the methods shown here is the work of Siegel (1956).

Student's t Test

One of the most well-known tests for equality between two distributions is Student's t-test, developed by Student (1908), although the name itself applies to a family of tests with slightly differing assumptions. Student's t-test applies the null hypothesis that the means of two normal distributions are equal, assuming equal variance of the distributions. An alternative to Student's t-test, known as Welch's t-test, relaxes this assumption of equal variance.

A fundamental assumption of all Student's t tests is that the two samples are drawn from normal distributions. This makes the test inapplicable to the analysis of the distribution of output pairing distances for the systems we have examined, as there is no guarantee that these distributions are normal.

χ^2 Homogeneity Test

The χ^2 test is an extremely popular nonparametric test for discrete or binned data that may be used to test the hypothesis that two samples are drawn from the same underlying distribution.

Although this test at first appears ideal for the sampled distributions produced by the output of the simulation experiments, the χ^2 test is largely applicable only for cases with a small number of bins that each contain a relatively high number of values. The underlying χ^2 distribution, on which the test is based, is parametrised by the number of *degrees of freedom* in the sample data. This parameter relates to the size and number of parameters in a given sample, representing the amount of information remaining in the sample as parameters are learnt.

The χ^2 test is mainly applicable to samples with relatively low degrees of freedom; it is thus not easily applicable to our simulation experiments with large sample sizes and a high number of bins. Indeed, the results of the simulator contain enough bins of data that they begin to approximate a continuous distribution, and the resulting degrees of freedom make the χ^2 test far less applicable. We therefore consider tests that treat data of this form.

Mann-Whitney U test

The Mann-Whitney U test (Mann and Whitney, 1947) is a nonparametric method for testing whether two samples are drawn from the same distribution, the null hypothesis being that this is the case. The test relies on testing that the probability of values from one sample being greater than that of the second sample is sufficiently close to 0.5.

The Mann-Whitney test is applied in situations where the underlying distribution is continuous or ordinal; that is, that there exists a total ordering amongst all observations. The fact that no two observations have the same value is of importance in the test itself, which relies on ordering the observations from both samples into a single ranked series.

This feature makes applying the Mann-Whitney test to the result of the simulation experiments awkward, as the binning of the data causes collisions between values of the two distributions. The Mann-Whitney test, therefore, is less forgiving of the discrete nature of the data from the simulation results.

Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is a nonparametric statistical test that allows for a variety of comparisons of two population samples where the underlying distribution of each is unknown. In the context of the simulation experiments detailed in this and the previous chapter, the Kolmogorov-Smirnov test allows for a degree of confidence in the differences observed between experimental runs with differing parameters.

The Kolmogorov-Smirnov test compares the cumulative frequency distributions of two samples in order to determine whether they were drawn from the same distribution. The null hypothesis in this case is that the distributions of the two samples are equal. If we observe a critically large difference between the two cumulative distributions then we can reject this null hypothesis and say that the distributions from which the two samples were drawn are different at the given confidence level.

5.4.3 Details of the Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test can be applied in a number of ways in order to reach various conclusions concerning sample data. In its one-sample case the Kolmogorov-Smirnov test is used to test goodness of fit between observed sample data and some hypothetical distribution, such as the normal distribution.

The two-sample case tests whether two sets of sample data are drawn from the same distribution. This test is based on the assumption that samples drawn from the same distribution exhibit similar cumulative frequency distributions. In the case that the two samples are drawn from different distributions, their cumulative frequency distributions are likely to differ by some critical value. The critical value in the Kolmogorov-Smirnov test, the D value, is drawn from a known distribution according to the level of statistical error that we are willing to accept for the test.

The two-sample test is performed in the following manner:

1. Build the cumulative frequency distribution for each sample.

2. Find the greatest difference observed between the value of the two cumulative frequency distribution.

3. Compare this maximum distance with the critical D value appropriate for the sample size and significance level.

The Kolmogorov-Smirnov two-sample test can be applied in two senses to ascertain different features of sample distributions. We examine these two methods here.

One-tail test

The *one-tail* test is used to determine whether the first sample's cumulative frequency distribution is *stochastically larger* than that of the second sample. A distribution X is said to be stochastically larger than a second distribution Y if, for the two cumulative distributions $F(X)$ and $G(Y)$, $F(A) \leqslant G(A)$ for all A.[4]

In the one-tail test both the sign and the value of the largest difference between the two distributions is considered, that is:

$$D = max\,(S_X(x) - S_Y(x))$$

As we are not generally concerned with whether a given sample is "better" than another in these tests, this information being more effectively gained from other sources, we do not apply the one-tail test here. The concept of one distribution being stochastically larger than the other would be misleading in this case as higher values are not "better" in our scale. Were we to plot the absolute, rather than signed, distance between output messages there would be some justification in applying the one-tail test, however we would have lost important information regarding the possibility of message positions being reversed in the output sequence.

Two-tail Test

The *two-tail* version of the Kolmogorov-Smirnov test is concerned solely with whether the two samples under investigation were drawn from the same distribution, and it is this test that we apply. The difference between the one-tail and the two-tail test occurs in the comparison of the maximum difference between the two cumulative frequency distributions. In the two-tail test, the absolute value of the largest difference is considered:

$$D = max\,|S_X(x) - S_Y(x)|$$

This test reveals whether the two samples are drawn from differing underlying distributions. The test makes no assumption of one distribution being better or larger than the other. It is this approach that we take towards analysing our data.

[4]The direction of the comparison operator initially appears counter-intuitive. For two cumulative frequency distributions, the distribution with a lower value at any given point has a higher proportion of its values concentrated in the higher end of the scale of possible results and is thus larger.

Significance Level	Critical D value
0.10	$1.22\sqrt{\frac{n_1+n_2}{n_1 n_2}}$
0.05	$1.36\sqrt{\frac{n_1+n_2}{n_1 n_2}}$
0.025	$1.48\sqrt{\frac{n_1+n_2}{n_1 n_2}}$
0.01	$1.63\sqrt{\frac{n_1+n_2}{n_1 n_2}}$
0.005	$1.73\sqrt{\frac{n_1+n_2}{n_1 n_2}}$
0.001	$1.95\sqrt{\frac{n_1+n_2}{n_1 n_2}}$

Figure 5.7: D values large enough to reject the null hypothesis at the given level of significance. n_1, n_2 refer to the number of observations in each sample. Values in this table taken from (Siegel, 1956).

Significance levels

When testing the null hypothesis, a statistical test is generally designed to inform us whether or not we have observed a result that allows us to reject that hypothesis. Taking a probabilistic approach allows the possibility that we will observe, by chance, results that represent the extremes of the underlying distributions from which we have drawn our samples. In these cases it is possible to gain a skewed perspective of the nature of such a distribution. Statistical tests therefore contain a parameter that takes into account the possibility of such extremely unlikely events occurring.

The Kolmogorov-Smirnov test compares the observed difference between the cumulative frequency distributions against a critical D value. If the difference between the cumulative frequency distributions exceeds the D value for a given significance level, the null hypothesis is rejected and the data is assumed to be drawn from different populations. A table of critical D values for the Kolmogorov-Smirnov two-sample test is given in Figure 5.7.

Again, it is important to note that null hypothesis in the Kolmogorov-Smirnov test is that the distributions of the two samples do *not* deviate. The results of the test are therefore that the null hypothesis can be rejected at the given α, implying that the probability that the distributions are different is at least $1 - \alpha$; or that the null hypothesis *cannot* be rejected at the given significance level, implying that the probability that the distributions are different is less than $1 - \alpha$. It is not possible for the test to accept or prove the null hypothesis.

In our tests, we choose a high significance level of 99%, $\alpha = 0.01$. This value represents a high confidence that the rejection of the null hypothesis is true, based on a large number of simulation runs for each variation in parameter.

5.4.4 Example Application

We show here an example application of the Kolmogorov-Smirnov test taken from the results of the simulation experiments.

In the first case, we do not expect the Kolmogorov-Smirnov test to reject the null hypothesis that two sets of results are drawn from the same distribution at the given confidence level. To demonstrate, we select the spread of output distances observed for consecutive messages under two runs of the simplest Mix experiment.

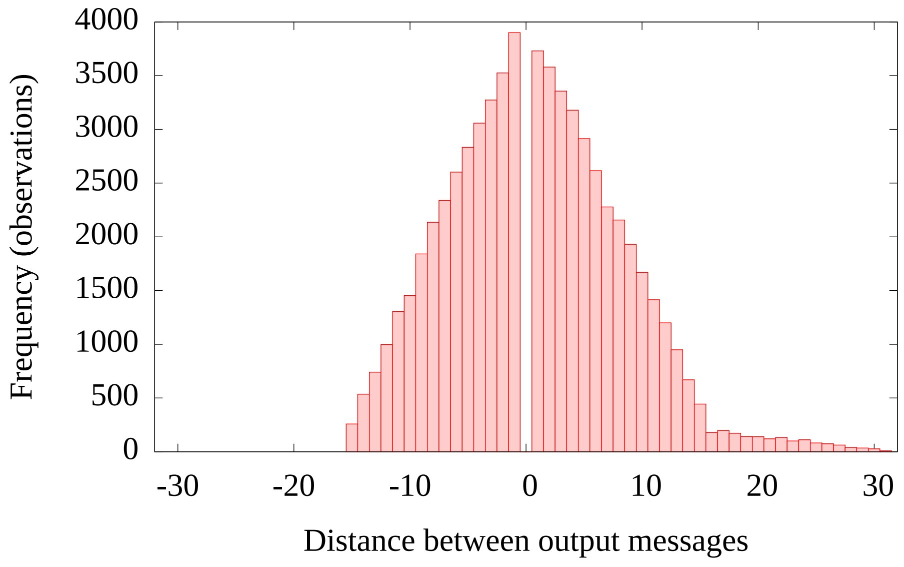
Equal distributions

Figure 5.8 shows the frequency of observed output distances from a single mix node with a threshold value of 16 messages. Over the course of the simulation, 64 messages are passed through the node. (Both histograms show the results of a separate single runs of the simulator.)

The output spread of the results is not of particular interest at this point, however it can be seen that the two graphs appear similar. Figure 5.9 shows the results of applying the Kolmogorov-Smirnov two-sample test to the two sets of aggregate data, the null hypothesis being that the two samples were drawn from the same distribution.

Note that the significance level is reported by the implementation as the probability of accepting an error, rather than the confidence level itself. We show several confidence levels for the purpose of illustration; our chosen confidence level, 99%, is marked with an asterisk. The values shown as a result of each test represent the D value for the samples, as described in Section 5.4.3, against the critical D value for that confidence level as shown in Figure 5.7. The null hypothesis is rejected if the D value for the samples is greater than or equal to the critical D value.

The null hypothesis that the two sets of data are drawn from the same distribution cannot be rejected for these two sets of results, even if we are willing to consider a relatively high probability that the observed result occurred by chance. This result is consistent with the idea that the observed graphs are representative of the underlying model.



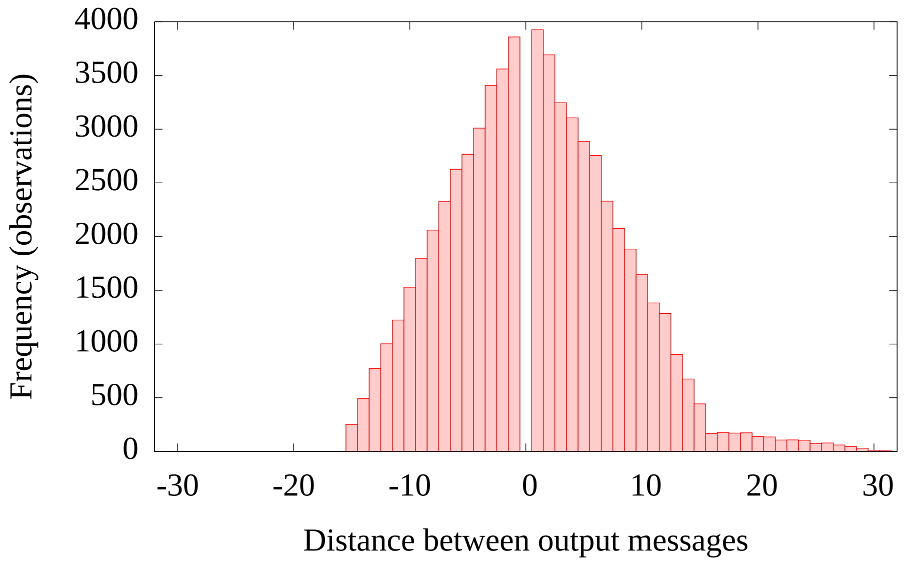


Figure 5.8: Comparison of result sets from individual experiment runs. Both show a single mix with flushing threshold 16.

```
Significance levels:
  0.1: Null hypothesis not rejected
 (distributions appear the same).
   => ( 0.0036892361111111 < 0.00679288989860745 ).
  0.05: Null hypothesis not rejected
        (distributions appear the same).
   => ( 0.0036892361111111 < 0.00757240185418536 ).
  0.025: Null hypothesis not rejected
         (distributions appear the same).
   => ( 0.0036892361111111 < 0.00824055495896642 ).
* 0.01: Null hypothesis not rejected
        (distributions appear the same).
   => ( 0.0036892361111111 < 0.00907574633994274 ).
  0.005: Null hypothesis not rejected
         (distributions appear the same).
   => ( 0.0036892361111111 < 0.00963254059392696 ).
  0.001: Null hypothesis not rejected
         (distributions appear the same).
   => ( 0.0036892361111111 < 0.0108574879526922 ).
```

Figure 5.9: Example output of Kolmogorov-Smirnov test implementation

Unequal distributions

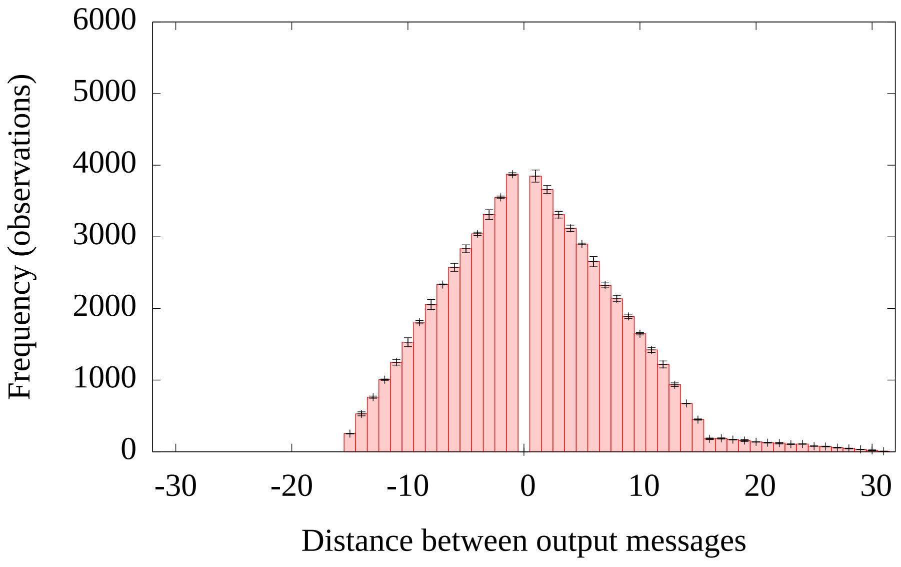
Here we demonstrate use of the Kolmogorov-Smirnov test to show that experiments conducted with different parameters result in samples that are drawn from statistically significantly different underlying distributions. This supports our view that the change of parameter has a statistically significant effect on the behaviour of the system.

We demonstrate here an example drawn once again from the single mix node experiments. In Figure 5.10 the top graph is a frequency plot of the output distances for a node with threshold 16. The bottom plot is of a node with threshold 12. Applying the Kolmogorov-Smirnov test reveals that the underlying distributions show a statistically significantly difference.

Figure 5.11 again shows results of the statistical significance test. Here, the test rejects the null hypothesis at the 99% level, providing a reasonable level of confidence that the two samples are drawn from differing underlying distributions and thus represent a statistically significant difference in the behaviour of the two systems.

5.5 Summary

In this chapter we have discussed the mathematical basis for the analyses that we apply to the results of our simulation experiments. We have detailed the notion of information-theoretic entropy that underlies our quantification of the effectiveness of the systems described in the model of Chapter 4, and shown the application of this notion to the results of the simulator. We have also shown the methods that we employ in order to gain confidence in the statistical significance of our analyses.



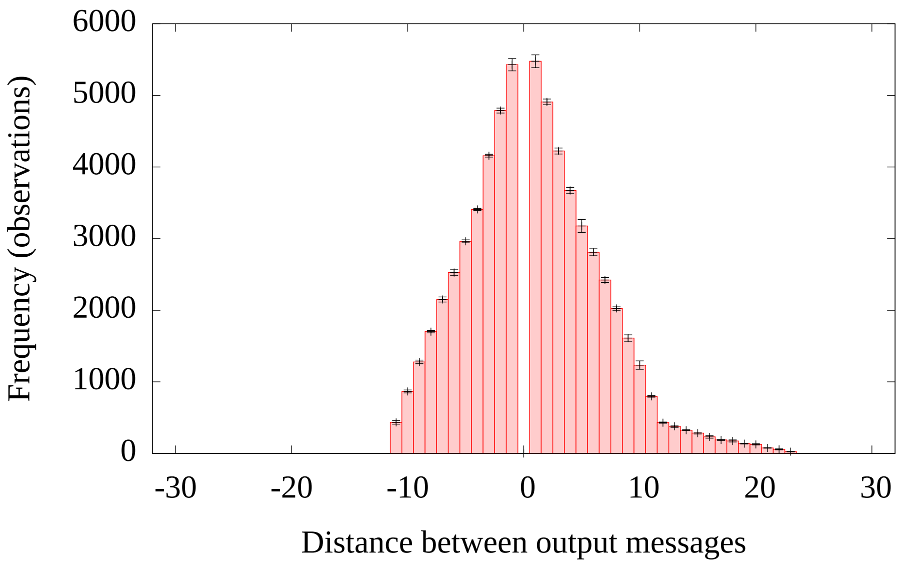


Figure 5.10: Comparison of result sets from different experiments. Top: a single mix with threshold 16. Bottom: a single mix with threshold 12.

```
Significance levels:
  0.1: Null hypothesis rejected
       (distributions appear different).
   => ( 0.0301959325396826 >= 0.00679288989860745 ).
  0.05: Null hypothesis rejected
        (distributions appear different).
   => ( 0.0301959325396826 >= 0.00757240185418536 ).
  0.025: Null hypothesis rejected
         (distributions appear different).
   => ( 0.0301959325396826 >= 0.00824055495896642 ).
* 0.01: Null hypothesis rejected
        (distributions appear different).
   => ( 0.0301959325396826 >= 0.00907574633994274 ).
  0.005: Null hypothesis rejected
         (distributions appear different).
   => ( 0.0301959325396826 >= 0.00963254059392696 ).
  0.001: Null hypothesis rejected
         (distributions appear different).
   => ( 0.0301959325396826 >= 0.0108574879526922 ).
```

Figure 5.11: Results of the Kolmogorov-Smirnov test for experiments with differing parameters.

Chapter 6

Simulation

6.1 Introduction

The model presented in Chapter 4 provides a classification of anonymity systems according to the method that they employ in order to introduce randomness into the output sequence of messages.

This approach allows us to check for the *existence* of anonymity in some system model, but does not immediately provide a quantification of this anonymity. To achieve a quantification based on this model, we must therefore calculate the level of randomness introduced by the strategies presented in the previous chapter. This allows us to judge the effectiveness of these strategies in confusing an observer attempting to link initiators and recipients.

By casting our systems into the form of a simulation we lose the purity of a highly abstract model but gain the ability to apply statistical analyses to the behaviour of the models. This approach has led to the development of the simulation and analysis tools described below.

6.2 Quantification Approach

There are a number of existing approaches towards quantifying the effectiveness of anonymity systems. The most widely referenced of these is the anonymity set of Chaum (1981), however this approach is appropriate only for a rough quantification of anonymity due to its assumption of uniform linkage probabilities between actors. To analyse the systems proposed in Chapter 4 we require a more detailed approach.

A significant development over the anonymity set was the information theoretic metric, proposed with slight variations independently by Diaz et al. (2002) and Serjantov and Danezis (2002). These quantifications take account of the varying probabilities in linkages between senders and recipients. Unfortunately, both proposed metrics are applicable mainly to mix-based systems and their use in more general strategies remains an open problem. As such, we cannot directly apply either of these metrics in comparing the systems we propose.

Other metrics, such as that of Wright et al. (2002) have relied on the ability of different systems to resist attacks designed to link senders to recipients. This can allow fundamentally different systems to be compared to an extent, however the attacks that have been employed in previous work have made particular assumptions regarding the nature of the systems to be compared. This restricts the ability of these approaches to compare the entire range of possible anonymity systems that we propose.

As such our approach to quantification attempts to be as general as possible both in term of its applicability to a wide range of systems, and to its practicality in implementation. Our quantification relies on an information-theoretic entropy measurement, however this is built by observing the distances introduced between messages as they pass from sender to recipient in simulated networks of differing anonymity strategies.

The specifics of the quantification method are discussed in detail in Chapter 5. In this chapter we describe the design and implementation of a simulator that allows for the strategies proposed in the abstract model of Chapter 4 to be subjected to statistical analysis.

6.3 Purpose

We simulate networks of actors described by the abstract model of Chapter 4 in such a way that we can observe their behaviour in an approximation of a genuine network. The purpose of these simulations is to allow for a quantification of the anonymity provided by the various strategies we propose. The networks in which we are interested consist of senders creating messages that are passed across a number of intermediary nodes to corresponding receivers.

The network nodes in our model typically implement some form of anonymity strategy as described in Chapter 4. Messages travel over a number of hops, being passed from node to node before being delivered to the receiving actor. An example network of this form is show in Figure 6.1.

Our simulations allow us to examine the flow of messages in these networks. These observations are then used to quantify the level of anonymity provided by the specific anonymity strategies.

