Improving Security and Efficiency of Mix-Based Anonymous Communication Systems

Shaahin Madani

A thesis submitted in fulfilment of the degree of
Doctor of Philosophy (Computer Science)

School of Computer Science and Information Technology
College of Science, Engineering and Health
RMIT University
Melbourne, Australia

April 2015

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis/project is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Shaahin Madani
April 2015

Acknowledgements

Nothing truly valuable can be achieved except by the unselfish cooperation of many individuals.
—Albert Einstein

Family is everything. My parents, Shirin and Taghi, who have for ever supported me in my ventures. My sister, Shiva, who is also my dearest friend, companion and ally. My brother-in-law, Wael, who is much more than a true brother I could have ever dreamed of. They have always been a constant source of endless and unreserved love and support, and have stood by me in every step of my journey. Without them, I could have not achieved what I have today. I am deeply proud of them and eternally grateful. They are my everything.

I would like to offer my substantial gratitude to my two supervisors, Associate Professor Ibrahim Khalil and Professor Timos Sellis. I am thankful for their patience, steadfast integrity, insightful feedback and dedication to both my personal and academic development. Ibrahim and Timos have always been available and willing to guide and help. I am very lucky to have been supervised by two deeply knowledgeable and extensively experienced and yet truly humble and open-minded human beings.

I wish to express my sincere love and appreciation to my friends and fellow PhD Candidates who made this journey especially pleasant and memorable: Mohammad Nabi Omidvar, Golnar Khomami, Arash Joorabchi, Adrian John Meade, Reza Soltanpoor, Borhan Kazimipour, Azadeh Ghari Neiat, Dhiah Al-Shammary, Poya Parvaneh, Jian Zhong, Ayman Ibaida, Alsharif Abuadbba and Conor Cameron. I dearly cherish the great time we have spent together, the interesting and intellectually stimulating conversations we have had, and the knowledge and experience we have helped each other gain.

This work was supported by an Australian Postgraduate Award (APA) scholarship. I am grateful to the Australian Government for providing this generous support.

At the end, one must acknowledge that it was all about the *journey* and not the *destination.* I did have a joyful journey, indeed.

Credits

Parts of the work presented in this thesis have been published in the following forums:

Peer Reviewed Papers

- S Madani and I Khalil. Multi-Binomial Mix: A Proposal for Secure and Efficient Anonymous Communication. *Computer Networks: The International Journal of Computer and Telecommunications Networking (2015).* DOI 10.1016/j.comnet.2015.10.007.
- S Madani and I Khalil. Garbled Routing (GR): A Generic Framework towards Unification of Anonymous Communication Systems. *Journal of Network and Computer Applications 44 (2014): 183-195.* DOI 10.1016/j.jnca.2014.05.005.
- S Madani and I Khalil. Garbled Routing (GR): Towards Unification of Anonymous Networks. *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).* IEEE, 2013.
- S Madani and I Khalil. Prevention of Denial-of-Service Attacks in Garbled Routing (GR) Network. *Proceedings of the International Conference on Communication, Networks and Satellite (COMNETSAT).* IEEE, 2013.

The thesis was typeset using the $\LaTeX\,2_\varepsilon$ document preparation system.

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.

—Article 12, The Universal Declaration of Human Rights

Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers.

—Article 19, The Universal Declaration of Human Rights

Abstract

The communication layer leaks important private information even in the presence of encryption, which makes anonymous communication a fundamental element of systems that protect the privacy of users. Traffic mixers have long been used to achieve communication anonymity, but the security challenges and the resulted inefficiencies hinder the path to a wide adoption of these systems. In this thesis, we take a step towards improving the security of traffic mixers and building a platform for efficient anonymous communication.

We begin by revisiting Binomial Mix, which is one of the most effective designs for traffic mixing proposed to date, and the one that introduced randomness to the behaviour of traffic mixers. When thoroughly examined in different traffic conditions, Binomial Mix proved to be significantly more resilient against attacks than previously believed.

We then build on the design of Binomial Mix and propose two new designs for traffic mixers. The first design, Multi-Binomial Shared-Pool Mix (MBSP Mix), employs multiple sources of randomness which results in a behaviour less predictable by the attacker and thus provides a higher degree of anonymity. The second design, Multi-Binomial Independent-Pool Mix (MBIP Mix), enables a single traffic mixer to anonymise multiple communication channels with potentially differing latencies. This additional property significantly improves the security and efficiency of the mix.

Moving beyond the design of traffic mixers in isolation, we propose the architecture and details of a generic framework for anonymous communication. The proposed framework consists of various parts designed to enable the integration of various Anonymous Communication Systems as plug-in components into a shared and unified system. In addition to achieving a larger user-base and enjoying its associated security benefits, this approach enables the reusability of components across multiple communication systems.

Finally, we also present techniques to make the circuit establishment facility of the framework resistant towards Denial-of-Service attacks. We believe that our work is one step towards building a fully developed generic framework for anonymous communication and our results can inspire and be used for the design of a robust generic framework.

Contents

List of Figures

List of Tables

List of Acronyms

ACS	Anonymous Communication System
CDF	Cumulative Distribution Function
DDoS	Distributed Denial-of-Service
DoS	Denial-of-Service
DRPT	Dynamic Route Processing Table
GRF	Garbled Routing Framework
GRP	Garbled Routing Proxy
GRR	Garbled Routing Router
GRS	Garbled Routing Server
HDD	Hard Disk Drive
IPP	Interrupted Poisson Process
MBIP Mix	Multi-Binomial Independent-Pool Mix
MBSP Mix	Multi-Binomial Shared-Pool Mix
MP	Message Processor
MPT	Message Processing Table
PET	Privacy Enhancing Technology
RAM	Random Access Memory
RPR	Route Processing Rule
TDPM	Timed Dynamic Pool Mix
TM	Timed Mix
TPM	Timed Pool Mix
URL	Unified Resource Locator

Chapter 1

Introduction

The right to be let alone is indeed the beginning of all freedom.
—Justice William O. Douglas

1.1 Motivation

The Internet has now infiltrated more or less every aspect of our lives. The human race is so heavily dependent on this technology that our methods predating it now seem excruciating and belong to the Stone Age. In the modern world, the Internet has made our daily communications so inexpensive and convenient that we no longer consider distance to be an obstacle in engaging with our peers. Business and retail, in the presence of the Internet, have been transformed to become significantly more efficient and globalised. Similarly, technology is changing many other aspects of our everyday lives. For example, the emergence and widespread use of electronic currencies (e.g. Bitcoin [1]) is revolutionising financial transactions, and technology companies have been steadily changing the way we use services (e.g. Uber [2]). Without the Internet, these significant technological improvements would have been inconceivable.

However, the Internet was not designed to protect the privacy of its users. At the time of its inception, it was impossible to imagine the impact of this technology and the omnipresence it has achieved to date. The lack of privacy and its widespread use have made the Internet a perfect platform to, unfortunately, invade the privacy of billions of users globally.

The metadata and unencrypted content of Internet-based applications are exposed to parties who happen to have access to the physical links of the network. These include a wide range of different parties, such as Internet Service Providers (ISPs), government agencies and infrastructure providers. Such entities may have various intentions and interests with which an Internet user may or may not be in sympathy. For example, while a user may support an act of law enforcement, it is highly likely that they will rebuff many other intentions such as the act of spying, Internet censorship, or profiling for targeted advertisements.

Access to and possession of important private information, such as the content and metadata of Internet communications, is a great power which must come with great responsibility. Nonetheless, conflicting personal and national interests often lead to the use of personal information in ways that the users, should they ever be informed about the act, would never have consented to.

It is crucial to note that *encryption* and *steganography* can be used to protect *only the content of the communication but not its metadata* such as Internet Protocol (IP) addresses and information about the times and volumes of the communications. The Internet metadata reveals a great deal of private information about an Internet user.

Anonymous Communication Systems (ACSs) are systems that enable their users to communicate anonymously, that is, in ways that not only the content but also the metadata of their communication is protected. ACSs are an important part of Privacy Enhancing Technologies (PETs) and can achieve different degrees of anonymity according to what is necessary in the circumstances. These systems are used for a diverse range of applications such as election schemes, Voice over IP (VoIP), file sharing, publishing and accessing online material, and sending instant email messages. The reasoning behind the use of ACSs is manifold and includes, e.g., protection of privacy and personal preferences, circumventing censorship and surveillance, whistle-blowing and freedom of speech. In Section 2.5.6 we present further details of the numerous applications of ACSs and enumerate some of the existing systems built specifically for these applications.

1.2 Background

In this section we briefly review the concepts and definitions which assist in understanding the immediately following sections, namely, the problem statement and research questions considered by this thesis. The topics discussed hereunder will be revisited in Chapter 2, where we provide more details and also discuss many other related issues.

1.2.1 Anonymity

We adopt the definition of *anonymity* proposed by Pfitzmann and Hansen which is widely used in the anonymity literature. That is, *anonymity* of a subject means that the subject is "not identifiable within a set of subjects, the *anonymity set*" [3, 4]. In other words, from the perspective of the adversary, he "cannot *sufficiently* identify the subject" within the *anonymity set*, which implies both that there is a possibility to quantify *anonymity*, and that certain applications may require a definition of a threshold where the anonymity begins [4].

The *anonymity set* constitutes the set of subjects with potentially the same attributes who might cause an action. The *anonymity set* can differ in accordance with the knowledge of the adversary, and is therefore relative with respect to the adversary [4]. It can also vary over time [4].

In a communication system, the sender of a message can enjoy *sender anonymity* only within a set of potential senders, namely the *sender anonymity set*. Likewise, the receiver of a message can enjoy *receiver anonymity* only within the *receiver anonymity set*. These two sets may be disjoint, be the same or may overlap [3, 4].

Anonymity can be considered both for an individual subject, referred to as *individual anonymity*, and for all the users of an ACS, referred to as *global anonymity*. Assuming other conditions are equal, the *global anonymity* of an ACS improves in two circumstances. Firstly, it improves when the number of members in the *anonymity set* grows and, secondly, when there is a more even distribution of senders (in the *sender anonymity set*) and receivers (in the *receiver anonymity set*) in the system [3–7].

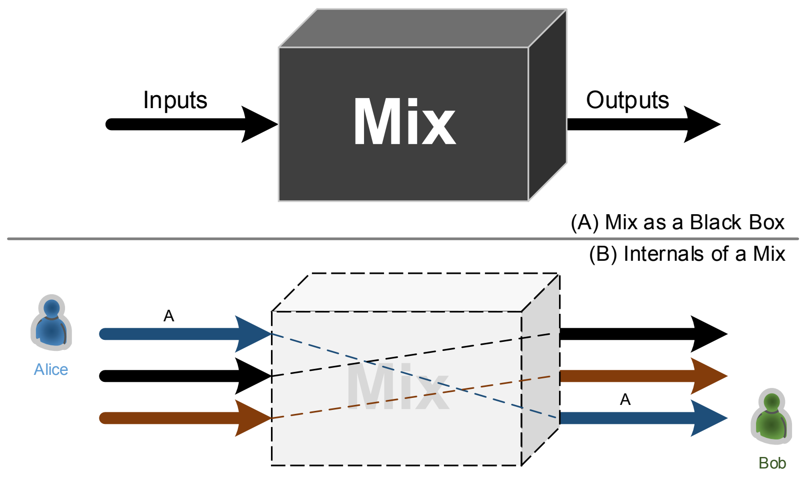


Figure 1.1: A *mix* is the first type of ACS which can be thought to be working as a black box (part (A)) in order to hide the link between the incoming and outgoing messages (part (B)).

Evidently, the number of participants in either of the *anonymity sets* is a function of the number of users of the ACSs in question. Therefore, assuming the anonymisation procedure maximises anonymity by creating the largest *anonymity sets* possible, a higher degree of anonymity achieved by an ACS is a function of a higher number of users using that system. In other words, the more users, the more likely it is for the system to offer a higher degree of *global anonymity*.

1.2.2 The Original *Mix* Design (Chaumian Mix)

Research activity to build ACSs commenced with the seminal work of Chaum in 1981 [8] whereby he proposed a special network router known as a *mix* which could provide an anonymous email service. A mix is a black box, similar to what is shown in Figure 1.1(A), which is designed to hide the link between the messages it receives and those it relays and thereby prevents an adversary from learning that link. Figure 1.1(B) shows the internals of a mix used by Alice in order to send an anonymous message to Bob. By mixing the communication of two other channels, the mix relays the message in a way unknown to and unexpected by the adversary.

The messages that a mix receives are encrypted and usually contain the address of the next mix that the messages should be sent to. After receiving a message, the mix performs a number of actions on the message such as, e.g., encryption, decryption, appending dummy bits (i.e. *padding*), randomising, batching and delaying. These actions are mainly intended to eliminate from the channel the information that an adversary could otherwise use to discover the link between the incoming and outgoing messages (e.g. timing and bitwise similarities). Eventually, the mix relays the message to its final destination or to another mix.

1.2.3 Mix Systems and Mixnets

Since the proposal of the original mix, research in the field of anonymous communication has been very active and many designs and improvements have been proposed. Some of these designs are based on the original Chaumian mix, which are generally referred to as *mix systems* [9, 10].

Mixes are used in anonymising ISDN telephone systems [11], real-time communication [12], web surfing [13, 14], email exchange [10, 15–19], and generic high- and low-latency routing [20–25]. Further details about the existing mix designs will be presented in Section 2.6.

Mixes are commonly used together in groups to achieve a higher degree of anonymity and/or to build a more resilient system. A network of connected mixes is commonly referred to as a *mixnet*. Over the years many mixnets have been designed and some of them now have a large user-base, such as Tor [26] and I2P [27, 28] networks. Further details about the existing mixnets will be presented in Section 2.7.

1.3 Problem Statement

As mentioned in Section 1.2.2, a mix should carry out a number of tasks on the messages it receives such as decrypting and encrypting messages, extracting routing information from the messages, unifying the size of messages, delaying them for certain periods, and executing a selection algorithm to choose a subset of messages to be relayed. Note that not every mix system is required to carry out all of these tasks, and that the strategy and actual implementation may vary in each system.

The above-stated tasks come with an inherent time cost. In other words, more sophisticated or time-consuming tasks translate into a higher delay imposed by the mix to the anonymised communication channel. It must be highlighted that some of these tasks (e.g. the selection algorithm) may have an intentional delay as part of their process. The resultant delays negatively affect the usability of the ACSs and may even render them unfit for the applications with low-latency requirements. There has always been a trade-off between the degree of security that a mix system can achieve and the corresponding delay imposed on the communication channel. Therefore, *building mix systems that can achieve a good balance of security and efficiency is important and challenging.*

The delay resulting from a strong anonymisation process can make a highly secure mix unfit for the communications with low-latency requirements. Therefore, there is a divide between the design of mixes and mixnets that offer anonymity to the low-latency communications and those that serve the high-latency ones. We discussed in Section 1.2.1 that a higher number of users in a mix system leads to a higher degree of *global anonymity* achieved by that system. An ACS that can anonymise both high- and low-latency traffic can attract a larger number of users and, therefore, can achieve a higher degree of *global anonymity*. Therefore, *it is important to create ACSs that can attract a large number of users.*

Furthermore, the development of multiple mixnets in isolation not only divides the user-base of all these systems, but also does the same to the efforts of the respective communities in research and development. Implementation and testing of these systems are conducted in isolation and reusability of code, simulation, test and deployment does not exist. Therefore, there is room for a unified approach to anonymisation by mixes that allows multiple systems to operate in a shared platform in order to benefit from a collective user-base and a unified software architecture.

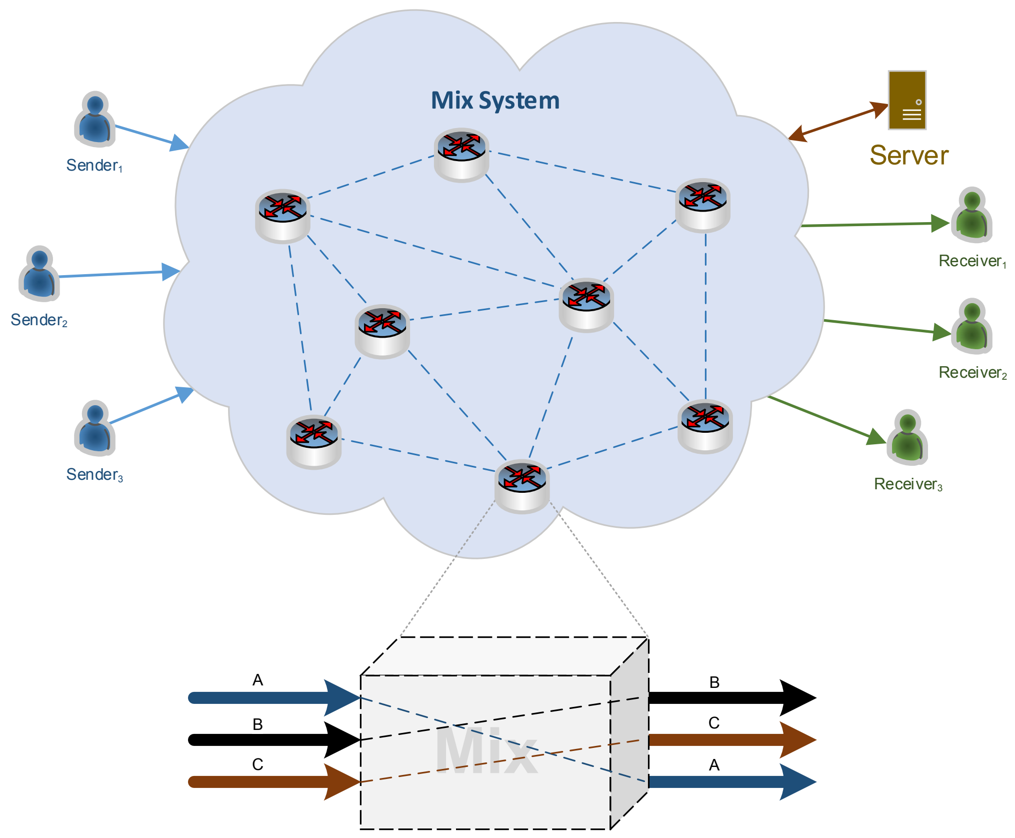


Figure 1.2: Schematic of a typical mix system with a number of senders, receivers, and remote servers. The routing scheme is arbitrary, and each router within the network is a mix.

1.4 Research Questions

As a step towards addressing the issues discussed in the previous section, this thesis aims to answer four related research questions which attend to various parts in the design of mixnets. Figure 1.2 shows an abstract illustration of a typical mix system with the typical components and users. The system, through the collaboration of a number of routers (i.e. mixes), anonymises the communications among the senders and receivers. The symbol of *server* shown in the diagram exemplifies an entity that both receives and sends data through the ACS. We start by focusing on the internal design of the mix. The first two research questions concentrate on this aspect of mixnets as shown in Figure 1.4. We first revisit the design of Binomial Mix [24] which was the first design that introduced randomness to the behaviour of a mix. Due

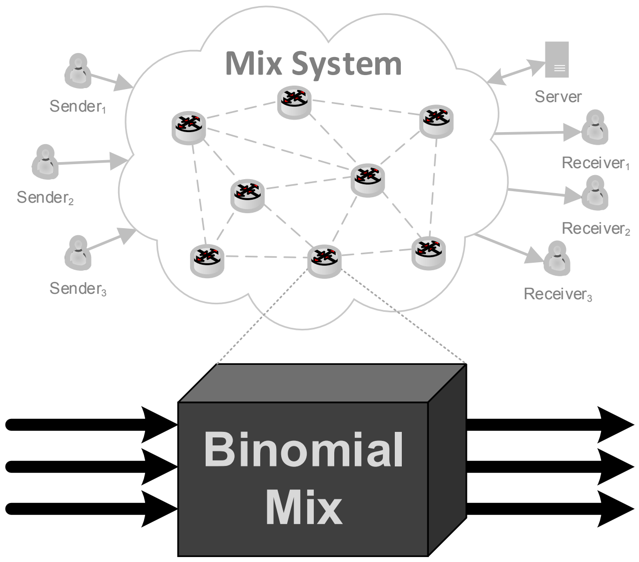


Figure 1.3: RQ1 and RQ2 focus on a particular kind of mix, and analyse and extend that design.

to the random behaviour, adversaries aiming at breaking the anonymity of this mix can succeed only probabilistically. We consider the security of Binomial Mix against passive attacks.

RQ1: *How resistant is Binomial Mix against passive attacks?*

We revisit Binomial Mix to measure its resistance towards the passive attack previously introduced, and give a particular attention to the operation of the mix under various traffic conditions. The passive attack against Binomial Mix and its success ratio was presented in [24], and here we consider whether the same level of resistance is present under various traffic conditions.

The design of Binomial Mix allows the operation of a single selection algorithm which leads to two main issues. Firstly, the behaviour of the mix is well-known to the adversary and closely linked to that of the selection algorithm. Consequently, the adversary is able to capitalise on that knowledge to build attacks targeting the weaknesses of specific selection algorithms. Secondly, the mix treats all the passing traffic equally and thus imposes unnecessary delays, depending on the properties of the selection algorithm in use, where a higher speed is preferable to a very high degree of

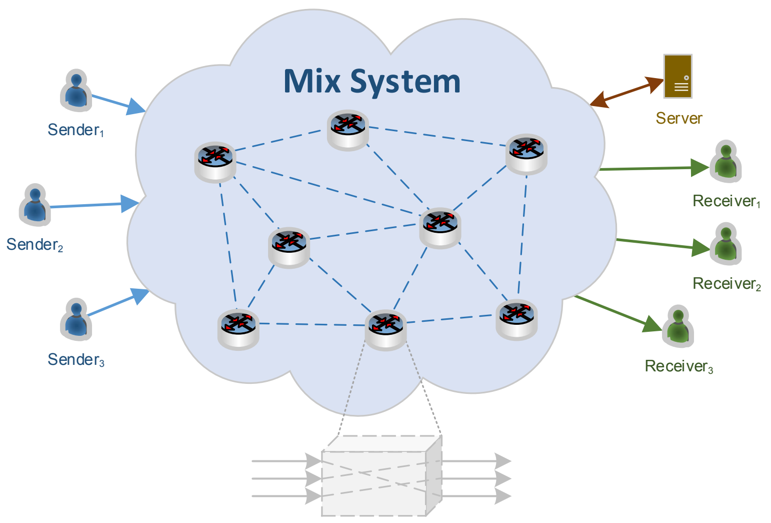


Figure 1.4: RQ3 and RQ4 focus on the users of the system, and framework routers and protocol.

anonymity. These issues lead to the second research question considered in this thesis.

RQ2: *How can Binomial Mix be made more efficient and secure?*

We consider building on the design of Binomial Mix with two goals. Firstly, to decouple the behaviour of the mix from that of a fixed selection algorithm and to introduce more uncertainty to the behaviour so as to frustrate the efforts of an adversary to break the anonymity. Secondly, to find mechanisms enabling the mix to relay multiple communication channels with desired degrees of security and delay.

Moving beyond the design of mixes in isolation, we then turn our attention to the mixnets as a whole as shown in Figure 1.4. As discussed in Section 1.3, a common framework for anonymous communication would allow mixnets to enjoy a range of benefits. This is the aim of the next research question.

RQ3: *How does one build a generic framework for the mix-based Anonymous Communication Systems (i.e. mix systems)?*

We consider the requirements that a generic framework needs to satisfy, and aim to build a framework accordingly. We do not aim to construct a new ACS, but, instead, seek to build an *anonymous network of anonymous networks* whereby various mixnets can merge, hiding against each other, can satisfy multiple security and efficiency requirements and address the challenges discussed in Section 1.3.

The architecture and design of a generic framework consist of multiple components. It is also important to build a system that is not only generic but is also secure and reliable. The next research question is related to one of the security challenges of such systems, namely, the protection against Denial-of-Service (DoS) attacks.

RQ4: *How does one protect the circuit-establishment facility of the generic framework against DoS attacks?*

This question is a dependent question on *RQ3*. We consider the features of the framework which together allow the creation of dynamic communication circuits through the network. We aim to identify the possible DoS attacks on this facility and propose countermeasures to protect the framework against possible attacks.

A thorough consideration of every aspect of a generic framework and all the associated attacks is beyond the scope of this thesis. We hope that our proposal stimulates further research in this domain.

1.5 Outline and Main Contributions

In our work towards answering *RQ1*, we discover that Binomial Mix is significantly more resistant towards the passive attack than previously reported. The results obtained from our attack simulator shows that the great majority of the adversary's findings is, in fact, not genuine results but only false positives. Further analysis of the findings of the adversary reveals a predictable pattern in the false positive results, and thereby the existence of additional information which may assist to improve the attack.

We propose two new mix designs which, as indicated by *RQ2*, are built on the design of Binomial Mix. Multi-Binomial Shared-Pool Mix (MBSP Mix) enables a mix to accommodate not one but multiple selection algorithms which operate one at a time. We have simulated a number of scenarios of using multiple selection algorithms in one MBSP Mix. The results show that the design of MBSP Mix reduces the link between the behaviour of any specific selection algorithm and that of the mix as a whole. It is also shown that, depending on the coexisting algorithms, the MBSP Mix may enjoy a significant decrease in the traffic delay in comparison with Binomial Mix.

The second design, Multi-Binomial Independent-Pool Mix (MBIP Mix), associates each selection algorithm with a separate pool of messages and allows the simultaneous execution of multiple algorithms. Hence, MBIP Mix supports multiple channels of communication with different selection algorithm and thus different degrees of security and delay. Analyses of multiple selection algorithms in MBIP Mix show that the mix can not only cause a significantly lower traffic delay, but can also significantly increase the cost of carrying out a *blending attack* to the extent that makes it impractical for an adversary to do so.

Aiming to answer *RQ3*, we propose the architecture and the detailed design of a generic framework for mix-based anonymous communications. This framework, which we named Garbled Routing Framework (GRF), consist of a dynamic routing scheme as well as a dynamic message processing mechanism. Various mix systems can partially or entirely join the framework in a plug-in based fashion. Three different types of Message Processors (MPs) enable the framework to host any message-processing logic that a *guest ACS* may require. Additional MPs can be deployed in the system through arbitrary Unified Resource Locators (URLs) or, in a trusted way, through a central network authority. We analyse the security of the framework, model certain popular mix systems within the framework, and provide proof-of-concept implementation and test.

Finally, and in order to answer *RQ4*, we analyse the circuit establishment facility of the generic framework to identify possible DoS attacks. We propose three techniques to make the framework resilient towards the identified attacks and provide analyses of what impacts resulted from employing the proposed techniques in the framework.

We show that implementing a combination of these techniques in the system makes the circuit establishment facility safe against the identified DoS and, the distributed counterparts, Distributed Denial-of-Service (DDoS) attacks.

1.6 Organisation

The rest of this thesis is organised as follows. Chapter 2 provides more information about the background of the field and the closely related issues. Chapters 3–6, respectively attend to research questions RQ1–RQ4 of this thesis. In each chapter we present a detailed account of the issue in question, the relevant background, the methodologies used to address the problems, the experiments and our findings. Finally, the thesis will be concluded in Chapter 7.

Chapter 2

Background

If privacy is outlawed, only outlaws will have privacy.
—Philip R. Zimmermann

2.1 Introduction

Anonymity research has grown in importance, and the recent revelations about mass surveillance programs [29] as well as the ever-increasing censorship of Internet access globally [30] only signify the critical role that research in this area can play in protecting privacy of global citizens. In this chapter we review the background of the field of anonymity research and enumerate the possible solutions for achieving communication anonymity. We will also discuss a set of related topics as noted below.

Organisation The rest of this chapter is organised as follows. We start by presenting the basic history of research activity in this domain and explaining the terminology commonly used. This will assist the reader to understand the rest of this thesis where technical terms are frequently invoked. Section 2.3 presents the adversarial models and discusses the importance of threat models. Section 2.4 presents various techniques which are used in this domain for measuring anonymity as provided by Anonymous Communication Systems (ACSs). In Section 2.5 various issues and structures of ACSs and the possible applications for these systems are discussed. Section 2.6 presents the currently available mix designs and discusses their operation.

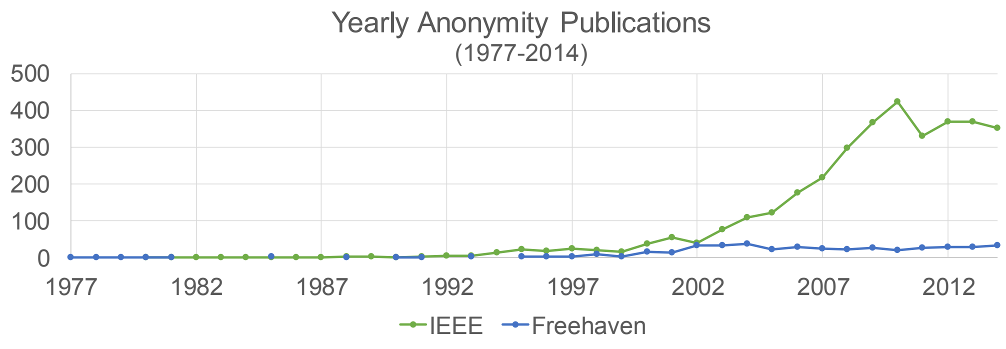


Figure 2.1: History of Anonymity Research

Lastly, in Section 2.7 we discuss the existing mixnets and present their architecture and operation.

2.2 History and Terminology

Central to the anonymity research is hiding the *identity* of an entity, typically that of a user or an agent. The most typical form of identity is the *name* of an entity, that is, a unique identifier of an individual that allows distinguishing that individual in a group [31]. In networked systems, the *name* of an entity may be taken to mean the Internet Protocol (IP) address, Media Access Control (MAC) address, email address, phone number or even, in a broader sense, the geographical location of that entity.

Anonymous is the state of an entity in the absence of information identifying its *identity*. The ultimate goal in anonymity research is to make the desired entities absolutely anonymous. Although few works predate it, the work of Chaum [8] is widely recognised as the first and most influential system to provide anonymity. Since then, anonymity has grown to become a very active field of research. Figure 2.1 shows the number of anonymity publications according to IEEE Xplore[1] [32] and the Freehaven bibliography[2] [33]. It can be seen that the interest in anonymity research has grown - particularly in the last decade.

Pseudonymity is the state of being, firstly, *anonymous* and, secondly, associated

[1]The results are obtained by searching for publications with 'anonymous' or 'anonymity' in their metadata.

[2]The Freehaven bibliography is an authoritative collection of selected publications maintained online.

with a false name. It is important to note that an entity that is associated with pseudonyms is not necessarily *anonymous* because the communication channel *per se* leaks information that can reveal the identity of the entity (e.g. the corresponding IP or MAC address) even when pseudonyms are in use. Pseudonyms are particularly useful in systems that need to track users, such as where reputation and history of users are essential to the operation of the system.

There are multiple forms of *anonymity* that a system can achieve, namely, sender, receiver, communication and location anonymity. *Sender anonymity* is achieved where none of the messages received by the communicating parties in the network can be linked to any particular sender identity. DC-Nets, which will be discussed in Section 2.5.4, achieve *sender anonymity. Receiver anonymity* is achieved where none of the messages sent by the communicating parties can be linked to any particular receiver identity. Systems that achieve *receiver anonymity* include broadcast systems [34, 35] and private information retrieval [36].

Communication anonymity, which is also referred to as *relationship anonymity*, is where no communication or message in the system reveals a relationship between two entities in the system. Further, no two communications or messages can be linked. *Sender* and *receiver* anonymity are stronger properties than *communication anonymity* because in the latter it is clear that the entity in question is indeed participating in some form of communication [37]. Mix systems or *mixnets*, which will be discussed in Section 2.7, achieve *communication anonymity.*

Other terms are also used in this domain which we briefly mention here. *Unobservability* is defined where not an identity but an item of interest is in question [3]; and can be achieved in a system that not only provides *anonymity*, but also uses cover (dummy) traffic. The role and impact of cover traffic will be discussed in Section 2.5.5. *Sender unobservability* and *receiver unobservability* exist where it is undetectable if a message is, respectively, sent by a sender or received by a receiver. *Communication unobservability* exists where communications between two groups cannot be detected.

2.3 Defining Adversary and Threat Model

In anonymity research, an agent whose aim is to degrade or eliminate anonymity is commonly referred to as an *adversary* or an *attacker*. The intent of an adversary is to link a message to a particular sender or receiver identity, find the link between senders and receivers, or find the origin or destination of messages.

Anonymity is achieved in relation to *a particular adversary* with specific capabilities and level of presence. Therefore, it is important to define the properties of the assumed adversary, also referred to as defining the *threat model*, prior to designing or evaluating a particular ACS. Standard terms are used to define various adversarial models which are reviewed in this section.

A *global* adversary is omnipresent and has access to all nodes and links in the network and therefore has complete knowledge about the system. This is in contrast with a *local* adversary who only has a limited presence in, access to, and knowledge about the network. A *passive* adversary is one who can only observe the links and nodes, as opposed to an *active* adversary who can also alter the content of the channels. An *external* adversary is one who does not have access to the internals of the system, whereas an *internal* adversary is an insider who has access to and knows about the internals of the system.

The ultimate capable adversary constitutes one who is *internal* and *external*, *local* and *global*, and *active* and *passive*. However, protecting against such an adversary leads to unrealistic designs, and ACSs are thus usually designed in accordance with a realistic threat model and aligned with the actual requirements in the circumstances.

2.4 Measuring Anonymity

A good deal of research has focused on various means to measure anonymity in different contexts. Protection of the anonymity of data is one of the most important problems to consider; that is, how data about individuals can be disseminated without compromising their privacy. The most popular approach to measure the anonymity of data is k-anonymity [38, 39]. In order to address the limitations of this work, numerous extensions and advancements were proposed later such as p-Sensitivity [40], l-diversity [41], t-closeness [42], an average version of t-closeness [43], δ-disclosure [44] and differential privacy [45].

Where anonymous communication is in question, it is important to note that the capabilities of the considered adversary affect the degree of anonymity that can be achieved by a system. For this reason, anonymity metrics are often modified according to the assumed threat model (see Section 2.3). Further details about different threat models are available in [46].