The abstract model presented in Chapter 4 is concerned with randomness in output sequences of systems. At a basic level, we assume that the greater

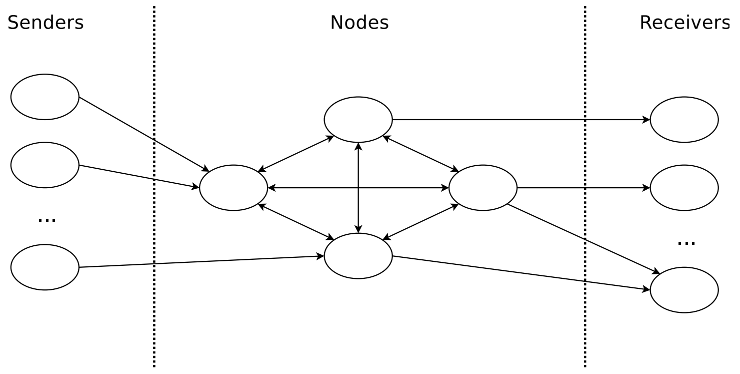


Figure 6.1: Example network structure.

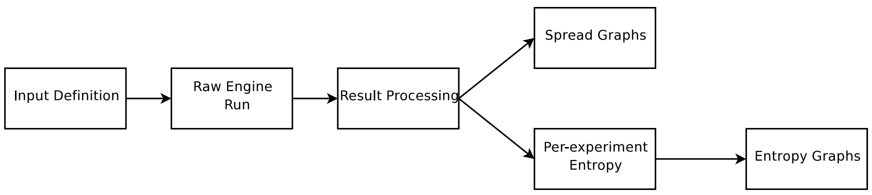


Figure 6.2: Flow of experiment processing.

the randomness in a system's sequence of output messages relative to its sequence of input, the more effective the system is at providing anonymity.

In order to quantify randomness, we calculate the entropy of the distribution of messages in the output sequence of each simulation for a number of fixed input sequences. This value represents the level of uncertainty that an attacker has for a system, given knowledge of the input sequence. The specifics of these calculations are discussed in more detail in Section 6.7. A full description and motivation of the quantification method is presented in Chapter 5.

The purpose of the simulator is to model a range of anonymity systems in a form suitable for statistical analysis. As we are concerned with the ability of an adversary observing network traffic, the simulator bases its analysis on sequences of observable communications.

6.4 Design

The systems that we analyse typically consist of multiple initiators, nodes and recipients. The simulator is built around *Actor* objects that fulfil these roles. To represent the network itself, communication between actors takes place across a universal *Environment* object that also governs the scheduling of parallel execution.

The simulator is divided into an event-processing engine that generates sequences of events showing the interaction of Actor objects, and a set of processing scripts that collate these events and perform statistical analyses.

Experiments are defined by the sets of actors that comprise the network and the messages that are sent. The simulation engine then generates the sequences of events resulting from the passage of messages across the network. Results are processed by an analysis script to describe both the spread of possible output sequences and the entropy of the distribution of messages in these sequences. Figure 6.2 shows this flow of execution.

6.5 Engine

The simulation engine produces the sequences of events that are used in our analyses. These events are generated by the interaction of node processes in an abstract network. Senders and recipients communicate with each other via a series of intermediary nodes. Simulations take place within an environment that regulates the actor processes and provides a medium of communication allowing messages to be passed from actor to actor.

The engine is therefore based around the interactions of *Actor* objects within the *Environment*. Each anonymity strategy defined in the model of Chapter 4 is implemented as a subclass of a basic *Node* class. This class is itself derived from the base actor. In addition to the various node classes there are *Sender* and *Recipient* actors that act as the endpoints of communication in the network. There are also a number of special senders with differing behaviour derived from the base sender class. The hierarchy of actor classes is shown in Figure 6.5.

6.5.1 Network Scheduling and Routing

The simulation engine applies random scheduling to the parallel processes in the simulated network, with a simple fairness criterion to prevent nodes from remaining idle.

The fundamental action taken by each actor occurs in its `step()` function, causing that actor to send a message from its internal queue, generate a dummy message, or to perform some other action.

The simulator divides time into *epochs*, in which each actor's `step()` function is called exactly once. The activation of actors is randomised, but

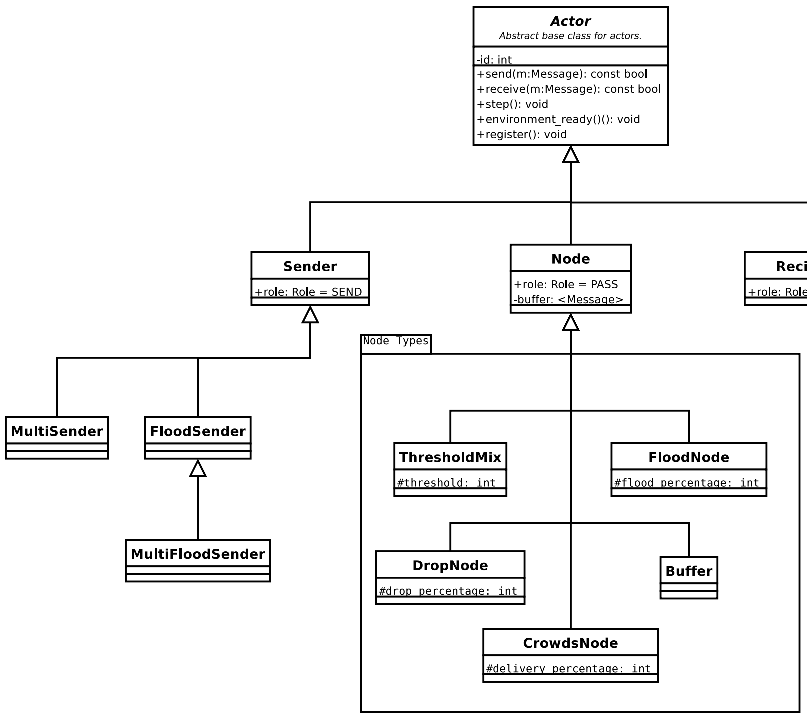


Figure 6.3: Simulator Actor Class Hierarchy

each is called only once per epoch. This approach to scheduling nodes' step functions is intended to approximate, with a simple level of enforced fairness, an asynchronous network of nodes that may experience delays in message routing.

The main processing loop of the simulation continues until all messages have been delivered to their ultimate recipient actor.

6.5.2 Input format

Experiments are defined by a series of *Actor* definitions, each of which defines zero[1] or more actors and their parameters. An Extended Backus-Naur Form (EBNF) grammar for the input file is shown in Figure 6.5.2. For the sake of simplicity, we omit definitions of the `alphanumeric`, `number` and `empty` type.

A simple valid input file is given in Figure 6.5. The actor definitions in this example show, in turn:

[1]A definition may specify that no actors of a defined type should be created in the simulation run. This is useful in those cases where batch processing of experiments requires a particular actor type to be removed from certain experiments.

```
<input-file>       ::= <input-block>*
<input-block>      ::= <comment> | <actor-definition>
<actor-definition> ::= Actor <actor-name> { <parameter-list> }
                       <quantifier>
<quantifier>       ::= : <number> | <empty>
<actor-name>       ::= Sender | Receiver | MultipleSender |
                       FloodSender | MultipleFloodSender |
                       ThresholdMix | DropNode | FloodNode
<parameter-list>   ::= <alphanumeric> = <number>;
                       <parameter-list> | <empty>
<comment>          ::= # <alphanumeric>
```

Figure 6.4: EBNF grammar for the simulation engine input file.

- A single sending actor that will transmit 32 messages, each of which travels over a single hop before reaching a recipient.
- Four threshold mix nodes, each with an internal pool of 16 messages. In order to prevent messages from becoming permanently stored in the node, a timeout is specified that will cause all messages in that node to be flushed after 32 epochs without the node having received any events.
- A single receiving actor that takes no special arguments.

Each actor is defined using the `Actor` keyword followed by an actor type that corresponds either to a sender, recipient or to a specific anonymity strategy as defined in the model.

An actor definition takes a number of parameters that define that actor's behaviour. These parameters control factors such as the pool size of a threshold mix, or the number of messages sent by a sender before it terminates. Following the definition of parameters, an optional argument allows the simulator to create multiple copies of the preceding actor definition.

The valid actor types implemented in the simulator, and their associated parameters, are shown in Figure 6.6.

6.5.3 Output format

The simulation engine outputs a time ordered sequence of events, where each event represents the sending of a message from one actor to another. Each instance of a message being transmitted between two actors is reported in the form:

```
# Experiment 01:
# Investigate a simple mix.

Actor Sender {
   message_hops = 1;
   send_amount = 32;
}

Actor ThresholdMix {
   pool_size = 16;
   tock_limit = 32;
} : 4

Actor Receiver {}
```

Figure 6.5: Sample simulator input file

```
( <Initiator>.<Epoch><EpochID>, <Sender>, <Recipient> )
```

where the `Sender` and `Recipient` actors refer to the actors involved in the specific hop of this message. The `Initiator` is the actor that ultimately created and sent the message. `Epoch` is a count of how many `step()` actions have been taken by the actor, while `EpochID` refer to the number of messages sent by the actor in this particular `Epoch`. These values together form a unique identifier for each message created by a given `Initiator`.

Dummy messages are not created with valid message identifiers. In such messages the identifier, usually the combination of `Epoch` and `EpochID`, is set to `-1-1`.

Example output from the simulation engine is given in Figure 6.7.

6.6 Implementation Details

Our classification describes four explicit methods for introducing randomness into the output sequence of a system. We investigate each of these, implemented as an actor in the network, via simulation.

The simulation necessarily introduces details that can be abstracted away in a more abstract model. We discuss here the implementation issues that arise from translating the model into an executable form, and from implementing the design shown above.

Actor	Description	Parameters
Sender	Simple sending actor.	send_amount, message_hops
MultiSender	A single actor that emulates multiple individual senders, allowing for a multiple-sender message queues that are consistent across experiment runs	send_amount, message_hops, senders
FloodSender	A sender that injects dummy traffic into the system along with its valid messages.	send_amount, message_hops, flood_percentage
MultiFloodSender	A multi-actor version of the `FloodSender`.	send_amount, message_hops, flood_percentage
Buffer	A simple deterministic buffer.	⟨none⟩
ThresholdMix	An implementation of the most basic mix anonymity strategy.	pool_size, tock_limit
FloodNode	A buffer that probabilistically chooses whether to pass a real message or inject a dummy message.	flood_percentage
DropNode	A buffer that probabilistically chooses whether to pass or to drop each message.	drop_percentage
CrowdsNode	A buffer that probabilistically chooses whether to pass or deliver each message.	delivery_percentage
Recipient	A simple receiving actor.	⟨none⟩

Figure 6.6: Valid actor types in simulation engine

```
Creating 1 sender(s). Messages: 16. hops: 1
Creating 1 threshold mix(es). Pool Size: 16, tock_limit: 32
Creating 1 receiver(s).
Actor 1: SEND
Actor 2: PASS
Actor 3: RECEIVE

Run 1 of 1024.

(0.00, 0, 1)
(0.01, 0, 1)
...
(0.05, 1, 2)
(0.013, 1, 2)

Run 2 of 1024.

(0.00, 0, 1)
(0.01, 0, 1)
...
(1.-1-1, 1, 2) <-- Dummy message
(0.04, 1, 2)
(0.015, 1, 2)

...
```

Figure 6.7: Sample output format from simulation engine with dummy message highlighted.

6.6.1 Language choices

The bulk of processing for experiments takes place in the simulation engine. For reasons of speed and efficiency, as well as convenient object-orientation, the engine is entirely written in Standard C++ with use of the Standard Template Library. Random-number generation algorithms are taken from the GNU Scientific Library (GNU, 2007).

For processing and analysis, the majority of the work consists of searching through and tallying raw events generated by the simulation engine. A language with strong support for text processing and regular expressions is preferable for ease of development and adaptation. The Ruby language (Thomas et al., 2004) is a convenient balance of object orientation, rapid development and acceptable speed.

A combination of perl scripts and simple shell scripting were used to orchestrate experiments and corresponding results processing.

6.6.2 Environment

The environment object is central to the simulator, acting both as a scheduler for the components of the system and as the basis for communication between the actors.

At program initialisation, each actor defined in the input file is created and registered with the environment. The registration process assigns each new actor a unique identifier and stores this, along with the actor's *role*, in the environment.

There are three roles within the environment: `SEND`, `PASS`, and `RECEIVE`, respectively corresponding to the *Sender*, *Node* and *Receiver* actors types shown in Figure 6.5. These roles define the flow of messages possible between actors: `SEND` actors may send messages to any `PASS` actor, but may not communicate directly with `RECEIVE` actors or themselves receive messages. `PASS` actors receive messages and send them either to each other or to an appropriate `RECEIVE` actor. `RECEIVE` actors perform no sending actions at all.

Register of Actors

The environment acts as a centralised register of actors. This is used by actors in generating messages and as a central means of communication to allow actors to pass messages between themselves.

To create a new message, an actor requests from the environment the identifier of appropriate actors to populate the path and recipient sections of the message. To directly contact another actor in order to pass a message, the environment supplies the sending actor with a pointer to the desired actor.

In each experiment, when all specified actors have been registered, the environment announces to each actor in turn that the environment is ready and that the simulation is due to start. Actors then perform any preliminary operations required before the start of the simulation run, such as the generation by a sender of its queue of messages.

Scheduling

The environment also functions as a scheduler for parallel actor processes in the simulation. Each actor implements a `step()` function that defines the actions that the actor performs in a given epoch; the environment's main processing loop during a simulation is to select a random actor and call its `step()` function. This may result in the sending of a message, the incrementing of internal timing variables or no action at all, depending on the internal state of the actor.

Scheduling in the simulator is random, but with restrictions to prevent undesirable statistical effects of the simple model: each actor is guaranteed to perform its step function exactly once per epoch, but the ordering of such calls is randomised. This approach to scheduling is taken both for simplicity and as an approximation to asynchronous networks with small possible delays.

A pseudocode description of the behaviour of the environment is given in Algorithm 1.

6.6.3 Sender

Sending actors generate messages that travel across the network to receiving actors. It is the events caused by the passage of these messages that are analysed for the purposes of determining the entropy introduced by the system. As a sender does not forward messages for other nodes, its behaviour is relatively simple.

Each sender is defined with a number of messages to send and a given number of hops for each message. At the initial run of an experiment, the actor generates messages according to these two parameters. Messages are created with a fixed path that consists of a sequence of one or more intermediary node identifiers, and a recipient identifier. These parameters are randomly assigned from the list of suitable actors retrieved from the environment. Messages store the identifier of the sender that created them, as well as an identifier unique to that message amongst those generated by that

Algorithm 1 Main environment actions

```
{Register actors.}
$id \leftarrow 0$
$active\_senders \leftarrow 0$
for all $actor$ in unregistered actors do
    assign $id$ to $actor$
    push $(id, actor.\textsc{Role})$ to actor register
    if $actor.\textsc{Role} = \text{SEND}$ then
        $active\_senders \leftarrow active\_senders + 1$
    end if
    call $actor.\textsc{Register}$
    $id \leftarrow id + 1$
end for

{Initialise system.}
for all $actor$ in registered actors do
    call $actor.\textsc{Initialise}$
end for
$sent\_messages \leftarrow 0$

{Main simulation loop}
while $active\_senders > 0$ and $sent\_messages > 0$ do
    $actor \leftarrow$ {random actor from actor register}
    $actor.\textsc{Step}$
end while
```

sender. The combination of sender identifier and message identifier is used in the analysis of results to uniquely identify messages.

After message creation, the sending actor's sole behaviour is to send a single message each time that its `step()` function is called. This injects the message into the network and registers with the environment that a message with the given identifier has been sent. When the sender has emptied its queue of messages it de-registers with the environment and takes no further part in the simulation.

The queue of sent messages is retained by the sender between multiple runs of each experiments. Senders therefore reproduce the same sequence of messages for each simulation run of a given experiment.

A description of the individual sender's flow of execution is given in Algorithm 2.

Algorithm 2 Sender behaviour

```
procedure INITIALISE($message\_count$)
    for $i \leftarrow 1, message\_count$ do
        $message \leftarrow$ GENERATEMESSAGE
        Push $message$ to internal queue
    end for
end procedure

procedure STEP
    if messages remain in queue then
        pop first message from queue and SEND
        increment environment $sent\_messages$ count
    else
        de-register with the environment
    end if
end procedure
```

In practise, a variation on the simple sender is used to ensure that the entire input sequence is reproducible. This implementation of the sender creates messages with differing sender identifiers, as well as registering with the environment as multiple actors. This behaviour mimics multiple individual senders, allowing the sequence of input for the entire network to be preserved between simulation runs.

The `send()` function is simple, and is given in pseudocode form in Algorithm 4.

Each message contains a sequence of actor identifiers that represent the path to be taken through the network. At each hop, the head of this sequence

Algorithm 3 Message generation

```
procedure GenerateMessage($path\_length$, $message\_id$)
    for $i \leftarrow 1, path\_length$ do
        $actor \leftarrow$ {get random PASS actor id from Environment}
        Push $actor$ onto $path$
    end for
    $actor \leftarrow$ {get random RECEIVE actor id from Environment}
    Push $actor$ onto $path$
    Return ($message\_id, path$)
end procedure
```

Algorithm 4 Send method

```
procedure Send($message$, $recipient\_id$)
    $receive\_actor \leftarrow$ {get reference to actor $recipient\_id$ from Environment}
    call $receive\_actor$.Receive($message$)
end procedure
```

is removed and resulting message is forwarded to the next actor identified in the sequence. Sending of a message is achieved by requesting a pointer to the named actor from the environment. This pointer is used to call the receiving actor's public `receive()` function. If allowed, this function copies the passed message to the called actor's internal queue for consideration in the next epoch of the simulation run. If the receiving actor is not in a suitable state to accept the message, it may decline to copy the message and notify the sending actor of this fact.

6.6.4 Receiver

A receiving actor's behaviour is the simplest of any actor in the simulator. The actor performs no actions as part of its `step()` function, and can neither generate nor pass messages. The sole purpose of a receiver is to act as a sink for messages.

Execution of the simulator continues until all genuine messages created by the sending actors have passed through the network. A receiver therefore de-registers each message that it receives from the environment. Upon the environment detecting that all messages have been received, it ceases to call actors' `step()` functions and ends the simulation.

A pseudocode description of the receiver's flow of execution is given in Algorithm 5.

Algorithm 5 Receiver behaviour

```
procedure Receive(message)
    if message is non-dummy traffic then
        decrease environment sent_message count
    end if
end procedure
```

6.6.5 Nodes

We discuss here the implementation details of nodes that implement the various strategies for introducing randomness into the output sequence of messages as described in the model.

Mix

The implementation of the Mix strategy applied in the simulator is the simplest form of threshold mix, with a simple timing criterion added to prevent messages from remaining in the node indefinitely once the flow of input messages has stopped. A basic threshold mix stores messages until a certain number accumulate in its buffer, whereupon all messages are flushed from the buffer in random order.

One problem with a pure threshold mix arises from the fact that no flushing action is possible until the threshold value is reached. Our simulations, in which a fixed number of messages enter the system, would result in messages indefinitely remaining in mixes for which the threshold value can never be reached. To counteract this effect we impose a timing condition that causes these nodes to flush after a given period of inactivity. A description of this node's behaviour is given in Algorithm 6.

On receiving dummy traffic the Mix node does not immediately drop the incoming message, as is the case for the other node types. To do so would trivially reveal certain messages as being dummy traffic. We do not directly apply such deductions in our analysis, however removing such trivial attacks serves to make the simulation more representative of the real-world systems in which we are interested.

To handle dummy traffic, the implementation of the Mix node therefore generates a new dummy message for each incoming dummy message. The original message is discarded and the newly generated messages is added to the tail of the internal queue. This behaviour is only required in those experiments where we combine Mix with Flood nodes. The GenerateDummy procedure referenced in Algorithm 6 is described under the Flood strategy in Section 6.6.5.

Algorithm 6 Mix (Threshold) implementation

```
procedure Initialise
    $tock\_count \leftarrow 0$
    $flushing \leftarrow false$
    $pool \leftarrow \langle\rangle$
    $pre\_buffer \leftarrow \langle\rangle$
end procedure

procedure Step
    if $flushing = true$ then
        if $pool$ is non-empty then
            $message \leftarrow$ {pop random message from pool}
            Send($message$)
        end if
        if $pool$ is empty then
            $flushing \leftarrow false$
            Move up to $threshold$ $pre\_buffer$ elements to $pool$
        end if
    else
        $tock\_count \leftarrow tock\_count + 1$
        if $tock\_count \geqslant tock\_limit$ then
            $tock\_count \leftarrow 0$
            $flushing \leftarrow true$
        end if
    end if
end procedure

procedure Receive($message$)
    $tock\_count \leftarrow 0$
    if $message$ is dummy traffic then
        $message \leftarrow$ GenerateDummy
    end if
    if $buffer\_size < threshold$ and $flushing = false$ then
        add $message$ to $pool$
    else
        add $message$ to $pre\_buffer$
    end if
    if $pool\_size \geqslant threshold$ then
        $flushing \leftarrow true$
    end if
end procedure
```

Flood

The FLOOD node performs no explicit mixing operation, and instead injects extra messages into the stream of messages passing through the node. The level of message injection is defined by a parameter that specifies the probability of injecting a dummy message rather than passing a valid message. As this parameter increases, more and more dummy messages are injected into the stream.

A FLOOD node stores each message that it receives in its internal buffer, implemented as a simple first-in first-out queue. When the actor's `step()` function is called the node determines, according to the injection probability, whether to send the genuine message at the head of the queue, or whether to pass a dummy message in its place. If a dummy message is sent, the genuine message remains at the head of the queue.

In the case that the FLOOD node's internal buffer is empty, a dummy message is always generated.

A description of the flood node's flow of execution is given in Algorithm 7.

Algorithm 7 FLOOD Node implementation

```
procedure STEP
    if buffer is empty then
        generate and send dummy message
    else
        random ← {random percentage value}
        if random < flooding_percentage then
            message ← GENERATEDUMMY SEND message
        else
            pop message from buffer and SEND
        end if
    end if
end procedure
```

Dummy messages are intentionally as simple as possible. Each message travels only a single hop, and does not contain any unique identifying information. Dummy messages are also never considered as part of the set of consecutive input pairs examined in analysing these experiments. As such, dummy messages do not require individual identification.

The path of a dummy message is generated by requesting a random actor of the appropriate type from the environment, typically an intermediary node or a recipient, and creating a message with that actor as the sole element of the message path. Handling of dummy messages is dependent upon the receiving node, but typically results in the message being silently dropped.

Generation of dummy messages is described in Algorithm 8

Algorithm 8 Message generation

```
procedure GenerateDummy
    $actor \leftarrow$ {get random $PASS$ or $RECEIVE$ actor id from Environment}
    Push $actor$ onto $path$
    Return ($dummy\_id, path$)
end procedure
```

Hide

The implementation of the Hide strategy is requires a different approach from that of the other node types. In a Hide system, individual nodes do not explicitly perform actions that can be said to introduce randomness. Instead, the confusion of an observer viewing the system is gained through the existence of hidden system elements. The nodes of the network for this strategy may be simple buffers, as in our experiments.

In order for the input sequence to be reproducible across multiple simulation runs, systems built around other strategies make use of a *MultipleSender*. This actor stores a fixed message queue for a number of individual senders, causing the sequence of input messages to be consistent across multiple runs of the experiment. For the Hide experiments, the existence of "unseen" senders is modelled by an increasing number of individual *Sender* actors in addition to the standard *MultipleSender*. The messages generated by these senders are ignored in the result processing script when considering inputs. This unseen behaviour achieves our goal of affecting the system's output ordering.

Drop

The behaviour of a Drop node can be considered the opposite to that of a Flood node. Rather than injecting fake messages, the Drop node removes genuine ones. When multiple messages exist in the node at the same time, it therefore becomes impossible to tell which messages have been dropped and which forwarded.

The Drop strategy clearly introduces severe reliability issues into the network. However, the properties of this node type are of interest for the sake of complete coverage of possible strategies. Examination of this strategy also has implications for systems that attempt to provide anonymity over unreliable networks.

Similarly to the FLOOD node, the DROP node makes a probabilistic choice regarding the action to take for a given message. In the case of the DROP node, the random choice determines whether or not to drop a message from the queue. A pseudocode version of this behaviour is shown in Algorithm 9.

Algorithm 9 DROP Node implementation

```
procedure STEP
    if buffer is non-empty then
        message ← pop message from buffer
        random ← {random percentage value}
        if random < dropping percentage then
            delete message
            decrease environment sent_message count
        else
            send message
        end if
    end if
end procedure
```

Crowds

The Crowds node, as described by Reiter and Rubin (1998), and discussed in detail in Chapter 3.4.3, is a practical variant of the DROP strategy. This strategy differs significantly from the other node types in that it has no fixed message path length and cannot function effectively as a single node.

On entering a Crowds node, as with other node types, each message is added to an internal queue. When considered for delivery, a random choice is made concerning whether to deliver the message immediately to its ultimate recipient or to forward the message to another randomly chosen node in the system.

At the single node level, if we assume that a node may not forward messages to itself, the behaviour of this system is equivalent to that of a simple buffer: the sequence of the output messages preserves that of the input. When considered as a network of nodes, however, the system produces a significant mixing effect. The behaviour of such a system is of interest both as a practical implementation of the DROP strategy, and as an example of a system that has been both implemented and deployed.

A pseudocode description of the Crowds node's behaviour is given in Algorithm 10.

Algorithm 10 Crowds Node implementation

```
procedure STEP
    if buffer is non-empty then
        $message \leftarrow$ pop message from buffer
        $random \leftarrow$ {random percentage value}
        if $random <$ delivery percentage then
            deliver $message$ to ultimate recipient
        else
            send $message$ to random other node
        end if
    end if
end procedure
```

6.7 Processing of Results

The processor scripts convert the results of the simulator into a form suitable for analysis. This largely consists of correlating the location of pairs of messages in the input stream with their corresponding location in the output stream.

Due to the size of the message sequences passing through systems, there are a large number of possible output sequences for systems that aim to confuse an attacker. Exhaustively exploring such a set of sequences is neither feasible nor particularly desirable. An attacker is unlikely to attempt to correlate each input message with an output message, and will typically be more concerned with identifying a given target message.

In analysing our simulations, we therefore choose to examine the effect that a system has on the distance in the output sequence between pairs of messages from the input sequence. By selecting multiple message pairings and calculating the distribution of their distance from each other in the output sequence, we gain an approximate but computationally efficient quantification of the level of reordering caused by the system.

The specific choice of which messages to examine, and details of how the results are to be analysed, are discussed in greater detail below. A full discussion of this is given in Chapter 5.

6.7.1 Choice of pairings

It is not feasible to consider all possible pairs in the sequence of inputs; the computational requirements to perform such an analysis would further restrict the range of possible simulations to extremely small systems with very few messages. We therefore require a choice of message pairings that demonstrate the level of randomness in the system without requiring an explicit calculation for each possibility.

We choose to examine the output distance between pairs of messages that are consecutive in the input stream. This provides a useful representation of the specific action of the anonymity system in introducing multiple output orderings of the input sequence, whilst keeping the number of possibilities to be examined relatively low. A number of other approaches, such as random pairings and representative spreads of messages, have been explored in preliminary experiments and rejected. The details of these preliminary experiments are discussed in Chapter 5.2.3.

6.7.2 Basic Operation

The output of the simulator consists of a set of simulation runs, in which each run is a time-ordered sequence of events analogous to a CSP trace. Each run consists of the same sequence of input events, defined as all those that pass between actors fulfilling the `SEND` role, and actors fulfilling the `PASS` role.

The sequence of input events is fixed across simulations runs, except in the case of the HIDE experiments that use hidden senders to disturb the input sequence. In these experiments, messages originating from the hidden senders are ignored when selecting consecutive pairs in the input sequence.

To process the results, the processor first creates a list of all input events by examining the first run of the simulation. For each consecutive pair in this list the processing code locates the corresponding pair of output events for each run of the simulation. The location of messages in the output sequence, along with the distance between the two, is used to form a probability distribution over all output orderings for each pair in the input. This distribution is used for the later entropy calculations.

The example in Figure 6.8 demonstrates a sample result from an iteration of Algorithm 11 for a simple set of output messages. The current consecutive pair is highlighted:

Algorithm 11 Execution flow of the results processor

```
for all consecutive $(i, j)$ in input do
    for all run in result file do
        $i', j' \leftarrow$ location of $i, j$ in output
        distance $\leftarrow i' - j'$
        Store $i', j'$ and distance
    end for
end for
```

Input	i	j	Output	i'	j'	Distance
$\langle a, \mathbf{b}, \mathbf{c}, d \rangle$	2	3	$\langle \mathbf{c}, \mathbf{b}, a, d \rangle$	2	1	-1
			$\langle d, \mathbf{c}, \mathbf{b}, a \rangle$	3	2	-1
			$\langle \mathbf{c}, d, a, \mathbf{b} \rangle$	4	1	-3
			$\langle \mathbf{b}, d, \mathbf{c}, a \rangle$	1	3	2

Figure 6.8: Example of input and output sequences for a simple simulation run. Target messages are highlighted.

6.7.3 Specific considerations for the Drop strategy

For each of the strategies Mix, Flood and Hide it is guaranteed that each input message will be present in the output. Analysis of the simulation results relies on this fact to calculate the respective distances between input pairing distance and output pairing distance for each consecutive pair.

In the case of the Drop strategy, however, the property that each input message appears in the output no longer holds. This presents the issue of how to account for messages that are removed from the system.

The purpose of the processing stage is to render the results of the simulator useful for the calculation of output entropy. This, in turn, is to provide a measurement of the uncertainty of an observer concerning the possible location of a message in the output stream.

With this consideration, the most useful approach is to tally the cases in which one or both target messages are dropped from the output sequence. We take the approach of recording separate results for the case of the first, second or both messages being dropped. An example of this is given in Figure 6.9.

A consequence of this approach is that many results produce the same output. Regardless of the position of the remaining message, the dropping of one message from a pair causes the result to be recorded either as First or Second. As we see in Chapter 7, this causes the entropy of the out-

Input	i	j	Output	i'	j'	Distance
$\langle a, \mathbf{b}, \mathbf{c}, d\rangle$	2	3	$\langle \mathbf{c}, \mathbf{b}, a, d\rangle$	2	1	-1
			$\langle d, a\rangle$	-	-	Both
			$\langle d, a, \mathbf{b}\rangle$	4	-	Second
			$\langle \mathbf{b}, a, d\rangle$	1	-	Second
			$\langle a, \mathbf{c}, d\rangle$	-	3	First
			$\langle \mathbf{b}, d, \mathbf{c}, a\rangle$	1	3	2

Figure 6.9: Example of input and output sequences for a simulation run with dropped messages. Target messages are highlighted.

put distribution for a Drop node to be significantly smaller than for other strategies.

6.8 Experimental Setup

The simulation experiments explore each of the four node types detailed in Chapter 4. For each node type, there exists a *characteristic parameter* that defines the extent to which that strategy is applied to the sequence of messages. These parameters correspond to the threshold in Mix nodes, the ratio of dummy messages to real messages in Flood nodes, the number of hidden senders in a Hide system, and the percentage of dropped messages in a Drop system. In our consideration of a Crowds-style node (Reiter and Rubin, 1998), the characteristic parameter is the probability of a message being immediately delivered rather than sent to another intermediary node.

Experiments are carried out for a variety of factors that affect the anonymity of the system. We examine each node type as its characteristic parameter varies; nodes are simulated both individually and as part of a four-node system. In four-node systems, the path length of messages is varied so that results are obtained for messages travelling over path lengths of one to four hops.

In each case we consider 64 messages passing from eight sending actors to a single recipient. As we consider only unorderings on messages, and not on sender identity, this is may be considered as equivalent to a single sender whilst allowing for analysis based on sender identity if required.

Each simulation is performed 1024 times with the same initial queue of messages travelling over a pre-generated path, resulting in a sample of the possible output sequences. This process is repeated three times, for different randomly generated input queues, in order to observe difference in behaviour

Strategy	Parameter	Value	Nodes – Hops
MIX	Threshold	16 → 1	1 node – 1 hop 4 node – 1 → 4 hops
FLOOD	Flooding Ratio	90% → 10%	1 node – 1 hop 4 node – 1 → 4 hops
HIDE	Hidden Senders	8 → 1	1 node – 1 hop 4 node – 1 → 4 hops
DROP	Drop Percentage	90% → 10%	1 node – 1 hop 4 node – 1 → 4 hops
Crowds	Delivery Percentage	90% → 10%	4 node – variable hop

Figure 6.10: Single-strategy experimental parameters

Strategy	Parameter	Value	Nodes – Hops
MIX/ FLOOD	MIX:FLOOD Ratio MIX Threshold: 16 FLOOD Ratio: 90%	0:4 → 4:0	4 node – 4 hop
Crowds / FLOOD	Crowds:FLOOD Ratio Crowds Delivery: 50% FLOOD Ratio: 90%	0:4 → 4:0	4 node – 4 hop
Crowds / MIX	Crowds:MIX Ratio Crowds Delivery: 50% MIX Threshold: 16	0:4 → 4:0	4 node – 4 hop

Figure 6.11: Combined-strategy experimental parameters

caused by random variation of message paths.

The experiments that we perform, and their varying parameters, are detailed in Figures 6.10 and 6.11. Figure 6.10 describes experiments that apply to single anonymity strategies. Experiments that combined more than one approach to anonymity in a single system are shown in Figure 6.11.

6.9 Summary

In this chapter we justify our simulation-based approach to analysing the systems described in the model of Chapter 4. We have detailed the motivation behind the development of the simulator, and the purpose of the analyses that we perform. The design of the simulator is shown, and its approach to implementing the concepts expressed in the model is demonstrated.

Using the simulator developed in this chapter, the simulation experiments allow the anonymity provided by anonymity strategy to be quantified. We explore this over a range of experiments that cover the fundamental anonymity strategies both individually and as part of networks that consist of multiple nodes with varying strategies. The results of these experiments and their analysis are shown in the following chapter.

Chapter 7

Analysis of Simulation Results

7.1 Introduction

For the purpose of investigating the level of randomness produced by a variety of communicating systems, we have constructed a simulation engine that demonstrates the raw behaviour of messages passing over archetypal implementations of each class of anonymity system as discussed in Chapter 4.

The processing of this raw data results in two sets of data from each experiment. The first of these is a histogram showing the distribution of output distances for consecutive input pairs; each variation in the experiment's parameter provides a different histogram. The second set of results shows, for an experiment, the entropy of the output distances as the experimental parameter varies. This allows the effectiveness of changing that particular parameter to be seen.

64 messages are sent through each system from the main sending actor. The Flood and Drop nodes clearly cause the number of output messages to alter. The design of the Hide experiments also mean that a larger number of messages are received due to the action of the hidden senders. In each case, however, the set of input pairings is considered to be those drawn from the 64 original messages.

For each experiment the defining parameter of each system is varied over an appropriate range. The range that we explore for the various systems are, briefly:

- Mix: the threshold size varies from 16 messages to 1 message.
- Flood: the percentage of dummy messages to real messages varies from 90 percent to 10 percent.

- Hide: the number of hidden senders varies from 1 hidden sender to 8 hidden senders. Each hidden sender sends 8 messages in total.
- Drop: the percentage of dropped messages varies from 90 percent to 10 percent.

For each of these systems a number of system setups are examined. These are consistent across each experiment and are designed to investigate the interaction of the anonymity method with the base level of randomness introduced by the network itself.

Each experiment is first carried out for a single node interacting with the sender and recipient. In this experiment each message may only travel a single hop. The network is then expanded to consist of four fully-connected nodes. In this system, the effects of messages travelling over 1, 2, 3 and 4 hops is simulated.

7.2 Control Experiments

The simulations investigate the randomness in the output sequence of messages for various combinations of nodes. We are interested in the level of randomness introduced by each specific node type. The network of communicating asynchronous nodes itself, as implemented in the simulator, also perturbs the sequence of messages due to the randomness of the system. In order to have some comparison of the effect of the node types, it is necessary to have an idea of how much randomness is added without their effects.

As a control experiment, we simulate a sequence of messages passing through simple buffer nodes. These buffers perform no action beyond forwarding messages that they receive; the intention is that they represent the simplest form of network traffic.

To investigate the effects that the use of multiple buffer nodes have on the sequence of input we individually examine systems comprised of one to four nodes, with messages travelling from one to four hops.

As expected in the case of a single node with a single hop, there is no confusion of the input sequence. This is the archetypal non-anonymous system. There are no possible permutations of the sequence of output messages, and consequently zero conditional entropy in the output distribution given the input distribution.

As the number of nodes in the system increases, however, a level of randomness is introduced into the output sequence due to the asynchronous interaction of the nodes running in parallel. Far from being a minor effect, passing messages through even a small number of asynchronous buffer nodes introduces a level of randomness into the output sequence that is comparable to the specific mechanisms employed by the nodes themselves.

Figure 7.1 demonstrates the effect of passing 64 messages through networks made up from one to four buffer nodes. In each case, the route for each message is randomly generated when the system is initialised. In each experimental setup 30 initial starting conditions were tested, and for each set of initial conditions the simulation was repeated 1024 times. The aim of this approach is to eliminate any dependence in the results on particular input conditions. The error-bars on each graph represent the standard deviation of observed values, whilst the value of the bar itself is the mean.

Pairs of messages that enter the system in a particular order are more likely to retain that order than to reverse their positions, although the distance between the two may change. As might be expected, therefore, there is a bias to the positive side of the distribution. This is reflected in the output spread in each graph.

7.2.1 Multiple Hops

Increasing the path length of a message to include a greater number of hops increases the level of confusion in the output ordering. Figure 7.2 shows the spread of messages for networks of two to four buffer nodes, with messages travelling over two hops in each. Recall that in most experiments, barring the crowds nodes that we examine later, we prohibit messages from including a single node twice in their path.

Figure 7.2 demonstrates that increasing the number of hops results in a notable increase in the spread of the output messages. Focusing solely on this parameter, the resulting spread of messages in a four-node buffer network as the number of hops is increased from one to four is shown in Figure 7.3.

It can be clearly seen that the frequency graphs for these control experiments become increasingly perturbed as we increase the number of hops and nodes. The computational limits on running the simulation experiments have not allowed for the much greater sample sizes that would be required to smooth these results, and as such we are forced to accept a level of statistical noise in individual features of the graphs.

Applying the Kolmogorov-Smirnov test to two halves of a simulation run fails to reject the null hypothesis that they are drawn from the same distribution at the 99% significance level; we may therefore consider each individual simulation run to providing a representative view of the behaviour of the system according to some fixed set of messages. However, comparing two runs of the same experimental setup with differing message queues causes the null hypothesis to be rejected at the 99% confidence level. This implies

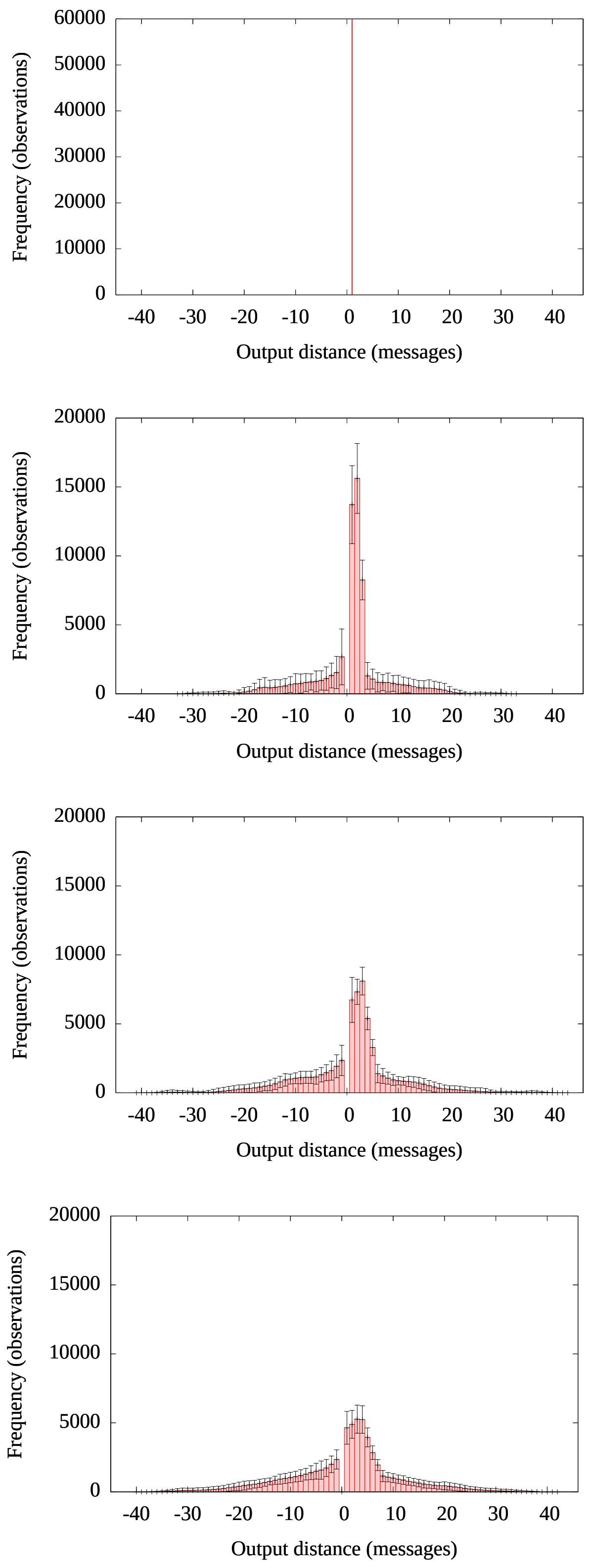


Figure 7.1: Distribution of spread in output for simple buffers. Number of buffers ranges from a single buffer (top) to four buffers (bottom).

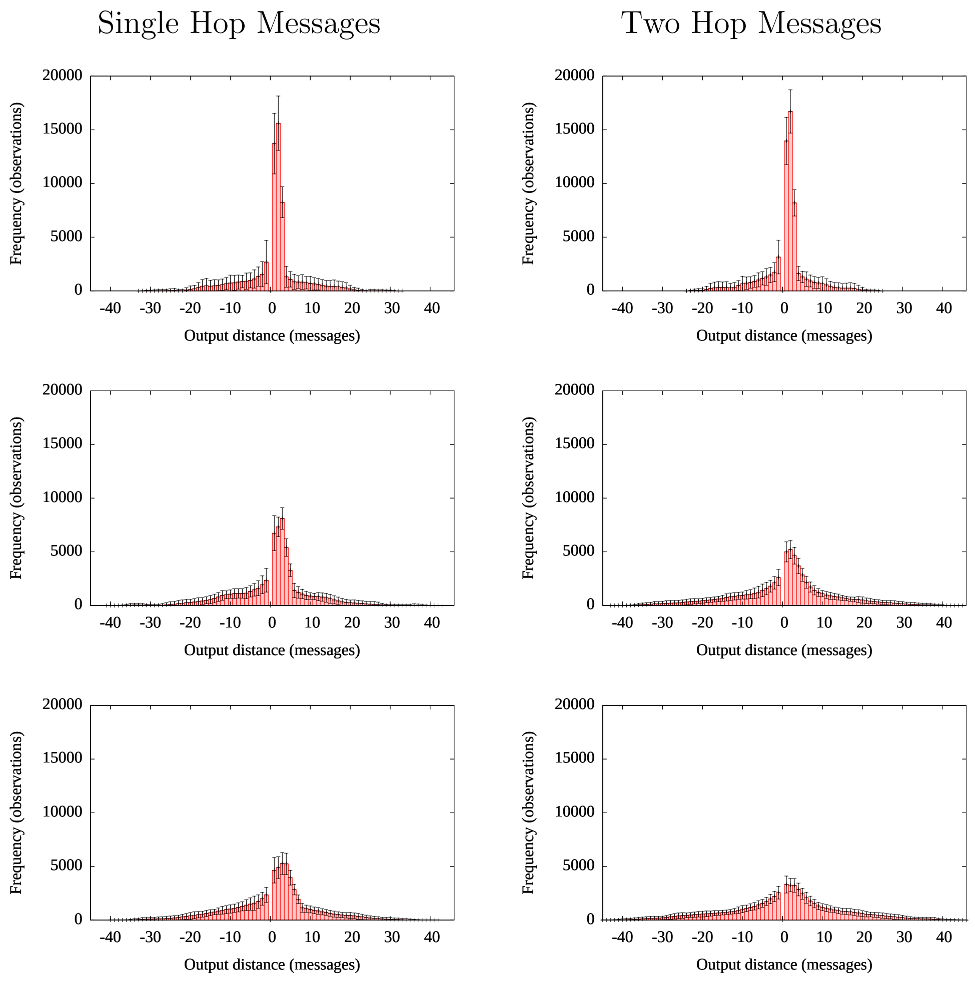


Figure 7.2: Spread in output sequence caused by buffers with multiple hops. Number of buffers in the network ranges from 2 (top) to 4 (bottom).