In this section, we present a brief review of the available metrics for measuring communication anonymity. There are three main approaches to measuring communication anonymity proposed in the literature; namely, anonymity set, individual anonymity degree, and entropy-based metrics. The detail of these approaches is presented hereunder and is followed by a brief mention of the less common alternative metrics.

2.4.1 Anonymity Set

The traditional way of measuring anonymity in an ACS is through measuring the size of the *anonymity set* [20, 47]. The size of the *anonymity set* is the size of the group of senders (for sender anonymity) or receivers (for receiver anonymity) which are equally likely to have sent or received a message.

Assuming that the adversary knows the number of potential senders (or receivers) N prior to an attack, and has compromised C senders during the attack, then the size of the *anonymity set* is $n = N - C$. This quantifies the level of anonymity achieved after an attack.

2.4.2 Individual Anonymity Degree

From the point of view of the adversary, the *individual anonymity degree* is defined in certain possible states shown in Figure 2.2. This qualitative description of anonymity was first presented in the proposal of Crowds [48].

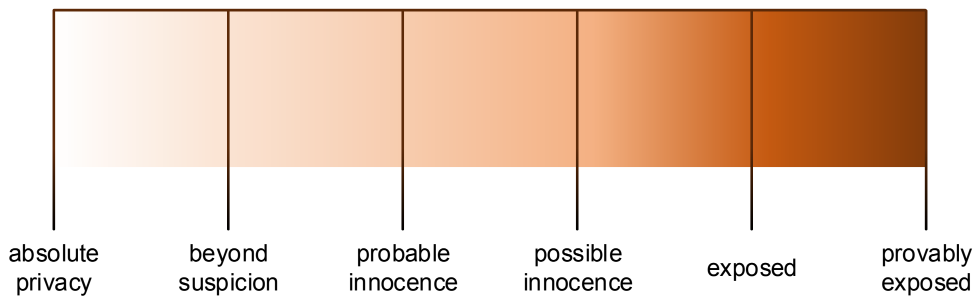


Figure 2.2: Individual Anonymity Degree Scale [48]

Assume that an adversary wants to discover the sender of a particular message. The highest possible degree of anonymity is *absolute privacy* where it appears to the adversary that no one in the ACS ever sent any message. The next state, *beyond suspicion*, which is the highest achievable degree of anonymity, is the state that no-one seems more likely to have sent a message than anyone else. *Beyond suspicion* is also referred to as *total anonymity* or *strongly probabilistic.*

In *probable innocence*, the entity in question appears to the adversary to have a 50% chance of being the sender; namely, it is equally likely for that entity to not be the sender of the message. The next state, *possible innocence*, is when there is a non-trivial chance that someone other than the entity in question has sent the message. *Exposed* is when there is a non-trivial chance that the entity in question has sent the message. Finally, *provably exposed* is when the attacker is absolutely certain and can prove that a certain entity is indeed the sender of the message.

2.4.3 Entropy-Based Anonymity Metrics

Shannon's entropy [49], which measures the level of uncertainty in a set of data, has been used to propose entropy-based metrics for measuring anonymity in ACSs [50, 51]. These metrics consider the global anonymity of an ACS, measure how random the probability distribution is, and can be used to describe the average degree of anonymity or uncertainty of an ACS.

Closely related to these approaches is another metric that uses Shannon's entropy to quantify the anonymity of mixnets considering the observations of some compromised nodes [52]. Further, another work formalised the notion of unlinkability by using Shannon's entropy [53].

On the other hand, it is also argued that metrics should consider the worst-case scenario instead of Shannon's entropy [54, 55] which contemplates the average case. In another work, *communication anonymity* is measured by using Shannon's entropy and min entropy [56]. Similarly, in [57] Shannon's entropy, min entropy and Hartley's entropy are used and a generalisation of those techniques, the Rényi entropy, is proposed.

2.4.4 Other Metrics

Alternative to probabilistic anonymity metrics are *possibilistic* or *nondeterministic* metrics such as those proposed in [58–60]. These metrics define anonymity to be equivalent to the inability of the adversary to link an entity to its actions with *absolute certainty.* A combinatorial approach is proposed in [61] which counts the number of possible one-to-one exchanges between a group of senders and a group of receivers. Limitations and extensions of this approach can be found in [62]. A detailed study of anonymity metrics and a theoretical framework for privacy preserving systems can be found in [63].

2.5 Anonymous Communication Systems

In this section, we review various models of ACSs and their architecture. We firstly look at the possible network structures that an ACS can have. We also discuss the differences between two distinct categories of ACS; namely, wireless and wired. We then present the details of the three main models of ACS; that is, trusted relays and the darknets, DC-Nets and mixnets. Lastly, we review the possible applications of ACSs.

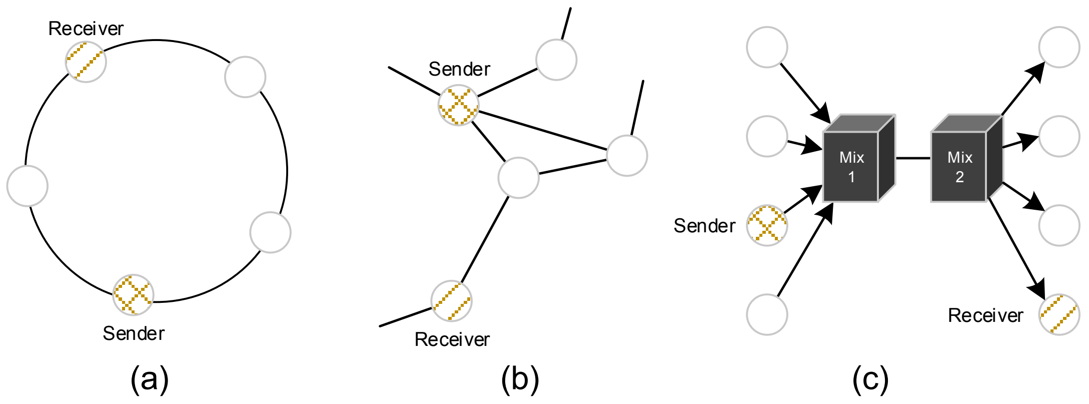


Figure 2.3: Solutions for Anonymity [9]

2.5.1 Network Structure

Regardless of the system in use, communication anonymity can be achieved mainly in any of the three ways as shown in Figure 2.3. In the peer-to-peer (P2P) model, as shown in Figure 2.3(a) and 2.3(b), the sender may be connected to multiple other peers. This approach achieves sender and communication anonymity provided that the peers are not compromised and the communication is not tapped [34].

The third approach, known as mixnets which will be further discussed in Section 2.7, does not achieve sender anonymity because the senders can be traced to the input of the mixes as shown in Figure 2.3(c). Mixnets only achieve communication anonymity.

The solution presented in Figure 2.3(a) is particularly suitable for broadcast communication as proposed in [47] and [34]. Such a system achieves unconditional sender and receiver anonymity [34, 35]. The solution presented in Figure 2.3(b) requires all the network nodes to participate in the system even if they do not have anything to communicate [9]. The resulting system is suitable for low-latency communication (such as [48] and [23]), but the communication channels can be disrupted by any single node that is participating in it.

When compared to mixnets (Figure 2.3(c)), the other two solutions provide less security against a powerful and omnipresent adversary. Further, mixnets are more scalable, robust and efficient in comparison with the P2P solutions [9].

2.5.2 Wireless vs Wired Anonymity

Research aimed at the protection of communication privacy broadly falls in two categories, namely, *wireless* and *wired*. The *wireless* anonymous communication research focuses either on the protection of privacy or location information in stationary sensor networks [64–66] or the protection of anonymity of mobile users in last-hop wireless networks [67–69].

In this research, we focus on the *wired* ACSs. These ACSs have been extensively studied [8, 26, 34, 47], and usually consist of a set of nodes collaborating to achieve communication anonymity. Contrary to the wireless networks, wired ACSs usually assume the network topology is fixed or known to the attacker [70].

2.5.3 Trusted Relays and Darknets

ACSs that rely on trusted nodes are the simplest form of systems that provide anonymity and have been used for a wide range of applications. In these systems, a sender trusts an intermediary node with its identity and that node relays the communication between the sender and the receiver. As the intermediary node learns the identity of the communicating parties, such systems are vulnerable to internal attacks. Moreover, the intermediary node may be forced to reveal the identity of communicating parties. In fact, such an event took place in relation to Penet remailer (anon.penet.fi) [15] where a court of law ruled that the identity of communicating parties be disclosed by the system administrator [10]. Other examples for the use of trusted relays include web proxies (e.g. [71]) and VPN servers utilised for re-routing traffic (e.g. [72, 73]).

A closely related class of systems are the *darknets* which are also referred to as *friend-to-friend* (F2F) networks [74–76]. Darknets, which must be distinguished from the *deep web* [77], are networks that operate based on various degrees of mutual trust and trust delegation [78] among the users, and serve a wide range of purposes such as digital content sharing [79–83] and traffic re-routing [84–86].

2.5.4 Dining Cryptographers Network (DC-Net)

Dining Cryptographers Network (DC-Net) [47, 87] enables a sender to use broadcasting to send a message to multiple receivers and achieves perfect sender and receiver anonymity [35, 47]. This approach does not rely on re-routing of messages to offer anonymity and is based on a peer-to-peer (P2P) network. Therefore, this protocol requires no extra information necessary for re-routing traffic, and achieves a lower delay [88].

No-one in the system will find out who the sender of a message is, and the receiver receives the message under certain circumstances (odd parity). However, due to the use of broadcasting, only one sender can send a message. There is also a scalability issue as the participants require significant coordination and synchronisation for sharing secret coin flips [37]. These shortcomings make DC-Nets unscalable and impractical.

2.5.5 Mixed-Based Schemes

In this thesis we focus on *mix*-based systems, a specific class of ACSs, which has been widely used and subjected to extensive analysis and research. In this section, we present a brief introduction about this class of ACS and the related common attacks. Later, in Sections 2.6 and 2.7, we will provide details about the existing systems.

The Mix (Chaumian Mix)

The original mix design proposed by Chaum [8] was briefly described in Section 1.2.2. This design is the most influential design which has had the greatest impact on research work developed thereafter. Most of the ACSs proposed later are variants of the original Chaumian mix.

Figure 2.4(a) shows a mix which receives messages through $I_1 \cdots I_5$ and relays batches of messages through $O_1 \cdots O_5$ such that there is no correlation between the incoming and outgoing messages in terms of appearance and flow. The inability of linking input and output messages based on their appearance, a.k.a.

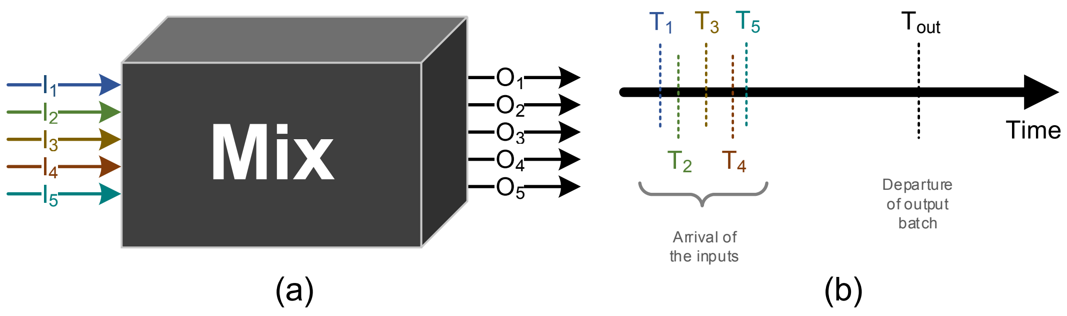


Figure 2.4: A Mix and its Inputs and Outputs [9]

bitwise unlinkability, can be achieved through the use of cryptographic techniques (encryption and/or decryption) and, where a change in the size of messages can disclose sensitive information, by message padding.

In order to remove the information that can disclose the link based on the flow of traffic (e.g. timing and volume), messages can be reordered or delayed. This is shown in Figure 2.4(b) where multiple messages arrived at a mix in various times ($T_1 \cdots T_5$), and are dispatched simultaneously and in a batch in T_{out}. The dispatch of the messages, according to the specifics of the design, may be triggered by different conditions such as time interval or threshold.

Mixnets

Based on the degree of anonymity required in a system, multiple mixes may be interconnected to form a network of mixes or *mixnets*. Each of the mixes in a mixnet performs mixing, as discussed in the previous section, and then relays the batches of messages to another mix or to the final destination.

Based on the design of the mixnet, the mixes may be connected in either of two different topologies to form a network, namely, cascades (fixed-route) and free-route. In mix cascades, as shown in Figure 2.5(a), the communication path goes through the same path and through a set of fixed mixes in a sequential order. The free-route mixnets, also known as peer-to-peer (P2P) mixnets, as shown in Figure 2.5(b), allow the communication to be anonymised by going through any of the several available paths in the system.

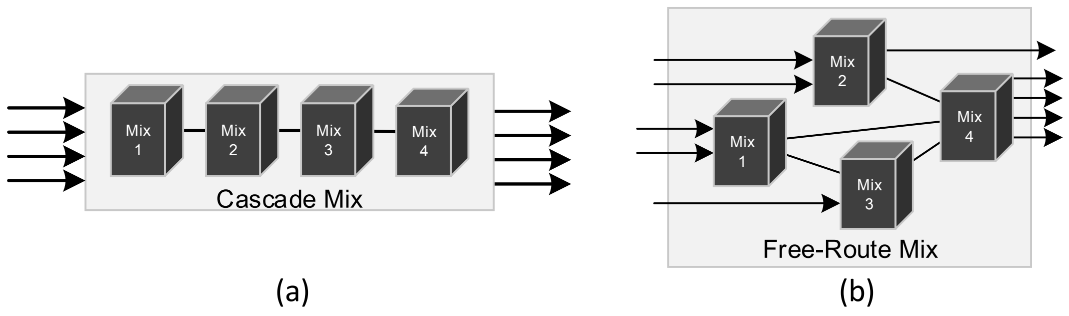


Figure 2.5: Mix Topologies [9]

A considerable body of previous works was dedicated to exploring the different aspects of these two mixnet topologies and weighing their advantages and disadvantages [25, 89–91]. Overall, the cascade topology is generally considered more secure. However, under certain conditions, the free-routing topology provides a more robust anonymity [90].

Attacks on Mixnets

In this section, we review the common types of attacks against ACSs; namely, passive attacks, Denial-of-Service (DoS) attacks, tagging attacks, probabilistic attacks and, finally, the infamous $(n-1)$ attacks.

Passive Attacks Mixes can be vulnerable to a number of different types of passive attacks. Most simply, the attacker may launch a brute force attack; that is, he may choose to follow every possible path the message could have possibly taken [46]. However, it is apparent that a brute force attack requires access to the entire communication channel while also imposing a great cost to the adversary. In a different type of attack, the adversary may also be able to learn about the communicating parties by watching the timing of the incoming and outgoing messages if such information is not eliminated by the mix. Similarly, it is possible to degrade anonymity by watching the communication patterns, counting packets, or relying on the observation that users typically communicate with a relatively small number of parties [46].

Probabilistic Passive Attacks An adversary can use probabilistic techniques to discover the relationship between the messages travelling through the network. When a single mix is the target of such attacks, the aim of the adversary is to find probabilistic correlations between the messages going into the mix and the ones which are flushed out. In Chapter 3, where we analyse the resilience of Binomial Mix [24] against passive attacks, we consider the probabilistic attacks. As we shall see in Section 2.6.7, Binomial Mix uses randomness in its selection algorithm and thus the adversary can only succeed probabilistically.

Denial-of-Service (DoS) Attacks Disrupting the network and rendering some mixes inoperable can make ACSs unreliable to use. Such a disruption in the network can also result in the change of behaviour of certain communicating parties and thus leak information about who is talking to whom [46].

Message Tagging If the first and last mixes are controlled by the adversary, he can tag the messages in the beginning of the path. When the tagged message reaches the end of its journey, the last mix can distinguish it from other messages and thus eliminate its anonymity [46].

$(n-1)$ **Attack** Many anonymity systems aim to protect their users against global and local active attackers, and regard the $(n-1)$ attack (a.k.a *blending attack*, *flooding attack* or *spam attack*) as a vulnerability [92]. The $(n-1)$ attack consists of two phases. In the first phase, the *emptying phase*, the attacker targets a message M_t that is heading a mix. He delays M_t as well as all the other messages heading that mix. Simultaneously, he floods the mix with enough messages to force the flush of the messages inside the mix. Afterwards, the attacker proceeds to the second phase of the attack known as the *flushing phase*.

In the *flushing phase*, the attacker sends the delayed message M_t along with $(n-1)$ other messages of his own to the mix. Once M_t leaves the mix, the attacker can distinguish it from the messages of his own and thus the anonymity provided by the mix will be eliminated.

The $(n - 1)$ or *blending attack* allows an attacker to identify the receiver of a message with absolute certainty [92]. It is important to note that this attack compromises the anonymity provided by a single mix and does not relate to the rest of the nodes in the mixnet. Therefore, it is an inherent assumption that the attacker is powerful enough to be able to isolate and trace a particular message [92].

Cover Traffic A mix may produce cover (dummy) traffic to protect the anonymity of real messages [93]. That is, one or more dummy messages, made to be indistinguishable from the real traffic to the eyes of the adversary, may be inserted into the stream of messages before the mix flushes. Different strategies may be used by mixes in producing cover traffic, some of which make the work of the adversary significantly more costly [92]. Although cover traffic is usually generated by the mixes, it can also be created by users which can improve the security of the network [94].

In the case of the $(n - 1)$ attack, although the adversary will be able to perform the *emptying phase* without any trouble, he cannot achieve the same level of certainty in the *flushing phase* in the presence of cover traffic [92]. When the mix flushes the targeted message (M_t), it will be accompanied by at least one other (dummy) message and thus the adversary's knowledge is reduced at least by half.

Nevertheless, if the adversary is able to keep tracking all of those messages to their destinations, he can eventually distinguish M_t because only that message will end up in the hands of a receiver and all the dummy messages will end up in other mixes [92]. Evidently, conducting this attack would be significantly more costly for the adversary.

It is interesting to briefly mention some other strategies for using cover traffic that can make attacks more difficult [92, 95]. Consider, for example, that the mix may specify different path lengths for each dummy message, or that some of the dummy messages may be destined for the receivers instead of other mixes.

It must nonetheless be highlighted that the use of cover traffic imposes significant overheads to the network because a considerable amount of bandwidth will be consumed not for exchanging real data but solely for providing more security. Further, some strategies may involve generating cover traffic even when the parties do not have anything to communicate (see, e.g., [93]), which may not be desirable for all the participants. Further analysis and discussion about cover traffic and its limitations can be found in [96].

2.5.6 Applications and Other Systems

ACSs are used for a wide range of applications and many existing systems have been developed as free or commercial products. Not all of these systems have been subjected to vigorous research and analysis; nevertheless, they mostly leverage one or more of the ACS-models discussed in the previous sections. Here we look at the potential applications of these systems, and enumerate a number of the systems which are currently in use by a considerable number of users.

Electronic Voting When Chaum published his seminal work about anonymous communication in 1981 [8], he cited voting as a possible application. A large body of research [97–107] has been dedicated to building a class of mixes which provides robustness and verifiability in order to assure senders that the network has processed their messages properly [10]. These properties are closely linked to voting because privacy and verifiability of vote delivery are crucial in this context.

Email Building systems to provide anonymous email communication has been an active field of research. Pseudonymous email relays, such as [15, 108], offered a lower degree of anonymity. Anonymous remailer came later, such as [15, 17, 19, 109], have gained popularity and addressed many shortcomings and security flaws. Another notable project in relation to the exchange of anonymous and/or pseudonymous emails is [110].

File Sharing Many software applications have been developed to address different levels of privacy required for file sharing. The Tor network [26] has been used by some for file sharing. However, it has been suggested that this methodology does not protect the anonymity of users [111]. Many other systems are available for file sharing that fully or partially anonymise users, such as [112–114].

Re-Routing, Copyright Protected Material, and Censorship Re-routing of traffic can be desirable for a variety of reasons such as gaining access to copyright-protected material and circumventing Internet censorship. Many providers restrict the access to the material they provide based on the location of the visitors due to copyright protection requirements. Location is usually determined from the Internet Protocol (IP) address of the visitors. In order to gain access to the otherwise inaccessible material, Internet users can use servers that re-route traffic. By so doing, these users pretend to be located where the re-routing servers are. Traffic re-routing is also one of the primary means to circumvent Internet censorship because it can simply deceive the censoring firewalls into believing that users are visiting legitimate destinations. That is, the firewall can be deceived by the act of re-routing into believing that the user is not communicating sensitive material as the (encrypted) packets are being exchanged with an unknown entity (i.e. the re-routing server). The OpenNet Initiative [30] regularly conducts research to expose and analyse Internet filtering and surveillance practices on a global scale [115–118].

Tracking and Profiling Avoidance Tracking and profiling the activities of Internet users is a widespread practice and a major invasion of privacy. Multiple advanced and sophisticated techniques exist to carry out these types of activities, such as the use of web cookies and digital fingerprinting [119, 120]. ACSs obfuscate communication patterns and thus protect Internet users from becoming the subject of tracking and profiling. Nevertheless, some attacks have been reported to be effective on ACSs [121–124].

Anonymous Payments Protecting the privacy of financial transactions has long been the goal of the cryptographic community, dating back to 1983 [125]. However, the traditional electronic transactions require a central and trusted entity, which have hindered the development and adoption of anonymous electronic transactions. With the rise of decentralised electronic currencies such as Bitcoin [1], the need for a central authority has been eliminated and ACSs can now play a more effective role in providing anonymity to financial transactions. A whole body of research has been dedicated to this need (see, e.g., [126–128]).

Other Applications There is a number of other applications for ACSs. It can provide a safe communication channel for many, such as journalists, military personnel, whistle-blowers, law enforcement officers, parents seeking online privacy for their children, and human rights and political activists. Freedom of speech, in particular, has been the reasoning behind building many systems (see, e.g., [129–131]). While noting the importance of ACSs and their various and numerous applications, we should also mention that this technological tool, akin to any other tool built by humanity, can be used for malicious and/or unethical reasons.

2.6 Existing Mixes

In this section we present a review of the mix designs that are most commonly cited in the literature. Our focus is to describe how each mix operates and how much delay is imposed by each mix on the transmission of messages.

2.6.1 Threshold Mix

Threshold Mixes wait until they receive n messages and then they flush all the messages at once [92]. The minimum delay caused by a Threshold Mix is ϵ for the scenario where a message arrives when the mix has already received $n-1$ messages and thus flushes instantly. The maximum delay can be infinite because the mix may not receive enough messages to meet the threshold and thus the messages may remain in the mix. The mean delay, assuming a constant rate of arrival r, can be computed according to $\frac{n-1}{2r}$.

2.6.2 Timed Mix

Timed Mix (TM) flushes in specific time intervals (e.g. every t seconds) whereby every message in the mix will be flushed [92]. The minimum delay is ϵ which occurs for a scenario where a message arrives just before the time interval. The maximum delay is $t - \epsilon$ which occurs if a message arrives right after a flush. The mean delay is $\frac{t}{2}$. In Chapter 4, we use TMs where we consider various combinations of mixes that can coexist within our proposed mix designs.

2.6.3 Threshold and/or Timed Mix

Threshold and time parameters can be combined in a mix. In Threshold *and* Timed Mixes, in every time interval t the mix flushes its messages but only if there are at least n received messages [92]. The minimum delay is similar to Timed Mix, ϵ, and the maximum delay is similar to Threshold Mix - infinite.

In Threshold *or* Timed Mixes, in every interval t the mix flushes all of its messages, but it also flushes all of its messages at any time if at least n messages have been received regardless of whether the interval has been met [92]. The minimum and maximum delays are similar to Timed Mix, that is, ϵ and $t - \epsilon$ respectively.

2.6.4 Threshold Pool Mix

Pool Mixes contain a pool inside the mix with the capacity of n messages, and in the flushing time only a subset of the messages inside the pool is chosen by the selection algorithm to be dispatched. A Threshold Pool mix always maintains f number of messages, i.e. the threshold, inside the mix. The retained messages are chosen uniformly at random [92]. The mix flushes n messages when $n + f$ messages accumulate in the mix.

The minimum and maximum delays are similar to those of Threshold Mix, namely, ϵ and infinite, respectively. Due to the probabilistic nature of the mix, there is a low chance that a message may remain in the mix for an arbitrarily long time. Therefore, the mean delay is $1 + \frac{f}{n+f}$. Assuming an average rate of receiving r messages per second, the average delay is $(1 + \frac{f}{n+f})\frac{n}{r}$.

2.6.5 Timed Pool Mix

Timed Pool Mix (TPM) [92], similar to TM, has a periodic flushing time. It does not, however, flush all its messages and retains f messages chosen in a uniformly random fashion in the pool in every round. If in the flushing time no more than f messages are accumulated in the mix, it will not flush any messages.

The minimum delay is ϵ, and the maximum delay is infinite because insufficient traffic may result in the mix never flushing the messages in its pool. Similar to Threshold Pool Mix, it is possible, though very unlikely, that messages can remain in the pool for an arbitrarily long time. TPMs will be used in Chapter 4 where we evaluate combinatorial models.

2.6.6 Timed Dynamic-Pool Mixes (Cottrell Mix)

Timed Dynamic Pool Mix (TDPM) [92] is similar to TPM in that its flushing time is according to a fixed time interval t. TDPM retains a number of messages in the pool in every flushing iteration. However, the number of flushed messages is not fixed but, instead, is relative to the number of messages inside the pool.

Assume TDPM contains $m + f_{min}$ number of messages in the pool. In the flushing time, the mix will flush messages only if the number of messages in the pool are higher than a threshold f_{min}, that is, where $m > 0$. If that condition is met, TDPM flushes either a fraction of $m + f_{min}$ or one (1) message, whichever is higher.

The minimum delay that can be imposed by a TDPM is ϵ, and the maximum delay is infinite. The mean delay is as high as that of a TPM and typically higher. TDPMs, too, will be used in our evaluations in Chapter 4.

2.6.7 Binomial Mix

Binomial Mix [24] is known as the first design that introduced randomness to the behaviour of the traffic mixers. This design is a particular type of TDPM which relies on the normal Cumulative Distribution Function (CDF) to choose the subset of messages that should be flushed in an iteration.

The operation of Binomial Mix is as follows [24]. Let the number of messages that the mix flushes be s, which, on average, is equivalent to $nP(n)$. s follows a Binomial distribution, and thus has a variance equivalent to $np(1-p)$, where p is the result of the function $P(n)$.

Using the normal CDF is suitable because of its desired properties. In low traffic conditions, the normal CDF increases the delay of the communication and retains the messages in the mix for a longer period. This increases the anonymity in low traffic conditions as the messages will not be flushed merely due to a forced time interval.

In heavy traffic conditions, the normal CDF decreases the delay of the communication and allows a higher throughput. Consequently, Binomial Mix achieves a lower delay than the TDPM in heavy traffic conditions, but this comes at the cost of providing lower anonymity because then the messages are very unlikely to stay in the mix for more than one round.

Due to the probabilistic nature of Binomial Mix, the adversary does not obtain much information about the number of messages inside the pool by observing the incoming and outgoing messages of the mix [24]. Therefore, attacks on Binomial Mix are not exact and can succeed only probabilistically. In Chapter 3 we will revisit the design of Binomial Mix and re-examine its resilience towards probabilistic passive attacks. In so doing, we will be taking into account the impact of various traffic conditions on the success rates of the attack. In Chapter 4 we extend the design of Binomial Mix and propose two new models which achieve greater security and efficiency. Further details about the operation of Binomial Mix and the relevant attacks will be presented in the respective chapters.

2.6.8 Stop-and-Go Mix

Stop-and-Go (SG) Mix [20] is different from other models discussed in the previous sections, in that it does not operate by batching messages. Instead, SG Mix relies on the delay built into the message by the sender. The delay is specified according to an

Exponential distribution. When an SG Mix receives a message, it delays the message for the specified time and then relays it.

The design of SG Mixes was aimed at minimising the potential for $(n-1)$ attacks. Another interesting property of SG Mixes is that, due to the independence of network messages, it is possible to blend high- and low-latency traffic in the same system. However, it is shown that SG Mixes provide no anonymity in extreme cases of low traffic because they do not adapt the delay to the traffic conditions [132].

2.6.9 Alpha Mix and Tau Mix

Alpha Mix [21] is an approach that generalised SG mixing with the aim of enabling mixes to blend high- and low-latency traffic. The sender considers its security requirements and accordingly makes a trade-off between security and communication delay.

In Alpha Mix, the sender of a message M specifies in the message the number of *batches* that a mix must process before it can relay M. Tau Mix differs in that M contains not the number of batches but the number of *messages* that must be processed before M can be relayed by the mix.

2.6.10 RGB Mix

RGB Mix [133] sends *heartbeat* messages to themselves through the network. This approach protects against the $(n-1)$ attack because the *heartbeats* will not be received if the mix is under the first phase of the attack, namely the *emptying phase.* As a countermeasure, the mix injects cover traffic to confuse the adversary. The details of $(n-1)$ attack and the impacts and limitations of using cover traffic were discussed in Section 2.5.5.

2.6.11 Verifiable Mixes

Mixes with universal verifiability properties are a dedicated field of research into the development of systems that assure the senders that their messages have been properly processed. More information about verifiable mixes can be found in [102, 103, 105–107, 134, 135].

2.7 Existing Mixnets

This section contains a brief overview of the most important mixnet designs that exist to date. It will discuss the details of the systems which have been widely adopted by users and/or been distinctly influential in the developments of the field.

2.7.1 Anonymous Remailers

The Chaum's mix was designed to anonymise email messages. Multiple systems were proposed later which used mixnets to anonymise emails. Early attempts to anonymise email messages (Type 0) resulted in the creation of pseudonymous remailers such as Penet remailer (anon.penet.fi) [15]. Second generation of remailers (Type I), known as Cypherpunk remailers, were a very weak version of the Chaumian mix [15]. In the third generation (Type II), Mixmaster [16, 17], provided protection against traffic analysis by using techniques such as message reordering and padding. Finally, the fourth generation (Type III), Maximinion [18, 19], additionally provided bidirectional communication and is extremely resistant to traffic analysis [10].

2.7.2 Onion Routing and Tor

Onion Routing [136–139] is a variant of mixnets which is suitable for low-latency and interactive communication (e.g., Secure Shell (SSH) and web browsing). This system provides sender, receiver and communication anonymity.

In an Onion Routing network, the sender encrypts each message multiple rounds, similar to the layers of an onion as shown in Figure 2.6, and then sends it to the first Onion Router in the cascade. The first router decrypts the message with its private key, and retrieves the routing information for the second router in the chain as well as the encrypted message that it must relay to the second router. There are five Onion Routers in the chain that repeat these actions until the last one sends the message to

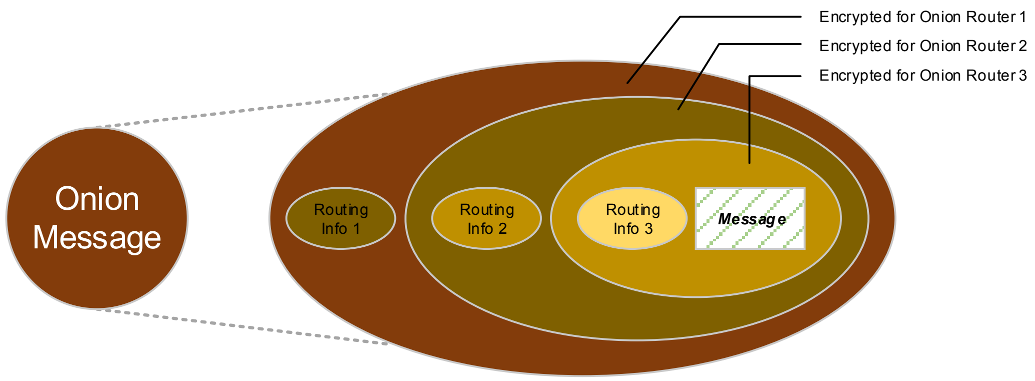

Encrypted for Onion Router 1
Encrypted for Onion Router 2
Encrypted for Onion Router 3
Onion Message
Routing Info 1
Routing Info 2
Routing Info 3
Message

Figure 2.6: Structure of an Onion Message: the message consists of multiple layers of encryption performed using the public keys of the intermediary routers. When each layer is removed, the Onion Router extracts the routing information for the next hop as well as the message that must be relayed.

its final destination. The system is implemented at the application or Transmission Control Protocol (TCP) layer, and forward secrecy is protected by relying on standard secure communication protocols.

The second generation of Onion Routing, The Onion Router (Tor) [26, 140, 141], uses free-routing instead of cascades. In the new design, the number of routers in a circuit has been reduced to three. At the time of writing, Tor is a popular anonymous network with a large user-base. Tor is a mature ACS and its various aspects and properties have been subject to vigorous and ongoing research (see, e.g., [142–149]).

2.7.3 Freedom Network

The Freedom Network closely follows the architecture of Onion Routing. Details of the architecture of the system were published in a series of technical papers [150–153]. It offered sender anonymity for web browsing and could also be used for electronic mail, Secure Shell (SSH), Telnet and Internet Relay Chat (IRC). The network topology used in this system was the cascade model.

Freedom Network was efficient and reasonably secure against DoS attacks but was vulnerable to generic traffic analysis. This system was a commercial service designed and operated by Zero-Knowledge Systems Inc., a Canadian company, and was discontinued in 2001.

2.7.4 Garlic Routing

Garlic Routing [154, 155] is a variant of Onion Routing [137]. A Garlic message, similar to an Onion message as shown in Figure 2.6, contains multiple layers of encryption which should be peeled by the intermediary routers while in transit. After each router peels a layer of the message, instead of an onion it finds multiple garlic *bulbs* to transmit.

Each *bulb* contains an alternative and complete path from the router to the destination. Hence, Garlic Routing provides path redundancy and, therefore, delivery reliability and robustness.

2.7.5 I2P Network

The Invisible Internet Project (I2P Network) [27, 28] is a mixnet suitable for low-latency communication. I2P is based on Garlic Routing and uses the term *cloves* to refer to the *bulbs*. The communication paths in the I2P Network are unidirectional, and the path establisher can choose the number of hops in a path and whereby makes a trade-off among anonymity, latency and throughput.