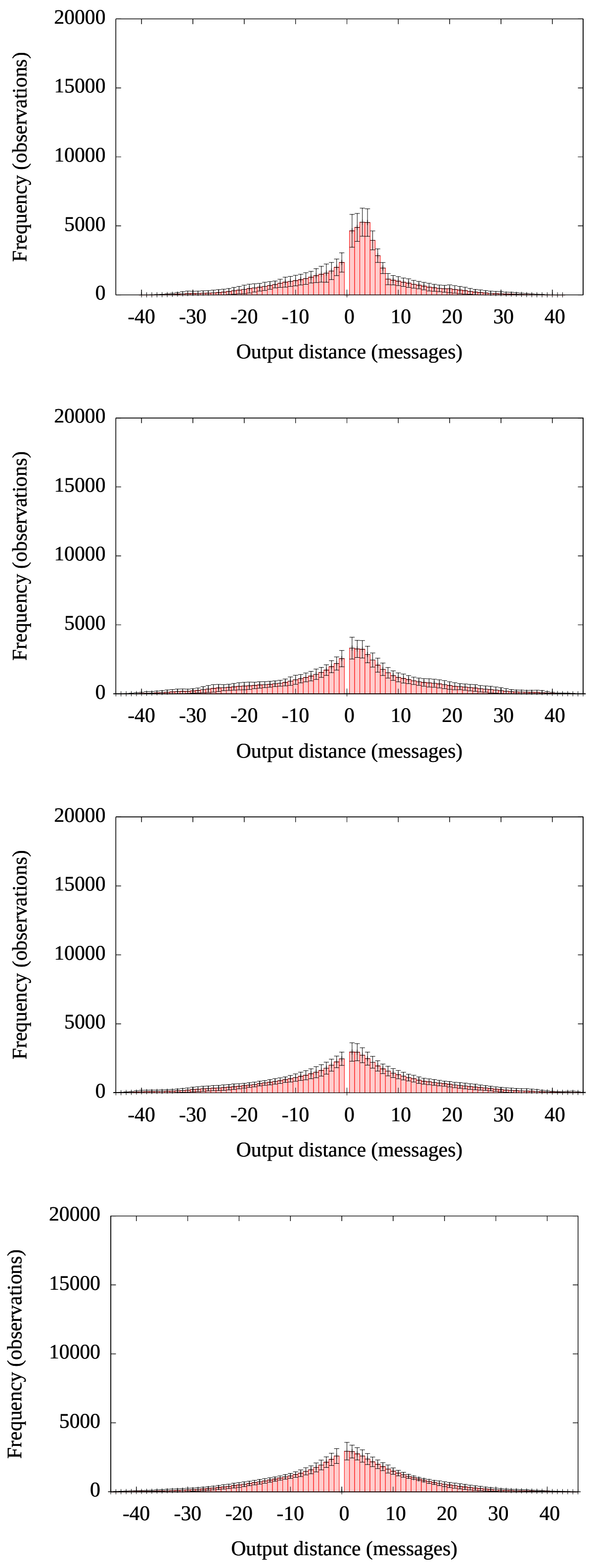


Figure 7.3: Frequency of spread in output sequence caused by a four-buffer network. Number of hops per message ranges from 1 hop (top) to 4 hops (bottom).

that varying the paths of messages travelling through the system does have a statistically significant effect on the overall distribution that has not been smoothed out by the number of simulation runs available to us. These results of the statistical tests cause us to consider the results for single node networks with a greater level of confidence as to the observed trends.

7.2.2 Entropy of output pairings

The control experiments show us the base level of confusion introduced into the output sequence by the nature of a simple communicating network. An attacker able to observe all network traffic will clearly be able to trace a message through such simple systems by observing the input and output sequences for each individual node rather than for the combination of all nodes in the system. However, restricting that view to the input and output sequences of the system as a whole demonstrates a level of randomness that is considerable even for small systems.

Although the graphs showing the spread of output distances between consecutive input pairs provide an overview of the systems' behaviour, they do not immediately provide us with a useful quantification of the randomness in the system. We therefore calculate the entropy of the observed distributions.

The entropy provides us with how many bits of information would be required to describe the output distance distribution, given that we know the input sequence. An ideal system maximises this entropy, and thus the confusion in the output ordering. As we consider 64 input messages, our systems contain the possibility of messages being up to 63 messages apart[1], there are 126 possible output distances. The maximum level of entropy that could possibly be observed is therefore that required to describe a uniform distribution over 126 values. The entropy of such a system can be calculated, according to Equation 5.1 to be slightly less than 7 bits.

Figure 7.4 shows an average change in entropy of the output distribution as the number of nodes in the network increase. For the single-buffer case there is clearly no confusion introduced by the system, however the entropy notably increases as extra buffers are added to the system. While we do not explore systems comprised of more than four nodes, we consider this entropy graph as the basis for comparison as we examine the alternative node types.

Figure 7.5 shows the change in entropy for a four buffer system as the message path is extended to include an increasing number of hops. In this figure, the level of statistical error in the graph is clearly significant in com-

[1]This statement is untrue in the case that messages are injected or removed from the system during a system run. This is discussed later.

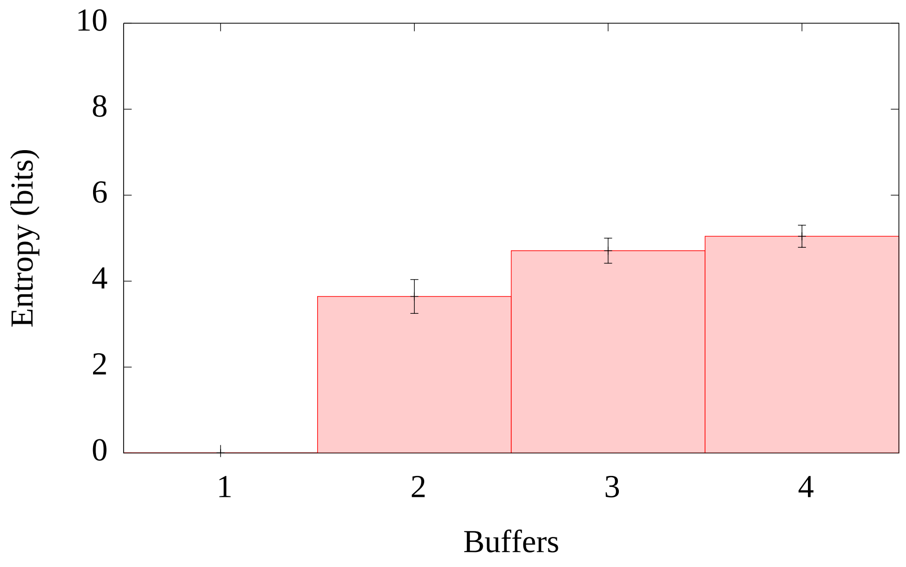


Figure 7.4: Entropy of the output distribution for single hop messages as the network size increases from 1 to 4 buffers.

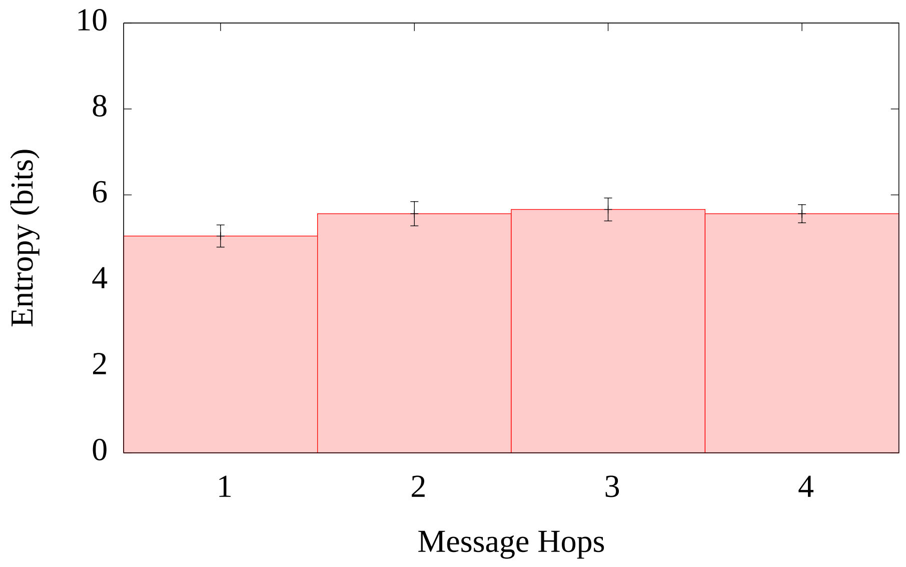


Figure 7.5: Entropy of the output pairing distribution for a four buffer network as the message paths increase from one to four hops.

parison to observed difference in levels for different path lengths. While we cannot gain much information from this graph, we can at least observe that increasing the message path length through simple buffers does not significantly increase the level of randomness beyond that already introduced by the number of nodes.

We now investigate the increase in entropy provided by each of the specific anonymity strategies.

7.3 Mix

The mix node is the most well-known, researched and popular form of anonymity strategy. There are many variations of the basic design, mainly concerned with the details of the flushing strategy to be employed.

In simulation, we take the simplest form of mix: the threshold mix. This system stores messages in an internal pool until a given threshold of messages is reached, then forwards all stored messages in random order. For convenience, we add a timing parameter to the mix that causes all messages to be flushed when a sufficiently large time period has passed without a message being received. This guarantees that all input messages are ultimately delivered, but that the timing parameter does not otherwise affect the behaviour of the mix.

Over the course of 64 messages, we investigate the effects of varying the threshold from 16 messages down to a single message. At the final stage, the mix should behave identically to a simple buffer process.

An advantage of the mixing strategy is that for a given single node we can achieve position swapping of messages; messages that enter the node in a particular order can leave the node in the reverse order. At the level of a single node, mixing is the only strategy that can achieve this. Due to this property, we expect to see a relatively wide spread, both positive and negative, in the output distance between pairs of messages.

For the case of a single node, we observe the spread of messages demonstrated in Figure 7.6. The distribution of spread is largely symmetrical, although clearly biased to the positive side of the distribution. This skew represents the bias caused by messages pairs that fall into separate flushings of the node not being mixed with each other. Such pairs of messages are thus unable to reverse their position in the output sequence. As the threshold become increasingly low, it becomes significant that a larger number of consecutive pairs are separated by not occurring in the same round of mixing. This bias becomes increasingly pronounced as the threshold approaches one.

For each change in threshold size the null hypothesis, that the two experi-

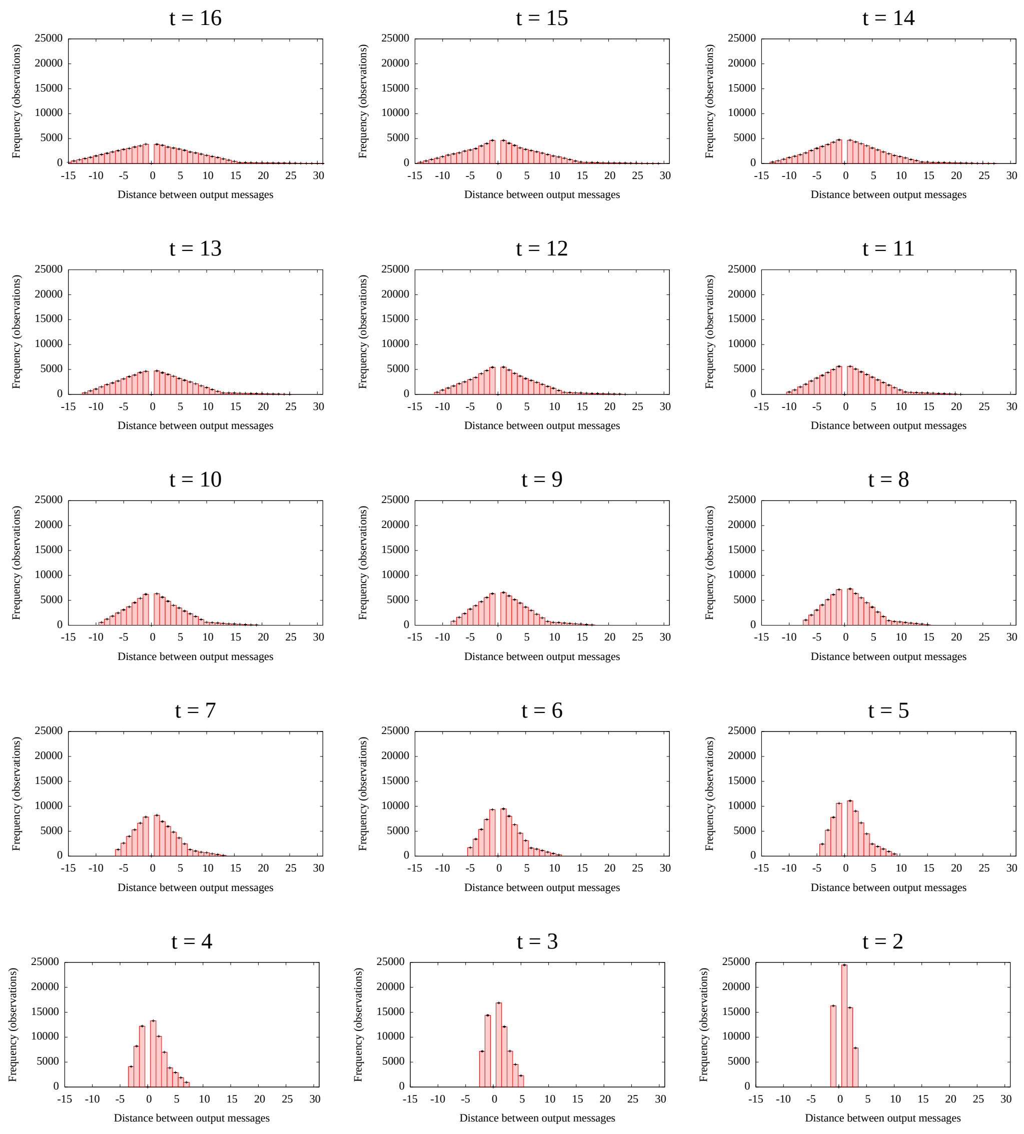


Figure 7.6: Output distribution for a single MIX node as threshold value (t) varies.

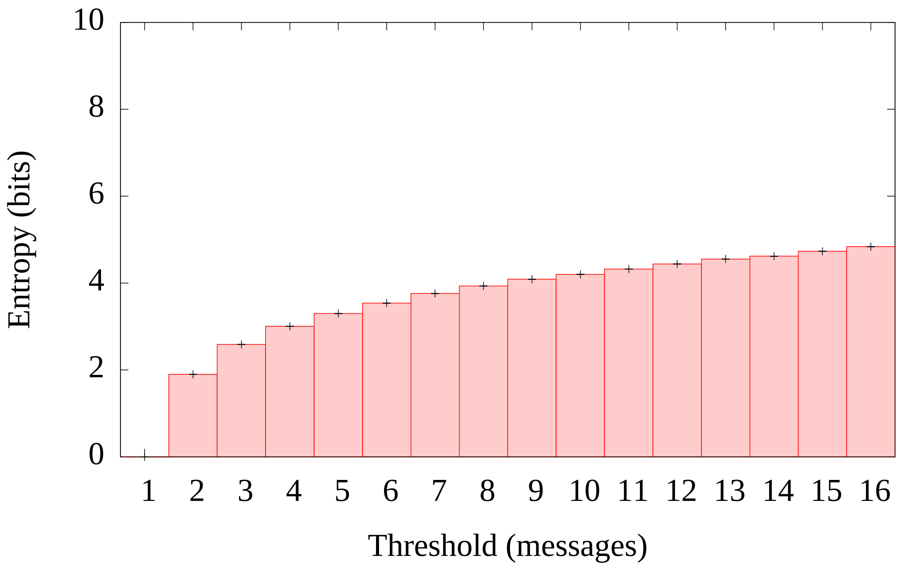


Figure 7.7: Entropy change for a single MIX node as the threshold varies.

ments were drawn from the same distribution, is rejected by the Kolmogorov-Smirnov test at a confidence level of 99%. In testing significance of the results, two sample runs of each experiment were not be judged to be drawn from different distributions at the same confidence level.

The level of entropy in the output distances of the simulation runs clearly follows the size of the mix threshold, as can be seen in Figure 7.7.

(Full results of all experiments are shown in appendix A.)

The variation in output sequence entropy as the number of hops between the mix nodes increases is shown in Figure 7.8. There is a noticeable increase in entropy as the threshold increases, as would be expected. However it must be noted that high levels of statistical error in the results of the simulation apply here, as in the four buffer-node experiments.

The MIX clearly demonstrates a desirable level of randomness in the case of a single node, and this continues as the number of nodes in the network increase. When considered in the traditional Dolev-Yao framework discussed in Section 2.6, where an attacker is aware of all traffic on the network, the MIX node maintains a high level of confusion as to the location of messages in the output.

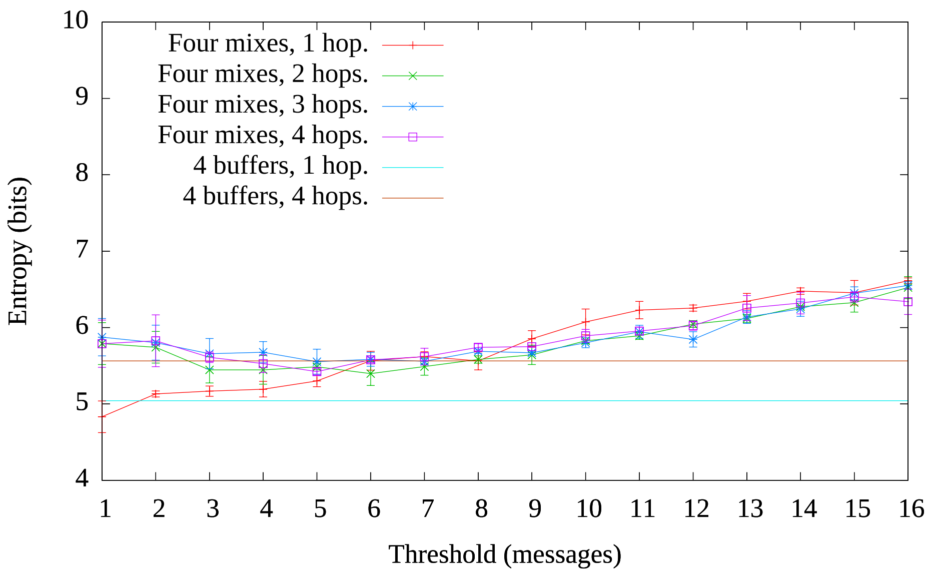


Figure 7.8: Entropy of Mix node networks as flushing threshold varies.

7.4 Flood

The introduction of dummy traffic into the stream of messages has been proposed as a method to improve other anonymity systems. One of the advantages of this approach is that, theoretically, it can prevent an attacker from knowing that a genuine message has been sent at all — it breaks the fundamental principle that one cannot hide the fact than an event occurred.

Our simulation does not address this aspect of the flooding strategy. We concern ourselves only with the increase in entropy in the output sequence as the ratio of flood messages to real messages increases.

We quantify the level of flooding in terms of the relative percentage of flood messages to real messages. Starting from 10% flood messages, in which one dummy message is sent for ever nine real messages, we examine the change in entropy and spread as we approach 90% flood messages, in which nine dummy messages are sent for each real message.

As the flooding strategy does not implicitly allow for pairs of messages to swap their positions in the output sequence, instead injecting extra messages into the stream, we expect to see a long-tailed right-skew to the possible output distances that increases with the percentage of flood messages. The spread of messages for a single flood node is show in Figure 7.9.

Examining the output entropies from the single Flood node, shown in Figure 7.9, we observe a greatly increasing level of entropy in the output

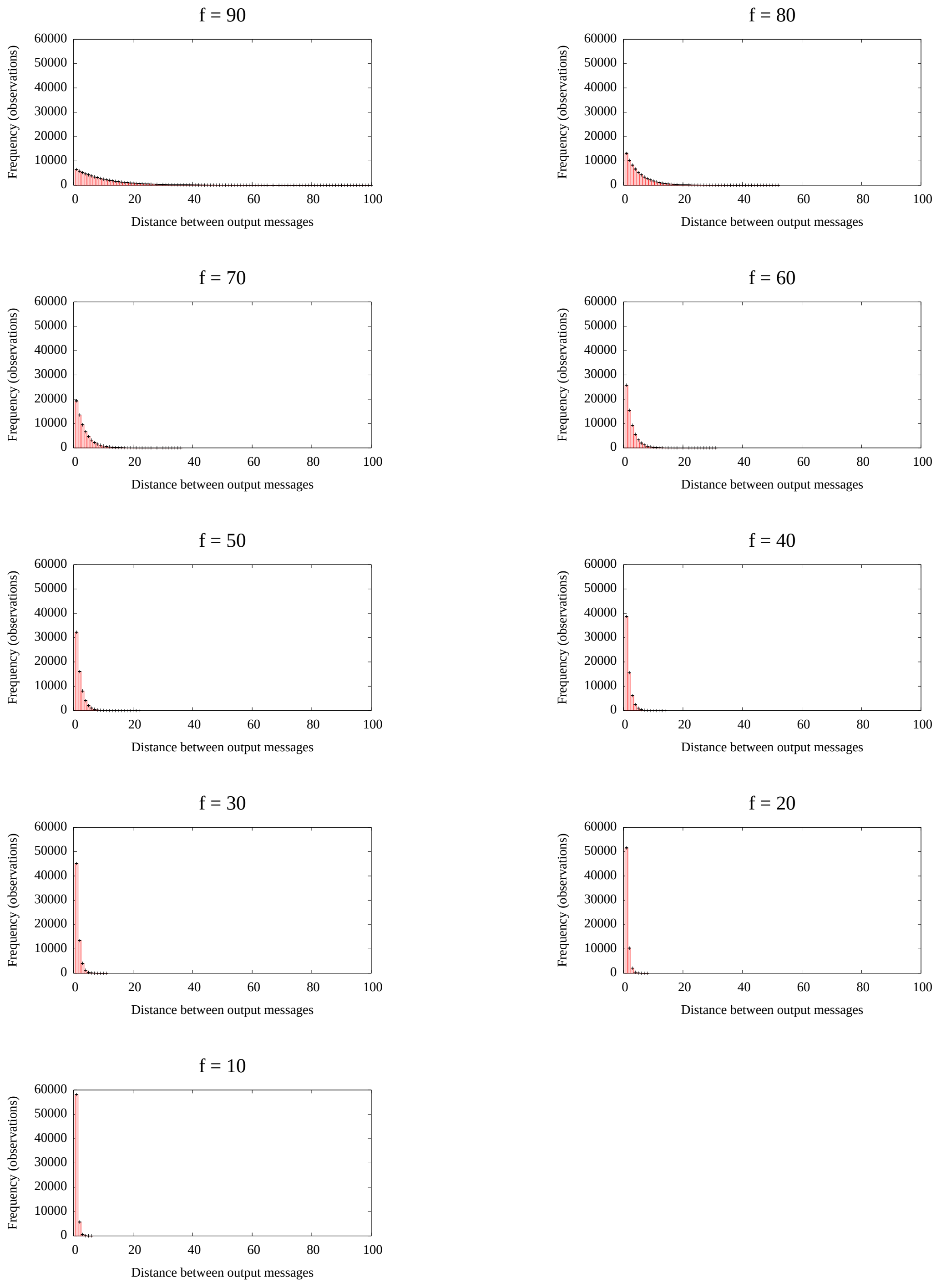


Figure 7.9: Output of single Flood node as percentage of flood messages (f) varies.

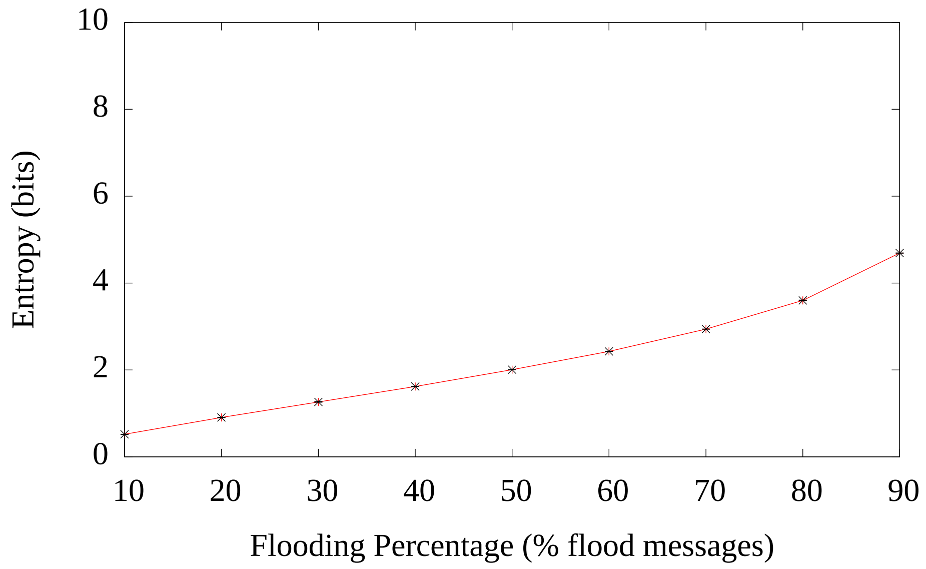


Figure 7.10: Entropy change for a single FLOOD node as the flooding percentage varies.

distances. Although limited by the great level of bandwidth required, this node type has no upper limit to the entropy in the output sequence. An increase in the flooding percentage increases the number of messages in the output sequence and thus increases the possible level of confusion.

In analysis of these tests we can again reject the null hypothesis at a significance level of 99% for each change of the flooding ratio, showing that varying the flooding parameter by 10% results in a statistically significant difference in the distribution of the output results.

A difference between the FLOOD node and other nodes is that each FLOOD node introduces messages into the system and thus causes an increase in the size, and therefore the maximum possible entropy, of the output sequence. Similarly, for each extra hop across which a message travels, each node in the system will add another level of cover traffic. The addition of a flood node into a system is therefore expected to strictly increase the randomness of any system, as will an increase in the path length of messages.

Figure 7.11 demonstrates the output spread of messages from systems made up from four FLOOD nodes as the message path increases. As can be seen, the spread of possible output message distances becomes extreme even at this low flooding percentage.

To better understand the level of entropy introduced into the output

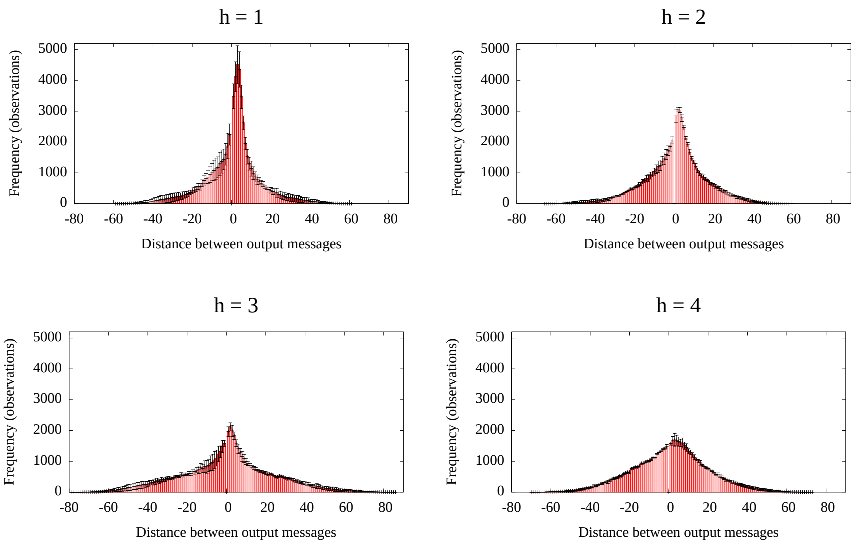


Figure 7.11: Output distribution for four FLOOD nodes as the number of message hops (h) varies with flooding rate $f = 20\%$.

sequence, Figure 7.12 shows the entropy of a four FLOOD node system as the message path length increases from one to four hops. These show that the increase in entropy over the basic buffer node system is significant, even for the case of single hop messages. Even in the case of a single hop flood node, a ratio of 30% flood messages to 70% genuine messages is sufficient to raise the confusion in the output sequence above the level of a simple buffer with messages travelling over four hops.

Despite the advantage of FLOOD nodes, the bandwidth required to support the strategy is very high. The advantages of the method are somewhat unfairly highlighted by our analysis as we do not consider the level of traffic that flooding produces. We do not analyse the levels of traffic required for a full FLOOD network, although they are sufficiently high to have prevented any implementations of a FLOODING strategy from being deployed. Exploration of this aspect of the FLOOD node would be a useful extension of the work presented here.

7.5 Hide

In a HIDE system, confusion is introduced into the relative orderings of the input and output sequences through the action of unobserved senders. By re-

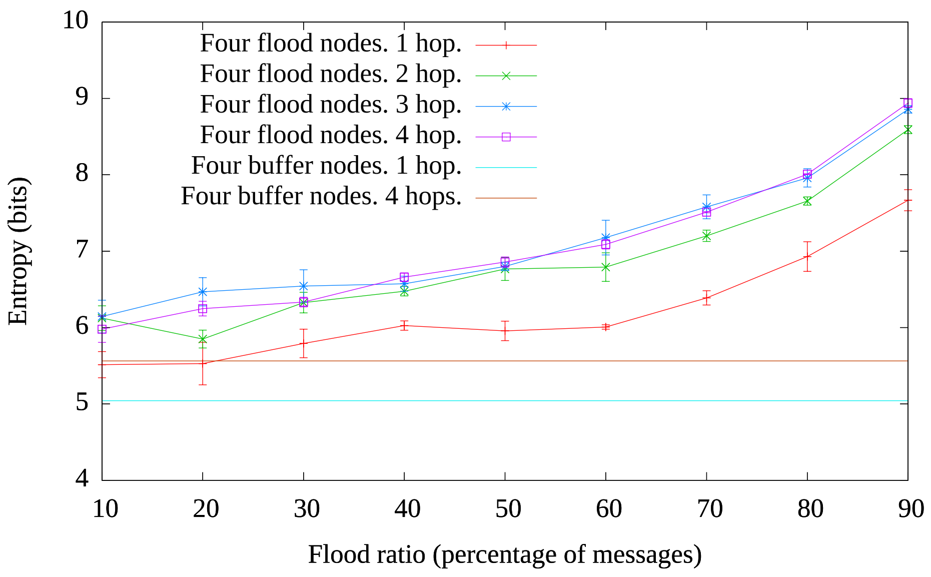


Figure 7.12: Entropy of Flood nodes as flooding percentage varies.

stricting the observer to consider only a certain subset of the possible senders we gain an effect similar to that of dummy traffic in its effect on the entropy of the output.

To simulate this effect, we consider a number of hidden senders acting alongside those that we maintain throughout the other experiments. These hidden senders are interleaved with the standard sending actor, and so introduce an element of randomness directly into the input sequence that is not present in other experiments. Each hidden sender adds 8 messages into the input sequence, that may be injected at any point. In considering the view of an attacker observing the node, we consider that the attacker will be unaware that the messages from hidden senders have been injected into the system at all. The nodes themselves in these Hide experiments are simple buffers.

When we consider consecutive pairs in the input, we do not perform any analysis of those messages that are produced by the hidden senders.

As Figure 7.13 shows, the spread of output messages is not greatly increased by the interaction of hidden senders. This is reflected by the variance in entropy as the hidden senders increase. It is worth noting that the hidden senders, as implemented in our simulation, is similar to a low level form of message flooding. As such, we would expect the entropy of the system to continue increasing indefinitely as the proportion of hidden senders increased.

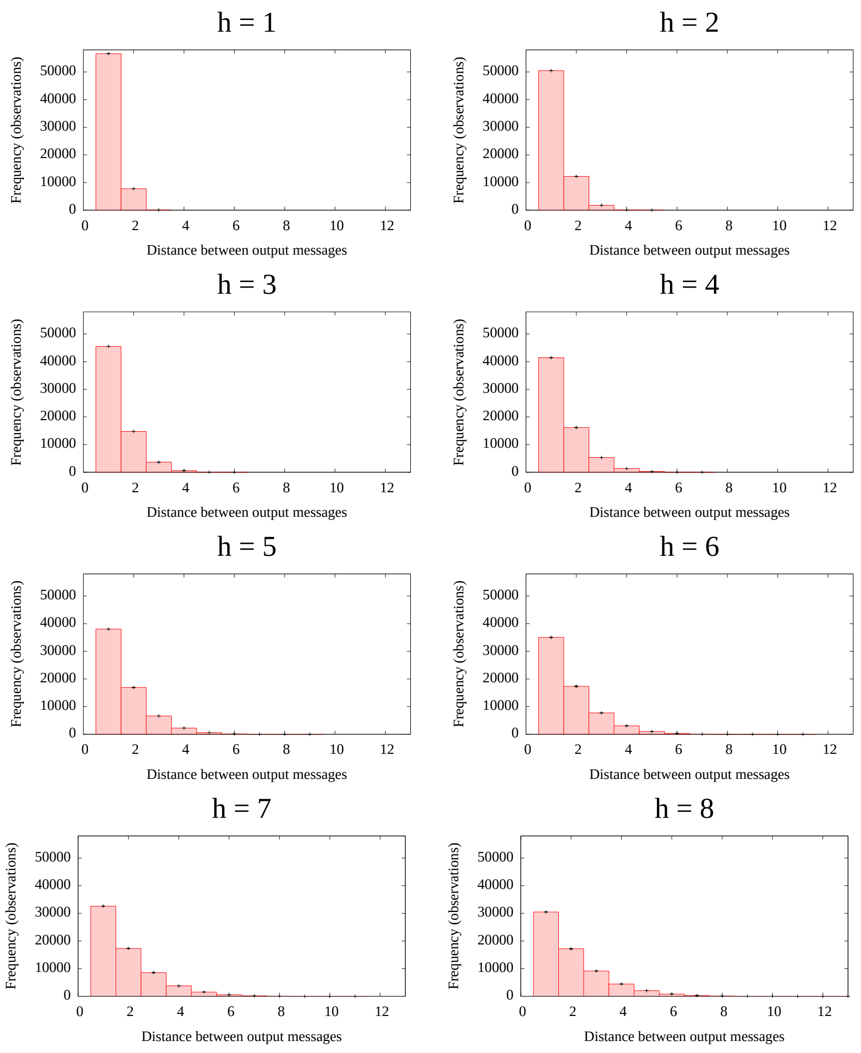


Figure 7.13: Spread of output messages for a single HIDE node as number of hidden senders (h) varies.

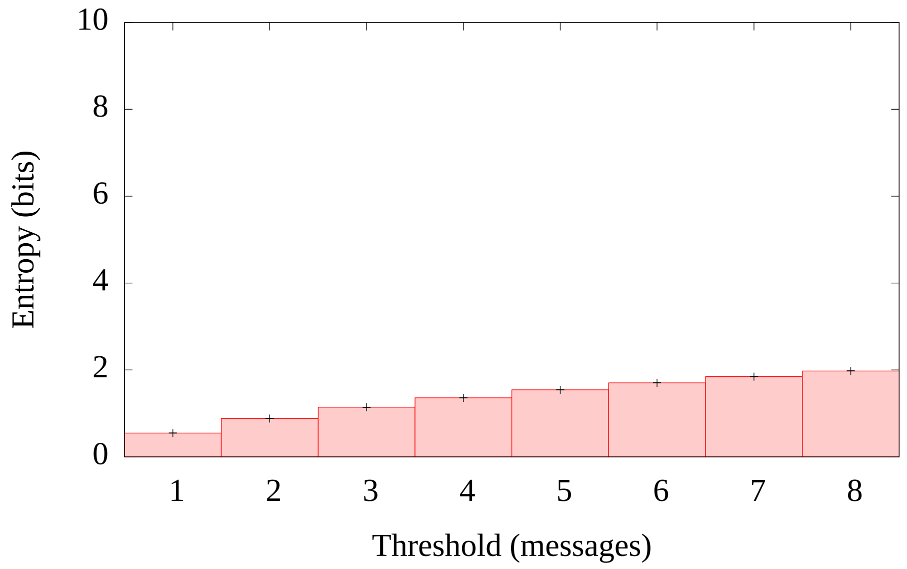


Figure 7.14: Entropy change for a single-node HIDE system as the number of hidden senders varies.

Despite this, Figure 7.14 implies that the number of hidden senders required to achieve a significant level of randomness is relatively high.

As the level of randomness is defined at the point of message entry into the system, we do not expect a great increase in this entropy as the number of nodes or hops increases. Figure 7.15 compares the entropy of a four node system with an increasing number of hidden senders. There is an apparent increase in the entropy of the system, despite a clear level of statistical noise in the results; however the overall increase is in the range of single bit as the number of hidden senders varies from 1 to 8.

The number of extra senders required to significantly introduce randomness into the output sequence is therefore relatively high. In practical consideration, however, it may be expected that a real-world network will contain a large number of senders that cannot be observed by an attacker. As a specific method for introducing randomness the HIDE strategy is not particularly efficient, but is a not insignificant advantage in systems where the attacker model is restricted to a non-global view of the network.

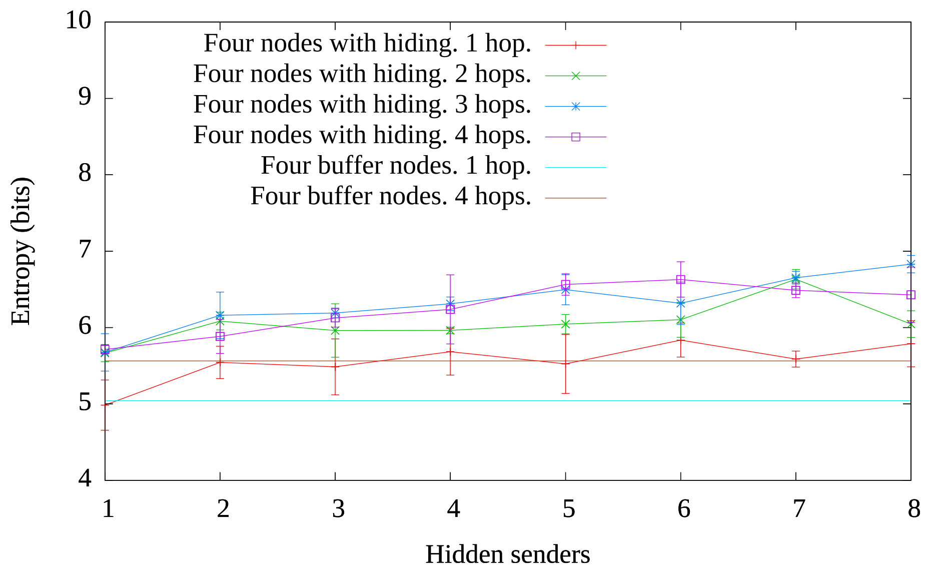


Figure 7.15: Overlaid entropies of four node Hide systems with increasing numbers of hidden senders. The base entropy of a buffer network is shown for comparison.

7.6 Drop

Pure Drop nodes are, almost certainly, the least useful basic form of anonymity system. Here, the observer is confused as to the possible relationship between input and output messages due to a given percentage of messages being dropped as they travel across the network. One problem with this system is clear: we lose the reliable message delivery that is desirable in most communicating systems.

The parameter that we vary in the Drop node experiments the percentage of messages that are dropped by each node rather than being forwarded to the next hop on their route. For the single node system, therefore, consecutive pairs of input messages can only exit the node consecutively. The alternative is that one or both of messages never appear in the output sequence. This makes a graph demonstrating the spread of output messages less appropriate than for our earlier systems. We will instead show only the entropy for a single node system, representing the range of possibilities including the chance that one of both messages are dropped in the output.

Figure 7.16 demonstrates that for a single node we cannot expect a high level of entropy in the output sequence. The entropy of the system maximises at a fifty percent probability of dropping a message. This represents the point

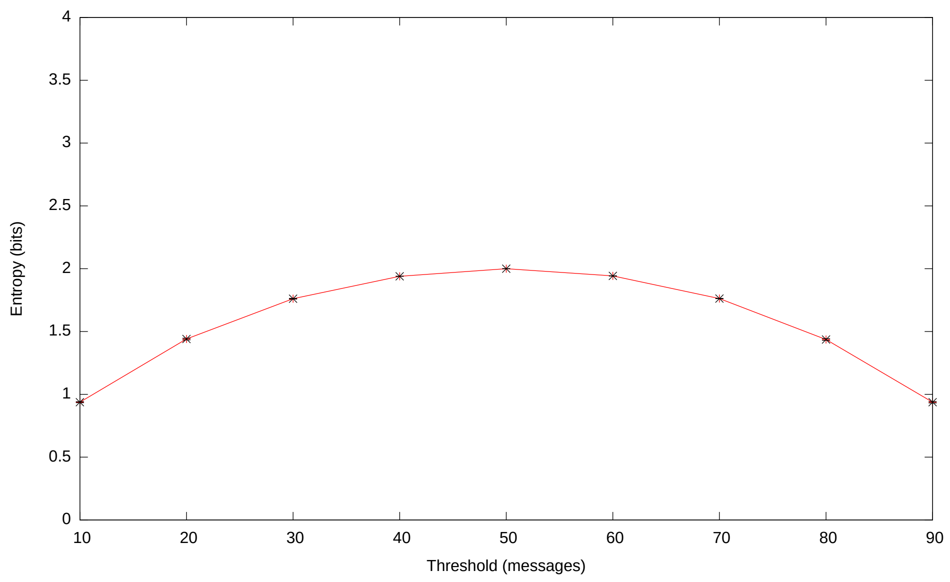


Figure 7.16: Single Drop node with increasing probability of dropping a message.

at which there is an equal chance of messages being classified as "dropped" rather than being one space apart in the output sequence.

We may consider the Crowds-style node as a more useful exposition of an anonymity strategy expressible via internal random choice.