I2P Network uses a Kademlia-like [156] Distributed Hash Table (DHT) as its network database to store and disseminate information about routers and particular destinations.

I2P Network is not designed to access an arbitrary destination over the Internet. However, it is possible for a node to become an *exit relay* to share its Internet access with the other anonymous nodes in the network.

Various aspects of I2P Network have been subject to analyses and research such as its security [157–160], availability [161] and usage [157, 162].

2.7.6 Tarzan

Tarzan [163, 164] is a peer-to-peer ACS that works in Internet Protocol (IP) layer and achieves sender, receiver and communication anonymity. The communication circuit through the network is dynamically created by the sender. Cover (dummy) traffic is used to provide additional security. Failure of a communication tunnel results in significant delays and computation overheads [165]. Tarzan provides application independence, is suitable for Web surfing and has the benefit of using less processor intensive symmetric keys.

2.7.7 MorphMix

Morphmix [22, 23] has a very similar architecture and threat model to that of Tarzan. The circuits through the network, however, are not created by the sender. Instead, intermediary nodes specify the routes, and their actions are observed by witnesses which are specified and trusted by the senders.

The choice of delegating circuit establishment to the other nodes is made in order to prevent a subverted node from choosing the path such that it only goes through other subverted mixes. Collusion detection and prevention mechanisms are built into the system but are not effective in every case [166].

2.7.8 Crowds and Hordes

Suitable for Web surfing, Crowds [48, 167, 168] is a mixnet that uses the name *jondos* to refer to its mixes. Hops of a communication channel in Crowds are extended by mixes in a random (but biased) fashion in each hop. This achieves sender anonymity and removes a single point of failure in the channel. Communication anonymity can also be achieved, although it can be nullified if the sender reveals identifying information in the request [169].

Hordes [170] improves the design of Crowds. Unlike Crowds, which uses HTTP proxies as mixes, Hordes uses User Datagram Protocol (UDP) proxies. Another difference is that Hordes achieves sender anonymity through using multicast reverse paths.

2.7.9 ISDN, Real-Time and Web Mixes

Mixes were used to anonymise telephone conversations over the Integrated Services for Digital Network (ISDN) [11]. This usage of mixes was practical as it met the requirements of ISDN networks. Later, a generalised version of this design was proposed which could be used for real-time and low-latency communication [12]. These designs were further improved and modified to form Web Mixes [13] which were suitable for anonymous web browsing and was implemented in a web anonymising proxy called JAP [14].

2.7.10 Other Mixnets

There are other systems which leverage the concept of traffic mixers. WonGoo [171] is a scalable P2P system based on Crowds for low-latency communication. This system relies on random forwarding and layered encryption, and provides strong anonymity and high efficiency. For a comparison of WonGoo, Crowds and the pure mix, refer to [172].

Cashmere [165] relays traffic through robust groups of mixes to provide a resistant anonymous layer on a structured P2P overlay. Cashmere achieves sender and communication anonymity, and its design can be extended to offer receiver anonymity.

2.8 Summary of the Chapter

In this chapter we have presented a broad overview about the important and relevant topics in relation to the anonymity research. We have seen that this field of research is becoming increasingly popular as the activity in the field clearly demonstrates. We reviewed the terminology and adversarial models commonly considered in this field, and considered the different anonymity metrics proposed in the literature.

As we have seen, ACSs can be built to address various degrees of anonymity according to their use cases. They range from simple and trusted traffic relays to sophisticated systems that use traffic mixers, or *mixes*, to achieve anonymity. We also briefly reviewed the wide range of applications for ACSs.

Due to the focus of this thesis, we presented a more detailed review of the existing mix designs. Among the available designs, Binomial Mix is capable of making traffic analysis more difficult by relying on a probabilistic selection algorithm. Consequently, the behaviour of the mix is less predictable to the adversary and therefore less knowledge is available for building attacks. In Chapter 3 we further analyse Binomial Mix and examine its resistance against probabilistic passive attacks. In Chapter 4 we take a step further and propose two new mix designs based on Binomial Mix.

Finally, in this chapter we also looked at the existing and most extensively researched mixnets. It is apparent that the discussed mixnets are architecturally different and use various message-processing techniques. We also know that a mixnet with a larger number of users can potentially offer greater anonymity. In Chapters 5 and 6 we propose a framework system for merging various mixnets such that they can protect their distinct algorithms and designs but also enjoy the company of the users of other ACSs.

Chapter 3

Passive Attacks on Binomial Mix

They who can give up essential liberty to obtain a little temporary safety deserve neither liberty nor safety.
—Benjamin Franklin

3.1 Introduction

Many years after the initial *mix* was proposed by Chaum [8], a number of improved designs have been proposed in the literature [5, 132]. Among these proposals, Binomial Mix [24, 173], which for the first time introduced randomness to the behaviour of a mix, is the most sophisticated design.

The focus of this chapter is to answer *Research Question 1* of this thesis (see Section 1.4). Specifically, we focus on the passive attack previously reported in [24] which aims to guess the number of messages contained in a Binomial Mix and, using additional computation, to break the anonymity provided by the mix. Since the operation of Binomial Mix involves randomness (see Section 3.3.1 for details) the attacker needs to observe the operation of the mix a number of times before it can guess, and only probabilistically so, the number of messages contained in the mix. Such an attack was simulated and it was reported that the attacker needs typically about 200 rounds of observation to succeed [24].

The degree of anonymity provided by Binomial Mix strongly correlates with the network traffic conditions. That is, a higher incoming traffic volume yields a lower degree of anonymity (and a lower delay). We note that the previously simulated attack did not take into account the correlation between the traffic volume and the resultant anonymity. In this chapter, we revisit the attack and closely examine the impacts of the noted correlation.

Our findings demonstrate that Binomial Mix is significantly more resilient to the attack than previously reported. That is, the reported success rate of the attack is verified only in low traffic conditions. The results also show the unreliability of the attacker's findings as the attack in fact never results in failure but in a large number of false positives instead. Nevertheless, the false guesses may provide additional knowledge which might be used to improve the attack. Lastly, the impact of dropped messages, in heavy traffic conditions, on the success rate of the attack is shown.

Organisation The rest of this chapter is organised as follows. The following section defines the *threat model* used in this chapter. Section 3.3 contains the required background about the operation of Binomial Mix, the terminology used in this chapter and the definition of the passive attack in question. The details of the simulation and incoming traffic ratios are presented in Section 3.4. In Section 3.5 we present the highlights of the experimental results and discuss the outcomes of the experiments and the corresponding observations. Section 3.6 elaborates on the effectiveness of the passive attack on Binomial Mix in light of our findings. In Section 3.7 we suggest possible venues for extending our work and examining other interesting situations. Lastly, Section 3.8 presents a summary of this chapter.

3.2 Threat Model

It is important to note that one assumes the capabilities of the adversary have fundamental impact on the anonymity that can be achieved by a particular Anonymous Communication System (ACS). In other words, the anonymity provided by an ACS can be measured only by assuming *a certain* threat model, and the results are no longer valid if the threat model changes.

We assume a passive attacker, that is, one who can observe all the incoming

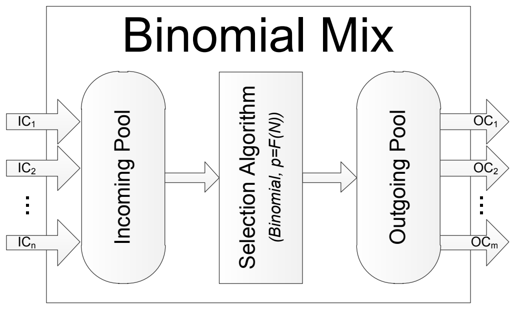


Figure 3.1: Schematic of Binomial Mix

and outgoing traffic of the mix without any limitations and knows all the internal parameters of the mix. However, he cannot actively tamper with the traffic, i.e. to add, remove, delay or modify messages. Neither can he carry out internal attacks - namely, break into a mix and see the internal flow of the traffic. We also assume that the attacker does not have *a priori* or contextual information about the correlation between the incoming and the outgoing messages.

3.3 Background

3.3.1 Binomial Mix

A schematic of Binomial Mix is shown in Figure 3.1. The mix receives a number of messages through different channels (i.e. $IC_1 \cdots IC_n$) and stores them in a pool. The batching strategy of the mix consists of one selection algorithm which in each flushing time tosses a coin for every message in the pool to decide whether it must be taken out and relayed. The coin is biased according to the Cumulative Distribution Function (CDF) of the Normal distribution, $\Phi(N)$, where N is the number of messages within the pool. The messages remaining in the pool along with the newly received messages will be tried in the next flushing time.

The effect of using the normal CDF to bias the coin is that an increase in the incoming traffic volume translates into a higher probability of success in the Binomial distribution, and thus into a higher outgoing traffic ratio and a lower anonymity. In low traffic conditions, however, the probability of messages staying in the mix rises and the outgoing traffic is thus relayed with a higher delay and a higher anonymity.

Table 3.1: Terminology Used for Defining Passive Attacks on Binomial Mix

Sign	Meaning
i	Number of the round of the attack
A	Number of incoming messages at the start of the round
N	Number of messages in the pool (assuming A)
N_{max}	Maximum capacity of the pool
S	Number of outgoing messages (assuming N)
a_i, n_i and s_i	Where multiple rounds of attack is in question, this notation represents the actual values of the corresponding random variables in a particular attack round i (e.g. a_1 denotes the actual number of received messages in the first round of the attack.

3.3.2 Passive Attack on Binomial Mix

As mentioned before, due to the probabilistic nature of its selection algorithm, attacks on Binomial Mix are also generally probabilistic; that is, the attacker will observe the incoming and outgoing messages of a mix and try to correlate them using probabilistic techniques. In more sophisticated attacks, i.e. the active attacks, the attacker may also tamper with the incoming messages to short-circuit the process.

In order to define the passive attack on Binomial Mix, we must first define the information which is of interest to the attacker. Table 3.1 contains the terminology used for representing one round of mix's operation.

The attacker can observe A and S, and we assume, according to the threat model (see Section 3.2), that he knows the value of N_{max}. He wants to find the value of N, or more specifically n_1, to be able to correlate, with some certainty, the incoming and outgoing messages and thus to degrade the anonymity provided by Binomial Mix. According to the known properties of the Binomial distribution [174], given $N = n$, the attacker is able to compute the probability of $S = s$ by:

$$P(S = s|N = n) = \binom{n}{s} p^s (1-p)^{n-s} \tag{3.1}$$

However, N is not known to the attacker and he must estimate N from his observations of S. According to the Bayes' theorem we have:

$$\begin{aligned} P(N = n|S = s) &= \frac{P(s|n)P(n)}{P(s)} \\ &= \frac{P(s|n)P(n)}{\sum_{i=0}^{N_{max}} P(s \cap i)} \\ &= \frac{P(s|n)P(n)}{\sum_{i=0}^{N_{max}} P(s|i)P(i)} \end{aligned} \tag{3.2}$$

According to the threat model, the attacker does not have any *a priori* knowledge and thus must assume that all of the possible values of n are equally probable (i.e. $P(n) = P(i)$). Hence, he initially must assume that:

$$P(N = n) = \begin{cases} \frac{1}{N_{max}-s+1} & \text{if } s \leq n \leq N_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Hence, the Equation (3.2) can be simplified as follows:[1]

$$P(N = n|S = s) = \frac{P(s|n)}{\sum_{i=s}^{N_{max}} P(s|i)} \tag{3.4}$$

Although with a single observation the attacker gains very little information about n_1, he can improve his knowledge by observing multiple outputs (i.e. $s_1, s_2, \cdots$) to eventually estimate n_1 with a certain probability. His goal would be to find a value for n_1 that maximises Equation (3.4). To this end, after each observation of s_i the attacker needs to find the probability of every possible value of n_1. That is, he must compute $P(N = n|S = s), \forall n \in \mathbb{N} : s \leq n \leq N_{max}$ according to Equation (3.4).

The attacker needs to combine what he has learnt from each observation and, since each observation is independent of the others, he only needs to multiply the results of different observations. It is important to note that, due to the differences in incoming and outgoing traffic in each round, the value of N is likely to be different from one round to another. Hence, the computed probabilities need to be shifted according to relative changes in the values of N before these values can be multiplied to the

[1] The denominator mentioned in Equation (2) of [24] is $\sum_{i=s}^{N_{max}} P(i|n)$. This is due to a typo in [24], because that denominator could only be derived based on a wrong assumption, i.e. $P(s) = P(n)$. Moreover, had that denominator been used as the basis of the simulation, the attack simulator may have generated no results.

results obtained from the previous observations. Further details about this algorithm are available in [24].

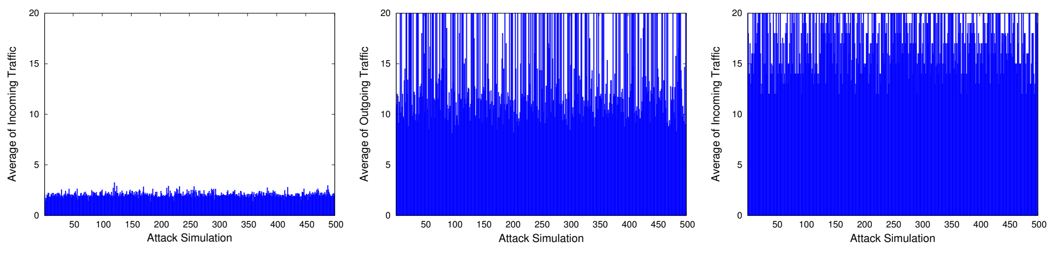
It has previously been reported [24] that one round of observation allows the attacker to guess the value of N with a probability of up to 15%. It has also been reported that the attacker needs approximately 200 rounds of observation to be able to guess the value of n_1 with a probability of at least 95%. These results were obtained from a simulator which combined the results of multiple rounds of attacker's observations.

We have implemented a simulator according to the described algorithm in [24] and [173], and conducted further experimentation. We have taken into account the impact of traffic volume on the success rate of the attack. Details of the findings are reported in Section 3.5. Let us first explain the experimental scenarios in the following section.

3.4 Incoming Traffic Relative to the Capacity of the Mix

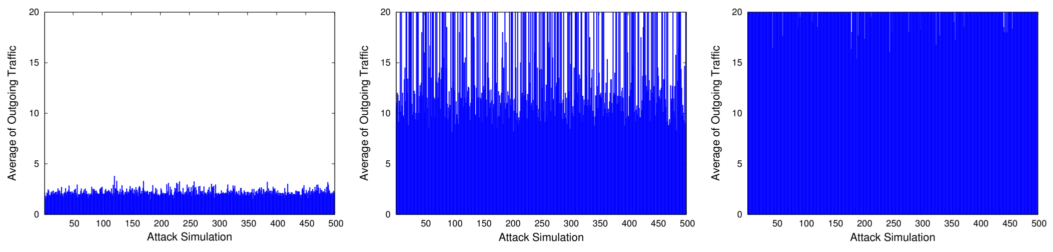
We pay particular attention to the incoming traffic (A) relative to the maximum capacity of the mix (N_{max}), and analyse its impact on the passive attacker's effort at estimating the number of messages in the pool (N). According to the *threat model*, the attacker is aware of the internal parameters of Binomial Mix, and we thus assume a fixed maximum capacity $N_{max} = 20$ which is also known to the attacker. We then study five scenarios with, respectively, an incoming traffic equal to 10%, 25%, 50%, 75% and 90% of the maximum capacity. The incoming traffic, in accordance to the widely accepted assumption in the field, is Poisson distributed.

A single round of attack starts from the attacker's first observation, continues with the consecutive observations which improve the attacker's knowledge, and ends when the attacker becomes capable of estimating the value of n_1 with at least 95% confidence. To obtain reliable results, we have conducted 500 rounds of attack simulation for each incoming traffic ratio. In each round the attacker is allowed up to 1000 observations of the traffic and the attack will be marked as failed if he cannot



(a) ≈ 10% of Capacity (b) ≈ 50% of Capacity (c) ≈ 90% of Capacity

Figure 3.2: Incoming Traffic Samples



(a) ≈ 10% of Capacity (b) ≈ 50% of Capacity (c) ≈ 90% of Capacity

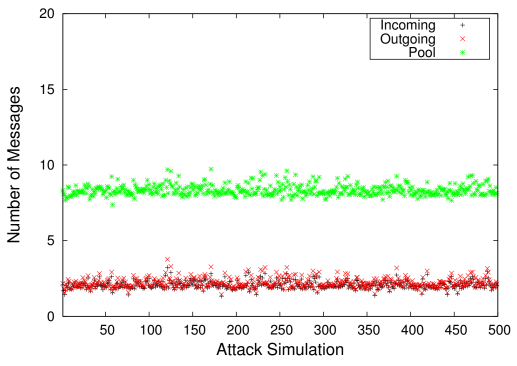
Figure 3.3: Outgoing Traffic Samples

estimate n_1 after exhausting all of his observations.

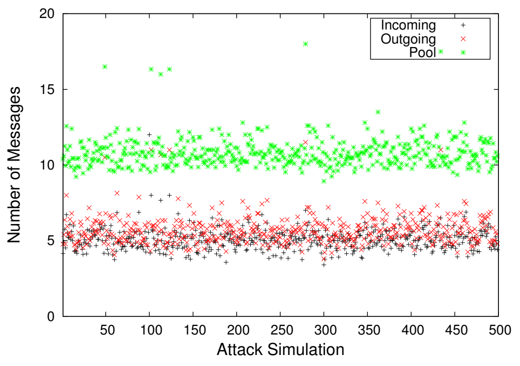
Figures 3.2(a)–(c) show the average of the incoming traffic in three of the five scenarios where the parameter of the Poisson distribution λ is respectively set to $\frac{N_{max}}{10}$, $\frac{N_{max}}{50}$ and $\frac{N_{max}}{90}$. Figures 3.3(a)–(c) show the average of the output traffic of the mix for the same scenarios. Figures 3.4(a)–(d) show the average of the overall performance of the mix in terms of the average number of messages in the pool in relation to the incoming and outgoing traffic.

3.5 Analysing the Experimental Results

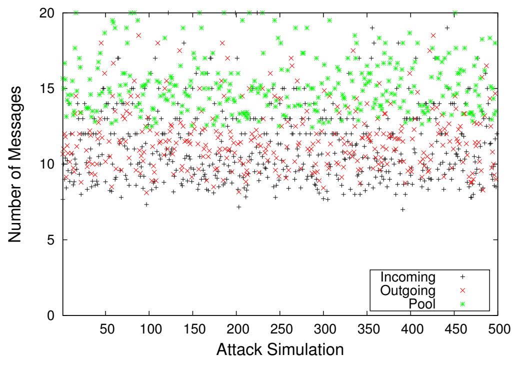
The previous sections presented the details of the attack as well as our experimental scenarios. We have built a simulator using the Java programming language and conducted the experiments. This section analyses the various results obtained from the simulator.



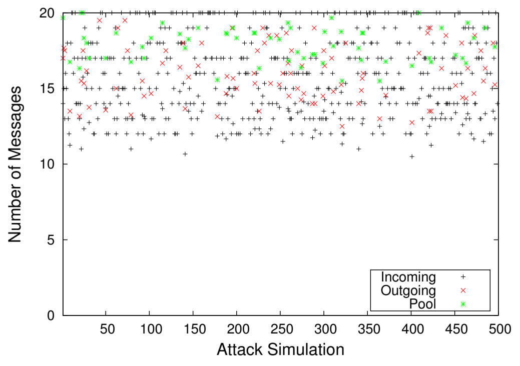
(a) ≈ 10% of Capacity



(b) ≈ 25% of Capacity



(c) ≈ 50% of Capacity



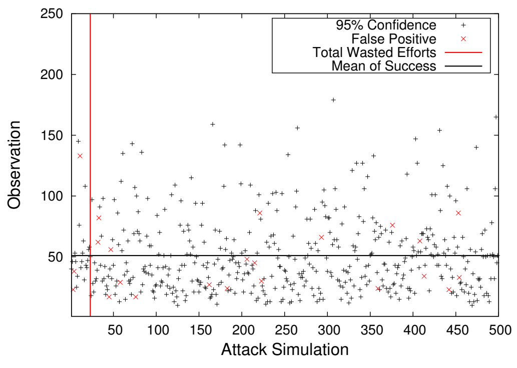
(d) ≈ 75% of Capacity

Figure 3.4: Average of Overall Performance of Mix in Experimental Scenarios. Each of the plots shows the volume of incoming messages (the black + signs), outgoing messages (the red × signs) and the number of messages in the pool (the green $*$ sign).

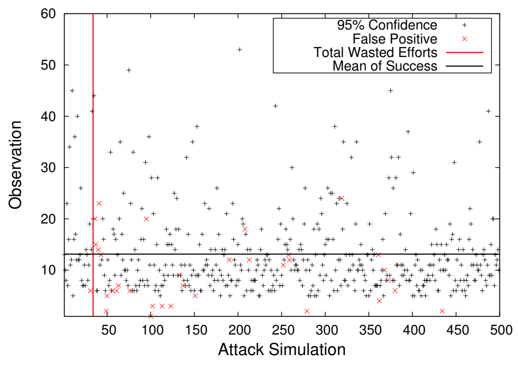
3.5.1 Impact of Traffic Volume on Attacker's Success

The results obtained from the scenarios mentioned in Section 3.4 are shown in Figures 3.5(a)–(e). Each figure contains a number of interesting items which can be understood with the following guides.

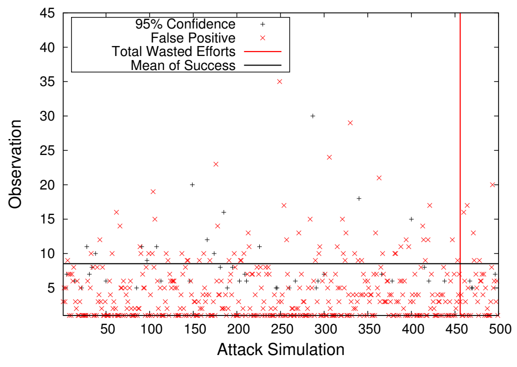
- The black dots show the number of rounds before the attacker could successfully estimate the value of n_1.
- The red dots show the false positive results; that is, when the attacker estimated the value of n_1 with a probability of at least 95% but his estimation was in fact wrong.
- The black horizontal lines show the mean of the number of times that the



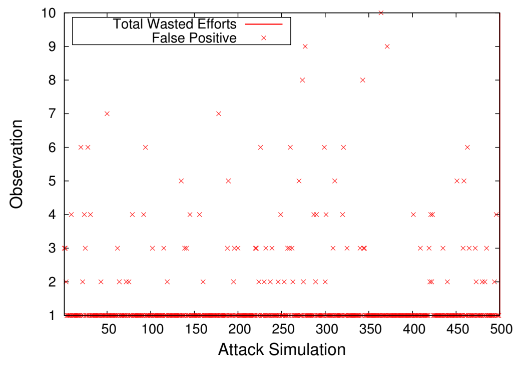
(a) ≈ 10% of Capacity



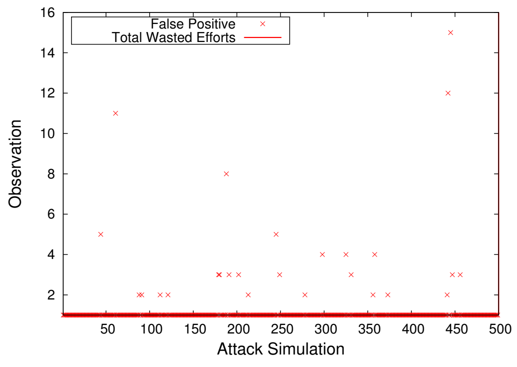
(b) ≈ 25% of Capacity



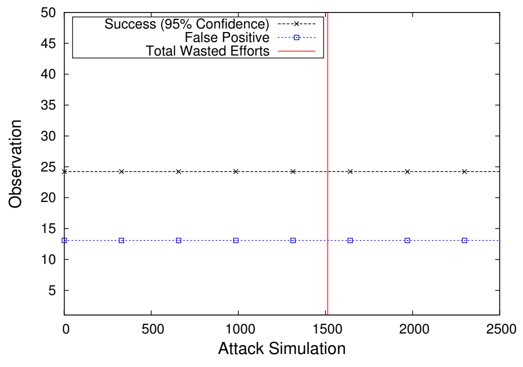
(c) ≈ 50% of Capacity



(d) ≈ 75% of Capacity



(e) ≈ 90% of Capacity



(f) Aggregated Results of 2500 Rounds

Figure 3.5: Attack Simulation Results. It is observable that an increase in the traffic ratio has a direct impact on the decrease of the successful attacks (i.e. the black + signs) and the increase of false positive results (i.e. the red × signs).

attacker is required to observe the mix before reaching a correct estimation.

- The red vertical lines show the total number of wasted efforts, i.e. false positives, relative to the total number of attack simulations.

Observing the outcome of the attack simulations leads to a number of interesting conclusions.

1. *Not every purported success is a true one.* The results show that a notable number of attacker's attempts lead to an incorrect value of n_1. The previous examination of Binomial Mix did not distinguish between the true successful attempts and the false positives [24] and the reported success rate is thus assumed to constitute only correct estimations. The recognition of the false positive values is important because it shows that the mix is significantly more resistant towards the passive attack than previously reported.

2. *The higher the traffic ratio, the more false positives.* Comparing the results of different scenarios shows that an increase in the ratio of the incoming traffic volume to the capacity of the mix, and especially where higher than 25%, leads to a decrease in the number of successful attacks an increase in the number of false positive results. This indicates that an increase in the incoming traffic ratio leads the attacker to believe that he is making better guesses, whereas in fact he is more likely to receive more false positive results.

3. *No failure, regardless of ratio of the traffic volume.* It is interesting to observe that the attacker will never fail to find an estimated value with 95% confidence. However, it can also be observed that an increase in the ratio of the traffic volume results in a lower number of required observations before the 95% confidence is achieved. Knowing this behaviour of the attack, the attacker must discard the results obtained from the attack found with a traffic ratio of higher than 25%. As we shall see in Section 3.5.2, these results can be improved by taking into account other factors.

4. *The lower the traffic ratio, the more observations required.* The earlier examination of the attack reported that the attacker requires approximately 200 observations to estimate the value of n_1 with 95% confidence [24]. Our experiments show that the number of required observations is, in fact, a function

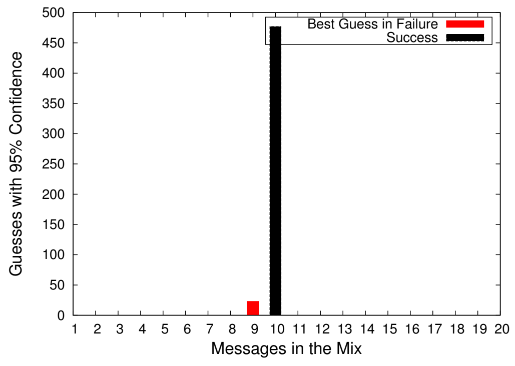
of the ratio of the incoming traffic to the capacity of Binomial Mix and thus may vary significantly in different traffic conditions. Where the incoming traffic is approximately equal to 10% of the capacity of the mix, the attacker may require up to 179 observations to reach an estimation; whereas with traffic equal to approximately 25% of the capacity of the mix, the attacker may require only up to 53 observations. This includes the correct guesses as well as the false positives.

5. *Approximately 60% of all the efforts are wasted.* If we combine the results obtained from all the attack simulations, it would be a total 2500 rounds. As shown in Figure 3.5(f), the total wasted efforts are $\frac{1513}{2500}$. That is, assuming a uniform randomness of the traffic ratio during the attacker's observations, approximately equal to 60% of the attacker's attempts. This is important as it signifies the low certainty in the findings of the passive attacker which is notably lower than was previously reported. Figure 3.5(f) also shows the average rounds of observation in successful attacks, i.e. 24.22, and in the false positives, i.e. 13.08.

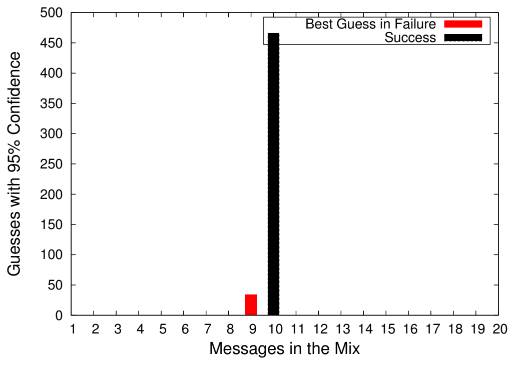
3.5.2 Attacker's Best Guess in Failure

The results presented in the previous section demonstrate that a large proportion of the attacker's attempts will lead to misleading results, and that the passive attack is thus not as effective on Binomial Mix as previously believed. In this section we pay closer attention to the results the attacker obtains in false positive guesses, and contrast them with what he must have found had the attack been successful.

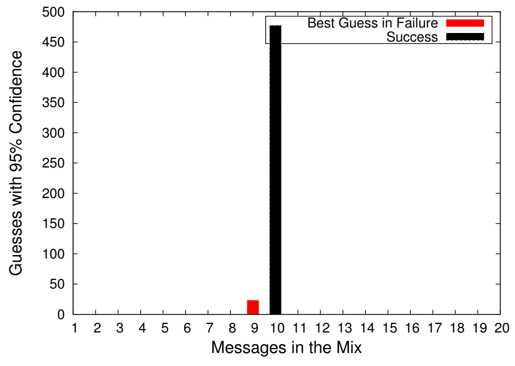
We seek to understand how far the wrong guesses usually are from the true value of n_1. To this end, we have assumed a constant designated value $n_1 = 10$ to be able to collect simulation results that allow aggregation. Note that we have executed the simulation with purely random values assigned to n_1 and reached consistent results with what was reported in Section 3.4. We then assumed a constant value for n_1 to be able to find the frequency distribution of the wrong guesses.



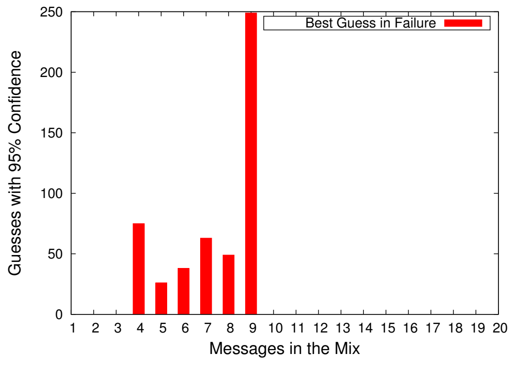
(a) ≈ 10% of Capacity



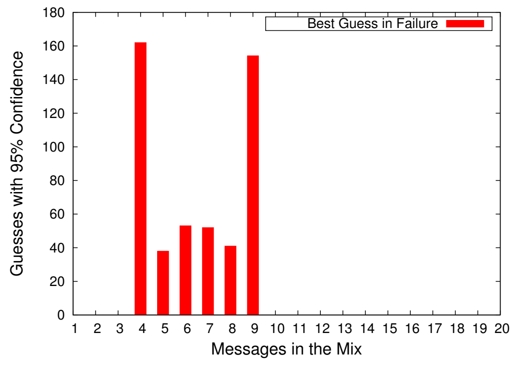
(b) ≈ 25% of Capacity



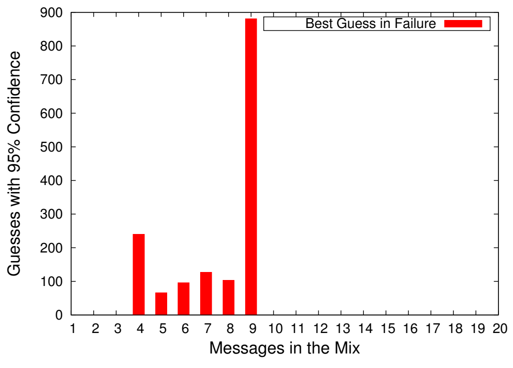
(c) ≈ 10% of Capacity



(d) ≈ 75% of Capacity



(e) ≈ 90% of Capacity



(f) Aggregated Results of 2500 Rounds

Figure 3.6: Attacker's Best Guess in Failure. The red vertical bars show the distribution of falsely identified results, and the black vertical bars show the successful attack results.

Figures 3.6(a)–(e) show the frequency distribution of the attacker's estimates for three of the examined scenarios. In every chart, the sum of the frequencies is equal to the total number of simulated attacks, i.e. 500. The black bars represent the number of correct estimates which, consistent with the designated value, always appears in the centre of the chart. The red bars represent the frequencies of estimates wrongly pointing to certain other values. Figure 3.6(f) shows an aggregated report of only the wrong guesses in all the 2500 rounds of experiment.

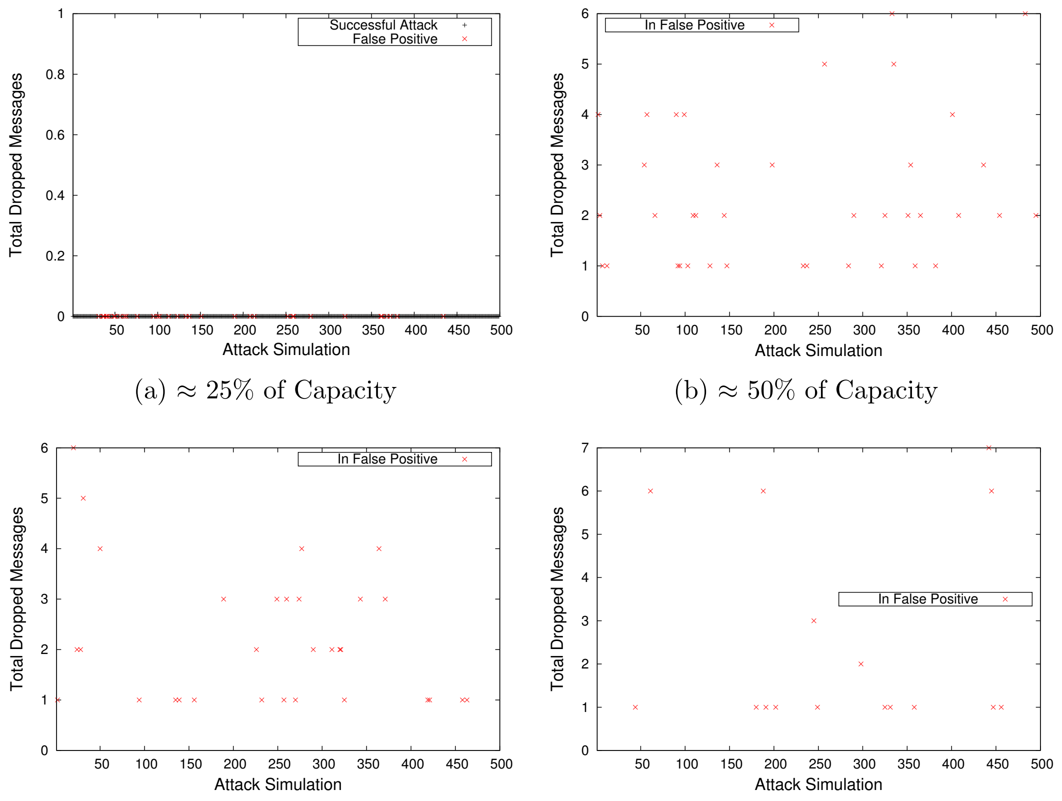
Our observations and findings are as follows:

1. *Wrong estimates are mostly very close to the true value.* When the traffic ratio is between 10% and 25% of the capacity of the mix, the wrongly estimated values are all equal to nine; that is, the closest possible lower estimate. This pattern persists with higher traffic ratios, albeit other wrong estimates also appear in the results. As shown in Figure 3.6(d), a significant portion of the total wrongly estimated values is only one unit less than the correct estimation.

2. *The higher the traffic ratio, the further the wrong estimates are from the true value.* As the traffic volume increases, the range of the wrong estimates widens in a non-uniform manner. The attacker's wrong guesses are never higher than the designated value; namely, in our examples no result fell within the range of 11–20. Neither do the wrong estimates ever fall below half of the designated value. That is, in our example no result is less than 5. Therefore, it can be concluded that the wrong estimates only fall within the range of $[\frac{n_1}{2}, n_1 - 1]$.

3.5.3 Dropped Messages

According to Equation (3.4), the maximum capacity of the mix (i.e. N_{max}) affects the estimates that the attacker computes. As in our experimental scenarios the λ of the Poisson distribution of the incoming traffic is a fraction of the N_{max}, it is conceivable that in high-traffic scenarios the number of incoming messages A, or the sum of the incoming messages and the ones still in the pool (i.e. $A + N$), exceeds N_{max}. In these cases the mix is forced to drop the excessive messages. This accords with real



(a) ≈ 25% of Capacity

(b) ≈ 50% of Capacity

(c) ≈ 75% of Capacity

(d) ≈ 90% of Capacity

Figure 3.7: Messages that the Mix Dropped

life scenarios because it cannot be guaranteed that the capacity of the mix is always larger than the sum of the messages in the pool and the incoming messages.

It is important to note that the attacker can only observe the incoming A and outgoing S messages, but the number of messages inside the mix N is unknown to him. Hence, he cannot tell which incoming message, if any, was dropped and which made it to the pool and may thus appear among the outgoing messages. It is also important to note that Binomial Mix uses the CDF function as the bias of the binomial distribution [24] and, consequently, the number of dropped messages is expected to be very low.

Analysis of the data generated by the simulator indicates that the mix does not drop any messages with a traffic ratio of less than 25% of the capacity of the mix. In higher traffic conditions, as shown in Figures 3.7(a)–(c), the number of dropped

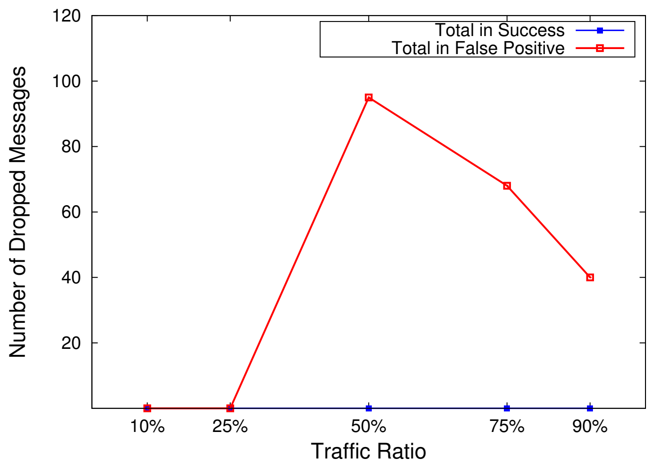


Figure 3.8: Aggregated Data on Traffic Ratio vs Dropped Messages

messages varies between 1 and 7.

3.5.4 Impact of the Dropped Messages

As discussed in Section 3.5.3, employing the CDF as the bias of the selection algorithm leads to dropping some messages during heavy traffic conditions. In this section, the overall impact of the amount of dropped messages on the success rate of the attack is analysed.

We have aggregated the total number of dropped message in all 500 simulation rounds for every traffic ratio and correlated the summations with the success rate of the attacks. The result is shown in Figure 3.8.

It is important to observe that *successful attacks occur only when there is no dropped message*. As Figure 3.8 shows, the number of dropped messages in all the successful attacks is equal to zero regardless of the incoming traffic volume ratio. It follows that a correlation exists between the number of dropped messages and the success rate of the attack.

The properties of the CDF does not allow a large number of dropped messages, but using a different bias function in the mix may result in a much higher loss of the messages, and thus a less reliable communication compared to the results shown in Figure 3.8.

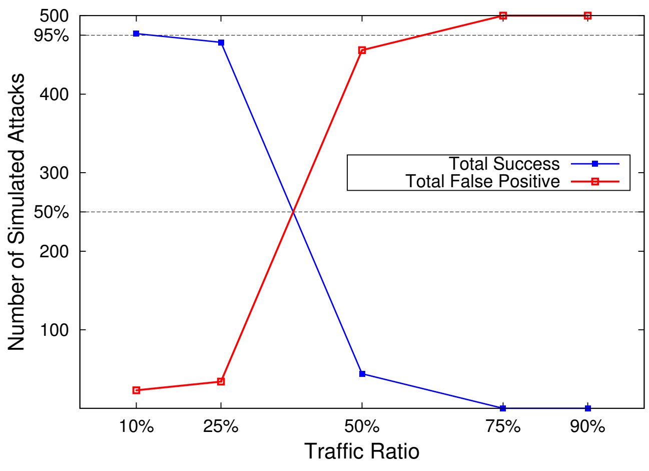


Figure 3.9: Aggregated Data on Traffic Ratio vs Attack Success

In the ACSs that use dummy traffic to further obfuscate the channels, network routers may generate and/or drop the dummy messages. Our results indicate that in such ACSs the passive attack will not produce any results for the attacker.

In order to build a more effective passive attack, the shortcoming of the attack in presence of dropped messages should be addressed. One possible solution would be to estimate the number of dropped messages based on the traffic volume. Nonetheless, a precise estimation, and generally adding other possible computations, imposes an extra processing cost on the attacker which can be prohibitive.

3.6 Effectiveness of the Attack on Binomial Mix

We have simulated 2500 attacks and an aggregated set of data about the success rate, as shown in Figure 3.9, reveals interesting information about the behaviour of the attack.

As mentioned in Section 3.5.1, approximately 60% of the attacker's attempts are wasted, which, as shown in Figure 3.9, mostly occur when the traffic ratio is more than 50% of the capacity of the mix.

It is interesting to observe that there is a drastic fall in the attack's success rate as the traffic volume grows from the ratio of 25% to 50%. A traffic volume equal to 37.50% of capacity of the mix results in an equal number of correct estimates and false positives.

It can also be observed that the 95% confidence in the correct results only exists where the traffic ratio is below 13.75% of the capacity of the mix. With a traffic ratio above 75%, the attacker does not have any realistic chance of making any correct estimation.

According to the stated observations, it can be concluded that the attacker can confidently rely on the results from the passive attack only when the traffic ratio is below 13.75% of the capacity of the mix.

In Section 3.5.2 we noted that when the traffic ratio is below 50%, the attacker can still be reasonably confident that the wrong estimations of n_1 are mostly one message less than the true value. Combining this knowledge with the results shown in Figure 3.9 leads to the conclusion that the attacker can be reasonably confident about the outcome of the attack as long as the traffic ratio is below 50% of the capacity of the mix. As the traffic ratio grows, however, so does the range of possible wrong estimates, and therefore the knowledge about the probability distribution of wrong estimates becomes less useful.

3.7 Possibilities for Further Research Work

In addition to our examination of Binomial Mix, explained in this chapter, further possibilities for research work in this area may be explored. In particular, we think that our work can be extended in at least in three ways.

- *Non-Poisson distributed incoming traffic.* In our experiments, a Poisson distribution is used to generate the simulated incoming traffic. This approach is widely accepted in the field and is also aligned with the experiments of the original Binomial Mix proposal [24]. Nevertheless, analysis of the traffic of two practical mix designs revealed that such an assumption may not be entirely correct [95]. It is therefore desirable to examine the effectiveness of the passive attack when the incoming traffic is distributed in a non-Poisson manner.

- *Randomness with other bias functions.* As stated earlier, the original design of Binomial Mix uses the normal CDF to determine the bias of the Binomial distribution. Other closely related designs were proposed: e.g. Binomial+, Logarithmic, and Square Root mixes. These differ from the original design mainly in the function used to determine the bias of the distribution [175]. It would be interesting to examine how effective the passive attack is on these designs.

- *Improving the attack algorithm.* In this chapter we focused on studying the success rate and the behaviour of a known attack. Given the results presented here, one might be able to improve the attack explained in Section 3.3.2 in order to produce a more efficient and effective attack.

3.8 Summary of the Chapter

In order to answer *Research Question 1* of this thesis (see Section 1.4), we have revisited and re-examined Binomial Mix, which was the first design to introduce randomness to the operation of a mix.

We built a Java-based simulator to analyse the effort of a passive attacker in breaking the anonymity provided by the mix. In particular, we focused on studying the impact of the incoming traffic volume as a function of the capacity of the mix. Five traffic volumes relative to the capacity of the mix were considered: 10%, 25%, 50%, 75% and 90% of the maximum capacity. In each traffic condition the rate of success of the passive attack was measured.

It was shown that the relative increase of the incoming traffic ratio significantly reduces the attacker's ability to launch successful passive attacks. However, we also found that the passive attack proposed in [24] never results in failure. Instead, it results in up to 60% false positive results. It follows that the known passive attack is not as robust as previously thought and that consequently Binomial Mix is much more resilient towards such attacks.

In spite of the existence of a high false positive rate, the attacker can improve the attack algorithm in order to achieve better results by taking into account the probability distribution of the false positives. This is possible because the wrong results always fall within a predictable range of values. This technique is effective for incoming traffic ratios of below 50% of the capacity of the mix.

It was also shown that a Binomial Mix which uses the normal CDF as its bias function will be forced to drop messages in heavy traffic conditions. We showed that the passive attack can only be successful where there are no dropped messages.

The analysis and experimentation presented in this chapter can be extended to consider other possible scenarios. One may consider non-Poisson distributed incoming traffic to examine other network traffic conditions. It would also be interesting to discover the behaviour of other bias functions (i.e. other than the CDF) towards the passive attack. Another interesting possibility is to leverage the findings of this research to build a more efficient attack algorithm.

In Chapter 4 we will focus on other aspects of this mix. Specifically, we will address the need for a higher efficiency and improve the design in order to make it resilient towards *active attacks*.

Chapter 4

Multi-Binomial Mixes

If the right to privacy means anything, it is the right of the individual, married or single, to be free from unwarranted governmental intrusion.
—Justice William J. Brennan

4.1 Introduction

The focus of this chapter is to answer *Research Question 2* of this thesis; that is, we aim to improve the efficiency and security of Binomial Mix design.

As noted in Chapter 2, Binomial Mix was the first design of a *mix* that used randomness in its operation. Consequently, the attacks on Binomial Mix can succeed only probabilistically. In Chapter 3 we examined that design against probabilistic passive attacks in various traffic conditions and noted its strengths. Nevertheless, choosing an appropriate function to specify the bias of the Binomial distribution used in the mix is challenging. There is a trade-off between the degree of anonymity a Binomial Mix can achieve and the delay it imposes on the communication channel.

The bias function used in the original design, for example, is the normal Cumulative Distribution Function (CDF) [24]. That function offers a desirably low delay but an undesirably low degree of anonymity in heavy traffic conditions [175]. Moreover, employing one bias function, and particularly when it strongly correlates with a publicly known parameter (i.e. the traffic volume parameter of the CDF), provides the adversary with credible information which forms the foundation on which various types of attacks can be built. We further elaborate on this issue in Section 4.4.

Another interesting aspect to consider is that Binomial Mix treats all the passing traffic equally and, consequently, low-latency channels suffer significantly more from the delay caused by the anonymisation process. Depending on the bias function, the delay may render the mix unusable for low-latency channels.

In this chapter we propose two new mix designs which extend the design of Binomial Mix to address these shortcomings. We name the first design Multi-Binomial Shared-Pool Mix (MBSP Mix) and the second design Multi-Binomial Independent-Pool Mix (MBIP Mix). Both these traffic mixers are built on the notion of using not one but multiple message selection algorithms. Each selection algorithm relays the messages randomly according to an independent Binomial distribution with potentially different bias functions. The following illustrate the advantages of this approach.

- Less credible information is made available to the adversary, and the mix is thus more resistant to both active and passive attacks.
- The shortcomings of one particular bias function (e.g. the delay and degree of anonymity) can be compensated for by the coexistence of multiple functions.
- In the case of MBIP Mix, a single mix may concurrently anonymise both high- and low-latency traffic channels.

Organisation The rest of this chapter is organised as follows. The following section defines the *threat model* used in this chapter. Section 4.3 presents further details about Binomial Mix and the related attacks. In Section 4.4 we discuss why making a less predictable mix increases the degree of anonymity, and how this may be achieved. Sections 4.5–4.6 respectively present the designs of MBSP Mix and MBIP Mix. Each of these sections also provides analyses of the predictability of the mix's behaviour, its traffic delay and its resistance towards the passive attack and the blending attack. Lastly, Section 4.7 presents a summary of this chapter.

4.2 Threat Model

We assume an active attacker as well as a passive attacker. That is, an adversary can observe all the incoming and outgoing traffic of the mix and knows all the internal parameters. He can also actively tamper with the traffic; that is, add, remove, delay or modify messages. However he can only carry out external attacks: he cannot observe the internal traffic flow of the mix in question and does not know the parameters dynamically set through the encrypted negotiations with other network nodes. We also assume that the attacker does not have *a priori* or contextual information about the correlation between the incoming and outgoing messages.

4.3 Background

The operation of Binomial Mix was described in Section 3.3.1. Since this chapter presents new mix designs which extend the design of Binomial Mix, in this section we recap the main related issues.

4.3.1 Binomial Mix

Binomial Mix contains only one selection algorithm. In each flushing time, the mix tosses a coin for each message in the pool to decide if that message should be relayed to the next node in the network. The next node may be another traffic mixer or the final destination of the message. The other the messages are kept in the pool to be tried in the next flushing iteration. A schematic of the main elements of Binomial Mix is shown in Figure 4.1.

The coin is biased so that messages will not be stalled in the pool for an unduly long period. The bias of the coin, and consequently the number of rounds messages may remain in the pool, is specified by a bias function which, naturally, has a direct impact on the traffic delay caused by the mix in various traffic conditions. As we shall see in the following sections, the bias function also has a direct impact on the degree of anonymity that a mix can achieve.

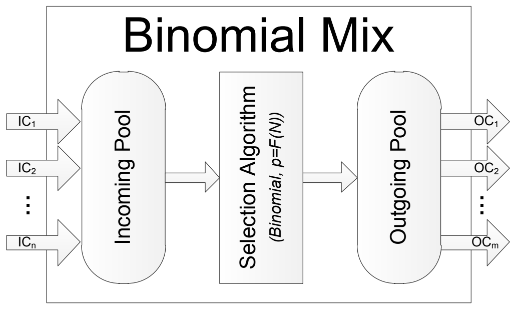


Figure 4.1: Schematic of Binomial Mix

The proposal of the Binomial Mix design used the CDF of normal distribution as a potential bias function [24]. That is, $\Phi(N)$, where N is the number of messages within the incoming pool in each flushing iteration. Using this function in low traffic conditions results in a lower probability of success and thus a higher traffic delay and a higher degree of anonymity. In heavy traffic conditions the normal CDF achieves the opposite results [175].

4.3.2 Passive Attack on Binomial Mix

Due to the probabilistic nature of Binomial Mix, attacks on that mix can succeed only probabilistically. In other words, the active attacker observes the incoming channels (i.e. $IC_1 \cdots IC_n$ in Figure 4.1) and outgoing channels (i.e. $OC_1 \cdots OC_m$), and attempts to break the anonymity of the mix by correlating the incoming and outgoing messages through probabilistic techniques.

The details of the passive attack were presented in Section 3.3.2. Here we adopt the same terminology (see Table 3.1) to only reiterate the main goals of a passive attacker.

The attacker can observe the number of incoming messages at the start of each round (A) as well as the number of outgoing messages (S). He aims to learn the number of messages in the pool (N) with some degree of certainty. If he can compute the number of messages in the pool in the first round of the attack, namely n_1, he will be able to break the anonymity provided by Binomial Mix.

According to the threat model (see Section 4.2), the attacker does not have any *a priori* knowledge. This implies that in the first round of attack the existence of any number of messages (i.e. between 0 to N_{max}) in the pool is equally probable. Hence, the probability of n number of messages being in the pool given the observation of s number of messages leaving the mix can be computed according to the following equation.[1]

$$P(N = n|S = s) = \frac{P(s|n)}{\sum_{i=s}^{N_{max}} P(s|i)} \tag{4.1}$$

The attacker requires multiple observations before he can compute a value for n_1 that maximises Equation (4.1). The attacker needs to combine the knowledge learnt from each observation as explained in detail in Section 3.3.2.

4.3.3 Blending and $(n-1)$ Attacks on Binomial Mix

An active attacker not only can see the messages passing through a mix but also can manipulate them. That is, to insert arbitrary messages, or modify, remove or delay the existing ones. One type of active attack, known as the *blending attack*, has proved to be notoriously difficult to foil. Most existing systems cite this attack or the $(n-1)$ *attack*, which essentially is one instance of the blending attack and is the most difficult to counter, as a vulnerability [92]. This attack was discussed in Section 2.5.5. As we will use this attack in our evaluations in this chapter, the following presents a recap.

In the $(n-1)$ attack, the attacker aims to compromise the anonymity of a particular message M_t by tracing it through the mix. There are two phases in the attack: the *emptying phase* and the *flushing phase.* During the *emptying phase* the attacker prevents any unknown message from entering the mix while also flooding the mix with a large number of self-generated, and thus identifiable, messages. This step is meant to force the mix to flush nearly all of the unknown messages already existing in its pool.

Secondly, during the *flushing phase*, the attacker allows the message M_t to enter the mix followed by n attacker-generated messages (and hence the name $(n-1)$ attack) while still preventing unknown messages from entering the mix. Once M_t leaves the mix, the attacker will be able to identify it and compromise its anonymity.

[1]For more information about how this equation is derived, refer to the detailed discussion in Section 3.3.2.

It was shown in [176] that, despite the aim of Binomial Mix to reduce the knowledge available to the attacker about the number of items in the pool, the mix is still vulnerable to the blending attacks. For a detailed analysis of the resistance of the currently available mix designs towards the blending attack, the curious reader is referred to [176].

4.4 Towards Reducing the Adversary's Knowledge

It is material to examine the underlying knowledge and assumptions that enable an adversary to carry out attacks on Binomial Mix. It must be noted that the design of Binomial Mix is publicly known and the attacker thus knows that:

1. there is a fixed and periodic flushing time;
2. the selection algorithm relies on randomly choosing messages from the incoming pool;
3. the randomness is according to a Binomial distribution;
4. the randomness is not fair;
5. the mix has a certain and known capacity (i.e. N_{max}); and
6. the bias function is strongly correlated with the number of incoming messages and in accordance with the normal CDF (or other potentially suitable bias functions such as Binomial+, Logarithmic, and Square Root mixes [175]).

This knowledge is the cornerstone of the attacks on Binomial Mix such as the one described in Section 4.3.2. Specifically, the knowledge items 1–3 above allow building an attack according to the known properties of the Binomial distribution; that is, the computations shown in Equation (4.1). The knowledge items 4–6 above inform the attacker that in certain conditions the mix may be offering a significantly lesser degree of anonymity, such as where the normal CDF results in a very low degree of anonymity under heavy traffic conditions as described in [175]. Such known properties may be exploited to improve existing or to build new attacks.

An important question to consider is whether the degree of knowledge available to the attacker could be *reduced*. This could be naively achieved by a design with inherent properties not readily available to the attacker, which in effect is to achieve *security through obscurity*. This approach is undesirable because the security of the traffic mixer will be compromised once the attacker learns the obscured properties.

A more robust approach would aim to remove or weaken the aforesaid knowledge items not by obscuring the design properties but by building properties with inherent dynamic behaviour so as to remove the grounds for such fundamental assumptions. For example, had Binomial Mix been designed to use not a fixed bias function but a function randomly picked from a set of suitable functions, the attacker would not have been able to make any firm assumptions about the known weaknesses of the used function. Further, had the bias function been chosen differently from one flushing time to another, he would not have been able to, or would have gained very little additional information to be able to, improve his knowledge about the bias function by observing multiple iterations. Naturally, each of these approaches has associated implications in terms of traffic throughput and delay which warrant careful consideration.

In the rest of this chapter we consider certain scenarios that decrease the knowledge available to the attacker by studying new variations of Binomial Mix.

4.5 Multi-Binomial Shared-Pool Mix (MBSP Mix)

MBSP Mix, shown in Figure 4.2, employs n Binomial distributions each of which have an independent and potentially differing success probability value denoted by $x_1, x_2 \ldots x_n$.

In each flushing time, an MBSP Mix randomly chooses one selection algorithm and tosses a coin for each message in the incoming pool. The coin is biased according to the bias functions associated with the Binomial distribution of the chosen selection algorithm. The selected messages are then relayed.

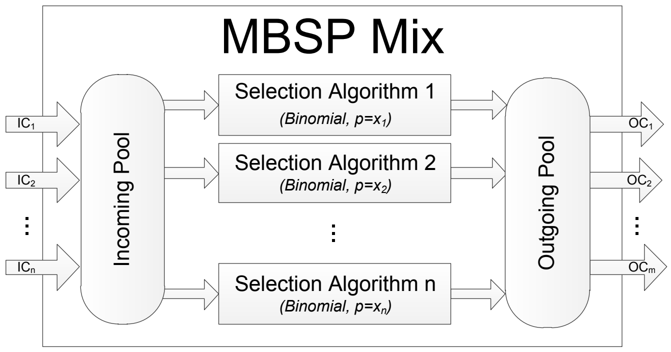


Figure 4.2: Schematic of Multi-Binomial Shared-Pool Mix (MBSP Mix)

4.5.1 Output Predictability

The original Binomial Mix design proposed to use the normal CDF to determine the bias of the coin. Thus, from the attacker's point of view, the probability of a particular message leaving the mix, $P(S)$, was independent of any other parameter except the number of messages in the pool. In the case of MBSP Mix, however, the mix may use a different bias function in each flushing time which may or may not have a correlation with the number of messages in the pool.

In MBSP Mix, the value of $P(S)$ in each flushing time may be equal to either of the values within the range $x_1 \cdots x_n$ depending on which selection algorithm (denoted by C) is in operation. The probability of a particular message leaving the mix is a function of one particular selection algorithm C_α being in operation and of its associated probability of success:

$$P(S) = P(C_\alpha) \cdot x_\alpha \tag{4.2}$$

MBSP Mix can randomly choose any available C, and the value associated with $P(C_\alpha)$ is thus determined based on a distribution which specifies how likely it is for a particular C to be selected in any given flushing time. Assuming that every C is equally likely, for n selection algorithms we have:

$$P(S) = \frac{1}{n} \cdot x_\alpha \tag{4.3}$$

More generally, from an external observer's point of view (such as an attacker) the

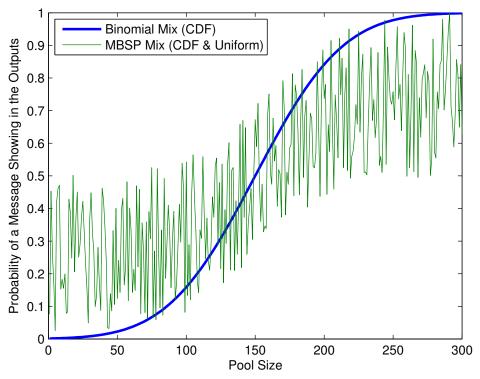


Figure 4.3: Performance of MBSP Mix (CDF and Uniform)

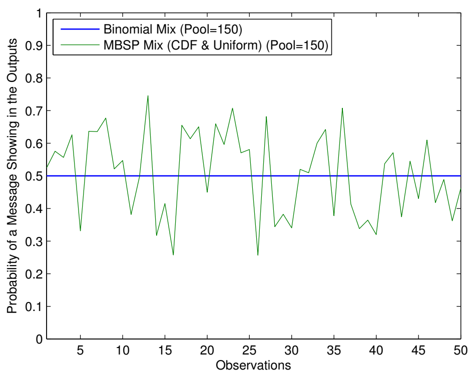


Figure 4.4: Attacker Observations of MBSP Mix (CDF and Uniform)

average probability of a particular message leaving the mix is:

$$P(S) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.4}$$

In order to understand the behaviour of MBSP Mix, we have compared its output predictability with that of Binomial Mix in a number of different configurations. We have assumed a Binomial Mix uses the CDF as its bias function, which is the function used in the original design of Binomial Mix. We have also assumed that all the available selection algorithms are equally likely to contribute to the output.

Figure 4.3 shows the predictability of the output of an MBSP Mix containing two selection algorithms: i) one which uses the CDF bias function, and ii) one which relies on a Uniform distribution to determine the bias.

It is observable that while MBSP Mix still relays a higher number of messages in high-traffic conditions, it behaves much less predictably than Binomial Mix. The unpredictability can be seen more easily in Figure 4.4 where multiple observations of the attacker are contrasted every time the message pool has a certain size (i.e. when 150 messages are in the pool). While Binomial Mix's expected behaviour remains the same in all the observations, MBSP Mix's behaviour is far less predictable.

It is seen that the MBSP Mix in Figure 4.3, in contrast with Binomial Mix, provides a higher degree of anonymity in heavy traffic conditions. This is because in heavy traffic conditions Binomial Mix simply relays the entirety of the incoming

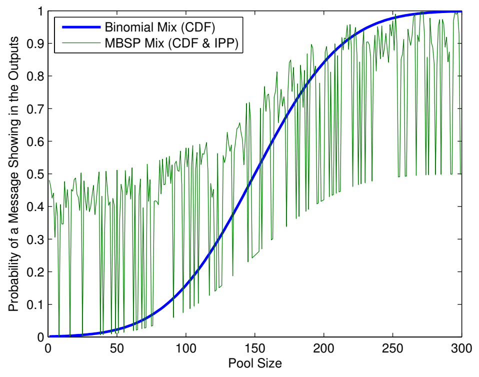


Figure 4.5: Performance of MBSP Mix (CDF and IPP)

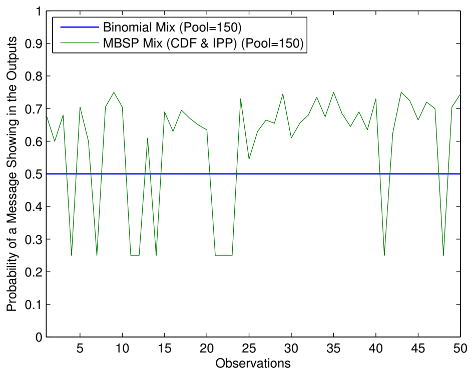


Figure 4.6: Attacker Observations of MBSP Mix (CDF and IPP)

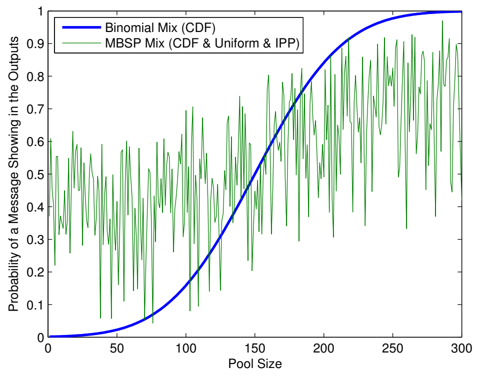


Figure 4.7: Performance of MBSP Mix (CDF, IPP and Uniform)

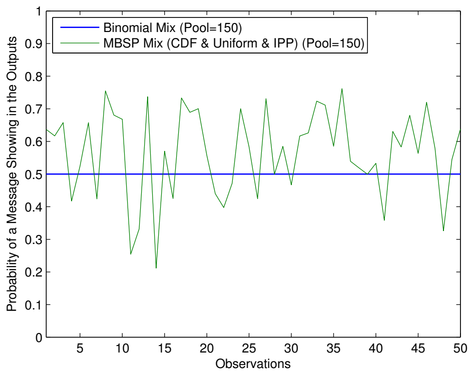


Figure 4.8: Attacker Observations of MBSP Mix (CDF, IPP and Uniform)

traffic and thus provides a significantly lower degree of anonymity. It is important to note that the MBSP Mix also imposes a lower latency in the low traffic conditions because the messages do not have to wait for a minimum threshold to be met.

Figures 4.5 and 4.6 show a similar scenario where an MBSP Mix consists of two selection algorithms: i) one which employs a Binomial distribution with a normal CDF as its bias function; and ii) one which employs an Interrupted Poisson Process (IPP). In this example, the IPP is interrupted 30% of the time and, when active, selects approximately 85% of the messages to be flushed. The traffic resulting from an IPP resembles the usual Internet traffic. As shown in Figure 4.5, using an IPP-based selection algorithm yields a notably less predictable behaviour pattern in comparison with Binomial Mix.

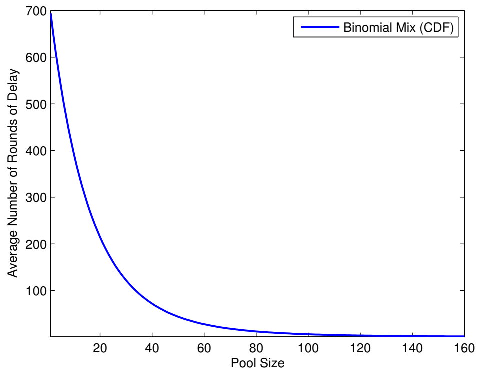


Figure 4.9: Average Message Delay in a CDF-Based Binomial Mix

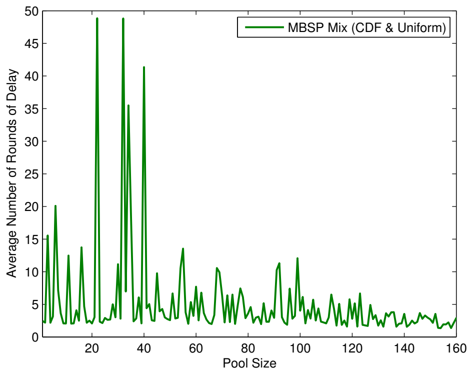


Figure 4.10: Average Message Delay in MBSP Mix (CDF and Uniform)

The third scenario shows an MBSP Mix that contains a mixture of the two previous scenarios. It now contains three selection algorithms: i) one which employs a Binomial distribution with a normal CDF as its bias function; ii) one which employs an IPP; and iii) one which uses Uniform distribution.

The results can be seen in Figures 4.7 and 4.8. It is observed that the overall throughput of the mix has increased, that the behaviour of the mix does not resemble the behaviour of any one of the selection algorithms in isolation, and that a higher degree of anonymity is achieved in heavy traffic conditions.

4.5.2 Traffic Delay

The average number of rounds that a message may stay in a Binomial Mix is $\frac{1}{p}$. Here p is the probability of success in the Binomial distribution and is, in a CDF-based design, computed according to the CDF of the normal distribution. That is, $CDF(m)$, where m is the number of messages inside the pool. Hence, in Binomial Mix the average delay of a message is $\frac{1}{CDF(m)}$ number of rounds. Note that this is *relative* to the ratio of incoming traffic as a higher incoming traffic ratio would result in a larger m and thus a lesser delay.

In MBSP Mix, however, p consists not of the probability of success in one selection algorithm but of the combination of such probability in all the selection algorithms. Precisely, it consists of the probability of a particular

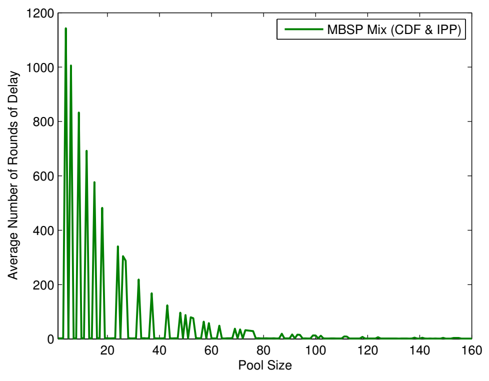


Figure 4.11: Average Message Delay in MBSP Mix (CDF and IPP)

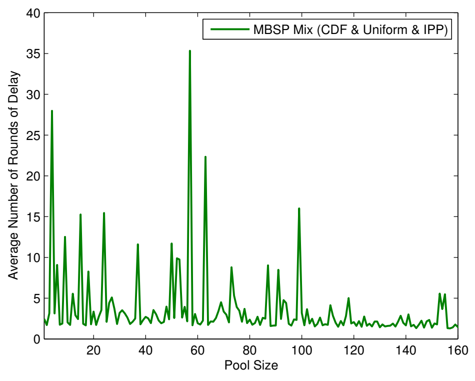


Figure 4.12: Average Message Delay in MBSP Mix (CDF, Uniform and IPP)

selection algorithm being in operation, $P(C_\alpha)$, and its associated probability of success x_α. Hence, in MBSP Mix with n selection algorithms, the average number of rounds that a message is delayed is:

$$\text{rounds} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} x_i} = \frac{n}{\sum_{i=1}^{n} x_i} \tag{4.5}$$

Figure 4.9 shows the average delay caused by Binomial Mix based on the pool size. Figures 4.10–4.12 show the average delay in the three MBSP Mix designs previously discussed in this section.