7.7 Crowds

While a pure Drop node may be considered the archetype for nodes that implement a strategy based on randomised internal choice, we consider that the Crowds node is a more useful subject for analysis. The inherent impracticality of the Drop node is not carried over to the Crowds approach, which has the added advantage of having been implemented as a real-world anonymity system. As such, we show here a similar analysis of the Crowds node to that conducted for the other node types.

The Crowds system is a realisation of random internal choice as a mechanism for introducing random reordering into a stream of messages. As this approach chooses to route messages either to another node or to deliver to the recipient the system intrinsically functions through the combination of nodes and cannot be employed as a single node. Each node, on receiving a message, probabilistically chooses whether to deliver that message to its

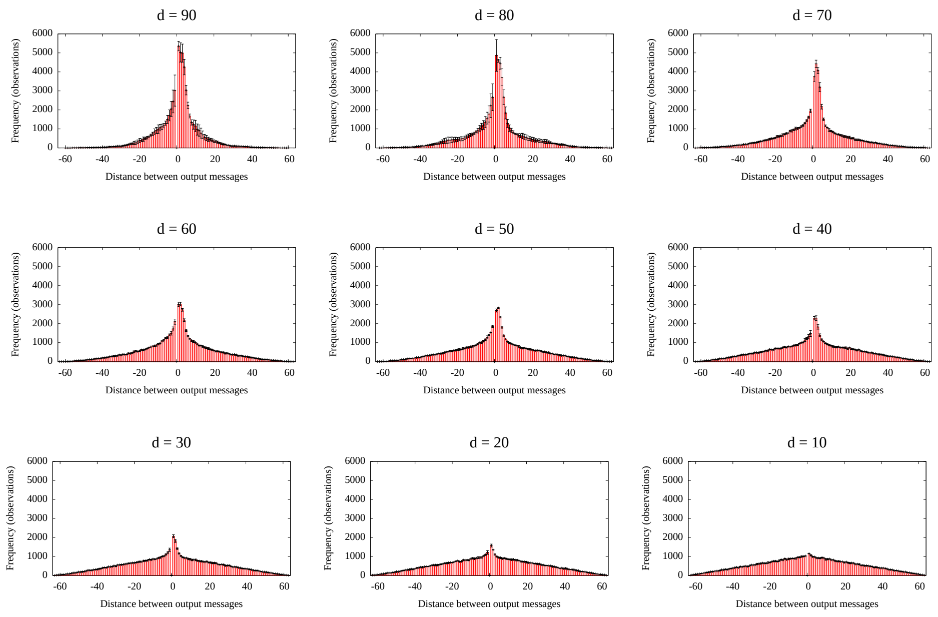


Figure 7.17: Spread of output for four Crowds-style nodes as the percentage probability of messages being delivered rather than forwarded (d) varies.

intended receiver, or to forward the message to another Crowds node. This approach causes messages to travel around the system for an indeterminate number of hops before being delivered to their ultimate recipient.

Due to the nature of the Crowds approach, we cannot vary our parameters as for the other experiment types in this chapter. The number of nodes in the system must be greater than one[2], and the path length of a message varies with the probability of a message being delivered. Figure 7.17 shows the results of a four node system as the probability of a message being randomly forwarded to a new node varies.

Although the system has four buffers, and so will inherently demonstrate a level of mixing in the output, the spread of messages is altered significantly from the corresponding simple buffer systems. As the probability of a mes-

[2]In fact, a Crowds-style approach could feasibly be applied a single node if one were to allow the node to deliver to itself. In such a scenario, each message would either be delivered to its intended recipient or returned to the end of the queue of messages when considered for delivery.

Such an approach mimics certain proposed MIX strategies, such as the Stop-and-Go Mix of Kesdogan et al. (1998) that delay messages individually rather than considering them in batches. One could argue that such "Mixes" are better classified as a form of single-node DROP strategy, and the abstract model of Chapter 4 would support this view.

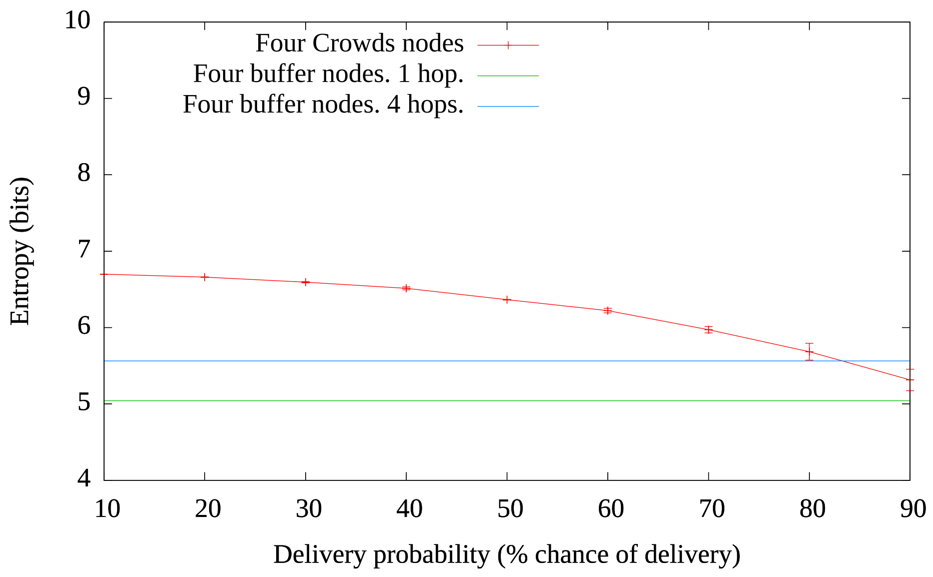


Figure 7.18: Entropy of four-node crowds-style network as delivery percentage varies.

sage being decreases, the number of messages remaining in the system at a given time also increases. We can see, as the chance of delivery becomes very low, that the system approaches a state where all messages that enter the system are effectively mixed. In this situation, the Crowds network behaves similarly to a Mix node with a threshold corresponding to the number of messages in the system.

Figure 7.18 shows the level of entropy in the system as the likelihood of delivering a messages rather than forwarding it varies.

7.8 Combined Systems

We have examined each of the fundamental anonymity node types. For each anonymity strategy, we have examined the node in isolation and as part of a small network comprised of multiple copies of that node. In this section we briefly examine the effect that combining multiple node types in a single system has on the level of randomness introduced into the output sequence.

The combinations that we have chosen each consist of two types of node. Every combination of Mix nodes, Flood nodes and Crowds nodes are examined. We choose to ignore the Hide anonymity strategy, as it is more easily conceived of as a restriction on the attacker's capabilities than a node type in itself; as such, it does not reflect a genuine combination of node types. The Drop node type is omitted due to its inherent impracticality, and we instead consider the more realistic Crowds-style node.

For each experiment, we choose a suitable value for the fundamental parameter of each node. The nodes are then combined in a system which varies the ratio of the two node types to each other. The results are processed as for the multiple node systems in the previous section.

7.9 Mix and Flood

In the combination of Mix and Flood nodes, a threshold of 16 messages was selected for the mix along with a fifty percent flooding ratio for the flooding nodes. These values were chosen as a useful representative level of the anonymity properties of each node type. As we are interested largely in the interaction between the nodes these systems, the message path length is set to four hops in order that all nodes will be visited.

Figures 7.19 and 7.20 provide a view of the combination of Mix nodes with Flood nodes. The level of entropy introduced into the output sequence by the two node types in isolation is highlighted in Figure 7.20. One point of interest concerning this graph is that the variation between combinations of the differing node types does not cause any great change in the level of entropy provided by the system. Note that entropies in Figure 7.20 range over less than a single bit in total; however the combination of the two node types results, in one case, in a higher level of entropy than that provided by either homogeneous system. A Mix node combined with three Flood nodes introduces a higher level of confusion into the output sequence than either four Flood nodes or of four Mix nodes.

Whilst the increase in entropy resulting from this combination is small in the example shown, it demonstrates that the combination of different types of anonymity system can show an improved level of confusion than that caused by a single approach.

7.10 Crowds and Flood

In combining a Crowds-style node with other node types there is the problem of indeterminate message path lengths. For the basic node types defined in the model, the number of hops taken by each message is a fixed number. Further, the implementation of nodes in the simulator defines the full message path on creation. In contrast to this, the Crowds approach implicitly relies

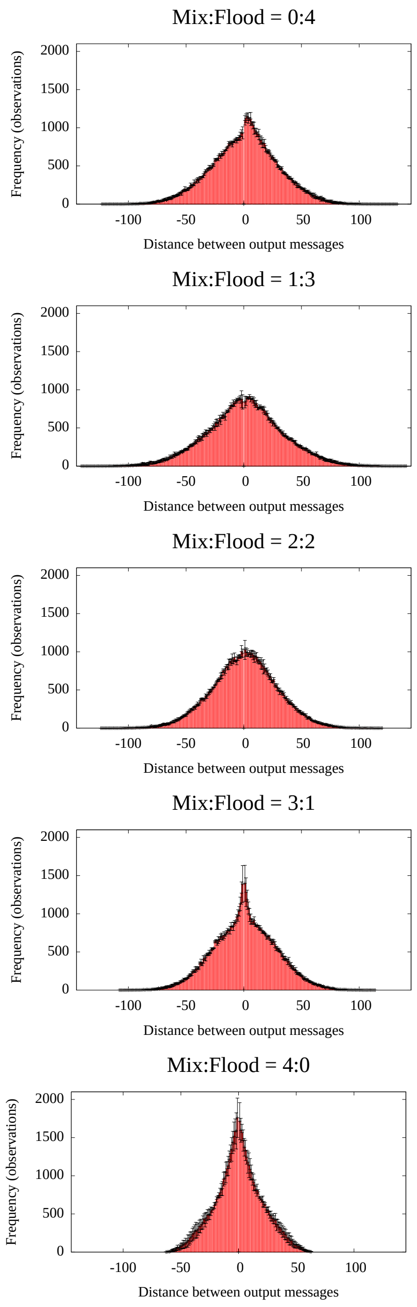


Figure 7.19: Spread of output for networks of Flood and Mix nodes as the number of each node type varies.

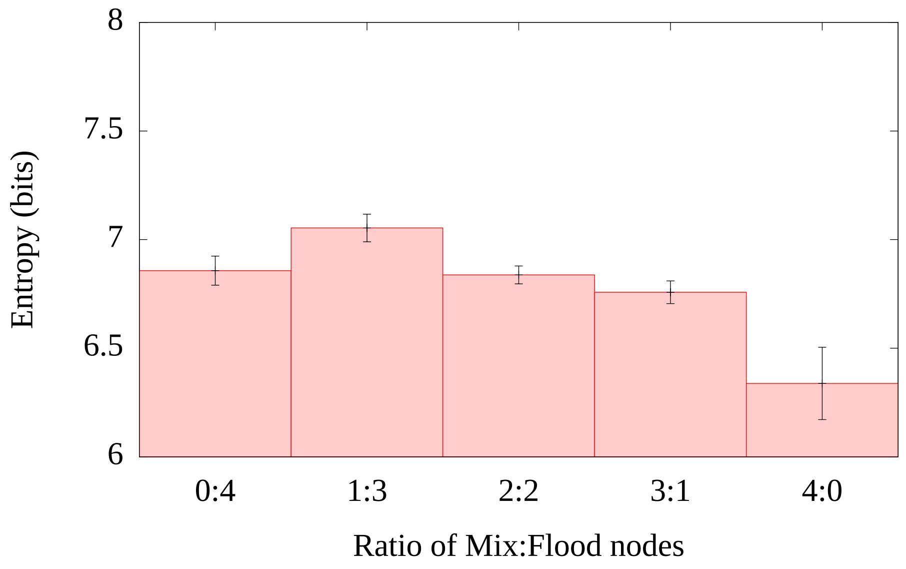


Figure 7.20: Entropy of networks of Mix and Flood nodes.

on a path that is variable in both length and route.

The variability of message paths in a Crowds system causes some difficulties in combining the nodes with other node types that rely on fixed message paths. There are a number of possible solutions to this problem, which we describe here.

7.10.1 Enforce hops

A possible approach in resolving the problem of variable message paths for Crowds nodes is to define a fixed message path for each message as usual. Different node types in the system would respect this *a priori* path to greater or lesser extents. A Crowds node would respect the *length* of the path, but not the routing information. On receiving a message a Crowds node would decrement the remaining number of hops on the path; if this indicated that the crowds node was the last hop on the route, the message would be delivered directly; if the value indicated that more hops were required the message would be forwarded to another random node, without respecting the route defined in the message.

This scheme is a violation of the principle of the Crowds node. Each message is no longer subject to being randomly dropped from the system, only the route of each message gains a random element. For a system in which messages already contain a randomly generated route, this is unlikely to prove any new or interesting behaviour. Considering an attacker who may function as a node on the route there could be an added layer of confusion in not storing routing information in each message, but this is both beyond the scope of our simulation and unlikely to be of any great benefit. Such a system would presumably already have a scheme in place to prevent a message's route from being known to all nodes on the path.

7.10.2 Isolate Crowds nodes

A second approach to combining Crowds nodes with other node types is to consider the step of entering a Crowds node as a single hop, and each subsequent entry into a Crowds node as a "null" hop. In such a system, a message would travel over its given number of hops. If the routing information results in the message entering a Crowds node during the path, the message will travel randomly between nodes until that random choice results in the message being forwarded to a non-Crowd node type. The message path would then continue as normal. In the case that entering a Crowds node was the last hop on the route, the message would be randomly forwarded between nodes with the added possibility of immediate delivery to the recipient.

This scheme is quite attractive in enforcing a given number of hops between nodes in the system, treating the entire body of Crowds nodes as a form of "super-node". An argument against implementing approach is that it again introduces an artificial element to the Crowds nodes by removing the option of immediate delivery if the predefined route has not been fully traversed. Further, the system adds a level of extra checks and special cases of the message state. It would be preferable to maintain the behaviour of individual nodes as closely to their "natural" state as possible.

7.10.3 No enforcement of path

The simplest approach, and the one that we have chosen to implement, is to allow the Crowds nodes to entirely disregard the path encoded in the message. On receiving a message a Crowds nodes will randomly forward the message to any other node in the system, or deliver it to the intended recipient. If the message is forwarded to another Crowds node, the process repeats. If the message is forwarded to a non-Crowds node, that node will respect the path encoded in the message and behave appropriately.

This preserves the most natural-seeming behaviour of each node type without adding extra layers of complexity or imposing an idea of the "best" behaviour of the system.

7.10.4 Results

Applying the approach discussed in Section 7.10.3 to systems comprised of varying numbers of Crowds and FLOOD nodes, we observe a spread of output messages as shown in Figure 7.21. The delivery probability of the Crowds nodes was set at fifty percent, and the FLOOD node was defined, as above, to inject a fifty percent ratio of delivery traffic into the message stream.

There is a notable spike in the frequency of output messages appearing within a small distance, likely caused by the possibility of a Crowds node immediately delivering a message to its recipient. Despite this, the spread of the graph is very widely distributed and approaching the desired symmetrical distribution.

Figure 7.22 again demonstrates a higher entropy in the output sequence of the combined system over that possible for either of its constituent node types. In the combination of FLOOD and Crowds nodes, this behaviour is particularly pronounced. In the most extreme case, that of a single Crowds node combined with three FLOOD nodes, the entropy of the output sequence is over one bit higher than that of the pure Crowds system. Exceeding the entropy of the lesser Crowds system is less surprising, however, than the fact that the system increases the entropy of the higher FLOOD node system by over half a bit.

This combination of node types appears to be a highly successful approach according to our criterion. This desirable behaviour would increase were we to allow the Crowds nodes to process the dummy traffic from the FLOOD nodes, that traffic being immediately dropped in the current implementation.

7.11 Crowds and Mix

Combining Crowds-style nodes with the more typical node types has been discussed in Section 7.10. The arguments made there are equally applicable to combining Crowds nodes with FLOOD or MIX nodes. We apply the same approach to the combination of Crowds and MIX nodes. The probability of delivery for Crowds nodes was set to fifty percent, and a threshold of 16 messages was selected for the MIX nodes.

Figure 7.23 again shows a symmetric distribution of output distance frequencies that demonstrates a clear spike for consecutive input messages to leave the system within a few messages of each other. This spike becomes more pronounced as the number of CROWDS nodes in the system increases. Again, the spike seems to correspond to the chance of a messages entering a Crowds node and being immediately delivered, thus removing any notable

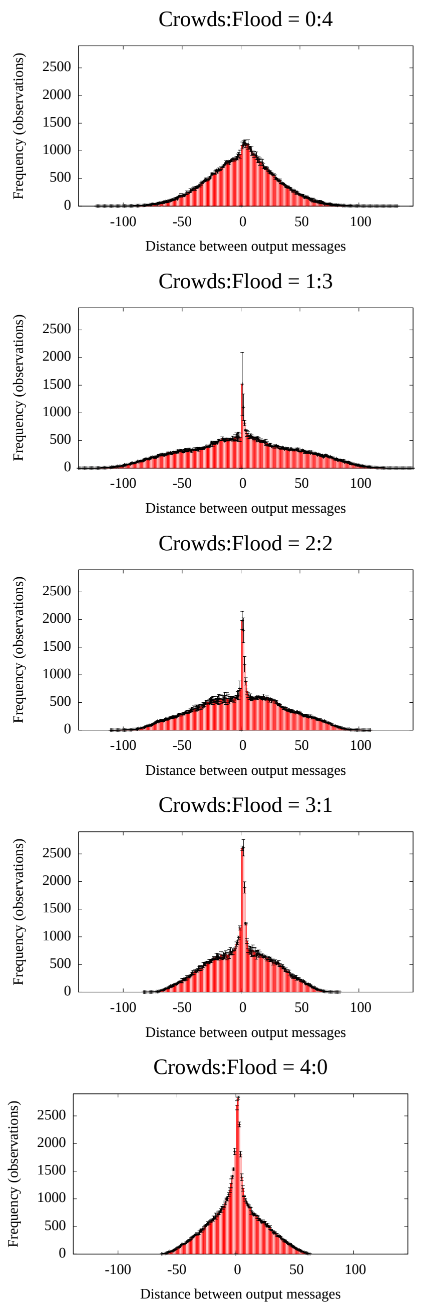


Figure 7.21: Spread of output for networks of Crowds and FLOOD nodes as the number of each node type varies.

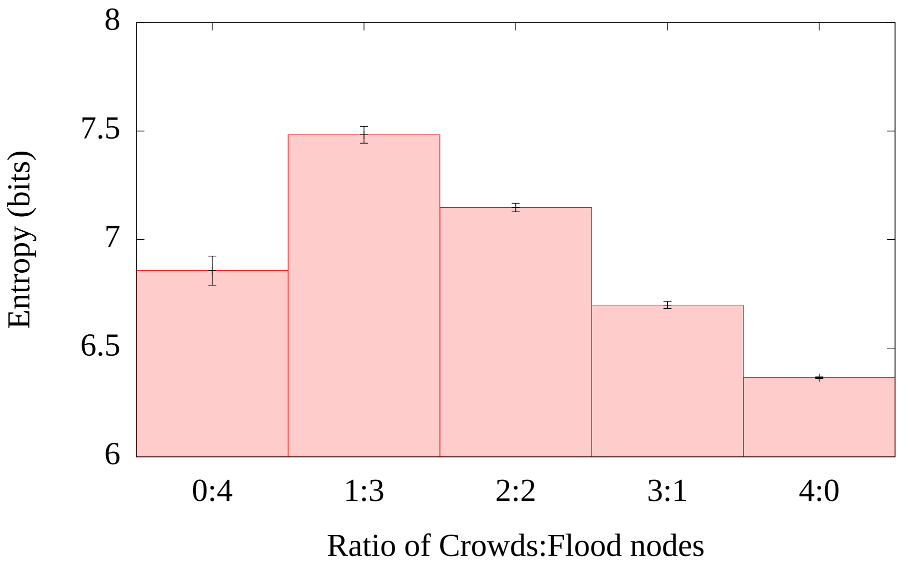


Figure 7.22: Entropy of systems comprised of varying numbers of Crowds and FLOOD nodes.

confusion in the output sequence. In other situations, however, the combination of a Crowds node with a number of MIX nodes appears to benefit the behaviour of the system by increasing the uniformity of the output distances.

Of the combined systems that we have examined, the pairing of MIX and Crowds nodes is notable in that any combination of the two node types out-performs a homogeneous system of either type. The overall change in entropy is small, ranging within half a bit, and is thus subject to some level of doubt from potential experimental error. Despite this the system appears to follow a trend suggested by the previous combinations, that the combining two different forms of anonymity node in a single system can have benefits for the amount of entropy introduced into the output sequence.

7.12 Summary

We have compared the level of randomness introduced into a sequence of messages by a variety of systems. The systems examined have been representative of the four basic anonymity strategies defined in Chapter 4, in each case exploring the behaviour of the system as the characteristic parameter of the strategy varies.

There are difficulties with directly comparing the level of entropy intro-

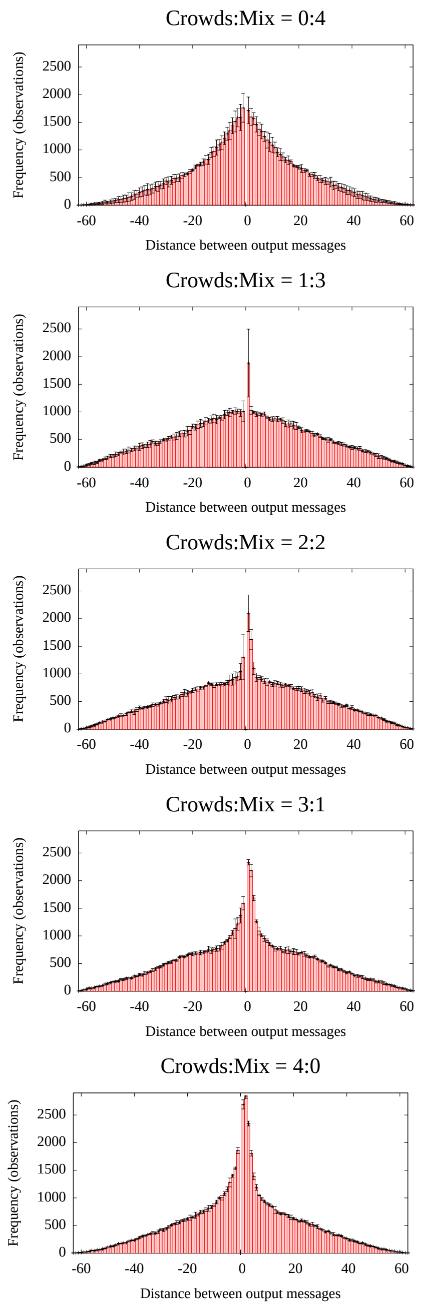


Figure 7.23: Spread of messages in systems comprised of differing numbers of Crowds and Mix nodes.