It is observed that two of the MBSP Mix designs, shown in Figure 4.10 and 4.12, cause significantly lower traffic delays in low traffic conditions. That is, approximately 50 rounds of delay in the worst case scenario as opposed to approximately 700 rounds in Binomial Mix.

A comparison between the two figures also reveals that a higher number of selection algorithms can lead to a lower traffic delay. It must be noted that the design shown in Figure 4.12 contains the same IPP which caused significant delays in the design shown in Figure 4.11. Nevertheless, the coexistence of a uniform distribution has compensated for the excessive delays.

4.5.3 Blending Attack

As stated in Section 4.3.3, a *blending attack* is perhaps the most notorious attack against mix systems. This section presents a comparison between the cost of that attack on the CDF-based Binomial Mix versus MBSP Mix. Specifically, we analyse the cost of attack in the *emptying phase* as well as in the *flushing phase*.

Number of Rounds Required in the Emptying Phase

We assume that during this phase the attacker sends N_T number of messages to the mix, and that the attacker chooses N_T such that it forces the mix's selection algorithm function to be equal to its asymptotic value, p_{asym}. This gives us an upper bound on how effective the attack can be.

We know from [177] that in Binomial Mix the number of rounds, k, necessary to flush all unknown messages in the mix with probability $1-\epsilon$ can be estimated using the following equation:

$$(1-(1-p_{asym})^k)^G \geq 1-\epsilon \tag{4.6}$$

where G is the number of unknown messages available in the mix. In the absence of *a priori* knowledge about the value of G, which is in accordance with the threat model (see Section 4.2), we consider the worst case scenario for the attacker where $G = N_{max}$, that is, assuming that at the beginning of the attack the mix is full of messages.

MBSP Mix allows multiple selection algorithms to operate in sequence, and each function has an independent asymptotic behaviour which may or may not correlate with the number of incoming messages. Therefore, in order to estimate the number of rounds necessary to flush the unknown messages in MBSP Mix with n number of selection algorithms, the Equation (4.6) must be changed to:

$$(1-(1-\frac{1}{n}\sum_{i=1}^{n} p_{asym_i})^k)^G \geq 1-\epsilon \tag{4.7}$$

which can also be expressed as:

$$k = \frac{ln(1-(1-\epsilon)^{\frac{1}{G}})}{ln(1-\frac{1}{n}\sum_{i=1}^{n} p_{asym_i})} \tag{4.8}$$

Contrary to the selection algorithms relying on the Binomial distribution, the value of p_{asym} cannot necessarily be computed for all possible selection algorithms. For instance, assume a design of MBSP Mix which contains two coexisting selection algorithms: i) a uniformly random function, and ii) a Binomial distribution with the CDF bias function. The asymptotic behaviour of the first selection algorithm does not correlate with the number of messages the attacker sends to the mix. Therefore, to analyse such an MBSP Mix designs we consider the entire range between the two possible extreme results of the uniformly random selection algorithm. In other words, we first consider the case where the selection algorithm chooses no message from the pool ($p = 0$), and secondly the case where it chooses the entire message pool contents ($p = 1$).

The comparison between the assumed MBSP Mix and the CDF-based Binomial Mix is shown in Figure 4.13. It is observable that the CDF-based selection algorithm will be flushed entirely in about one round of attack, which is similar to only the worst case of the MBSP Mix; that is, where $p = 0$. In other circumstances the MBSP Mix demands a significantly larger number of rounds of attack.

Number of Messages Required in the Emptying Phase

We know from [177] that the cost of the attack in terms of the number of messages that the attacker has to send to a CDF-based Binomial Mix can be computed as:

$$\text{Messages} = N_T + (k-1)(N_T + G)p_{asym} \tag{4.9}$$

As discussed earlier, in MBSP Mix the value of p_{asym} should be computed differently and therefore the number of messages that the attacker has to send to MBSP Mix is:

$$\text{Messages} = N_T + (k-1)(N_T + G)\frac{1}{n}\sum_{i=1}^{n} p_{asym_i} \tag{4.10}$$

Equations (4.9) and (4.10) can be used to compute the cost of the attack for the attacker in the *emptying phase*. The results are shown in Figure 4.14. In this example, the maximum capacity of the mix (i.e. N_{max}) is 300 messages.

We observed in Figure 4.13 that the CDF-based Binomial Mix requires only one round of attack in the *emptying phase* and, as shown in Figure 4.14, the number of

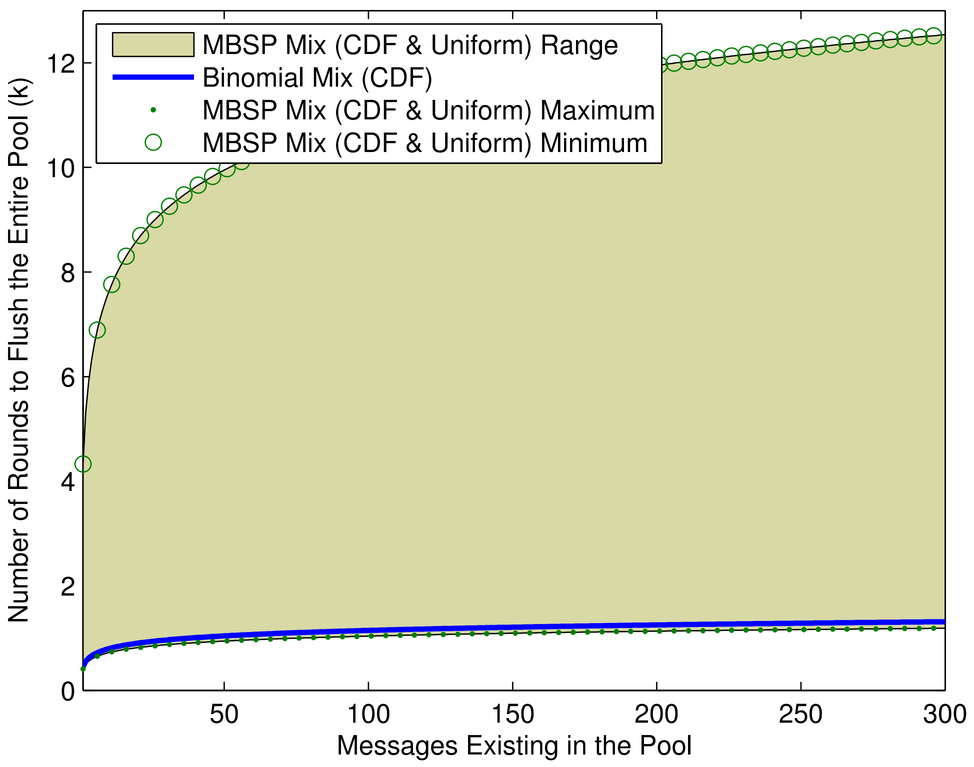


Figure 4.13: MBSP Mix, Rounds of Blending Attack in Emptying Phase

messages that the attacker has to send is approximately 300 messages. This resembles the worst case scenario in MBSP Mix. The highlighted area in the figure shows the range of the number of messages that the attacker has to send. It is observable that the cost of the attack for the attacker may increase up to approximately 20 times more than that of the CDF-based design.

Total Cost of the Attack

We know from [177] that the number of rounds to flush the messages from a CDF-based selection algorithm is $k' = \frac{1}{p_{asym}}$. In an MBSP Mix with n selection algorithms, k' changes to:

$$k' = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} p_{asym_i}} \tag{4.11}$$

Assuming that the attacker has chosen to empty the mix in k rounds and then flush the message in k' rounds, the total number of messages that the attacker must send

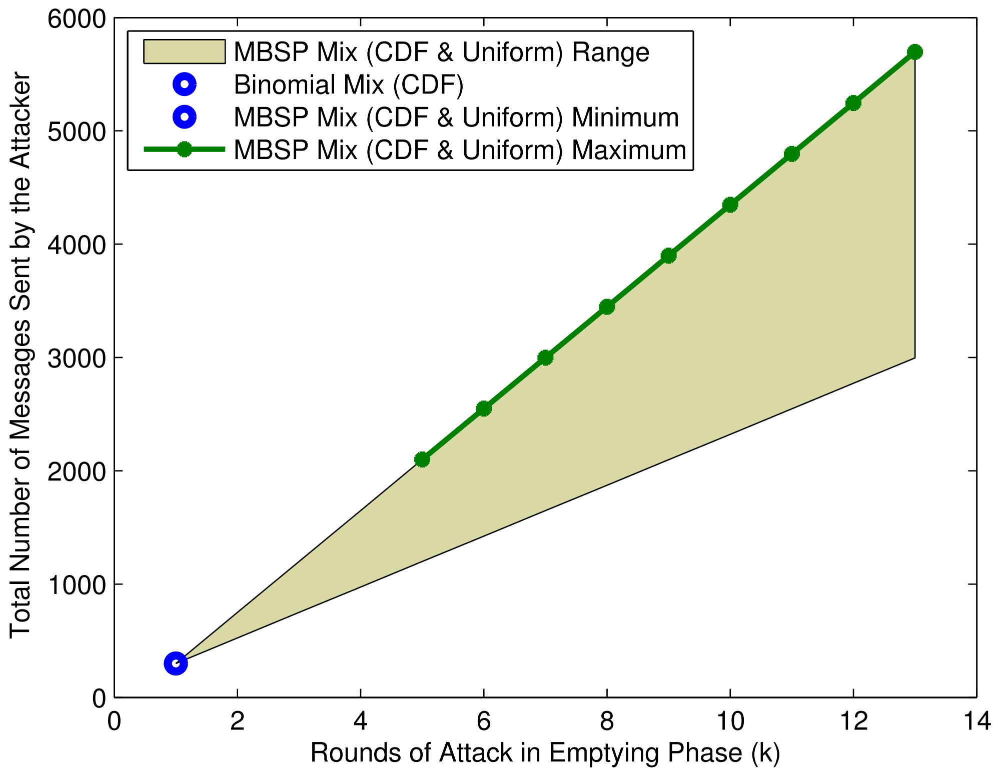


Figure 4.14: MBSP Mix, Messages in Blending Attack in Emptying Phase

to the mix is equal to:

$$\text{Messages} = (N_T + (k-1)(N_T + G) + (k'N_T + k'G - 1))\frac{1}{n}\sum_{i=1}^{n} p_{asym_i} \tag{4.12}$$

MBSP Mix is a Timed Mix (TM) and therefore the attacker theoretically needs to wait t units of time before each round. Hence, the total number of rounds that he must wait for the two phases of the attack is $(k + k')t$.

The total cost of the attack in terms of the messages that the attacker has to send to the mix is shown in Figure 4.15. The figure compares a CDF-based Binomial Mix and an MBSP Mix containing two selection algorithms: i) one based on the Binomial distribution with a CDF bias functions; and ii) one with a uniformly random algorithm.

It is observable that Binomial Mix has a lower cost for the attacker even if compared with the worst scenario of the MBSP Mix in which the random function

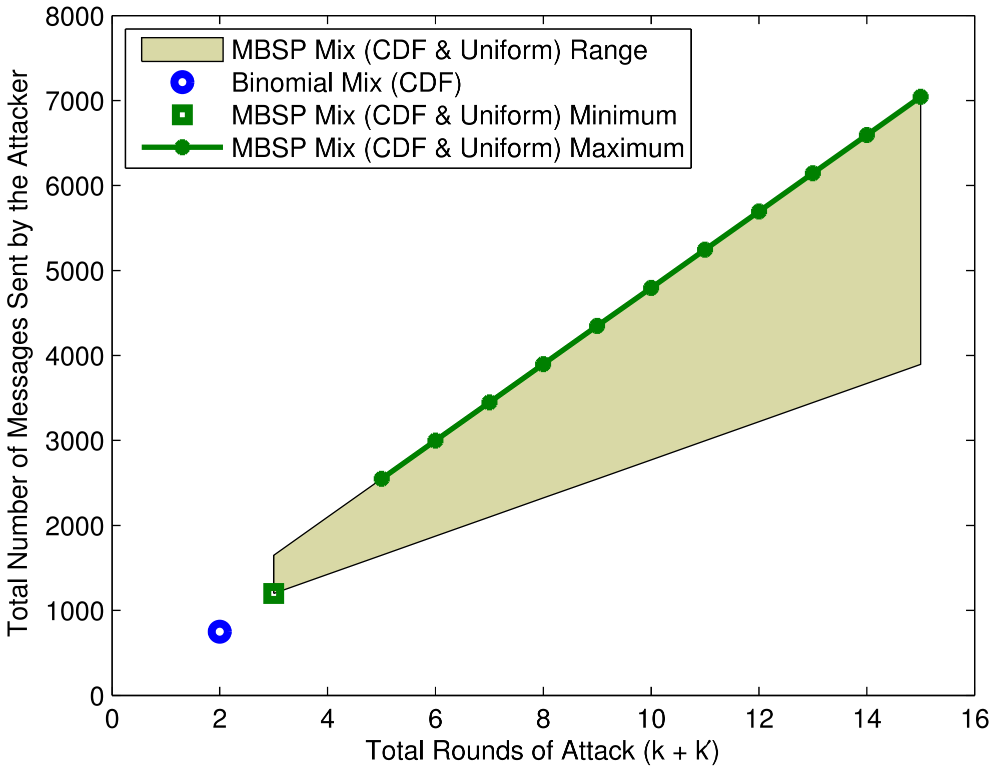


Figure 4.15: MBSP Mix, Total Messages in Blending Attack

selects the outputs with $p = 1$. The area highlighted in Figure 4.15 shows the range of the number of messages that the attacker must send depending on the assumed value of $k + k'$. It is shown that the total cost of the attack is approximately eight times more than that of the MBSP Mix.

4.6 Multi-Binomial Independent-Pool Mix (MBIP Mix)

MBIP Mix, as shown in Figure 4.16, differs from MBSP Mix in that each selection algorithm has its own separated pool of incoming messages.

In each flushing time, each selection algorithm decides the fate of every message in the incoming pool associated with that particular algorithm. Akin to MBSP Mix, the coin is biased according to the success probability associated with the Binomial distribution of the selection algorithm in question. The chosen messages are then relayed to the final destination or another mix.

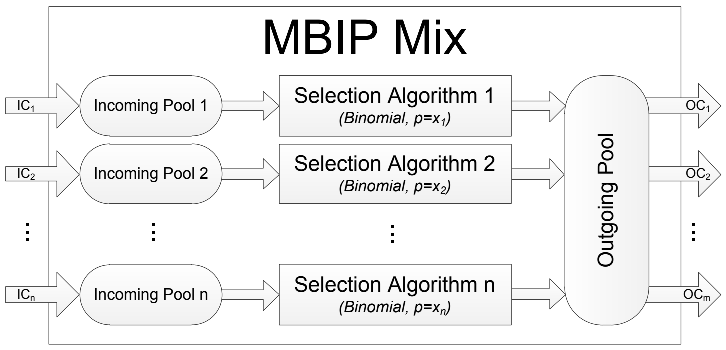


Figure 4.16: Schematic of Multi-Binomial Independent-Pool Mix (MBIP Mix)

4.6.1 Output Predictability

In every flushing iteration of MBIP Mix, the probability of a message leaving the mix, $P(S)$, is equivalent to the product of $x_1 \dots x_n$. As the attacker does not know to which *incoming pool* a message will be added, an outgoing message could thus, from his point of view, be the result of any selection algorithm. Therefore, from an external observer's point of view, the probability of a particular message leaving the mix, on average, is:

$$P(S) = \prod_{i=1}^{n} x_i \tag{4.13}$$

In order to understand the behaviour of MBIP Mix, we analysed four hypothetical designs which combine some well-known selection algorithms. In every scenario, the behaviour of the MBIP Mix is contrasted with that of a CDF-based Binomial Mix.

Figure 4.17 shows the expectation of the attacker from an MBIP Mix design which consist of two selection algorithms: i) one based on the Binomial distribution and the normal CDF bias function; and ii) one with a different bias function, namely the Timed Pool Mix (TPM).

Although the resultant combination shows a lower traffic throughput and a higher delay, it must be noted that the actual performance of the mix is not the same as the attacker's expectation. In fact, the mix has two separate communication channels that do not interfere with each other, and the performance of each depends solely on the traffic ratio of that particular channel. The attacker's expectation, which is

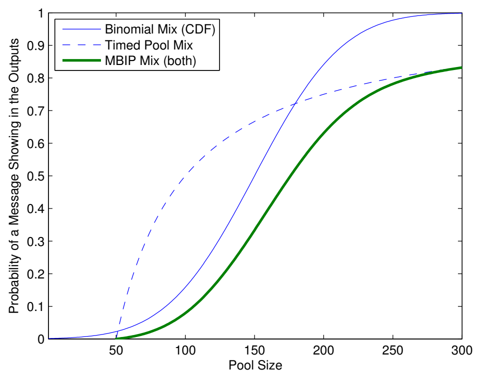


Figure 4.17: Attacker's Expectation from MBIP Mix (CDF and TPM)

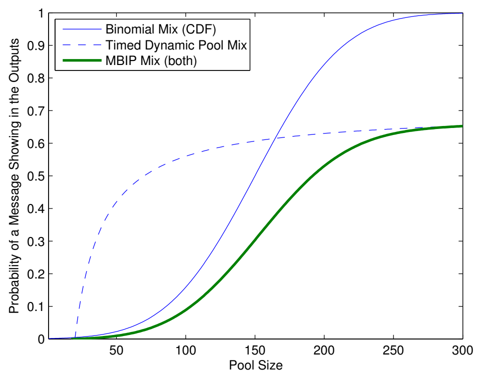


Figure 4.18: Attacker's Expectation from MBIP Mix (CDF and TDPM)

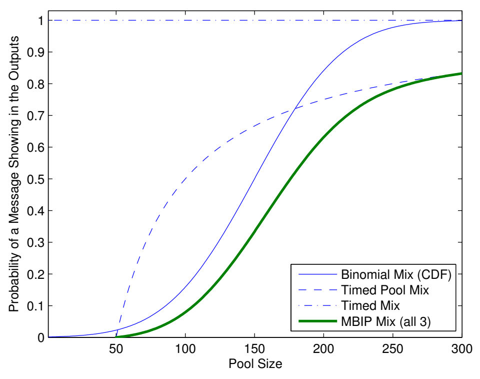


Figure 4.19: Attacker's Expectation from MBIP Mix (CDF, TPM and TM)

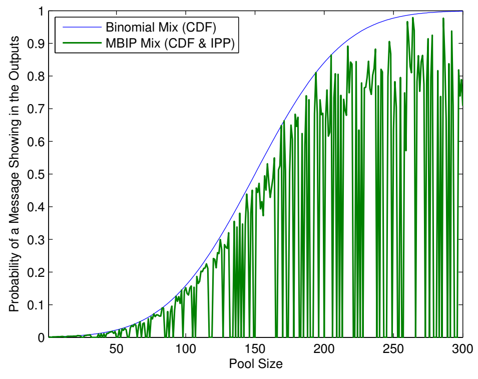


Figure 4.20: Attacker's Expectation from MBIP Mix (CDF and IPP)

computed according to Equation (4.13), demonstrates how far his guesswork could be from the actual behaviour of the mix.

Figure 4.18 shows a design of MBIP Mix whereby a CDF-based selection algorithm coexists with one based on Timed Dynamic Pool Mix (TDPM). It is observed that, due to the lower probability of success in TDPM versus the TPM (shown in Figure 4.17), the expectation of the attacker is further from the actual behaviour of the mix.

We have also simulated the expected behaviour of MBIP Mix where it contains three coexisting selection algorithms: i) the normal CDF; ii) a TPM; and iii) a TDPM. The results are shown in Figure 4.19.

Figure 4.20 shows that where at least one of the selection algorithms provides randomness, as provided by the IPP in this example, the attacker's expectation is even further from the actual behaviour of the mix.

4.6.2 Traffic Delay

Each message arriving at MBIP Mix will end up in one of the available *incoming pools*. Each pool has an independent probability of success associated with its selection algorithm $(x_1 \ldots x_n)$. Hence, the average number of rounds that a message stays in MBIP Mixdepends on the probability of success of the selection algorithm associated with the incoming pool in which a message is positioned. This may or may not be dependent on the traffic ratio and the current size of the pool. Hence, the average delay for each incoming channel $IC_1 \ldots IC_n$ can be computed as:

$$\text{Average rounds of delay for } IC_i = \frac{1}{x_i} \tag{4.14}$$

The independence between the communication channels means that the low- and high-latency communication channels could be anonymised using the same mix. This is a unique and important property of MBIP Mix and is exemplified in Figure 4.21.

4.6.3 Blending Attack

In this section we examine the cost of the *blending attack* on MBIP Mix including the number of rounds of attack, the number of messages, and the total cost for the attacker.

Number of Rounds Required in the Emptying Phase

The number of rounds that the attacker has to flood an MBIP Mix with messages can be computed according to:

$$(1-(1-\prod_{i=1}^{n} p_{asym_i})^k)^G \geq 1-\epsilon \tag{4.15}$$

where G is the number of unknown messages available in the mix; n is the number of selection algorithms operating within the MBIP Mix; $1-\epsilon$ is the probability that the attacker successfully flushes all the unknown messages in the mix; k is the number

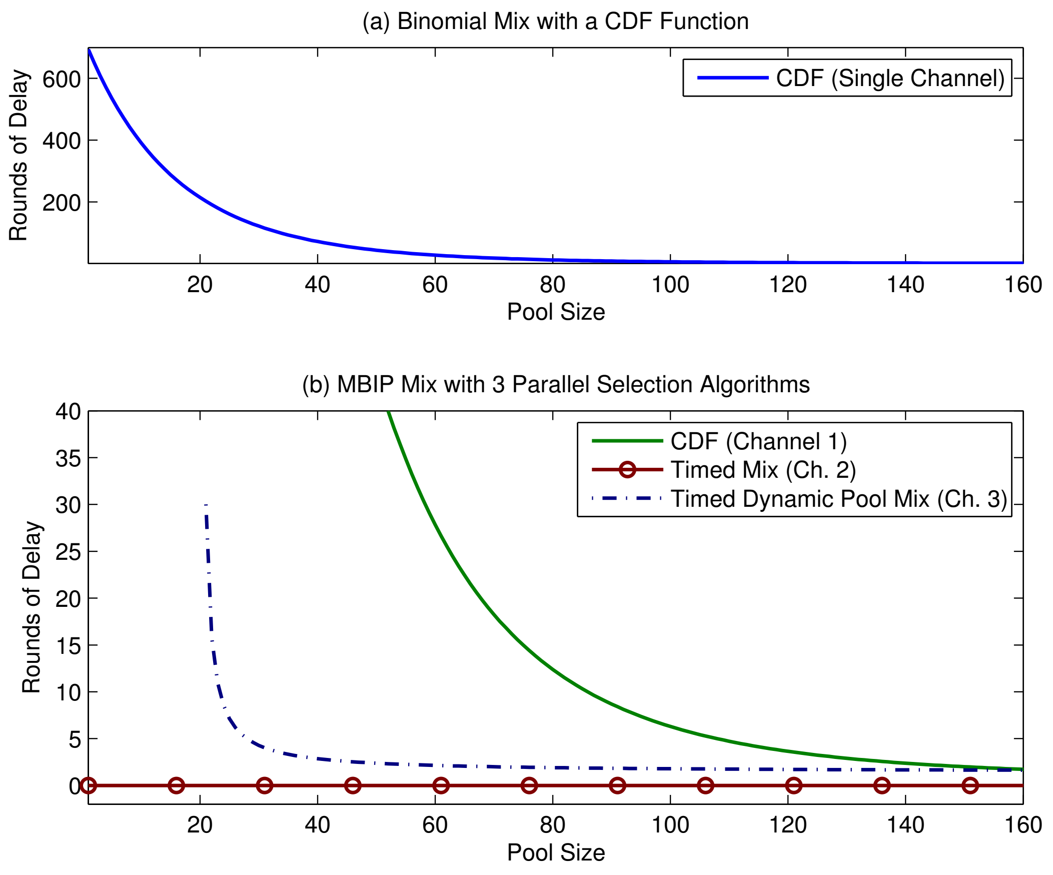


Figure 4.21: Average Message Delay in MBIP Mix

of rounds; and p_{asym} is the asymptotic value of the selection algorithm. This can be expressed as:

$$k = \frac{ln(1-(1-\epsilon)^{\frac{1}{G}})}{ln(1-\prod_{i=1}^{n} p_{asym_i})} \tag{4.16}$$

This in exemplified and contrasted with the CDF-based Binomial Mix in Figure 4.22. The MBIP Mix shown in the figure, similar to the example used for the MBIP Mix in Figure 4.13, contains two Binomial-based selection algorithms: i) a CDF-based and ii) a uniformly random algorithm.

It is observable that, the number of rounds of flooding in the MBIP Mix can increase to approximately 900 times more than that of the CDF-based Binomial Mix. This result is approximately 12 times greater than that of the MBSP Mix as shown in Figure 4.13.

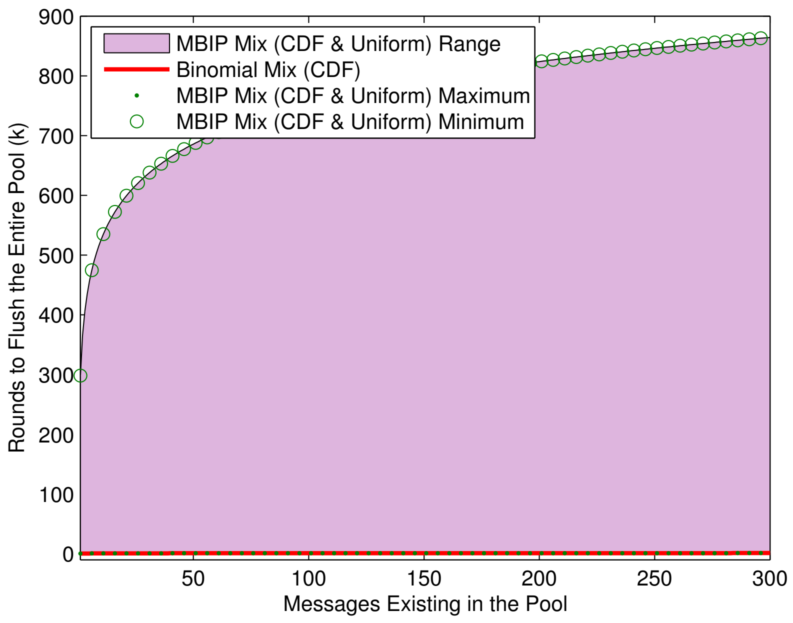


Figure 4.22: MBIP Mix, Rounds of Blending Attack in Emptying Phase

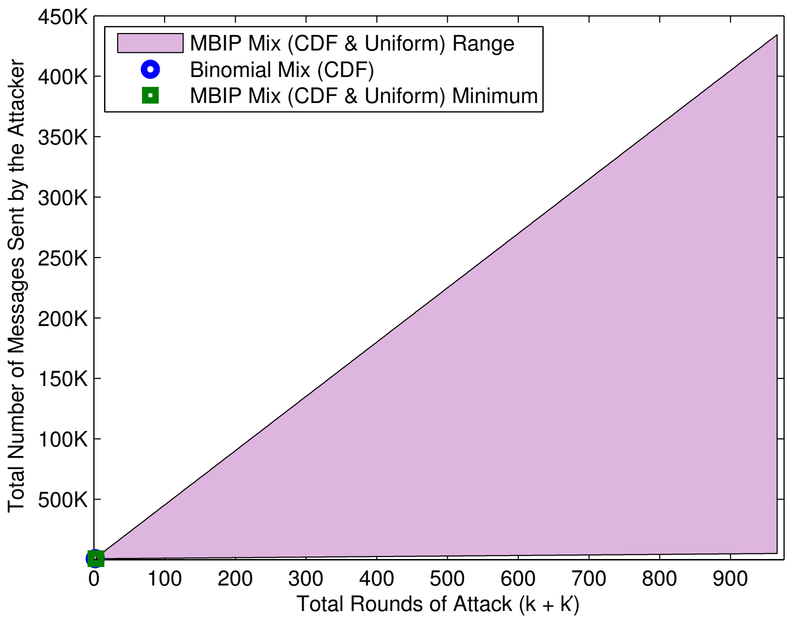


Figure 4.23: MBIP Mix, Total Messages in Blending Attack

Total Cost of the Attack

The number of messages that the attacker has to send in the *emptying phase* can be computed as:

$$\text{Messages} = N_T + (k-1)(N_T + G)\prod_{i=1}^{n} p_{asym_i} \tag{4.17}$$

where N_T is the number of messages to forces the selection algorithms to be equal to the corresponding asymptotic value (p_{asym}), and n is the number of selection algorithms in operation.

The number of rounds, k', necessary to flood MBIP Mix in the second phase of the attack, namely the *flushing phase*, can be computed according to the following formula:

$$k' = \frac{1}{\prod_{i=1}^{n} p_{asym_i}} \tag{4.18}$$

MBIP Mix is a TM and the total number of rounds required for the attack is $(k + k')$ rounds. The total cost of the attack, in terms of the total number of messages that the attackers must sent to the mix, can be computed as:

$$\text{Messages} = (N_T + (k-1)(N_T + G) + (k'N_T + k'G - 1))\prod_{i=1}^{n} p_{asym_i} \tag{4.19}$$

This is exemplified in Figure 4.23 where it is shown that the attacker must flood the mix with a significantly larger number of messages in a significantly larger number of rounds. Given the cost of the attack, it can be concluded that a properly designed MBIP Mix could be completely resilient toward the blending attack.

4.6.4 Requirement for Signalling Protocol

Design of MBIP Mix requires that the mix be aware of the *incoming pool* that an incoming message is destined for.

The design does not enforce a theoretical limit on the number of selection algorithms that MBIP Mix can host. In fact, our analysis shows that the greater number of coexisting selection algorithms, the greater is the uncertainty in the eyes of the attacker, and thus a higher degree of anonymity.

On the other hand, and as shown in Section 4.6.2, the selection algorithm associated with each channel results in a different traffic delay. This makes the mix capable of simultaneously anonymising network traffics with differing latency requirements (e.g. email vs SSH communication). For the best results, this property of MBIP Mix should be purposefully utilised.

These requirements can be met with a signalling protocol designed to set up the configurations associated with an MBIP Mix channel. These configurations must be set prior to the start of data transfer in the channel and should specify: i) the desired selection algorithm (e.g. CDF or Uniform); ii) the properties associated with the chosen algorithm (e.g. parameters of the distribution); and iii) a mechanism to link certain incoming messages to a specific channel.

Where a communication path involves more than one MBIP Mix, the signalling protocol needs to accommodate for a complete circuit establishment through the network. These requirements can be achieved using the architecture and signalling protocol provided by the Garbled Routing network (see Chapter 5).

4.7 Summary of the Chapter

In this chapter we sought to answer *Research Question 2* (see Section 1.4); namely, how can Binomial Mix be made more efficient and secure?

We proposed MBSP Mix and MBIP Mix which improve Binomial Mix in order to achieve a less predictable behaviour. This will reduce the knowledge available to the adversary which is the cornerstone of building various types of attacks. We have considered the passive attack presented in Binomial Mix's original proposal as well as the *blending attack*.

MBSP Mix employs not one but multiple Binomial-based message selection algorithms each of which has a potentially different bias function. The selection algorithms operate one at a time and can make the mix significantly more resistant to the passive and active attacks. The combination of bias functions used in a MBSP Mix affects the efficiency and security achieved by the mix.

MBIP Mix differs in that it contains isolated message pools associated with each selection algorithm. Multiple selection algorithms may operate concurrently and thus allow multiple channels with differing latency requirements to pass through a single mix. The desired latency can be adjusted by the circuit establisher in the initial phase. The resultant multi-tenancy further reduces the knowledge available to the attacker. It also renders carrying out the blending attack impractical.

In Chapter 5 we will divert our attention to the architecture of Anonymous Communication Systems (ACSs). Specifically, we will explain the properties of a generic framework that will enable various mix-based ACSs to converge in a shared system. As we shall see, this approach can achieve a higher degree of anonymity and reduce the development efforts.

Chapter 5

A Framework for Anonymous Communication

The best way to protect your privacy is through a flood of misinformation obscuring the truth.
—Larry Lambert

5.1 Introduction

Chapters 3 and 4 focused on the design and evaluation of mixes in isolation. Moving beyond that, we now turn our attention to the mixnets as a whole. The focus of this chapter is to answer *Research Question 3* of this thesis, that is, to build a generic framework for the mix-based Anonymous Communication Systems (ACSs).

In Chapter 2, we reviewed the commonly-used and/or cited mix and mixnet designs such as Tor, I2P, Crowds, Tarzan and Mixmaster. The variety of these designs, which all build on the notion of Chaumian mix (see Section 2.5.5), shows that while these systems employ many different routing and message-processing algorithms, they nevertheless have much in common as Peer-to-Peer (P2P) systems that seek similar goals. We also reviewed various types of *anonymity* and saw that, presuming other conditions are equal, *global anonymity* of an ACS increases as a result either of growth in the size of the *anonymity sets*, or more even distribution of sending or receiving subjects within the sets [4–7]. A unified framework for anonymous communication can have multiple benefits, described as follows.

Firstly, due to the variety of algorithms and design decisions in different ACSs, implementation and testing of each system has been an independent practice because developers need to build everything from scratch. This translates into lower *reusability* in terms of, e.g. code, simulation, test and deployment. Moreover, independent systems disperse both the research community and the end-users; the former hindering the development and the latter opposing the cause [4, 6, 7]. Amalgamating all the algorithms, designs, user-bases and development efforts into one generic system is therefore an imperative need. The framework proposed in this chapter, referred to as Garbled Routing Framework (GRF), addresses this need through leveraging the principles of Component-based Design.

Secondly, practical ACSs (such as [26, 48, 136, 163]) while concealing the identities, do not conceal the fact that a certain user is indeed using a particular network. This feature, referred to as *network unobservability* [4], is desirable as it further complicates traffic analysis. GRF achieves some degree of *network unobservability*.

Thirdly, foiling timing attacks relies on the existence of cover (dummy) traffic in the network [93] which in turn imposes a significant overhead. Ideally, high-latency traffic could assist with reducing the overhead by transmitting real data. Nevertheless, the precondition is the existence of a system with the capacity for hosting various types of traffic with different latency requirements. GRF facilitates such a mixture by offering an environment within which components of networks with potentially differing latencies can coexist.

Fourthly, further resistance to traffic analysis may be achieved by allowing secret algorithms to operate in the system and, consequently, reduce the attacker's knowledge about the *expected behaviour* of the network routers. Such an amalgam of secret and public algorithms may also introduce new vulnerabilities which need to be thoroughly studied and analysed. Through GRF, we take a step forward in enabling the use of secret algorithms by offering a design that supports this feature. We hope that our work stimulates further research in this area and paves the way.

The schematic in Figure 5.1 shows a comparison of the current practice in the

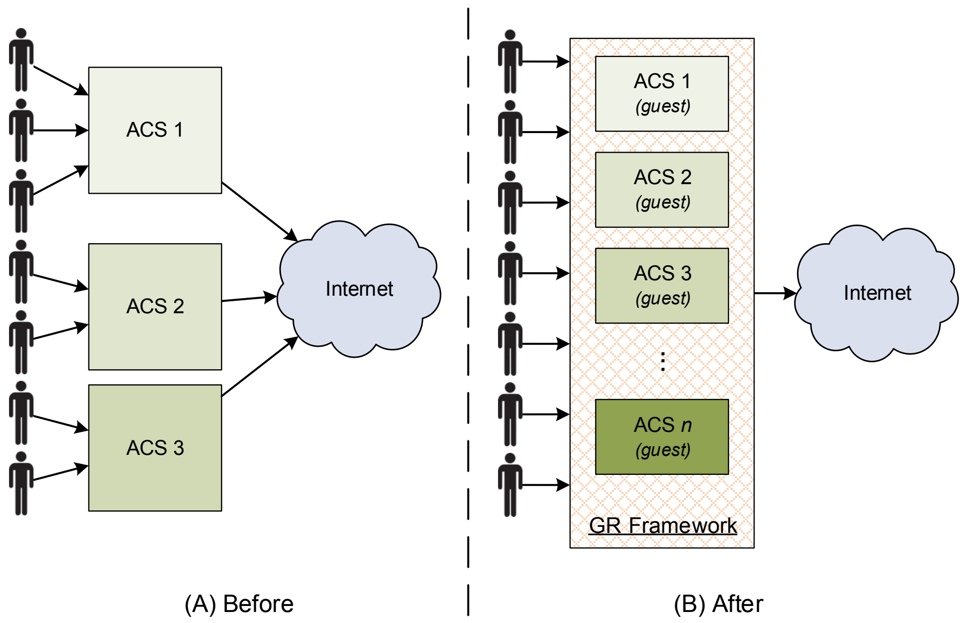


Figure 5.1: Part (A) shows three ACSs and their respective users. This is the current model where user-bases and communication channels are distinguishable. Part (B) shows GRF hosting multiple ACSs and therefore unifying the user-bases and communication channels.

design of ACSs, and how GRF changes the architectural standpoint. GRF can be thought of as an *anonymous system of anonymous systems*. Such an architectural approach unifies the user-bases and decreases the ACS distinguishability, both of which have either actual or potential positive impacts on the degree of anonymity.

There have been prior studies on providing unobservability of anonymous communications by hiding them amongst other Internet traffic, which rely on the cooperation of ISPs [178, 179] or popular routers [180]. We make no such assumption here, and build solely upon the cooperation of network peers, which is an inherent property of ACSs. We take the approach of creating an overlay convergence architecture that aims to bring the existing and future systems together.

There have been previous attempts at blending the traffic with different latencies, such as the Stop-and-Go-MIX [20] and Alpha-mixing [21] which offer such mixing through time intervals in the mix nodes. Beside the additional aims of our research as stated in the previous paragraphs, GRF accommodates such traffic mixing through the concept of Message Processors (MPs).

Organisation The rest of this chapter is organised as follows. The following section presents the definition of the threat model that we will use through the rest of this chapter. Various components of GRF are detailed in Section 5.3. Strategies for the adoption of GRF alongside the sample scenarios are presented in Section 5.4. In Section 5.5 we discuss various aspects of security analysis, followed by the experimental results in Section 5.6. Finally, in Section 5.7, the open questions, limitations and future direction are discussed.

5.2 Threat Model

Inspired by the model used for the practical low-latency systems, we assume a local attacker who is also an active attacker; that is, he can control and observe only some fraction of the network. Specifically, the attacker can monitor the communication to and from a user's computer; can add and remove arbitrary messages to the communication channels, and can run his own routers. However, the attacker is not a global attacker: he is incapable of monitoring the entirety, and particularly the edges, of the network. This model takes into account the fact that any ACS could eventually be broken were the attacker able to observe the edges of the network [132].

To limit the scope of this work, we also assume that the network authorities and the executable code running on the routers are fully trusted and free of implementation defects. Likewise, the same assumption has been made for the components that are downloaded through the network authorities. Finally, we also assume that the adversary does not have *a priori* or contextual information about the correlation between the incoming and the outgoing messages.

5.3 Design of Garbled Routing Framework (GRF)

The framework presented here is an overlay network that can host ACSs, referred to as *guest ACSs*, in the form of plug-in components. The components of guest ACSs are deployed through the network authorities, are publicly available and, according to the threat model (see Section 5.2), are trusted and safe.

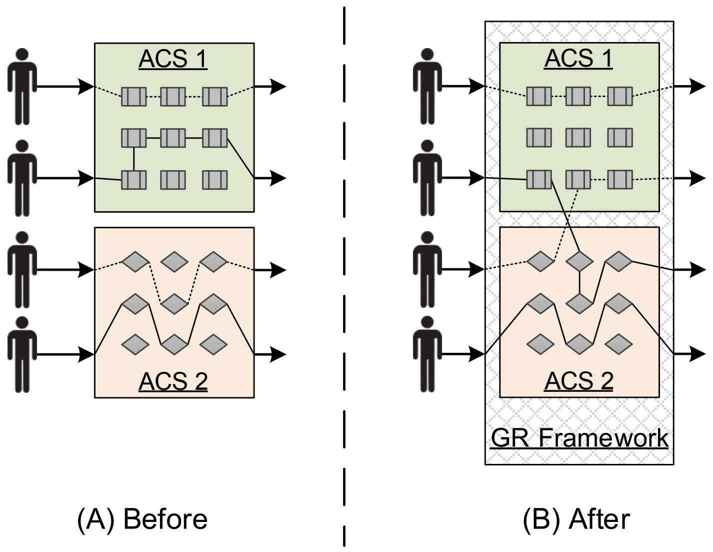


Figure 5.2: Part (A) shows the users of two ACSs and their respective circuits. The separation of networks does not allow circuits to travel inter-ACS and the routers are thus not shared. Adoption of GRF allows the circuits to travel beyond their home ACS as shown in Part (B).

It is also possible to deploy secret components in GRF through arbitrary sources. However, at this stage, the secret components are considered to be only used for experimental and research purposes. In the absence of GRF, the existing incompatibilities amongst the ACSs prevent them from sharing routers beyond ACS borders. A unified framework, as shown in Figure 5.2, enables inter-ACS circuit-establishment and router-sharing, and unifies the user-bases.

Section 5.3.1 presents an overview of GRF's major components and their roles in relation to other parts of the system. We will then present the technical details about the operation of each component in the subsequent sections.

5.3.1 System Architecture

The operation of GRF relies on three main parts of the systems, as shown in Figure 5.3: namely, Garbled Routing Server (GRS), Garbled Routing Proxy (GRP) and Garbled Routing Router (GRR). GRR is the most important part and consists of four sub-components. Dynamic Route Processing Table (DRPT), as we shall see in Section 5.3.2, is a data structure that allows GRR to decide on how the incoming

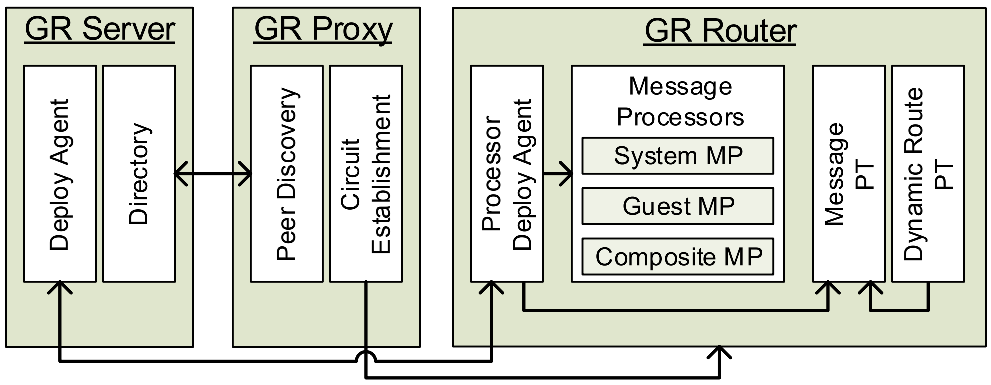


Figure 5.3: An architectural look at GRF. The direction of dependencies among the system components is shown by the arrows.

pc	pid	id	seq	size	payload

Figure 5.4: Network Message Format of GRF

messages should be dealt with. To assist this process, the network messages have a header as shown in Figure 5.4. It is important to observe that this header serves only to deliver the messages to the processor components. Message structures required for the operation of guest ACSs shall be encapsulated within the payload.

A Message Processing Table (MPT) assists with keeping track of the locally available processors. MPs, which are divided into three categories, are small processing units that perform unique actions to the messages directed to them. The details of MPT and MPs are presented Section 5.3.3. The Processor Deploy Agent is the client-side component that handles the deployment of MPs on the routers.

GRP is responsible for peer-discovery and creating circuits. The complexity of GRF mainly relies on the components of the router (GRR), and particularly on MPs and the two processing tables (i.e. DRPT and MPT). Circuit-establishment and peer-discovery are common practices in peer-to-peer systems which, in our system, are provided by the guest ACS designers. The deployment process of MPs, as indicated by the threat model (see Section 5.2), is a trusted process.

5.3.2 Dynamic Route Processing Table (DRPT)

DRPT, as shown in Table 5.1, contains a set of Route Processing Rules (RPRs) with a structure which resembles the rows shown in the table. If the incoming network message is destined for a specific MP, the header fields *pc* and *pid* will identify a unique processor. The messages that do not carry valid *pc* and *pid* values will be checked against the active RPR entries and forwarded to the correct MPs accordingly.

Each RPR consists of a set of values and are explained as follows. *Source* denotes the network address of sender of the message, and is used alongside the *id* and the range specified by *sequence* to match a message with a specific RPR. *Active time* specifies a time span during which the RPR is in effect. The time span also eliminates the need to have sequential *sequence* values in the messages of a single flow, which could potentially leak information to the adversary. The *status* is used during the path-establishment process, as we shall see in Section 5.4.

The *pc* and *pid* fields in the DRPT specify how incoming messages should be processed in case of a matching and active RPR. These values point to a unique MP listed by MPT. The *processor params* contain the parameters that should be passed to the MP alongside the incoming messages. Network routers can create RPR entries in the DRPT of other routers, and the *change secret* is used to protect RPR entries against unauthorised modification.

5.3.3 Message Processors (MPs) and Message Processing Table (MPT)

MPs are isolated and stateful execution processes that receive messages and perform specific tasks. The messages may be either carrying control commands or communication data. MPs react to the messages by changing state, modifying the messages and/or generating new messages.

MPT, as shown in Table 5.2, contains the list of locally available MPs. MPT is maintained by Processor Deploy Agent as shown in Figure 5.3. MPs are categorised in three main groups which are described in the following sections.

Table 5.1: Dynamic Route Processing Table (DRPT)

	Source	ID	Sequence		Status	Active Time		Change Secret	PC	PID	Processor Params
			Start	End		Start	End				
RPR_1	77.36.192.128	123	1	230	active	10:20	10:30	some-secret-1	SP	7	131.170.40.30
RPR_2	77.36.192.128	332	231	500	active	10:20	10:40	some-secret-1	SP	7	69.147.125.65
RPR_3	202.149.224.128	524	750	900	active	10:21	10:07	some-secret-2	SP	7	74.125.237.19

Table 5.2: Message Processing Table (MPT)

	PID	Processor Name	Description	Parameters
System Processors	1	Reserve RPR	Reserve a RPR entry to be finalised later	source|id|$\text{seq}_{s,e}$|$\text{time}_{s,e}$
	2	Finalise RPR	Finalise a reserved RPR entry and set its processors	source|id|$\text{seq}_{s,e}$|$\text{time}_{s,e}$|secret|pid|params
	3	Delete RPR	Delete RPR by its creator	source|id|$\text{seq}_{s,e}$|$\text{time}_{s,e}$|secret
	4	Create Composite	Create mix of existing processors	pid_1|param_1|pid_2|param_2 ...
	5	Add Public	Install public guest processor	pid
	6	Add Secret	Install secret guest processor	Unified Resource Locator (URL)
	7	Relay Message	Forwarding messages as is	IP
	8	Encrypt Message	Encrypt message	algorithm|key
	9	Decrypt Message	Decrypt message	algorithm|key
	10	Change Header	Alter the header values of message	pc|pid|id|seq|size
Public Guest Processors				
Available for everyone; algorithm is publicly known; downloaded from the network authority				
Secret Guest Processors				
Secret algorithms; downloaded via arbitrary URLs or other out-of-band means				
Composite Processors				
Created by the *PID=4* System Processor; in-memory and short-lived				

System Message Processors

The basic and essential services of GRF are offered by this type of MP and exist on every GRR. A network message having a *pc* value of *System* and a valid value for *pid* will be delivered to the specified System MP. Alternatively, an RPR in DRPT may direct a message with a particular *id* and *sequence* to a specific System MP.

The first three System MPs allow the modification of DRPT. They respectively offer the functionality to reserve, finalise and delete RPR entries on a remote router. The *change secret* may be locally generated by router or specified by a remote router.

The *Create Composite* MPs create a dynamic composition of the existing processors on a router. This is an important element of the design and will be further explained in the following sections. Composite MPs depend on the existence of a set of processors that allow being mixed with others. The *Add Public* and *Add Secret* MPs are responsible to deploy the two types of Guest MPs on a router.

A *Relay Message* MP is responsible for relaying the messages to other network nodes. Basic ciphering and deciphering techniques may be implemented in the form of *Encrypt Message* and *Decrypt Message* MPs. These basic methods may be extended by Guest MPs. The *Change Header* MP alters the header values of a network message while it travels through routers.

Guest Message Processors

As opposed to System MPs that form the basis of GRF, the existence of Guest MPs is not obligatory. However, any extension to the basic functionality of GRF, including integration with guest ACSs, is implemented by the Guest MPs. Figure 5.5 shows an instance where a communication path is going through three routers $< a, b, c >$, with each GRR performing some operations on the messages.

The *Public* Guest MPs are made available through the trusted network authorities. These Processors are open-source and available to the general public, which is the property of most existing and reputable ACSs [26, 163, 181, 182].

Secret MPs differ in that they are not necessarily well-known and open-source but

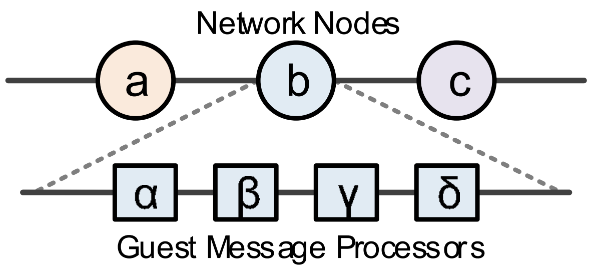


Figure 5.5: Guest MPs inside GRRs. Here GRR b contains a Composite MP encompassing four MPs, namely, the α, β, γ, and δ.

rather can be anything made available via an arbitrary source. This type of MP, which is currently only an experimental part of GRF, serves two purposes. Firstly, it allows GRF to accommodate the integration of guest ACSs which operate using secret algorithms. As a result, the user-bases of all ACSs will grow and uncertainty about the behaviour of routers will increase. Secondly, this type of MP accelerates the process of testing the developing algorithms by allowing designers to conduct experiments using an operational and large network.

Various security aspects of hosting secret algorithms within GRF are discussed in this chapter. Specifically, from the attacker's point of view, the additional uncertainty that these algorithms bring to the operation of routers is analysed in Sections 5.5.2 and 5.5.4; and the new types of vulnerabilities introduced by this technique and the corresponding remedies are discussed in Section 5.5.1. As this feature is introduced in GRF for the first time, it is necessary to invite more critical analyses from the community. Meanwhile, we rely on this feature only as an experimental one.

Composite Message Processors

This type of MP is created in-memory and only as a result of other remote nodes invoking the *Create Composite* MP. *pid* is specified during the creation phase, using a random generator algorithm, and returned to the remote creator. Composite MPs, as opposed to other MP types, are not permanent and will be automatically deleted when no longer in use. The flexibility offered by GRF mainly relies on the Composite MPs for they allow virtually any message processing algorithm to be hosted by GRF.

The existing ACSs can integrate with GRF and become guest ACSs in either of two ways. The first is to write a single and complex *Guest* MP which contains the entirety of the logic necessary for the operation of a router in the ACS in question. Though seemingly more convenient, this approach is discouraged as it is not aligned with the goals of GRF.

Alternatively, various features of the guest ACS and all the required logic for processing network messages in that ACS can be broken down and implemented as a set of small and simple MPs. These small MPs can then be combined into complex processing units using the *Composite* MPs. This second approach is encouraged because the frequently-used message processing techniques (e.g. encryption and integrity checking) would then have a higher chance of being shared and reused.

Here we present a simple example for using Composite MPs. Assume node N_A wants to *reserve* an RPR entry on node N_C without a direct communication with N_C. N_A would need to first create a Composite MP on an intermediary node N_B such that it would be a combination of *Change Header* and *Relay Message* MPs. In this example, the new *pid* of the message would be equal to 1 (i.e. to reserve a path), and relay destination would be N_C. N_A then could create an RPR on N_B and set its processor to the newly-created Composite MP. Having done that, N_A could then send a message to N_B's newly-created Composite MP, causing N_B to change the *pid* of the message to 1 and relay it to N_C.

5.3.4 Internal Message Flow of the Routers

In order to bring together the concepts mentioned in the previous sections, a schematic of various processor categories and their interactions is shown is Figure 5.6. Direction of the arrows shows the flow direction. Message Dispatcher resorts to DRPT and MPT to decide which processor is responsible to process a certain message. Only system MPs are allowed to access the system services, whereas the other types of MPs can only access the system services through creating new messages destined for the corresponding system MPs. Algorithm 1 shows the logic of message dispatching.

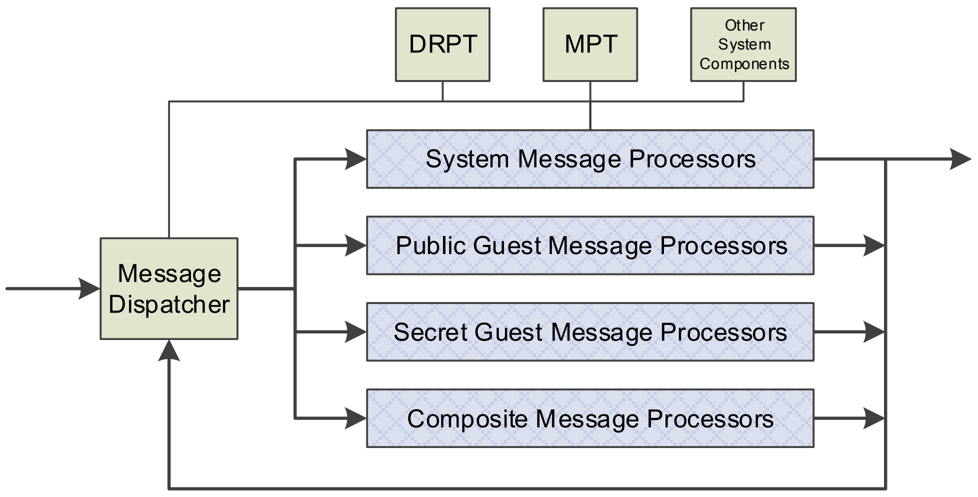

DRPT
MPT
Other System Components
System Message Processors
Public Guest Message Processors
Secret Guest Message Processors
Composite Message Processors
Message Dispatcher

Figure 5.6: Internal Message Flow of GRRs. Message Dispatcher receives network messages and routes them to the processors as per the existing rules in DRPT and MPT. Processing the messages may result in the creation of new messages which, in turn, will be returned to Message Dispatcher. Accessing and altering DRPT, MPT, and other system resources as well as sending messages to other network nodes are all available only through the System MPs.

Algorithm 1 Message Dispatching inside the Framework Routers

```
1: sender, msg ← AwaitMessage()
2: processor, params ← empty
3: processor ← FindProcessor(msg.pc, msg.pid)
4: if processor $\neq$ ‘valid processor’ then
5:     processor, param ← FindInDRPT(sender, msg.id, msg.seq)
6: end if
7: if processor = ‘valid processor’ then
8:     ProcessMessage(processor, params, msg)
9: else
10:    DropMessage(msg)
11: end if
```

5.3.5 Garbled Routing Server (GRS)

GRS is the network authority that offers two services. Firstly, GRS acts as the network directory assisting with the peer-discovery procedure. Secondly, it serves as a trusted source for distribution of System and Public Guest MPs; that is, it stores the list and the executable components of such MPs.

As mentioned in the description of the architecture (see Section 5.3.1) and aligned with the threat model (see Section 5.2), GRS constitutes a trusted authority and distributes only trusted code. Hence, this chapter does not include a security analysis with the aim of evaluating the resilience of GRS against attacks. Here, attention is paid to the services provided by GRS that assist with the experimentation of GRF.

5.4 Modelling the Operation of Guest ACSs

The technical details of GRF were presented in the previous section. The aim of this section is to put the presented details into context and to describe how GRF can be put into practice. To this end, we first explain how GRF can be adopted in various guest ACSs with differing requirements. Afterwards, we present three examples. The first example is a simplified look at the adoption requirements, whereas the second and third examples focus on modelling two real-world ACSs.

5.4.1 Adoption of the Framework

When it comes to sharing the user-bases, various ACS designs have different requirements and limitations. For instance, some ACSs may want to share their user-bases only partially in an attempt to hide the network topology or to prevent the attacker from enumerating and thus attacking all the participating routers. Adoption of GRF can be gradually implemented in various guest ACSs according to the requirements and design strategies.

Figure 5.7 shows three structurally different ACSs where A and B allow nodes to communicate with external entities (i.e. E_1 and E_2), and C is an example of a storage and retrieval system sharing data only within the system boundaries. Four

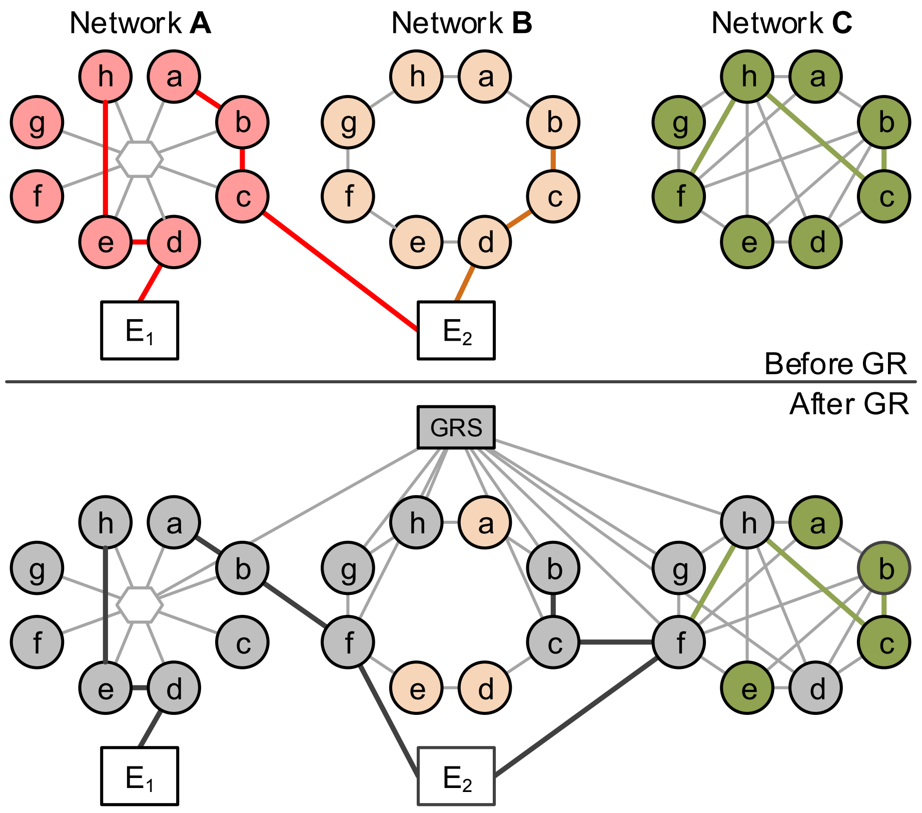


Figure 5.7: Different ACSs may expose the entirety of their nodes (e.g. Network A) or do so partially (e.g. Networks B and C). This allows a controlled sharing of user-bases and conceals the topology of networks as desired. GRS can be used as the network directory. The grey nodes represent GRRs; that is, nodes that adopted GRF's protocol.

sample communication paths are shown, namely $A_h \leftrightarrow E_1$, $A_a \leftrightarrow E_2$, $B_b \leftrightarrow E_2$, and $C_f \leftrightarrow C_b$. The lower half of Figure 5.7 shows the same ACSs after the adoption of GRF; that is, these ACSs have become guest ACSs of GRF. The nodes shown in grey represent GRRs. As shown, an ACS can expose its entire set of routers or choose to share only a limited portion.

GRF allows the communication paths in the guest ACSs to go beyond the traditional network boundaries. For instance, the communication path $A_a \leftrightarrow E_2$ is changed from $< A_a, A_b, A_c, E_2 >$ to use also the nodes in the ACS B and become $< A_a, A_b, B_f, E_2 >$. Similarly, the path $B_b \leftrightarrow E_2$ is also changed to use nodes in ACS C to become $< B_b, B_c, C_f, E_2 >$.

Routers operating inside GRF (i.e. GRRs) cannot know to which guest ACS other

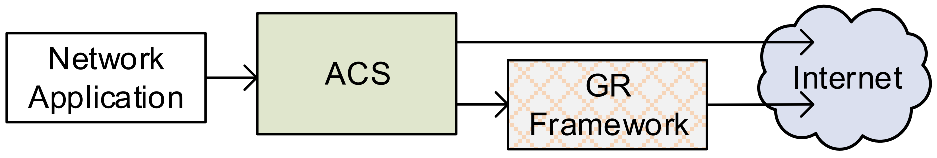


Figure 5.8: GRF shown as an extra communication layer, allowing the unification of multiple guest ACSs. The guest ACSs benefit from joining GRF whether fully or partially.

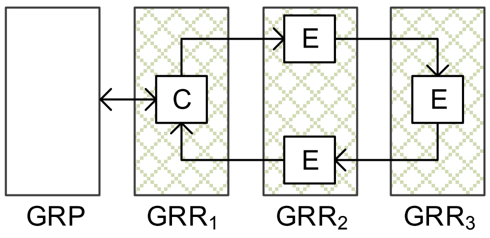


Figure 5.9: Sample Echo Circuit through GRF. Here GRP creates a circuit through three routers (GRRs) in order to send messages through this circuit that will be echoed back to its source.

routers truly belong, and GRRs' knowledge is limited to whether or not other GRRs are able to route messages for them. Figure 5.8 shows how GRF operates as an extension to the existing communication layers.

5.4.2 The Echo Circuit Example

An echo circuit is a very simple circuit that returns messages to the sender after they have travelled through a number of intermediary nodes. We assume an external entity GRP wants to establish an echo circuit through GRF. As shown in Figure 5.9, the path goes through the two routers, GRR_1 and GRR_2, and is then echoed back by a final router GRR_3 to its source.

Algorithm 2 shows a simple sequence of messages that GRP should send to the participating GRRs. In the first step the message is destined for GRR_1 which is instructed to relay messages received from GRP, with an assumed $id = 12$, to GRR_2. The rest of the routers are similarly instructed to forward the messages without any further processing. If all routers return success results, in Steps 8–10, GRP sends another round of messages to all routers for finalising the circuit creation. If the returned results indicate failure, GRP would have to try to address the problem and retry the circuit creation.

Algorithm 2 Echo Circuit Establishment by a Proxy (GRP)

```
1: Send($GRR_1$, pc:sp, pid:1, param[id:12, src:GRP, pid:7, dst:$GRR_2$])
2: Send($GRR_2$, pc:sp, pid:1, param[id:12, src:$GRR_1$, pid:7, dst:$GRR_3$])
3: Send($GRR_3$, pc:sp, pid:1, param[id:12, src:$GRR_2$, pid:7, dst:$GRR_2$])
4: Send($GRR_2$, pc:sp, pid:1, param[id:12, src:$GRR_3$, pid:7, dst:$GRR_1$])
5: Send($GRR_1$, pc:sp, pid:1, param[id:12, src:$GRR_2$, pid:7, dst:GRP])
6: results $\leftarrow$ AwaitReply(timeout, $GRR_1$, $GRR_2$, $GRR_3$)
7: if results = 'path succeeded' then
8:   for all $GRR_i$ in path do
9:     Send($GRR_i$, pc:sp, pid:2, param[finalise RPR params])
10:  end for
11: else
12:   retry with different settings (e.g. $id \leftarrow id + 1$)
13: end if
```

Existence of *reserved* RPRs allows having an abstract path negotiation state such that the logic of GRF is made independent of the logic of the guest ACSs. This makes the circuit-establishment process very flexible by allowing the negotiations to hide behind the reserved RPRs.

5.4.3 Tor-Like Circuit Establishment in the Framework

As a real-world and low-latency guest ACS modelling, we adopted the circuit-establishment in Tor, which is shown in Figure 5.10. Alice, who wants to browse a website through two intermediary routers, needs to create the communication path one hop at a time. In the first phase, she negotiates a symmetric key with GRR_1 by sending a *create* request, and then asks GRR_1 to *extend* the path to GRR_2.

The message exchange required to reproduce this phase is shown in part (a) of Figure 5.10. The $negotiator_n$ is an MP that wraps the logic of Diffie-Hellman key exchange, and $TorExtender$ represents another MP that wraps the logic of extending a Tor-like path to the next hop.

In the second phase, Alice sends the data transmission request to GRR_1, which is wrapped in multiple layers of encryption. GRR_1 unwraps one encryption layer and passes the resulting message to GRR_2. The final layer of encryption is removed by GRR_2 and the actual request is sent to the final destination. The responses received

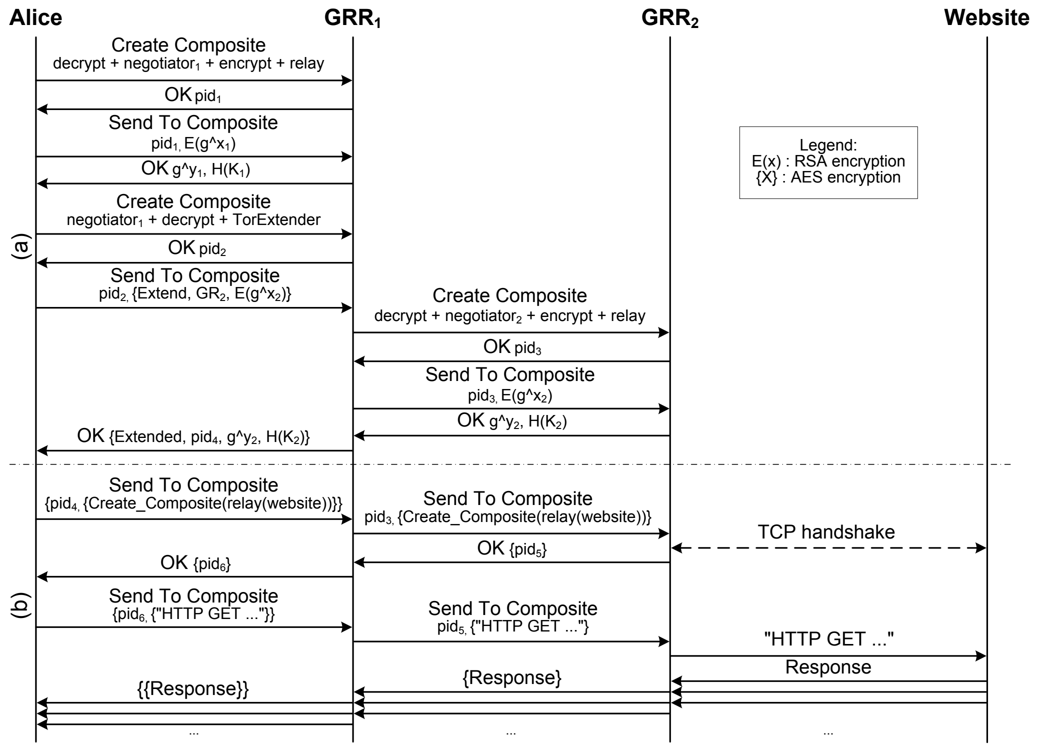


Figure 5.10: Tor-like Circuit Establishment through GRF. The original form of this model was presented in [26] and is extended here to show how a real-world ACS becomes a guest ACS in GRF.

from the website are sent through the path back to Alice. This phase is shown in part (b) of Figure 5.10.

5.4.4 Modelling Mixmaster in the Framework

Mixmaster [182] is a high-latency communication protocol that protects email messages against traffic analysis. In this section, we model the operation of this protocol as another real-world ACS example which can operate within GRF.

The relay nodes of the Mixmaster protocol are known as *remailers*. When Alice wants to send an anonymous email to Bob, her user-agent performs a number of tasks: compressing the message; splitting it into smaller pieces; building one independent circuit for each message piece through the ACS; encrypting each piece multiple times with public-key of relay nodes, and finally sending each piece to the first relay in the corresponding circuit. A simplified version of this process in GRF has been shown in Algorithm 3.