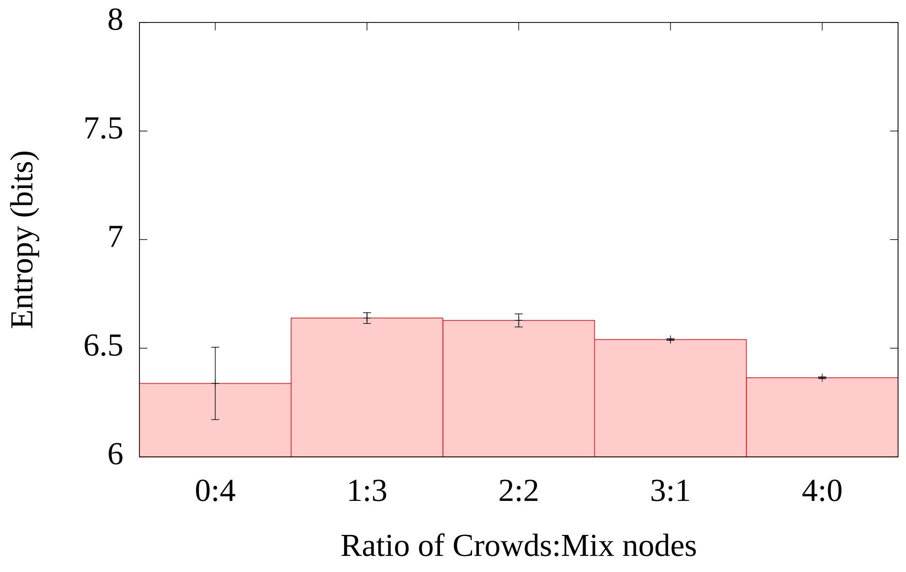


Figure 7.24: Entropy of systems comprised of varying numbers of Crowds and MIX nodes, compared against entropies of pure MIX and pure Crowds systems.

duced into the output sequence by the different anonymity strategies, as their characteristic parameters are not themselves comparable. The level of traffic introduced by a FLOOD node cannot easily be compared against the delivery probability of a Crowds node. Despite this, we have selected ranges of values for each node that cover a realistic set of possible values within the context of the simulations.

The FLOOD node, as might be expected, is capable of demonstrating a higher level of entropy in the output sequence than any other node type. The introduction of extra messages into the output stream removes the upper bound on the entropy that is enforced by a fixed number of messages. This advantage is reduced, however, by the strategy's high bandwidth requirements.

The MIX node demonstrates a useful level of entropy even with relatively small mixing thresholds, and results in a far more uniform distribution of output probabilities than that demonstrated by the single FLOOD nodes.

Employing the HIDE strategy causes a similar effect to that of a FLOOD node, but is more restricted in its ability to randomise the message stream. The effects of a HIDE system would be more apparent in larger multi-node systems in which several intermediate nodes, as well as sections of the output stream, could be hidden from view. Our approach to comparing and measuring anonymity systems based on the entropy of the output sequence is not the most suited to these systems.

The DROP strategy in its purest form has, as expected, proved to be almost useless for the desired goals of an anonymity system. The unreliability of message delivery and the small level of entropy that is introduced into the output sequence renders this strategy by far the least desirable of the basic forms. Despite this, a view of the conditional entropy of the system that related input message distances greater than one to output message distances could produce a different view of these systems, but this is unlikely to produce any startling results.

As a representative of the DROP strategy, the Crowds node demonstrates interesting behaviour. This system does not function as a single node, and we can thus only compare it against equivalent multi-node systems of the other strategies. In this context, a combination of Crowds nodes appear to function effectively as a multi-node Mix. As the probability of a message being delivered becomes increasingly low, messages are more likely to leave the system in a random order rather than that imposed by the ordering of the input sequence. The comparison between a MIX strategy of sufficiently high threshold and a series of Crowds node with sufficiently low delivery probability is striking; each approaches the ideal uniform distribution of output probabilities constrained by the possible combinations of pairs in the system.

7.12.1 Multiple Nodes

Extending systems of single nodes to multiple node systems introduce a surprisingly high level of entropy, even when the nodes are deterministic buffers as demonstrated in Figure 7.1. This effect is pronounced even for the small four-node systems that we consider. Perhaps due to the already high level of entropy introduced by such multi-node systems, we do not observe a great increase in output sequence entropy as the message path length is increased from one to to four hops. It is possible that in larger systems that consider a greater number of messages this effect would be more noticeable.

For each system comprised of multiple homogeneous nodes, the distribution of output sequences is "smoothed" to more closely represent the desired symmetrical distribution. This effect is most notable, and beneficial, in systems such as that of FLOOD nodes, where the mixing effect of a multiple node system allows message positions to become reversed.

The most interesting results from multiple node simulations have come from combining multiple node types. In each combined system that was simulated, a higher level of entropy in the output sequence than that possible from equivalent systems of either node type was observed. While there are several anonymity considerations beyond the level of mixing in a system of multiple nodes, these results suggest that the combination of various node types in a single system could be a viable approach towards building future anonymity systems. When considered in light of other properties, such as the amount of time and level of traffic that each node type requires, a combination of different node types in a single system could be used to yield a system able to meet a more flexible set of message delivery requirements.

Part III

Conclusions

Chapter 8

Conclusions

8.1 Overview

This thesis is concerned with the definition of anonymity as a property of communicating systems, the classification of systems that provide this property and the analysis of the methods employed by these systems.

We define anonymity in communicating systems as an observer's confusion concerning the linkage between observed events. Using this definition, we identify a number of fundamental strategies that introduce or enhance this property. We then present a classification of anonymity systems based on these strategies. Finally, we use simulations to analyse the effectiveness of these methods according to an information theoretic quantification of the anonymity that they provide.

8.2 Model

The fundamental requirement for anonymity in a communicating system is the existence of uncertainty as to linkages between events. In our model of communicating systems this uncertainty is brought about by enforcing multiple possible explanations for the state of a system with respect to the ownership of observed events.

For a system to meet our definition of anonymity, it must fulfil two requirements:

- **Data Independence:** Messages entering and leaving some system must be unlinkable by an observer that has access to the content of messages.

- **Ordering Independence:** The ordering of messages must not reveal unique linkages between observed events.

The first of these properties is achieved through appropriate use of cryptography, and we have chosen not to focus on this aspect of anonymity. The second property, *ordering independence*, is the focus of the work presented in this thesis.

To prevent a unique correlation between messages entering and leaving a system based on the flow of traffic we assert that a system must allow for multiple possible output sequences for a given sequence of input. Such systems must therefore exhibit some randomness in their message forwarding behaviour.

We demonstrate that existing systems for anonymising messages may be represented as one or more fundamental mechanisms that introduce randomness in communicating systems. This realisation allows us to classify anonymity systems according to the strategy that they employ to introduce randomness into the flow of messages.

Treating the various anonymity strategies as separate operators allows us to examine in isolation approaches that are typically viewed only as ways to improve the anonymity of other systems.

8.3 Methods of Providing Anonymity

We explore four strategies for providing anonymity representative of fundamental mechanism for resistance to traffic analysis:

- FLOOD: The introduction of dummy messages into a system's traffic.
- MIX: The storage and random reordering of batches of messages.
- HIDE: The obscuring of some portion of a system.
- DROP: The dropping of messages from the flow of traffic.

The most widely studied strategy is the MIX. Other fundamental approaches to providing anonymity are typically overlaid onto a MIX-based system in order to improve its anonymity or to overcome attacks against the system.

The FLOOD system provides a form of anonymity that cannot be matched by the other strategies. In most systems, observing an event informs the observer that a message has been sent. The FLOOD system, by introducing events that do not relate to "genuine" messages, prevents an attacker from making this deduction. This behaviour in the absence of genuine input messages allows the FLOOD node to meet the strongest form of the ordering independence property; such a system may be considered anonymous for any sequence of input.

The DROP system, in its basic form, is of little practical use due to its lack of reliable delivery. For the purpose of analysis, we therefore also consider a variant on the strategy that does not delete messages, but instead delivers them immediately to their destination rather than forwarding them to other intermediary nodes. This models the *Crowds* system of Reiter and Rubin (1998).

Having defined the basic forms of anonymity system, we analyse their effectiveness both individually and in combination.

8.4 Simulation

In order to quantify the effectiveness of the strategies described in our model, we construct simulations of small networks that implement each strategy. Our analysis is based on an information theoretic quantification of the level of reordering in the sequence of messages. We consider consecutive pairs of messages in the sequence of input and examine the distribution of their distance from each other in the output stream over multiple runs of a system.

8.4.1 Individual Strategies

The simulations demonstrate the behaviour of the varying methods of achieving anonymity as we vary their defining parameters. Of these, the FLOOD node is the most effective in increasing uncertainty. There is no upper limit on the amount of entropy in a FLOOD system; adding dummy messages allows the number of messages to endlessly increase and so increase the confusion between message locations.

From both the theoretical and simulation results, the FLOOD node appears to be the ideal system if we consider only its ability to introduce uncertainty into the ordering of the traffic flow. There are, however, high bandwidth requirements for the FLOOD strategy that present significant implementation difficulties in many cases.

In the case of the MIX, the entropy of the system increases until all messages are handled in a single batch. At this point, initially consecutive messages may appear at any point in the output sequence. Unlike other approaches, the MIX strategy allows for pairs of messages to reverse their position in the output stream rather than simply altering the distance between them.

A pure Drop system is shown to be ineffective both in its ability to introduce entropy into the output stream and in the number of messages that are delivered. The Crowds approach is a practical alternative that maintains reliable message delivery; unlike other methods, however, the Crowds system cannot function effectively as a single node and must be employed as part of a network. As we discuss later, the existence of multiple nodes introduces an extra level of entropy into the message sequence due to the interaction of the parallel processes.

The Hide system decreases the scope of an attacker's observations to increase that attacker's uncertainty as to the sequence of output messages. As a method for providing anonymity, the Hide system is difficult to implement. The approach relies on an attacker being unable to observe certain aspects of the system, and thus leaves the system open to attack by stronger observers. The Hide strategy is therefore more appropriately considered as the assumption of a limited attacker than as an explicit method in its own right.

8.4.2 Network Effects

Simulations show that even small networks introduce a notable level of uncertainty into the sequence of messages. This behaviour could be classed as a form of the Hide strategy as it relies on an observer who views only the input and output streams of traffic and not links between nodes. Despite this, the entropy increase caused by small numbers of nodes can be comparable to that of single nodes employing a specific anonymity strategy.

8.4.3 Combination of Strategies

Combining nodes that make use of different strategies is shown, in some cases, to improve the entropy of a system's output distribution above that of systems comprised entirely of a single node type. Each simulation combining multiple anonymity strategies shows, for certain combinations, higher levels of entropy than their homogeneous counterparts.

The use of multiple strategies in a system has the potential to introduce avenues of attack for an observer due to varying message paths causing different behaviours for messages. However, such an approach could also result in systems that meet more flexible requirements for message delivery whilst simultaneously increasing the confusion of an attacker. The implications of such an approach are a potential avenue for future study.

8.5 Future Work

This thesis presents a definition of anonymity and a model for classifying anonymity systems. This model is then partially explored through simulation. There are several areas of interest that would extend this work.

8.5.1 Formal Modelling

Our model of anonymity relies on the existence of randomness, and is inspired by the introduction of nondeterminism in abstract CSP processes. The presence of nondeterminism is an indicator of anonymity in the system. While this allows for the presence of anonymity in a system model to be tested via a simple determinism check, it does not provide a useful quantification of the anonymity properties of a system. As our approach focuses on simulation we have gained a quantitative analysis of the randomness in the system. Extending and formalising our underlying models, however, to result in a more provable analysis would be a useful development of our ideas.

A probabilistic extension to such a model could allow for a formal analysis of the systems that we propose. Assuming an appropriate model checker, such a model would allow systems to be checked for anonymity properties with given probability. This would lend itself naturally to verifying the *sufficient anonymity* requirements, incorporating cardinality and certainty, as described in Chapter 2.

8.5.2 Data Independence

We have largely ignored the problem of data independence, assuming that appropriate use of encryption prevents messages from being linked via their content as they pass through the network. If attackers function as message-passing nodes, however, the information revealed by a partial knowledge of the route can be used to compromise the anonymity of users.

Combining the elements of data independence and content independence would result in a more complete model for the verification or design of anonymity systems, but would require significant further development.

8.5.3 Attacker Models

The level of anonymity that a system can provide is strongly linked to the capabilities of the attacker. Our simulations assume that the attacker views the entire input and output sequences, except when we specifically reduce this capability for HIDE-based systems. This approach entirely ignores links between nodes in larger systems.

Extending the analysis to reflect an attacker's imperfect knowledge would provide a more useful analysis of the effectiveness of the various systems. Such a view would also allow us to consider attackers concerned with messages that are merely passing between nodes, and not only those delivered to receivers.

The simulation, being based on the model, also fails to consider corrupt nodes in which the linkage between input and output events is known for those nodes controlled by the attacker. As for the model, adding this consideration would allow a more complete evaluation of anonymity properties.

8.5.4 Improved Quantification

The quantification employed by the simulation experiments is based on the relative distance between pairs of messages that were consecutive in the input sequence. This quantification provides a computationally feasible approximation to the level of uncertainty concerning the distribution of outputs for a given input.

Our quantification is not effective at analysing the pure DROP strategy, in which messages may be dropped from the output. This is not a great concern, as the DROP strategy presents too many problems to be seriously considered as the basis of future work. The lack of a satisfactory analysis for this case, however, highlights a weakness of our method.

Quantifications of the effectiveness of anonymity systems, such as those presented by Diaz et al. (2002), Serjantov and Danezis (2002) and Wright et al. (2002) are typically based on a specific attack that applies to the system being examined. The attack selected by Serjantov and Danezis (2002) is examined in detail with respect to mixes, but is applicable to other types of system. Wright et al. (2002) base their work on an attack that applies to a variety of systems. Our quantification attempts to be entirely independent of the system under consideration, but consequently is less representative of specific attacks. Choosing an alternative focus for quantification, or exploring attacks focused against each strategy, may result in more effective analyses.

Recent work by Kesdogan et al. (2006) is also aimed at identifying fundamental limits of the strength of anonymity systems regardless of specific attacks, but focuses solely on mix-based systems. Extending this approach to the full range of strategies that we propose in this thesis could provide a useful alternative to the quantification that we employ.

A further consideration for the development of a quantifications is their ability to be applied in real-world systems. The current quantification is appropriate for simulations in which the entire state of a given system is known, but would be of less use as the knowledge of the system decreases. A quantification that requires only a limited knowledge of the system would be applicable beyond simulations in deployed systems.

8.5.5 Detailed Modelling

A direct extension to the work presented in the later chapters of this thesis would be to extend our simulations to larger networks. The experiments conducted in this work analyse relatively low numbers of messages passing over networks consisting of up to four nodes. Extending the simulations would allow for more detailed analyses of the behaviour of these systems.

The combination of multiple node types in a single system has produced interesting results in small networks. Larger scale models would allow for a greater number of combinations and their effects to be examined.

The simulation experiments, although at a lower level than the model, remain an abstraction of real systems and do not consider factors such as patterns of user traffic and bandwidth requirements. An important addition would be to incorporate such factors into our simulations and thus more closely approach a true implementation.

8.6 Summary

This thesis has presented a framework in which the properties of anonymity systems can be expressed, and the systems themselves modelled. This model has been used to classify anonymity systems according to the fundamental strategy that allows them to provide anonymity to users. Each of these strategies has been individually presented and related to real-world implementations. Importantly, we show that each strategy is individually capable of providing some level of anonymity.

We have developed a simulation that allows for each anonymity strategy to be implemented in a form suitable for statistical analysis. A quantification of the level of anonymity based on information theoretic entropy of the distribution of messages has been proposed and applied to the results of the simulator.

The results of our quantification allow us to present a comparison of the effectiveness of the varying fundamental anonymity systems. These systems have been explored individually and in small networks, as well as in systems comprising multiple anonymity strategies. We show that certain strategies are clearly preferable to others with respect to the uncertainty that they introduce into the distribution of messages, and that combining strategies in a single system can further increase the level of anonymity.

The work presented here demonstrates that there are fundamental features shared by all anonymity systems, and that the various strategies for achieving this may usefully be treated both individually and in combination. This view allows for a fuller understanding of existing approaches to providing anonymity, and provides insight into the design of future systems.

Part IV

Appendices

Appendix A

Results

A.1 Introduction

We present here full results of the simulation experiments, Each table shows the entropy (H) and standard deviation (σ) derived via simulation as the characteristic parameter of the node type changes.

A.2 Single Strategy Networks

A.2.1 Mix

Threshold	H	σ
1	0.000	0.000
2	1.899	0.001
3	2.586	0.003
4	3.004	0.001
5	3.298	0.002
6	3.540	0.003
7	3.759	0.002
8	3.932	0.004
9	4.089	0.001
10	4.200	0.002
11	4.323	0.001
12	4.439	0.002
13	4.553	0.004
14	4.619	0.003
15	4.732	0.005
16	4.838	0.003

A.2.2 Flood

Flood %	H	σ
90	4.690	0.005
80	3.601	0.007
70	2.940	0.004
60	2.428	0.005
50	2.007	0.002
40	1.619	0.004
30	1.264	0.009
20	0.906	0.006
10	0.519	0.002

A.2.3 Hide

Hidden Senders	H	σ
1	0.546	0.001
2	0.883	0.001
3	1.142	0.001
4	1.357	0.000
5	1.541	0.001
6	1.703	0.001
7	1.847	0.001
8	1.976	0.001

A.2.4 Drop

Drop %	H	σ
90	0.937	0.006
80	1.437	0.007
70	1.762	0.001
60	1.943	0.001
50	2.000	0.000
40	1.940	0.001
30	1.761	0.004
20	1.440	0.007
10	0.937	0.004

A.2.5 Four Nodes

Mix

Four Nodes – 1 Hop

Threshold	H	σ
1	4.831	0.207
2	5.130	0.040
3	5.167	0.067
4	5.192	0.102
5	5.301	0.076
6	5.566	0.123
7	5.619	0.063
8	5.563	0.118
9	5.853	0.105
10	6.074	0.168
11	6.228	0.113
12	6.254	0.040
13	6.344	0.103
14	6.477	0.043
15	6.456	0.161
16	6.612	0.037

Four Nodes – 2 Hop

Threshold	H	σ
1	5.790	0.274
2	5.741	0.207
3	5.446	0.172
4	5.446	0.187
5	5.486	0.038
6	5.397	0.157
7	5.490	0.115
8	5.582	0.048
9	5.642	0.127
10	5.826	0.036
11	5.891	0.048
12	6.046	0.045
13	6.118	0.053
14	6.272	0.011
15	6.326	0.124
16	6.523	0.144

Four Nodes – 3 Hop

Threshold	H	σ
1	5.874	0.245
2	5.802	0.229
3	5.658	0.198
4	5.677	0.139
5	5.553	0.163
6	5.583	0.091
7	5.557	0.034
8	5.689	0.029
9	5.669	0.023
10	5.800	0.064
11	5.944	0.084
12	5.844	0.100
13	6.138	0.084
14	6.245	0.099
15	6.448	0.084
16	6.550	0.049

Four Nodes – 4 Hop

Threshold	H	σ
1	5.788	0.309
2	5.827	0.340
3	5.610	0.070
4	5.527	0.116
5	5.423	0.045
6	5.576	0.036
7	5.619	0.111
8	5.741	0.048
9	5.749	0.009
10	5.891	0.084
11	5.954	0.051
12	6.024	0.060
13	6.254	0.165
14	6.320	0.142
15	6.401	0.067
16	6.338	0.167

A.2.6 Four Nodes – All Hops

Threshold	1		2		3		4	
	H	σ	H	σ	H	σ	H	σ
1	4.831	0.207	5.790	0.274	5.874	0.245	5.788	0.309
2	5.130	0.040	5.741	0.207	5.802	0.229	5.827	0.340
3	5.167	0.067	5.446	0.172	5.658	0.198	5.610	0.070
4	5.192	0.102	5.446	0.187	5.677	0.139	5.527	0.116
5	5.301	0.076	5.486	0.038	5.553	0.163	5.423	0.045
6	5.566	0.123	5.397	0.157	5.583	0.091	5.576	0.036
7	5.619	0.063	5.490	0.115	5.557	0.034	5.619	0.111
8	5.563	0.118	5.582	0.048	5.689	0.029	5.741	0.048
9	5.853	0.105	5.642	0.127	5.669	0.023	5.749	0.009
10	6.074	0.168	5.826	0.036	5.800	0.064	5.891	0.084
11	6.228	0.113	5.891	0.048	5.944	0.084	5.954	0.051
12	6.254	0.040	6.046	0.045	5.844	0.100	6.024	0.060
13	6.344	0.103	6.118	0.053	6.138	0.084	6.254	0.165
14	6.477	0.043	6.272	0.011	6.245	0.099	6.320	0.142
15	6.456	0.161	6.326	0.124	6.448	0.084	6.401	0.067
16	6.612	0.037	6.523	0.144	6.550	0.049	6.338	0.167