Algorithm 3 Mixmaster User Agent in GRF

```
1: receiver, message ← AwaitMessage()
2: compressedMessage ← CompressMessage(message)
3: messagePieces[] ← BreakIntoPieces(compressedMessage)
4: GRR$_{final}$ ← ChooseFinalRelay()
5: Create Mixmaster final remailer composite on GRR$_{final}$ to relay to receiver
6: for all messagePiece$_i$ in messagePieces[] do
7:     GRRList$_i$ ← SelectRandomRelays(GRR$_{final}$)
8:     for all GRR$_z$ ← GRRList$_i$ do
9:         Create Mixmaster remailer composite on GRR$_z$ to end to GRR$_{final}$
10:    end for
11:    Send messagePiece$_i$ to first router in GRRList$_i$
12: end for
```

The remailers, which may be implemented within GRF, perform a number of tasks shown in Figure 5.11. All the remailers along the way, except the final one, perform the following tasks: decrypting the received messages with their private key; integrity checking; decrypting the message using the embedded secret-key; shifting bytes up to update the header; appending random bytes to maintain a constant message size; adding random dummy messages; putting the outgoing messages into a message pool, and finally relaying the message to the next remailer.

As shown in Figure 5.11, these tasks may be performed by a set of MPs. It is important to note that, in accordance with the aim of providing reusability in GRF, multiples of the suggested MPs may be developed in a generic fashion and thus be shared amongst various guest ACSs.

The final remailer is responsible for merging the pieces of the message and forwarding it to the final destination. It is also responsible for identifying and discarding duplicated messages, decompressing the message after reconstruction, and maintaining a message pool similar to that of the intermediary remailers. These steps are expressed in Figure 5.11 as per GRF's MP design.

5.5 Security Analysis

In this section, we elaborate on the security aspects of GRF. The risks involved in deploying the MP components are discussed first. The topics that will follow are

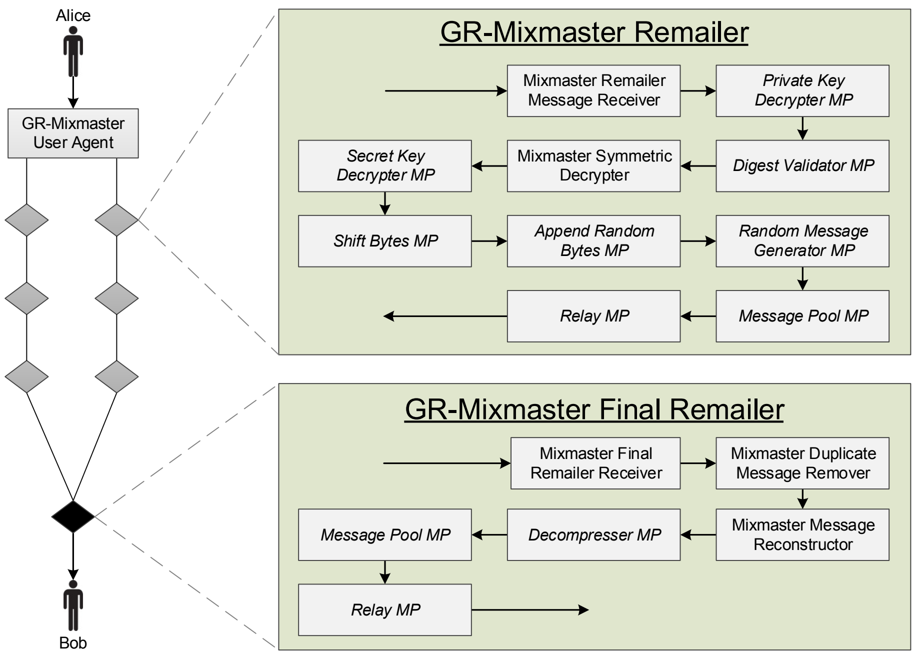

Alice
GR-Mixmaster User Agent
Bob
GR-Mixmaster Remailer
Mixmaster Remailer Message Receiver
Private Key Decrypter MP
Digest Validator MP
Mixmaster Symmetric Decrypter
Secret Key Decrypter MP
Shift Bytes MP
Append Random Bytes MP
Random Message Generator MP
Message Pool MP
Relay MP
GR-Mixmaster Final Remailer
Mixmaster Final Remailer Receiver
Mixmaster Duplicate Message Remover
Mixmaster Message Reconstructor
Decompresser MP
Message Pool MP
Relay MP

Figure 5.11: Mixmaster Remailers Algorithm in GRF. The two types of remailer are broken down to their potential MPs. The processors shown with italicised names are generic and can be shared amongst various guest ACSs.

mainly concerned with the ability of the attacker to predict GRF's states. This section also discusses the features of GRF that, if considered in the design of future ACSs, will have a positive impact on the security of those designs starting from the very early stages of their operation.

5.5.1 Susceptibility to Malicious Code

The term *malicious code* is used to refer to any MP deployed in GRF that either compromises the anonymity of users or disrupts the normal activity of GRF. As specified by the threat model (see Section 5.2), the software packages offering the core functionality of GRF are fully trusted. Therefore, malicious code refers only to the code that is deployed through the *Guest* MPs.

The *Public* MPs are deployed through the network authority, and thus only a compromised authority would allow the distribution of such malicious MPs. This issue could be avoided through voting and signing techniques used by the network authorities which, according to the threat model, is beyond the scope of this chapter.

The *Secret* Guest MPs can be deployed via arbitrary sources. It is important to observe that the execution of this type of processors depends on the existence of corresponding DRPT entries. Naturally, honest nodes never instruct other peers to invoke malicious components in their communication paths. However, this may change if the path goes through nodes controlled or subverted by the attacker.

Trusting the truthful execution of the protocol is a common challenge in the design of ACSs, which deteriorates by relying on volunteer network participants. Verifiable Mixes eliminate this problem but they are yet to be made efficient for low-latency communication [132]. The widely-accepted countermeasure is to create paths with geopolitically distant nodes, which significantly narrows the odds of them all being controlled by the same adversary. This guarantees some level of anonymity, even if the messages travel through malicious routers.

The attacker may also remotely create DRPT entries to invoke malicious components that do not intercept a channel but otherwise disrupt the network. This applies only where GRF routers (GRRs) allow the execution of *Secret* Guest MPs, and can be contained by effective sandboxing and resource management techniques within GRRs. Specifically, MPs must be allowed limited computation and memory resources, and be isolated from other MPs and system resources. This can be achieved, for instance, via the sandboxing and access control techniques that Java offers.

5.5.2 Information about the Expected Behaviour of Routers

Inspired by previous work [21], here we elaborate on attacker's knowledge about the expected behaviour of the system. The traditional ACS routers have a fairly predictable behaviour although it is somewhat obfuscated by embedding random elements in their design. For instance, nodes in Crowds [48] flip a biased coin to determine whether they should forward packets to another node. Although this increases uncertainty, the attacker is still able to benefit from the probability of possible behaviours.

GRRs must be able to disable Secret MPs if they so wish. Hence, at any given time, from the total number of routers $\mathcal{R}$, a portion run only Public MPs ($\mathcal{R}_p$) and the other routers allow both Public and Secret MPs ($\mathcal{R}_{p,s}$), where $\mathcal{R} = \mathcal{R}_p + \mathcal{R}_{p,s}$.

The total number of MPs deployed in GRF is denoted by $\mathcal{A}$, which includes $\mathcal{A}_p$ number of Public and $\mathcal{A}_s$ Secret MPs, where $\mathcal{A} = \mathcal{A}_p + \mathcal{A}_s$. The probability that a certain router runs a particular MP α and, consequently, behaves expectedly is calculated as:

$$\begin{aligned}
\mathcal{P}_\alpha &= Pr(\text{GRR running MP } \alpha) \qquad (5.1)\\
&= Pr(\text{GRR running a specific composition})\\
&= Pr(\text{GRR being only a relay})\\
&= Pr(\text{MP-based attacks match the right GRR})\\
&= \frac{\mathcal{R}_p}{\mathcal{R}\mathcal{A}_p} + \frac{\mathcal{R}_{p,s}}{\mathcal{R}\mathcal{A}}
\end{aligned}$$

This equation also includes terms about the probability of a GRR being only a relay, running a specific composition, and being targeted by the right algorithm-based attack. Note that the extra cases mentioned here are in fact different expressions of the same problem. In other words, being *only a relay* is one instance of α; targeting an attack to *the right MP* is searching for a particular instance of α; and running *a specific composition* is yet another instance of α.

As an example, assume a scenario where no router allows Secret MPs, no Public MP has any built-in randomness, and only two Guest MPs exist in GRF. Although the assumptions are overly strict, yet there is significant decrease in an attacker's certainty about the value of $\mathcal{P}_\alpha$. These assumptions are somewhat relaxed in Figure 5.12 where it is assumed that 10% of GRRs allow Secret MPs. Furthermore, observe that the number of Secret MPs (i.e. $\mathcal{A}_s$) is in fact unknown, and hence the attacker cannot compute the value of $\mathcal{P}_\alpha$ with absolute certainty.

5.5.3 Probability of Successful Attacks

We define $\mathcal{T}_\alpha$ to be an attack on MP$_\alpha$, with some probability of success $Pr(\mathcal{T}_{\alpha S})$. In the previous section it was shown that for an attack to succeed, it must first be matched to the right MP. We denote the probability of success of any given attack,

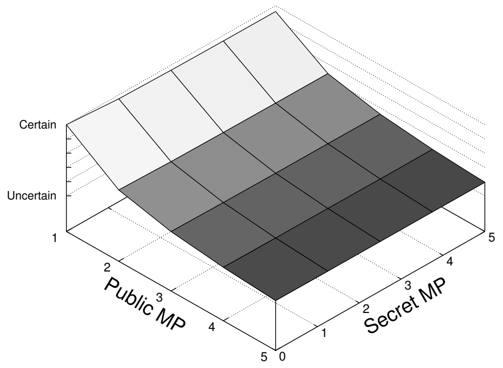


Figure 5.12: Certainty about MPs ($\mathcal{P}_\alpha$) when only 10% of GRR run Secret Guest MPs.

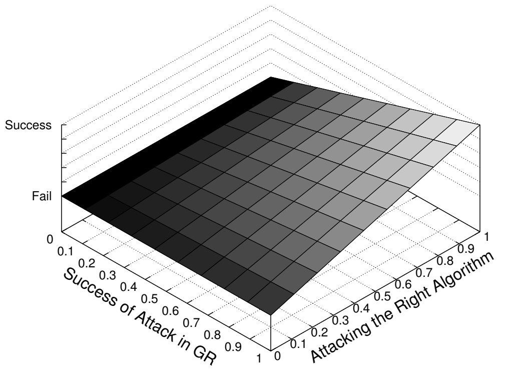


Figure 5.13: Probability of attack succeeding when ACSs migrate to GRF ($Pr(\mathcal{T}_{\alpha_{S'}})$).

after the targeted MP$_\alpha$ becomes a *guest* in GRF, as $Pr(\mathcal{T'}_{\alpha_S})$, and calculate it as:

$$Pr(\mathcal{T'}_{\alpha_S}) = Pr(\mathcal{T}_{\alpha_S}) \cdot \mathcal{P}_\alpha \tag{5.2}$$

This is shown in Figure 5.13, where the worst case is when attack $\mathcal{T}_\alpha$ is successful every time and the MP$_\alpha$ is identifiable with absolute certainty. Given the value of $\mathcal{P}_\alpha$ as shown in (5.1), it can be seen that as the total number of MPs in GRF grows, the probability of success of the attack decreases, i.e. $\lim_{\mathcal{A}\to\infty} Pr(\mathcal{T'}_{\alpha_S}) = 0$.

It is interesting to observe that the probability of n nodes being involved in the same communication path in GRF, denoted by $Pr(\mathcal{S'}_n)$, can be calculated as shown in (5.3). $\mathcal{N}$ is the total number of guest ACSs within GRF, and $Pr(\mathcal{S}_{n_i})$ is the probability of n nodes being involved in the same communication path in the guest ACS$_i$, irrespective of GRF.

$$Pr(\mathcal{S'}_n) = \prod_{i=1}^{\mathcal{N}} Pr(\mathcal{S}_{n_i}) \tag{5.3}$$

Equation (5.3) shows a relative decrease in the attacker's certainty about the total number of nodes typically involved in a circuit. Consequently, the attacker also knows less about the potential position of a GRR in the circuits.

In the traditional ACS design, the behaviour of a group of n consecutive routers is fairly likely to be known to the attacker. In contrast, in GRF the probability of a path of n nodes running any certain MP or composition of MPs is:

$$\begin{aligned}
&Pr(n \text{ nodes run algorithm } \alpha) &(5.4)\\
&= Pr(n \text{ nodes run a certain composition})\\
&= \mathcal{P}_\alpha{}^n
\end{aligned}$$

This shows a significant decrease in the attacker's certainty about the behaviour of the routers. Similar to (5.2), in (5.4) we can observe that $\lim_{\mathcal{A}\to\infty} \mathcal{P}_\alpha{}^n = 0$.

5.5.4 Elements of Obscurity

The attacker's elementary step is to gather information about the network operation and the expected behaviour of the nodes. For instance, to launch a tagging attack (see Section 2.5.5), the attacker must first ensure that cryptographic techniques used in the system are indeed vulnerable to a certain type of tagging attack. Although the operation of GRF is publicly known, GRF still introduces additional obscurity to the operation of ACSs as explained below.

Hidden Secret MPs Secret MPs, as presented in Section 5.3.3, can be deployed in GRF through an arbitrary source. This feature is currently experimental and subject to further study. It is important to note that if the source of deployment is public knowledge (e.g. through a publicly accessible URL), the MP is in fact no longer a *secret* entity.

Although the MPs made available through publicly accessible sources no longer constitute *secret*, they have other applications for the community. This feature enables the deployment of MPs which are yet to be approved by the network authorities for the purpose of wide distribution and use. Such a capability is useful for the phases in which MPs are being developed and tested because it can offer the development team an opportunity to test the effectiveness and performance of their anonymisation algorithm inside a real system.

It is important to highlight that these MPs, although not effectively *secret*, are deployed in the network through methods unknown to the adversary. Hence, to ensure a particular router supports such MPs, the attacker must undergo the task of probing that individual router for its available services.

If the Guest MPs are distributed through genuinely secret means, then both the logic and the expected behaviour of those MPs are concealed from the attacker. This is also true about the pattern of distribution of those MPs in the network. These are indeed *secret* MPs that limit the knowledge that the attacker can possibly have about the behaviour of GRF and its routers. This also provides for the existence of GRRs that serve other peers by offering the services of the System and Public MPs, while their own communications are in fact transmitted through Secret MPs and the associated *secret* ACS.

Furthermore, the existence of the Secret MPs limits the attacker's ability to position his routers within arbitrary guest ACSs. It is safe to assume that a secretly-operating guest ACS has out-of-band means of peer-discovery and component-distribution, as well as obscured means of node-selection.

Active Processors At any given time, it is not obvious whether a specific single or composite MP is *in operation* within a certain channel on a GRR. This is known only to the circuit establisher and owner of the router. According to the threat model (see Section 5.2), the attacker has only a partial view of the network. Besides, due to the geographical distribution of nodes, even if the communication path travels through certain GRRs controlled by the attacker, he will still not be able to gain information about the MPs that are active on other GRRs involved in the path.

5.5.5 Composite Processors

In GRF, composition of MPs is dynamic and at the request of other nodes. Harvesting the full capacity of this feature needs a careful design of guest ACSs and, in particular, a fine break-down in the design of their Guest MPs. This is shown in Figure 5.14 where three permutations of Guest MPs are illustrated.

In permutation (a), which resembles a conventional network, the behaviour of GRRs are fixed and therefore predictable. In permutation (b) the same order of MPs

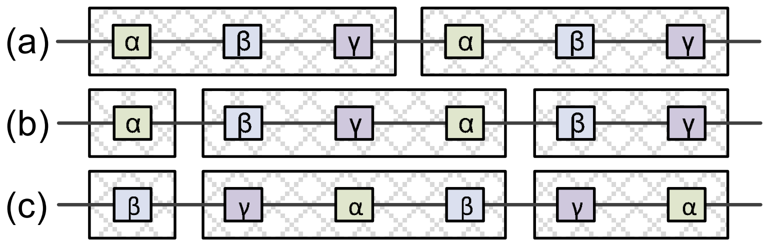


Figure 5.14: Three Possible Permutations of Composite MPs

is distributed among three GRRs, and consequently certainty about the behaviour of the GRRs is decreased. Permutation (c) is the most flexible design where neither the distribution nor the order of MPs is a constraint, and thus the behaviour of the routers is least predictable.

Nevertheless, design and implementation barriers may not allow such perfection. The theoretical limit λ on the possible number of permutations for an ACS with n Guest MPs is calculated according to Equation (5.5).

$$\lambda = n! + \sum_{i=0}^{n-1} \frac{n!}{(n-i)!} \tag{5.5}$$

Note that n solely counts the Guest MPs that can actually be freely distributed among network nodes in any order. A well-balanced distribution of permutations yields a lower chance that attackers can infer the type of ACS in use by analysing the behaviour of the composite MPs.

5.5.6 Predicting the State of the Routers

An active GRR, i.e. a router that sends and receives traffic, can be in a limited number of states listed below. Here the total number of guest ACSs is $\mathcal{N}$, where $\mathcal{N} \geq C + R$; $C \geq 0$, and $R \geq 0$.

$$\begin{cases} \text{1) Communicating through } C \text{ guest ACSs;} \\ \text{2) relaying traffic of } R \text{ guest ACSs; or} \\ \text{3) both 1 and 2.} \end{cases}$$

Inferring through which guest ACSs a node is sending and receiving data requires computing the probabilities of these states. The attacker cannot gain enough knowledge about the distribution of guest ACSs in GRF, and hence is not able to compute the probabilities. This is because GRF unifies and blends the packet structure of all its guest ACSs. Consequently, even if the attackers can monitor the traffic of a certain GRR, he can infer only that the router is connected to GRF but will not be able to identify the guest ACSs to which the router is connected.

In the ideal state, the probability of a GRR being connected to a specific ACS$_{\mathcal{G}}$ is $\frac{1}{\mathcal{N}}$. The attacker may be able to gain knowledge by analysing other traffic properties such as timing and volume. Uncertainty exists only when traffic patterns are indistinguishable. We use $\mathcal{E}_{\mathcal{G}}$ to denote the total number of guest ACSs with traffic properties similar to ACS$_{\mathcal{G}}$, and assume that they are equally likely, i.e. $Pr(\mathcal{G}) = \frac{1}{\mathcal{E}_{\mathcal{G}}}$. Knowledge of attackers about the states of active GRRs can be modelled as shown in be following distribution:

$$\text{Distribution} = \left\{ \prod_{i=1}^{C} \frac{1}{\mathcal{E}_i}, \prod_{i=1}^{R} \frac{1}{\mathcal{E}_i}, \prod_{i=1}^{C} \frac{1}{\mathcal{E}_i} \times \prod_{i=1}^{R} \frac{1}{\mathcal{E}_i} \right\} \tag{5.6}$$

The total number of ACSs with similar traffic patterns, i.e. $\mathcal{E}_i$, includes both the Public and Secret MPs, whereas the degree of similarity with the Secret MPs remains unknown. This is because there are no means to count and compare the Secret MPs unless an attacker can compromise all GRRs. The attacker can thus not have sufficient knowledge about the correct distribution of MPs with similar patterns (i.e. $\mathcal{E}_{\mathcal{G}}$ and $\mathcal{E}_i$).

Distribution (5.6) also implies that an increase in the number of guest ACSs, and particularly an increase in similarity of the traffic patterns of various guest ACSs, yields a lower chance of inferring the type of ACSs in use. The ideal would be to have similar traffic patterns for all guest ACSs (i.e. $\mathcal{E}_{\mathcal{G}} = \mathcal{N}$). In other words, this means that more reusability of Guest and Composite MPs leads to more uncertainty for the attacker. Hence, it is to the benefit of all guest ACSs to join GRF, to make the traffic patterns as similar to other guest ACSs as possible, and to reuse the existing MPs.

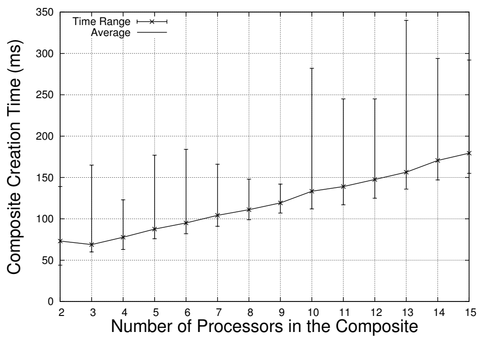


Figure 5.15: Creation of Composite MPs on remote GRRs accessible via LAN

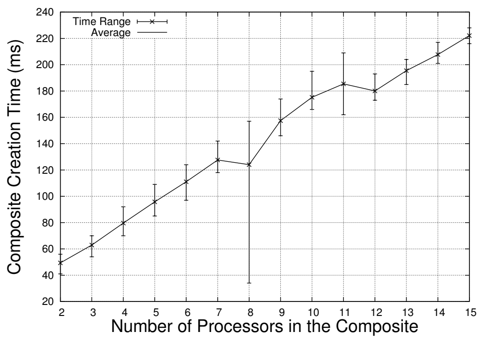


Figure 5.16: Creation of Composite MPs on remote GRRs accessible via loopback

5.6 Experimental Results

This section presents the results obtained from conducting experiments using an actual implementation of GRF. The implementation language is JAVA, and the experiments are conducted using machines with Intel Core 2 Due 2.66 GHz processor, 2 GB physical memory, and Microsoft Windows 7 operating system.

GRF network messages are 512 bytes, which consist of 8 bytes of header and 504 bytes of payload. This does not include the standard packet headers (e.g. TCP and IP). The LAN network of our lab is set up in a star network topology, and the network round-trip delay between any two machines is approximately equal to 1 millisecond. The round-trip delay of the loopback device used in the experiments is less than 1 millisecond. The indicated delays have a trivial impact on our results and are thus disregarded.

GRF builds on the notion of Guest MPs and takes the approach of a modular system that can be extended by additional modules. Ideally, guest ACSs will break their functionality down into a set of finely grained MPs, so that they can be reused in the paths built by other guest ACSs. As almost any nontrivial message processing involves creating Composite MPs, we paid particular attention to testing Composite MPs. To this end, the tests include building communication paths employing various numbers of MPs within each Composite MP inside GRR.

Figure 5.15 shows the average delay caused by creating composite processors based

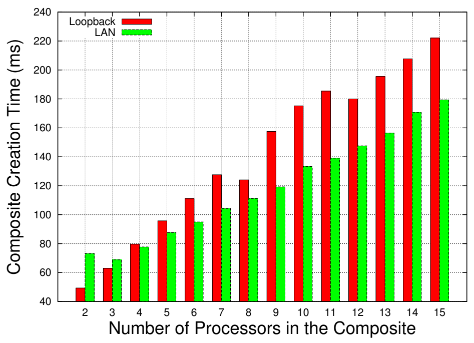


Figure 5.17: Comparison of Delays in Creation of Composite MPs

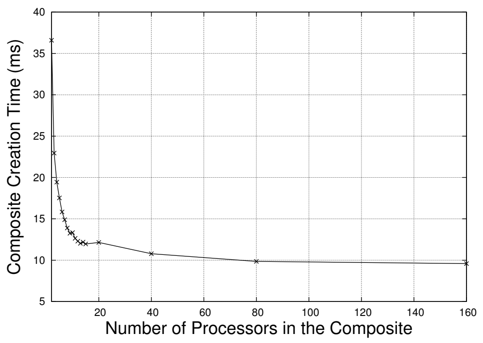


Figure 5.18: Average Delay Caused by Creation of Composite MPs via LAN

on the number of components employed in each composite. The results of a similar experiment are shown in Figure 5.16 where the path establisher used a loopback address to create composites on a local router.

The comparison of the two result sets is presented in Figure 5.17, which shows that creating more complex composite MPs on a local router results in relatively greater delays. These delays are the result of management and sharing of resources which are required to be performed by the operating system of the host machine. Breaking down the delay caused by the existence of each extra MP in a Composite MP is challenging due to the notable communication and JVM overheads. To minimise the overheads, we conducted the experiment with a very large number of MPs (up to 160 in a Composite MP), and present the results in Figure 5.18. Under the conditions of our lab, adding each new processor to the Composite MP adds roughly 9.5 milliseconds to the initial path establishment process.

The experiments are extended by first creating unidirectional paths, and then by creating complete bidirectional circuits that start from Alice, go through n GRRs, reach Bob in the middle, and return through n other GRRs to Alice. Figures 5.19 and 5.20 show the delays in creating the paths and transferring messages through them.

The times necessary for Alice to create each circuit, as well as the round-trip time of the data packet, are shown in Figure 5.21. Finally, we measured the transfer time of six real files over the established circuits, as shown in Figure 5.22. The reported times are calculated from the moment Alice sends the message carrying the first portion of

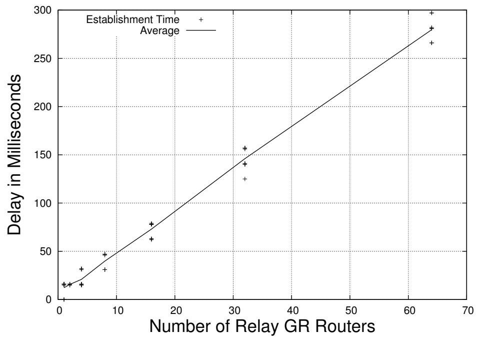


Figure 5.19: Unidirectional Path Establishment

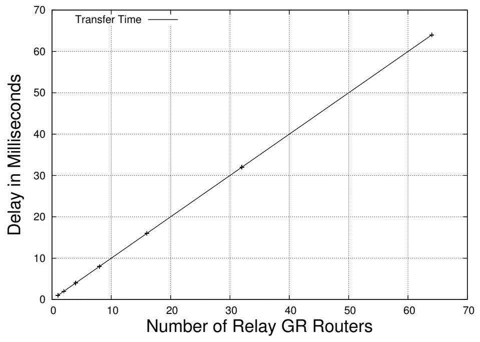


Figure 5.20: Unidirectional Message Transfer

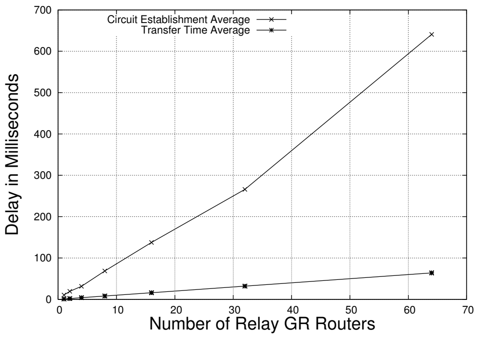


Figure 5.21: Bidirectional Circuit Establishment

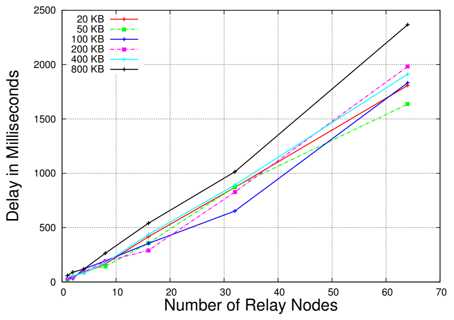


Figure 5.22: File Transfer Rates

the file to the moment Bob receives the last portion.

The conducted experiments show that the implementation of GRF is feasible. It also shows that so long as MPs do not impose excessive delays to the communication paths, the overheads on the data transmission is not significant. We also showed the average delay that is caused by each additional MP in a path. The limitations of our work and future directions will be discussed next.

5.7 Limitations and Future Directions

Design of ACSs involves consideration of many requirements and demands thorough analysis. The framework proposed in this chapter is the first attempt aimed at bringing together various ACSs. The next steps for strengthening the design of GRF are as follows.

Making the Framework Resilient towards Various Attacks The features of GRF which were presented in this chapter focus on making GRF as generic as possible so that it can allow as many guest ACSs to join its platform as possible. Special attention needs be paid to the various types of attacks that may disrupt not the operation of the guest ACSs but that of GRF. This requirement must be highlighted by referring to the fact that exploiting the vulnerabilities of GRF will negatively impact not on just one ACS but perhaps all the guest ACSs. In the next chapter, we focus on certain weaknesses of GRF towards the Denial-of-Service attacks, and note that further analysis to identify and address the potential vulnerabilities of GRF is required.

Further Implementation and Analysis We used an experimental implementation and simple Guest MPs to conduct our tests and analysis. Implementation of a sufficiently generic framework is a challenging and time-consuming undertaking. This chapter focuses on the functionalities provided by the framework and measures its feasibility. In order to build a full-fledged architecture, more research, development, and analysis are required. It is of particular importance to integrate the existing ACSs into GRF, which would allow improving the design of GRF to better match the requirement of real-world systems.

Network Authority Design In a generic framework, issues such as bootstrapping, peer-discovery, trust management, scalability, and network authority's resistance to different attacks, demand special attention. There are many requirements and models to be addressed which are beyond the scope of this chapter.

Large-scale Scalability Analysis Real-life analysis of a global peer-to-peer network can be best conducted with deployments with similar settings. Our analysis is based on the limited resources at our disposal, and this area demands future attention.

Malicious Secret Processors Employing Secret MPs in an ACS is a new approach that introduces more uncertainty about the behaviour of the network. In this chapter, we discussed the potential advantages, threats and counter measures. Nevertheless, at this stage we believe this feature should only be used only for experimental purposes. Further analysis and testing is required before being used in ACSs.

Coexistence of Different Latency Traffics GRF allows the mixture of different latency traffics, while finding the most efficient setting for mixing these traffics demands further work. This issue is closely linked to the need for dummy traffic in some guest ACSs, and reconciling them in a meaningful and efficient way is an interesting area of future work.

5.8 Summary of the Chapter

In order to answer *Research Question 3* of this thesis, we have proposed a generic framework which we named Garbled Routing Framework (GRF).

GRF is a new peer-to-peer overlay network architecture that can host the building blocks of ACSs and thus allow various ACSs to converge. Message Processor (MP) components, and their compositions which can be dynamically deployed on GRRs, are core to hosting various message processing logics in GRF. Dynamic routing mechanisms allow a simple, effective and customisable circuit-establishment process.

A unified framework enables sharing the volunteer routers beyond the traditional boundaries of ACSs, and thus grows the user-bases of all the ACSs that choose to become *guests* within GRF. It also enables code-base sharing that facilitates and accelerates the research and development. GRF enables the coexistence of Public and Secret MPs in a shared infrastructure which can potentially lead to a whole new trend in the design of ACSs.

In addition to the technical details, this chapter presents how GRF should be adopted by the guest ACSs. Routers of GRF offer a significantly less predictable behaviour, and therefore a relatively higher resistance to the attacks. Feasibility of the design is tested through a Java implementation which is used for various circuit establishment and data transmission scenarios. This chapter also discussed the existing challenges and limitations that must be tackled in order to harvest the great potentials of GRF.

Chapter 6

DoS Resistant Circuit Establishment Facility

Every man should know that his conversations, his correspondence, and his personal life are private.
—Lyndon B. Johnson

6.1 Introduction

In the previous chapter we presented our work towards creating a generic framework for anonymous communication. The proposed framework, referred to as Garbled Routing Framework (GRF), enables Anonymous Communication Systems (ACSs) to be hosted in a shared platform. As discussed in Section 5.7, the design of a sufficiently generic and robust system is a major undertaking and many dimensions of such a system require extensive analysis and development.

This chapter covers *Research Question 4* of this thesis which aims to improve the reliability in one aspect of GRF; that is, securing the circuit establishment facility of the GRF against Denial-of-Service (DoS) attacks.

In GRF, routers maintain a collection of data elements referred to as Dynamic Route Processing Table (DRPT). This is used by the routers to associate the incoming messages with the respective Message Processors (MPs) they are destined for. DRPT consists of a set of entries referred to as Route Processing Rule (RPR).

Table 6.1: Sample of Dynamic Route Processing Table (DRPT)

Source	ID	Sequence		Status	Active Time		Change Secret	PC	PID	Processor Params
		Start	End		Start	End				
77.36.192.128	123	1	230	active	10:20	10:30	some-secret-1	SP	7	131.170.40.30
77.36.192.128	332	231	500	active	10:20	10:40	some-secret-1	SP	7	69.147.125.65
202.149.224.128	524	750	900	reserved	10:21	10:07	some-secret-2	SP	7	74.125.237.19
202.149.224.128	221	$*$	$*$	active	$*$	$*$	some-secret-2	SP	7	74.125.237.20

As shown in Table 6.1, each RPR is essentially a filter for the incoming messages as well as a pointer to a set of instructions (i.e. MPs) indicating how the matching messages must be processed. The RPRs are created on a router as per the requests of other nodes in the network, which takes place during the circuit establishment process whereby a node creates a desired path through the available routers.

In the design of GRF which was presented in the previous chapter, requests for the creation of RPR entries can be sent by any node in the network and without any limitations. This allows malicious nodes to launch DoS attacks by sending too many requests for new RPRs and thus exhausting all available resources of the routers.

A sufficiently large number of requests sent by an attacker imposes a significant processing load as well as memory overhead on the receiving router. The abuse of processing power can cause a significant delay in the affected router's response time. The overconsumption of memory can consume the entirety of physical memory on a router and eventually kill that node. In this chapter the vulnerabilities of the circuit establishment facility of GRF to certain DoS attacks are identified, and counter measure techniques are proposed and evaluated to eliminate these threats.

Organisation The rest of this chapter is organised as follows. Section 6.2 presents the details of the identified DoS attacks and their impact on the functionality of GRF. The countermeasure techniques are presented in Sections 6.3–6.5, which are analysed through mathematical modelling. The proposed techniques make GRF resilient against the identified attacks.

6.2 DoS Attacks on Circuit Establishment Facility

The ability of Garbled Routing Routers (GRRs) to create new RPRs, which is exercised on the basis of the requests received from remote routers, can be exploited by attackers to launch DoS attacks on the circuit establishment facility. A malicious node can send a large number of requests for creation of RPRs[1] to other nodes in GRF, causing them to run out of either non-colliding RPR entries or available Random Access Memory (RAM). Consequently, the attacked routers will no longer be able to provide service to the requests of non-malicious network members.