Flood

Four Nodes – 1 Hop

Flood %	H	σ
90	7.667	0.138
80	6.929	0.194
70	6.388	0.093
60	6.007	0.028
50	5.956	0.128
40	6.025	0.061
30	5.792	0.187
20	5.528	0.279
10	5.513	0.171

Four Nodes – 2 Hop

Flood %	H	σ
90	8.591	0.052
80	7.656	0.054
70	7.200	0.074
60	6.791	0.187
50	6.767	0.151
40	6.476	0.062
30	6.326	0.135
20	5.849	0.117
10	6.123	0.162

Four Nodes – 3 Hop

Flood %	H	σ
90	8.856	0.048
80	7.959	0.119
70	7.580	0.157
60	7.178	0.227
50	6.800	0.054
40	6.573	0.099
30	6.544	0.212
20	6.467	0.185
10	6.144	0.214

Four Nodes – 4 Hop

Flood %	H	σ
90	8.939	0.058
80	8.007	0.005
70	7.511	0.008
60	7.089	0.061
50	6.857	0.067
40	6.659	0.057
30	6.333	0.062
20	6.247	0.096
10	5.978	0.173

A.2.7 Four Nodes – All Hops

Flood %	1		2		3		4	
	H	σ	H	σ	H	σ	H	σ
90	7.667	0.138	8.591	0.052	8.856	0.048	8.939	0.058
80	6.929	0.194	7.656	0.054	7.959	0.119	8.007	0.005
70	6.388	0.093	7.200	0.074	7.580	0.157	7.511	0.008
60	6.007	0.028	6.791	0.187	7.178	0.227	7.089	0.061
50	5.956	0.128	6.767	0.151	6.800	0.054	6.857	0.067
40	6.025	0.061	6.476	0.062	6.573	0.099	6.659	0.057
30	5.792	0.187	6.326	0.135	6.544	0.212	6.333	0.062
20	5.528	0.279	5.849	0.117	6.467	0.185	6.247	0.096
10	5.513	0.171	6.123	0.162	6.144	0.214	5.978	0.173

Hide

Four Nodes – 1 Hop

Hidden Senders	H	σ
1	4.984	0.329
2	5.543	0.211
3	5.485	0.366
4	5.683	0.306
5	5.523	0.388
6	5.835	0.223
7	5.587	0.104
8	5.788	0.302

Four Nodes – 2 Hop

Hidden Senders	H	σ
1	5.665	0.112
2	6.084	0.116
3	5.961	0.350
4	5.962	0.043
5	6.044	0.126
6	6.102	0.230
7	6.632	0.128
8	6.045	0.175

Four Nodes – 3 Hop

Hidden Senders	H	σ
1	5.675	0.245
2	6.160	0.305
3	6.191	0.064
4	6.311	0.089
5	6.495	0.196
6	6.318	0.278
7	6.652	0.083
8	6.830	0.114

Four Nodes – 4 Hop

Hidden Senders	H	σ
1	5.713	0.059
2	5.885	0.224
3	6.125	0.120
4	6.237	0.452
5	6.564	0.141
6	6.629	0.232
7	6.487	0.095
8	6.428	0.362

A.2.8 Four Nodes – All Hops

Senders	1		2		3		4	
	H	σ	H	σ	H	σ	H	σ
1	4.984	0.329	5.665	0.112	5.675	0.245	5.713	0.059
2	5.543	0.211	6.084	0.116	6.160	0.305	5.885	0.224
3	5.485	0.366	5.961	0.350	6.191	0.064	6.125	0.120
4	5.683	0.306	5.962	0.043	6.311	0.089	6.237	0.452
5	5.523	0.388	6.044	0.126	6.495	0.196	6.564	0.141
6	5.835	0.223	6.102	0.230	6.318	0.278	6.629	0.232
7	5.587	0.104	6.632	0.128	6.652	0.083	6.487	0.095
8	5.788	0.302	6.045	0.175	6.830	0.114	6.428	0.362

Drop

Four Nodes – 1 Hop

Drop %	H	σ
90	0.963	0.011
80	1.576	0.005
70	2.091	0.024
60	2.574	0.011
50	3.087	0.041
40	3.582	0.039
30	3.857	0.110
20	4.532	0.053
10	4.881	0.030

Four Nodes – 2 Hop

Drop %	H	σ
90	0.156	0.004
80	0.489	0.006
70	0.896	0.008
60	1.354	0.016
50	1.877	0.017
40	2.425	0.016
30	3.151	0.070
20	3.952	0.134
10	5.055	0.128

Four Nodes – 3 Hop

Drop %	H	σ
90	0.023	0.002
80	0.132	0.004
70	0.368	0.006
60	0.695	0.006
50	1.141	0.012
40	1.691	0.005
30	2.406	0.023
20	3.317	0.042
10	4.664	0.034

Four Nodes – 4 Hop

Drop %	H	σ
90	0.003	0.001
80	0.031	0.003
70	0.132	0.003
60	0.343	0.004
50	0.678	0.008
40	1.171	0.009
30	1.837	0.006
20	2.784	0.018
10	4.107	0.085

A.2.9 Four Nodes – All Hops

Drop %	1		2		3		4	
	H	σ	H	σ	H	σ	H	σ
90	0.963	0.011	0.156	0.004	0.023	0.002	0.003	0.001
80	1.576	0.005	0.489	0.006	0.132	0.004	0.031	0.003
70	2.091	0.024	0.896	0.008	0.368	0.006	0.132	0.003
60	2.574	0.011	1.354	0.016	0.695	0.006	0.343	0.004
50	3.087	0.041	1.877	0.017	1.141	0.012	0.678	0.008
40	3.582	0.039	2.425	0.016	1.691	0.005	1.171	0.009
30	3.857	0.110	3.151	0.070	2.406	0.023	1.837	0.006
20	4.532	0.053	3.952	0.134	3.317	0.042	2.784	0.018
10	4.881	0.030	5.055	0.128	4.664	0.034	4.107	0.085

A.2.10 Crowds-style

Delivery %	H	σ
90	5.313	0.140
80	5.683	0.110
70	5.972	0.042
60	6.222	0.029
50	6.364	0.004
40	6.513	0.019
30	6.593	0.008
20	6.660	0.004
10	6.698	0.001

Deterministic Buffers

Number of Buffers	Hops	H	σ
1	1	0.000	0.000
2	1	3.643	0.394
2	2	3.587	0.381
3	1	4.709	0.292
3	2	5.217	0.296
3	3	5.083	0.301
4	1	5.042	0.258
4	2	5.563	0.282
4	3	5.662	0.267
4	4	5.563	0.211

A.3 Mixed Strategy Networks

A.3.1 Mix with Flood

Node Parameters:
Mix node thresholds: 16
Flood node flooding percentage: 50

Mix Nodes	Flood Nodes	H	σ
4	0	6.338	0.167
3	1	6.757	0.052
2	2	6.837	0.041
1	3	7.053	0.064
0	4	6.857	0.067

A.3.2 Crowds with Flood

Node Parameters:
Crowds node delivery percentage: 50
Flood node flood percentage: 50

Crowds Nodes	Flood Nodes	H	σ
4	0	6.364	0.004
3	1	6.699	0.015
2	2	7.148	0.020
1	3	7.483	0.039
0	4	6.857	0.067

A.3.3 Crowds with Mix

Node Parameters:

Crowds node delivery percentage: 50

Mix node threshold: 16

Crowds Nodes	Mix Nodes	H	σ
4	0	6.364	0.004
3	1	6.540	0.004
2	2	6.628	0.030
1	3	6.639	0.025
0	4	6.338	0.167

A.4 Statistical Significance Tests

Below are the results of the statistical significance tests for each experiment. As the parameter varied in each experiment, the Kolmogorov-Smirnov test was applied in order to judge whether the change in parameter introduced a significant difference between distributions.

The resulting p-value for each experimental variation is reported below. The level of statistical significance chosen for these experiments was 99%, allowing a 1% chance that a successful rejection of the null hypothesis was incorrect.

The appropriate p-value for the tests below, at the 99% confidence level and given to 3 decimal places, is 0.009. In almost every case, the experiments rejected the null hypothesis. Those cases where this is not the case are marked in bold.

A.4.1 Single Mix

Parameter Change	p-value
$16 \to 15$	0.029
$15 \to 14$	0.024
$14 \to 13$	0.012
$13 \to 12$	0.031
$12 \to 11$	0.023
$11 \to 10$	0.027
$10 \to 9$	0.021
$9 \to 8$	0.029
$8 \to 7$	0.036
$7 \to 6$	0.043
$6 \to 5$	0.044
$5 \to 4$	0.055
$4 \to 3$	0.079
$3 \to 2$	0.111
$2 \to 1$	0.368

A.4.2 Four Mixes, 1 Hop

Parameter Change	p-value
$16 \to 15$	0.039
$15 \to 14$	0.022
$14 \to 13$	0.026
$13 \to 12$	0.047
$12 \to 11$	0.021
$11 \to 10$	0.036
$10 \to 9$	0.038
$9 \to 8$	0.052
$8 \to 7$	0.019
$7 \to 6$	0.020
$6 \to 5$	0.052
$5 \to 4$	0.032
$4 \to 3$	0.026
$3 \to 2$	0.035
$2 \to 1$	0.101

A.4.3 Four Mixes, 2 Hop

Parameter Change	p-value
$16 \rightarrow 15$	0.046
$15 \rightarrow 14$	0.018
$14 \rightarrow 13$	0.032
$13 \rightarrow 12$	0.013
$12 \rightarrow 11$	0.030
$11 \rightarrow 10$	0.020
$10 \rightarrow 9$	0.034
$9 \rightarrow 8$	0.015
$8 \rightarrow 7$	0.019
$7 \rightarrow 6$	0.018
$6 \rightarrow 5$	0.019
$5 \rightarrow 4$	0.024
$4 \rightarrow 3$	0.016
$3 \rightarrow 2$	0.059
$2 \rightarrow 1$	0.030

A.4.4 Four Mixes, 3 Hop

Parameter Change	p-value
$16 \rightarrow 15$	0.026
$15 \rightarrow 14$	0.044
$14 \rightarrow 13$	0.030
$13 \rightarrow 12$	0.051
$12 \rightarrow 11$	0.026
$11 \rightarrow 10$	0.027
$10 \rightarrow 9$	0.022
$9 \rightarrow 8$	0.010
$8 \rightarrow 7$	0.031
$7 \rightarrow 6$	**0.007**
$6 \rightarrow 5$	0.016
$5 \rightarrow 4$	0.018
$4 \rightarrow 3$	0.025
$3 \rightarrow 2$	0.023
$2 \rightarrow 1$	0.049

A.4.5 Four Mixes, 4 Hop

Parameter Change	p-value
$16 \rightarrow 15$	0.020
$15 \rightarrow 14$	0.021
$14 \rightarrow 13$	0.011
$13 \rightarrow 12$	0.040
$12 \rightarrow 11$	0.018
$11 \rightarrow 10$	0.015
$10 \rightarrow 9$	0.022
$9 \rightarrow 8$	0.012
$8 \rightarrow 7$	0.016
$7 \rightarrow 6$	0.016
$6 \rightarrow 5$	0.031
$5 \rightarrow 4$	0.015
$4 \rightarrow 3$	0.017
$3 \rightarrow 2$	0.043
$2 \rightarrow 1$	0.013

A.4.6 Single Flood

Parameter Change	p-value
$90 \rightarrow 80$	0.271
$80 \rightarrow 70$	0.168
$70 \rightarrow 60$	0.130
$60 \rightarrow 50$	0.109
$50 \rightarrow 40$	0.100
$40 \rightarrow 30$	0.101
$30 \rightarrow 20$	0.099
$20 \rightarrow 10$	0.102

A.4.7 Four Flood, 1 Hop

Parameter Change	p-value
$90 \rightarrow 80$	0.088
$80 \rightarrow 70$	0.068
$70 \rightarrow 60$	0.056
$60 \rightarrow 50$	0.026
$50 \rightarrow 40$	0.020
$40 \rightarrow 30$	0.032
$30 \rightarrow 20$	0.059
$20 \rightarrow 10$	0.025

A.4.8 Four Flood, 2 Hop

Parameter Change	p-value
$90 \rightarrow 80$	0.134
$80 \rightarrow 70$	0.074
$70 \rightarrow 60$	0.070
$60 \rightarrow 50$	0.010
$50 \rightarrow 40$	0.049
$40 \rightarrow 30$	0.033
$30 \rightarrow 20$	0.068
$20 \rightarrow 10$	0.050

A.4.9 Four Flood, 3 Hop

Parameter Change	p-value
$90 \rightarrow 80$	0.136
$80 \rightarrow 70$	0.063
$70 \rightarrow 60$	0.057
$60 \rightarrow 50$	0.071
$50 \rightarrow 40$	0.040
$40 \rightarrow 30$	0.012
$30 \rightarrow 20$	0.040
$20 \rightarrow 10$	0.046

A.4.10 Four Flood, 4 Hop

Parameter Change	p-value
$90 \rightarrow 80$	0.146
$80 \rightarrow 70$	0.075
$70 \rightarrow 60$	0.071
$60 \rightarrow 50$	0.037
$50 \rightarrow 40$	0.033
$40 \rightarrow 30$	0.054
$30 \rightarrow 20$	0.011
$20 \rightarrow 10$	0.050

A.4.11 Single Hide

Parameter Change	p-value
$1 \rightarrow 2$	0.096
$2 \rightarrow 3$	0.077
$3 \rightarrow 4$	0.064
$4 \rightarrow 5$	0.052
$5 \rightarrow 6$	0.047
$6 \rightarrow 7$	0.038
$7 \rightarrow 8$	0.035

A.4.12 Four Hide, 1 Hop

Parameter Change	p-value
$1 \rightarrow 2$	0.084
$2 \rightarrow 3$	0.022
$3 \rightarrow 4$	0.041
$4 \rightarrow 5$	0.031
$5 \rightarrow 6$	0.058
$6 \rightarrow 7$	0.055
$7 \rightarrow 8$	0.057

A.4.13 Four Hide, 2 Hop

Parameter Change	p-value
$1 \rightarrow 2$	0.072
$2 \rightarrow 3$	0.040
$3 \rightarrow 4$	0.036
$4 \rightarrow 5$	0.023
$5 \rightarrow 6$	0.029
$6 \rightarrow 7$	0.102
$7 \rightarrow 8$	0.125

A.4.14 Four Hide, 3 Hop

Parameter Change	p-value
$1 \rightarrow 2$	0.092
$2 \rightarrow 3$	0.020
$3 \rightarrow 4$	0.020
$4 \rightarrow 5$	0.058
$5 \rightarrow 6$	0.062
$6 \rightarrow 7$	0.052
$7 \rightarrow 8$	0.044

A.4.15 Four Hide, 4 Hop

Parameter Change	p-value
$1 \rightarrow 2$	0.031
$2 \rightarrow 3$	0.052
$3 \rightarrow 4$	0.034
$4 \rightarrow 5$	0.050
$5 \rightarrow 6$	0.034
$6 \rightarrow 7$	0.046
$7 \rightarrow 8$	0.024

A.4.16 Four Crowds

Parameter Change	p-value
$90 \rightarrow 80$	0.055
$80 \rightarrow 70$	0.045
$70 \rightarrow 60$	0.044
$60 \rightarrow 50$	0.030
$50 \rightarrow 40$	0.032
$40 \rightarrow 30$	0.018
$30 \rightarrow 20$	0.017
$20 \rightarrow 10$	0.011

A.4.17 Mix / Flood

Parameter Change	p-value
4-0 $\rightarrow$ 3-1	0.070
3-1 $\rightarrow$ 2-2	0.017
2-2 $\rightarrow$ 1-3	0.037
1-3 $\rightarrow$ 0-4	0.031

A.4.18 Crowds / Flood

Parameter Change	p-value
4-0 $\rightarrow$ 3-1	0.064
3-1 $\rightarrow$ 2-2	0.070
2-2 $\rightarrow$ 1-3	0.059
1-3 $\rightarrow$ 0-4	0.121

A.4.19 Crowds / Mix

Parameter Change	p-value
4-0 $\rightarrow$ 3-1	0.038
3-1 $\rightarrow$ 2-2	0.029
2-2 $\rightarrow$ 1-3	0.019
1-3 $\rightarrow$ 0-4	0.071

Bibliography

Ross Anderson. The eternity service. In *Proceedings of Pragocrypt '96*, 1996.

K. J. Arrow. A difficulty in the concept of social welfare. *Journal of Political Economy*, 58:328–346, 1950.

Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257. LNCS 2137, Springer, April 2001.

Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In Dingledine and Syverson (2002b).

Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In Federrath (2000).

Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

Louis Brandeis and Samuel Warren. The right to privacy. In *Harvard Law Review*, volume 4, December 1890.

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

David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Federrath (2000), pages 46–66.

Richard Clayton. Improving onion notation. In Dingledine (2003).

Thomas McIntyre Cooley. *A Treatise on the Law of Torts*. Callaghan, 2 edition, 1888.

Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0471062596. URL `http://portal.acm.org/citation.cfm?id=129837`.

Cypherpunks. Cypherpunks mailing list. `http://www.csua.berkeley.edu/cypherpunks/Home.html`.

Wei Dai. Pipenet 1.1. `http://www.eskimo.com/~weidai/pipenet.txt`, August 1996.

George Danezis. Mix-networks with restricted routes. In Dingledine (2003).

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

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

Claudia Diaz and Bart Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In *Proceedings of the 6th Information Hiding Workshop, IH 2004*. LNCS, Springer, May 2004a.

Claudia Diaz and Bart Preneel. Taxonomy of mixes and dummy traffic. Presented at the 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems, August 2004b.

Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Dingledine and Syverson (2002b).

Roger Dingledine, editor. *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, March 2003. LNCS 2760, Springer.

Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, *Proceedings of Financial Cryptography (FC '02)*. LNCS 2357, Springer, March 2002a.

Roger Dingledine and Paul Syverson, editors. *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, April 2002b. LNCS 2482, Springer.

Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. In Ira S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 126–141. Springer-Verlag, LNCS 2137, April 2001.

Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004a.

Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. `http://freehaven.net/doc/sync-batching/sync-batching.pdf`, 2004b.

D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1056650`.

European Union. European union directive on data protection, 1998.

H. Federrath, editor. *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, July 2000. LNCS 2009, Springer.

GNU Scientific Library. The GNU Project, 2007. URL `http://www.gnu.org/software/gsl/`.

Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, December 2000.

Ian Goldberg and David Wagner. TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*, 3 (4), August 1998.

David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In R. Anderson, editor, *Proceedings of the 1st International Workshop on Information Hiding*, pages 137–150. LNCS 1174, Springer, 1996.

HMSO. Data protection act. HMSO, 1984.

HMSO. Data protection act. HMSO, 1998.
`http://www.hmso.gov.uk/acts/acts1998/19980029.htm`.

C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

ICCPR. International covenant on civil and political rights. `http://www.hrweb.org/legal/`, January 1997.

JAP. Jap – anonymity and privacy. `http://anon.inf.tu-dresden.de/index_en.html`, 2004.

JAP Ruling, 2004. Ruling on storage of JAP data. `http://www.datenschutzzentrum.de/material/themen/presse/anonip3_e.htm`, 2004.

Dogan Kesdogan and Lexi Pimenidis. The Hitting Set Attack on Anonymity Protocols. In *Proceedings of Information Hiding, 7th International Workshop*. Springer Verlag, 2004. URL `citeseer.ist.psu.edu/677044.html`.

Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. LNCS 1525, Springer, 1998.

Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.

Dogan Kesdogan, Dakshi Agrawal, Vinh Pham, and Dieter Rautenbach. Fundamental limits on the anonymity provided by the mix technique. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 86–99, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2574-1. doi: http://dx.doi.org/10.1109/SP.2006.17.

H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(1):50–60, 1947.

Mixmaster. Mixmaster project development page. `http://mixmaster.sourceforge.net`.

Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. `http://www.abditum.com/mixmaster-spec.txt`, July 2003.

Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. `http://freehaven.net/anonbib/papers/Anon_Terminology_v0.14.pdf`, July 2000.

Andreas Pfitzmann and Michael Waidner. Networks without user observability – design options. In *Proceedings of EUROCRYPT 1985*. LNCS 219, Springer, 1985.

Richard Posner. The economics of privacy. In *American Economic Review Papers and Proceedings*, 1981.

Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In Federrath (2000), pages 10–29.

Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

Marc Rennhard and Bernhard Plattner. Practical Anonymity for the Masses with Mix-Networks. In *Proceedings of the IEEE 8th Intl. Workshop on Enterprise Security (WET ICE 2003)*, Linz, Austria, June 2003.

P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, 1998. URL `citeseer.ist.psu.edu/samarati98protecting.html`.

Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Dingledine and Syverson (2002b).

Andrei Serjantov and Richard E. Newman. On the anonymity of timed pool mixes. In *Proceedings of the Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems*, pages 427–434, Athens, Greece, May 2003. Kluwer.

Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. LNCS 2578, Springer, October 2002.

Claude E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27(3):379–423, July 1948. Continued 27(4):623-656, October 1948.

Vitaly Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, forthcoming.

Sidney Siegel. *Nonparametric Statistic for the Behavioural Sciences*. McGraw-Hill, 1956.

Frank Stajano and Ross J. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In *Information Hiding*, pages 434–447, 1999.

Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002. ISSN 0218-4885. doi: http://dx.doi.org/10.1142/S0218488502001648.

Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In Federrath (2000), pages 96–114.

Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods (1)*, pages 814–833, 1999.

Brad Templeton. A watched populace never boils. `http://www.templetons.com/brad/watched.html`, 2001.

David Thomas, Chad Fowler, and Andrew Hunt. *Programming Ruby. The Pragmatic Programmer's Guide.* Pragmatic Programmers, 2004.

United Nations. Universal Declaration of Human Rights, 1948.

United States Department of Commerce. United States Department of Commerce Safe Harbor Program, 2004.

Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceablility with computationally secure serviceability. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89, Belgium*, page 690. LNCS 434, Springer, 1990.

Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.