The packet size used in the experimental implementation of GRF is 512 bytes (see Section 5.6 for the details of the experiments) which consists of 8 bytes of header and 504 bytes of payload. If we assume that the creation of each RPR entry involves the transmission of only two network messages (i.e. request and response), then only one message (i.e. the request) can be used to carry all the necessary parameters to set up a new RPR entry.

With the current packet size, this leads to the size of each RPR being $\leq$ 504 bytes. Hence, in order to occupy 1 MB of RAM on a router, an attacker must remotely build more than 2000 RPR entries on that router. Consequently, she needs to exchange more than 4000 messages with the node, which may be understood as the amount of load imposed on the attacker.

The amount of occupied RAM on the routers, based on the number of exchanged messages, can be computed according to Equation (6.1). Here, r_n is the total number of messages the attacker exchanges, m_n is the minimum number of messages involved in the creation of each RPR (i.e. assumed to be equal to 2), m_s is the GRF message size (i.e. assumed to be 512 bytes), and h_s is the GRF header size (i.e. currently 8 bytes).

$$\mathcal{M} = \frac{r_n}{m_n} \cdot (m_s - h_s) \tag{6.1}$$

[1]Requests for creating new RPRs include all the requests with $PID \in \{1, 2, 4\}$. A comprehensive list of PID values is available in Section 5.3.2.

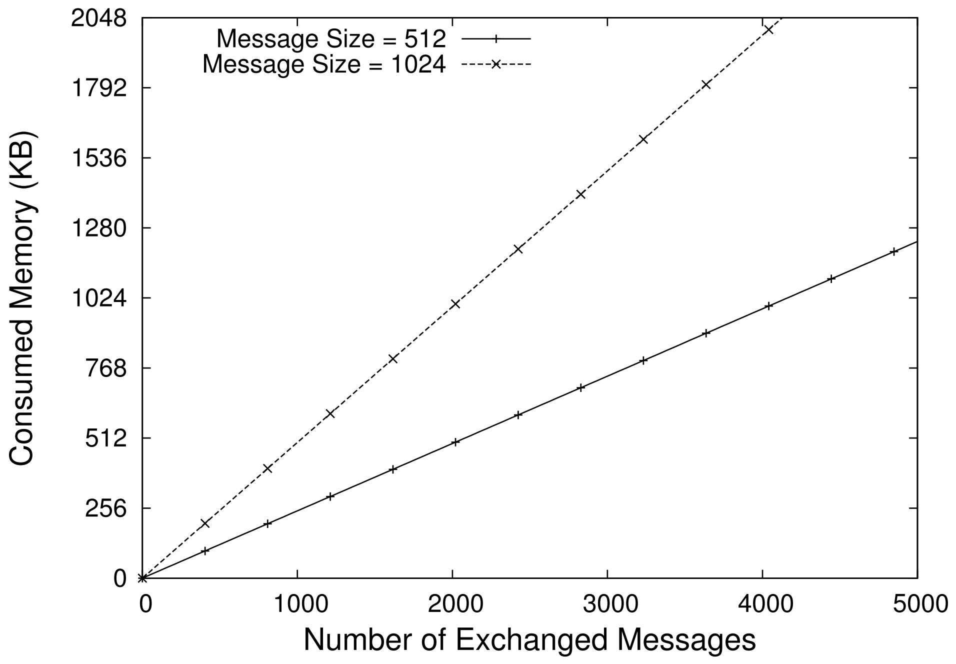


Figure 6.1: Memory consumption on GRRs based on the number and size of the exchanged messages when each RPR is created by one request message.

Figure 6.1 shows the ratio of memory consumption to the number of exchanged messages, assuming a best case scenario where each request leads to the creation of one RPR entry. Here the memory size of each RPR is limited to the capacity of a single GRF message. Nevertheless, it must be noted that 504 bytes of data is not sufficient for building every type of RPR (e.g. Composite MPs). Bigger packet sizes (m_s) can significantly increase memory consumption.

According to Equation (6.1), the load imposed on the attacker based on the number of messages she must exchange (r_n) to occupy a certain amount of memory is:

$$r_n = \frac{\mathcal{M}}{m_s - h_s} \cdot m_n \tag{6.2}$$

This is important as it represents the processing and network power that the attacker must spend in order to consume a certain amount of memory on a remote node.

It must be noted that if multiple nodes attack a single GRR, as shown in Figure 6.2, the amount of consumed memory multiplies by the number of attacker nodes.

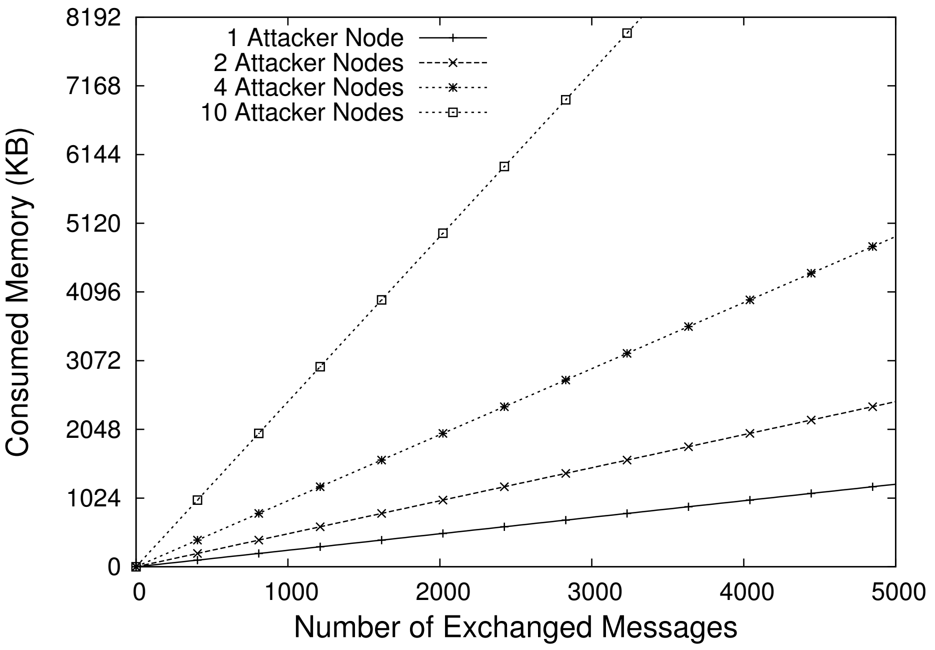


Figure 6.2: Memory consumption on GRRs based on the number of attackers, where each RPR is created by one request message, and message size is 512 bytes.

6.3 Limited Acceptance Based on Source

In the current model, one malicious node can exhaust the resources of a large number of other nodes by continuously sending requests for new RPR entries. As shown in Figure 6.1, this requires that the attacker exchanges many messages with the nodes under attack. Nevertheless, few resources are necessary in order to carry out this attack because the nodes can be targeted sequentially and the attack can be carried out *ad infinitum*.

One method to counter this problem is for GRRs to limit the acceptance of requests for the creation of new RPR entries based on a threshold ($\mathcal{T}$) associated with the source of the GRF messages. This prevents the malicious nodes from being able to exhaust the resources of other nodes by sending the requests directly to the targeted nodes.

Hence, in order to occupy the same amount of memory on a remote node, the attacker must go through an additional $((r_n - m_n)/\mathcal{T}) - 1$ number of intermediary nodes. In this scenario, the attacker must first attempt to create RPR entries on the intermediary nodes, and then use those nodes to send the RPR requests to the

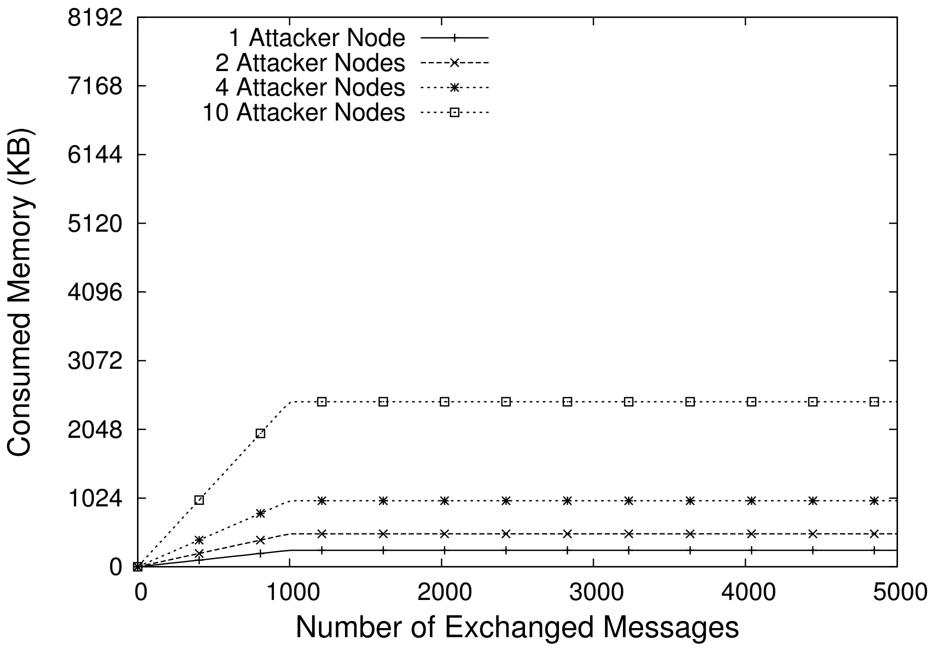


Figure 6.3: Memory consumption on GRRs based on the number of attackers, where each attacker is attacking with the highest capacity, and $\mathcal{T} = 500$.

targeted node. This extra step increases the number of exchanged messages previously shown in the Equation (6.2) to:

$$r'_n = r_n + \left[\left(\frac{r_n - m_n}{\mathcal{T}} - 1 \right) \times 2 \right] \tag{6.3}$$

This limitation affects the memory consumption for RPRs, previously shown in Equation (6.1), as below, where i denotes the number of attackers targeting a single router.

$$\mathcal{M}' = (m_s - h_s) \sum_{k=1}^{i} f(\frac{r_{n_k}}{m_n}) \tag{6.4}$$

$$f(x) = \begin{cases} x & \text{if } x < \mathcal{T} \\ \mathcal{T} & \text{otherwise} \end{cases}$$

The impact of this limitation is to significantly decrease the memory consumption as shown in Figure 6.3, which must be compared with the consumption of the current model which was shown in Figure 6.2. With this improvement, a large number of attacker nodes must focus on a single GR router to exhaust its RAM.

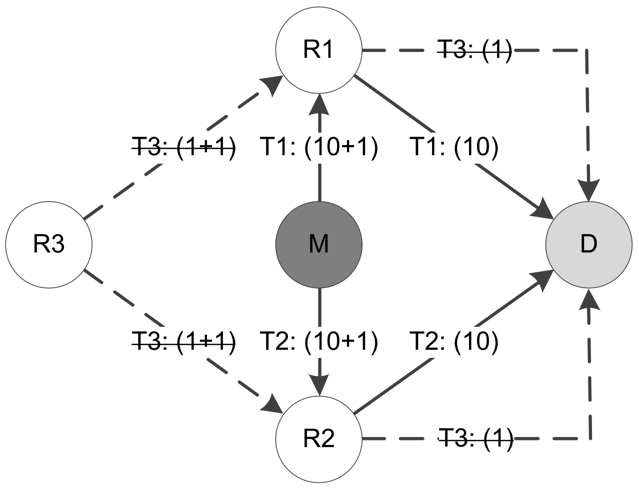


Figure 6.4: Router D allows only 10 RPRs per source $\mathcal{T} = 10$, and M attacks D by creating 20 RPRs via R1 and R2. Further legitimate requests by R1 and R2 will be ignored by D.

Another approach by an attacker is to employ a large number of other routers to send requests to the targeted node. This allows the attacker to perform a different kind of DoS attack which is illustrated in Figure 6.4.

In this attack scenario, the malicious node M sends the requests through two intermediary nodes *R1* and *R2* to the destination node D which is the target of the attack. Router D accepts requests from *R1* and *R2* up to the threshold $\mathcal{T} = 10$ and then ignores further requests from those two nodes. Consequently, at this stage neither *R1* nor *R2* will be able to route their traffic through *D*. Furthermore, any other non-malicious node, such as *R3*, will not be able to create routes on D through *R1* or *R2*.

6.4 Automatic Removal of Unused Paths

Limiting the acceptance of the requests for new RPRs based on their source forces the attacker to use more nodes in the network to launch a DoS attack. However, this is insufficient to fully eliminate the threat of DoS attacks on the circuit establishment facility. Moreover, depending on the value assigned to the threshold $\mathcal{T}$, the attacker may still be able to waste a notable amount of RAM on GRF routers because she

Table 6.2: Memory Consumption after Automatic Removal of Unused RPRs

Time (t)	1	2	3	4	5
Message (r_{n_t})	1000	1000	1000	1000	1000
New RPRs	500	250	125	62	31
Total RPRs	500	750	825	887	918
Memory (KB)	246.09	369.14	406.05	436.57	451.82

incurs little processing and transmission costs to do so.

The situation can be improved by automatically removing unused RPR entries from the GRF routers. To bypass this limit, the attacker is forced to not only create new RPRs on the targeted routers, but also maintain the existing RPRs by regularly transferring data through those nodes.

In order to understand the effect of this limitation, the requests sent by the attacker need to be analysed according to the maximum amount that can be sent within a certain period. We assume that the attacker can send approximately r_{n_t} number of requests per a certain time interval t. For the duration of T intervals, the amount of consumed memory can be calculated as shown in Equation (6.5). Here, the $f(t)$ denotes the number of messages necessary to maintain the dummy RPR entries during the interval t.

$$\mathcal{M}' = (m_s - h_s) \sum_{t=1}^{T} \frac{r_{n_t} - f(t)}{m_n} \tag{6.5}$$

$$f(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ \frac{r_{n_{(x-1)}}}{m_n} + f(x-1) & \text{otherwise} \end{cases}$$

For example, if the attacker can send 1000 new messages per each time interval t, as shown in Table 6.2, a growing proportion of these messages must always be dedicated to maintaining the existing RPRs and prevent their auto-removal.

This yields fewer new RPR entries that the attacker is able to create in each time interval and thus limits the total amount of memory that can be consumed by the attacker. Figure 6.5 shows the significant decrease in the memory usage even when multiple attackers target one router.

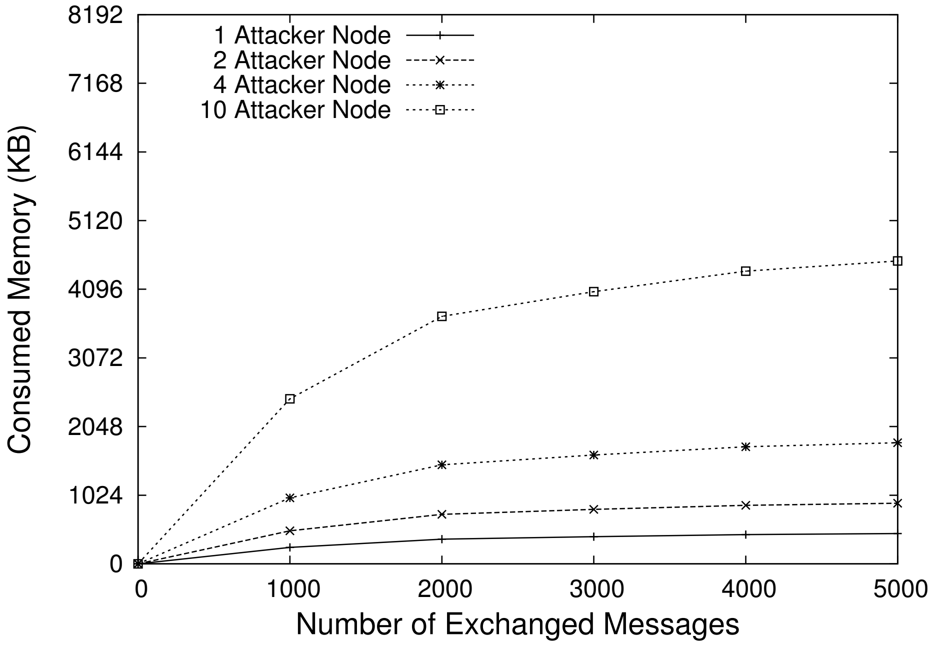


Figure 6.5: Memory consumption on GRRs based on the number of attackers, where attackers can spend up to 1000 requests per time interval $r_{n_t} = 1000$, after automatic removal of unused RPRs.

It must be noted that this technique forces the attacker to keep flooding the targeted node with messages in order to maintain the dummy RPR entries. Consequently, this approach limits the maximum number of nodes an attacker can attack at any given time.

Figure 6.6 shows the difference between this technique and the original design, where attackers keep sending 1000 messages per time interval and up to 5 intervals ($T = 5$). As shown in the figure, the automatic removal of the unused RPR entries causes the occupied memory to be released shortly after the attack stops.

This automatic removal of unused paths also addresses the DoS attack which was shown in Figure 6.4 because creating indirect RPR entries do not help the attacker to maintain the dummy entries on the attacked node. However, the shortcoming of this approach is that the value of the time threshold $\mathcal{T}$ is very sensitive. That is, a relatively strong attacker with sufficient resources can overflow a GRR before the auto-removal algorithm can detect the unused RPR entries. In this case, the GRR may completely fail and not be able to recover by eliminating the dummy RPRs.

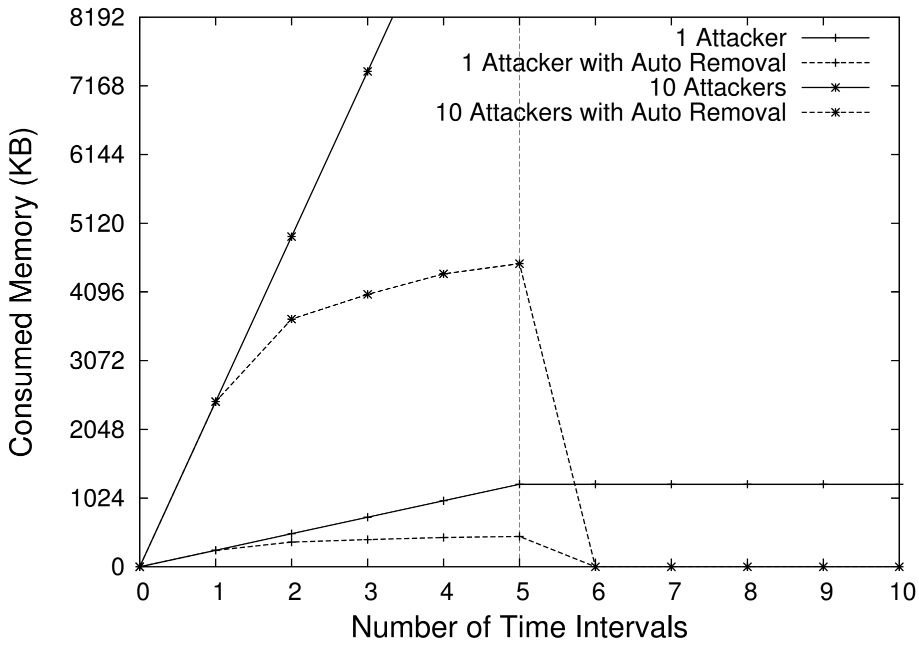


Figure 6.6: Memory consumption on GRRs when attackers send 1000 requests for 5 intervals, and then stop.

Choosing a very small $\mathcal{T}$ will not address the problem because then the RPRs may expire and be removed even before the path is fully established or the actual data starts flowing. Moreover, in order to reduce the delay caused by the circuit establishment, the establisher may attempt to create multiple reserved paths to make them readily available to use. This technique has been employed in the implementation of Tor proxy [26]. A very small $\mathcal{T}$ value will also hamper such delay-avoiding techniques.

6.5 Caching Active Route Processing Rules

To counter the issues discussed in the previous section, the RPR entries must be stored on Hard Disk Drive (HDD) as opposed to RAM. With this approach, even if the attacker is able to overflow a router with a large number of requests prior to the activation of auto-removal algorithm, the dummy RPR entries only waste some HDD space. Since the amount of available HDD space is usually significantly greater than RAM, this is a waste that all routers can easily bear.

Storing RPRs on HDD naturally introduces extra lag when the routers are

Table 6.3: Usage of RAM after Caching

Time (t)	1	2	3	4	5
Message (r_{n_t})	1000	1000	1000	1000	1000
New RPRs	500	250	125	62	31
Total RPRs	500	750	825	887	918
HDD (KB)	246.09	369.14	406.05	436.57	451.82
RAM (KB)	0	0	246.09	369.14	406.05

matching the incoming messages against the RPRs. To address this issue, routers need to cache the frequently used RPRs; that is, those which receive more traffic more often. The caching technique reduces the time required to access these active entries while the dummy RPRs created by the attackers will remain in HDD and be automatically removed.

In order to prevent the dummy RPRs from auto-removal, the attacker must keep sending r_d number of dummy messages, for each individual entry, for the duration of T_c intervals, before the router loads them into RAM (i.e. caches them). The amount of occupied RAM can be calculated according to the following equation that differs from Equation (6.5) in the definition of $f(x)$.

$$\mathcal{M}' = (m_s - h_s) \sum_{t=1}^{T} \frac{r_{n_t} - f(t)}{m_n} \tag{6.6}$$

$$f(x) = \begin{cases} 0 & \text{if } x \leq T_c \\ \left(\frac{r_{n(x-1)}}{m_n} \cdot r_d \right) + f(x-1) & \text{otherwise} \end{cases}$$

Table 6.3 shows the impact of this technique on the consumption of HDD and RAM, where the GRR expects to receive one data message per interval for each existing RPR ($r_d = 1$) for at least two consecutive intervals ($T_c = 2$). With an increase in the number of expected messages per interval, i.e. $r_d > 1$, more of an attacker's messages must be devoted to renewing the dummy RPRs; and thus the attacker will have less chance to create new RPRs.

This technique is also resistant to Distributed Denial-of-Service (DDoS) attacks, whereby multiple attackers simultaneously target one router. As shown in Figure 6.7, irrespective of the amount of resources at the attacker's disposal, or the number of

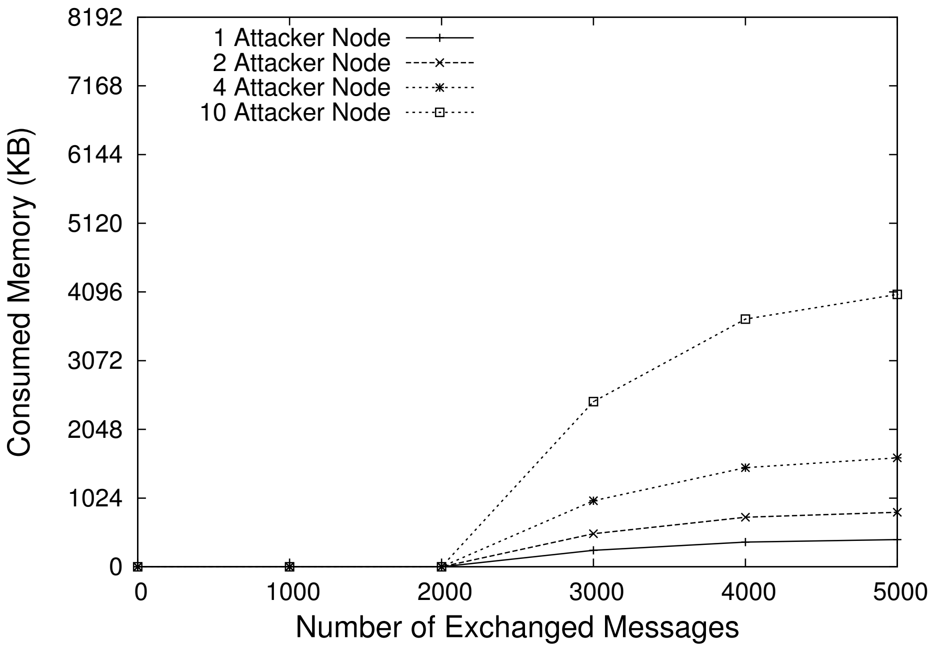


Figure 6.7: RAM consumption on GRRs based on the number of attackers, where attackers can spend up to 1000 requests per time interval $r_{n_t} = 1000$, after caching only the active RPRs.

simultaneous attacks, there is actually zero memory consumption for the period of T_c intervals. This provides the time required for the automatic removal algorithm to be able to identify and remove the dummy RPRs.

Note that the caching technique does not enforce automatic removal of the RPRs and thus the occupied memory will not be freed if the attack is stopped. This is unlike the effect of the auto-removal technique which was shown in Figure 6.6.

6.6 Summary of the Chapter

In order to answer *Research Question 4* of this thesis, this chapter has analysed the circuit establishment facility of the anonymous communication framework which we proposed in the Chapter 5. The analysis aimed to identify the weaknesses of that facility against DoS and DDoS attacks.

We identified a number of weaknesses that an attacker is able to exploit in order to disrupt the circuit establishment facility of GRF. In this chapter we proposed three techniques to address these weaknesses, and evaluated the impact of each technique on the consumption of the resources of framework routers while under attack.

The first technique aims to limit the number of acceptable requests for new RPRs based on the source of the messages. Consequently, a limit is enforced on the ability of the attacker to directly exploit the targeted router. Secondly, we proposed that the routers automatically remove unused RPRs, which forces the attacker to keep sending an additional and large number of messages to maintain the previously-created RPRs. The third technique is to store the RPRs in HDD as opposed to RAM and employs caching to cure the resultant delay.

Employing a combination of these techniques enables a GRR to resist the DoS and DDoS attack aimed to destabilise the circuit establishment facility of GRF. The generic anonymous communication framework proposed in this thesis is the first attempt to build such a system and therefore requires further research and development to become a fully-fledged platform. In this chapter we aimed at improving the reliability of the circuit establishment facility. Analyses of the weaknesses of GRF towards similar and other attacks are venues for future work.

Chapter 7

Conclusion

Information is power.

Anonymous communication is one of the fundamental requirements in systems that must protect the privacy of users, and has a wide range of applications such as electronic voting schemes, anonymous storage and retrieval systems and anonymous remailers. Systems that provide anonymous communications, known as Anonymous Communication Systems (ACSs), constitute an important part of Privacy Enhancing Technologies (PETs) and are widely used to protect against surveillance, to circumvent Internet censorship, to gain and retain freedom of speech, or to otherwise protect the privacy of users against arbitrary interference.

In the absence of dedicated ACSs, the existence of metadata along with leaks from the communication layer can disclose a great deal of private information about Internet users. Mix-based ACSs, otherwise known as *mix systems* or *mixnets*, are an important family of systems that provide protection against such unintended disclosures. In this thesis we have paid particular attention to *mix systems* and took a step towards making them more secure and efficient.

Firstly, we revisited one of the most important mix designs; namely, Binomial Mix. We examined the security of that mix against a passive attacker. To that end, we developed a simulator using Java, which we then used to simulate the probabilistic passive attack known to be effective against Binomial Mix. Our study focused particularly on the behaviour of the mix under various traffic conditions relative to the capacity of the mix. We defined multiple scenarios with the ratio of the incoming traffic ranging from 10% to 90% of the capacity and measured the attacker's success rate in every traffic condition. Analysing the results showed that the previously reported probabilistic passive attack is significantly less effective than is believed. Observation of the results also led to the discovery that additional, but so far neglected, information is available to the adversary that can be used to improve the reported attack. We presented the rates of success that the attacker can achieve under various traffic conditions, and reported the distribution of false-positive results which may be used to build a more efficient attack.

Secondly, we attempted to improve the design of Binomial Mix in terms of security and efficiency, and proposed two new mix designs. These mixes run multiple *selection algorithms*, each of which chooses messages from the pool according to a Binomial distribution. An independent bias function operates in association with each selection algorithm. Therefore, the behaviour of the mixes is not closely linked to the known behaviour of any single mathematical function. The first design, Multi-Binomial Shared-Pool Mix (MBSP Mix) executes one selection algorithm at a time and has a single pool of messages shared amongst all coexisting selection algorithms. The second proposal, Multi-Binomial Independent-Pool Mix (MBIP Mix), executes multiple selection algorithms simultaneously, and contains a separate pool of messages associated to each selection algorithm. We measured the efforts of both active and passive attackers on MBSP Mix and MBIP Mix and compared the results with that of the attackers on Binomial Mix. Both mixes show a significantly more secure build against both active and passive attacks. It is also shown that these designs are robust even against *blending attacks* which are notoriously difficult to foil. We further evaluated these designs by comparing delays imposed on the communication channel and showed that, depending on the combination and properties of coexisting selection algorithms, both mixes can achieve significantly less delay than that resulting from the operation of Binomial Mix. We examined a variety of different coexisting selection algorithms and reported their delays in various traffic conditions.

We then turned our attention to the operation of mix systems as a whole, and proposed a framework for building mix-based ACSs. The framework, referred to as Garbled Routing Framework (GRF), provides various facilities required by mix systems to build arbitrary circuits in the network, to route traffic, and to perform arbitrary processing of the messages in transit. Each ACS can use the facilities of GRF by meeting certain architectural requirements and can then join the platform as a collection of plug-in components. In order to showcase the generic nature of the design of GRF, we modelled the operation of two well-known ACSs which serve different purposes; namely, Tor and Mixmaster. We also analysed and discussed the security aspects of GRF such as its susceptibility to distribution of malicious code in the system and the likelihood of success of various attacks after a specific mix system adopts the design of GRF. Additionally, we proved the feasibility of the design by developing a Java-based implementation, and conducted tests for establishing various types of circuits and transferring data through the system.

Lastly, we attended to one of the important security challenges of GRF, that is, the Denial-of-Service (DoS) and the distributed counterpart, the Distributed Denial-of-Service (DDoS) attacks. We considered how the circuit establishment facility of GRF can be made resistant towards the DoS attacks. Noting that malicious network nodes are able to launch both DoS and DDoS attacks on GRF and put the targeted routers under undue load and pressure, we first identified the weaknesses of GRF which would allow such attacks. We then proposed certain techniques that, if implemented by the routers of GRF, will fully equip them against the identified DoS and DDoS attacks. We evaluated the impacts of the proposed techniques through mathematical modelling of the attack situation before and after employing the proposed techniques.

Limitations and Future Directions

Where we evaluated the design of Binomial Mix, we used the definition of probabilistic passive attack previously reported in the literature. Our work led to the discovery of additional information that the attacker may use to improve the attack; namely, the predictable distribution of false-positive results.

As an interesting future work, one could consider building models to utilise this additional knowledge in order to improve the effectiveness and accuracy of the passive attack. Such an improvement is likely to require fewer rounds of observations, which translates into less effort required by a passive attacker to compromise the anonymity provided by Binomial Mix.

As our experiments with Binomial Mix showed, the attacks known to be effective on mixes may have very different rates of success depending on traffic conditions. Therefore, it is important to examine the impact of traffic volume on other selection algorithms of Binomial Mix (e.g. not based on normal CDF), other mix designs, and against attacks of a different nature (e.g. active attacks).

On a related topic, we proposed the design of MBIP Mix, which allows mixing high- and low-latency traffic by hosting coexisting selection algorithms with corresponding properties. We have provided multiple examples of such coexistence. As another interesting future work, one may consider finding an optimal combination of selection algorithms and their corresponding properties to coexist within an MBIP Mix. An optimal design would offer a high degree of anonymity as well as a low latency. In the context of MBSP Mix, the same question can be asked in relation to communications with only one kind of latency.

Finally, we note that building a fully-fledged framework for generic anonymous communication is a major undertaking and involves consideration of various and many reliability, scalability and security aspects. Our work is a step towards creating such a system. The possibilities for future work in this domain are numerous. For example, one could analyse other facilities of our proposed framework (e.g. routing, design of routers and services of the network authority) and study their resilience against a variety of attacks common to ACSs. It is also necessary to build components for the existing mix systems (e.g. Tor and I2P) enabling them to migrate to the framework.

Another aspect of our proposed framework is that it allows *secret* message-processing techniques to operate alongside the publicly-known solutions. At this stage, we considered this feature to be experimental and only note its potential benefits; namely, the uncertainty it adds to the behaviour of the routers, which further impedes traffic analyses. More analyses and research work are required to understand and evaluate the impact of this feature so as to inform its further development.

We hope that our results stimulate further research in this domain and guide future endeavours.

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Consulted*, 1(2012):28, 2008.

[2] Uber. https://www.uber.com/. [Accessed April 2015].

[3] Andreas Pfitzmann and Marit Köhntopp. Anonymity, Unobservability, and Pseudonymity A Proposal for Terminology. In *Designing Privacy Enhancing Technologies*, pages 1–9, 2001.

[4] Andreas Pfitzmann and Marit Hansen. A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf, August 2010.

[5] J. Ren and J. Wu. Survey on Anonymous Communications in Computer Networks. *Computer Communications*, 33(4):420–431, 2010.

[6] Roger Dingledine and Nick Mathewson. Anonymity Loves Company: Usability and the Network Effect. In *Workshop on the Economics of Information Security (WEIS 2006)*, 2006.

[7] Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the Economics of Anonymity. In *Financial Cryptography and Data Security*, 2003.

[8] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24:84–90, 1981.

[9] K. Sampigethaya and R. Poovendran. A Survey on Mix Networks and Their Secure Applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006. ISSN 0018-9219. doi: 10.1109/JPROC.2006.889687.

[10] George Danezis, Claudia Diaz, and Paul Syverson. Systems for Anonymous Communication. *Handbook of Financial Cryptography and Security, Cryptography and Network Security Series*, pages 341–389, 2009.

[11] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead. In *Kommunikation in Verteilten Systemen*, pages 451–463. Springer, 1991.

[12] Anja Jerichow, Jan Muller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4):495–509, 1998.

[13] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In *Designing Privacy Enhancing Technologies*, pages 115–129. Springer, 2001.

[14] JAP. http://anon.inf.tu-dresden.de/. [Accessed April 2015].

[15] Sameer Parekh. Prospects for Remailers: Where is Anonymity Heading on the Internet? http://firstmonday.org/ojs/index.php/fm/article/view/476/397, August 1996.

[16] Lance Cottrell. Mixmaster and Remailer Attacks, 1994.

[17] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol Version 2 IETF Internet Draft. https://tools.ietf.org/html/draft-sassaman-mixmaster-03, 2004.

[18] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Symposium on Security and Privacy*, pages 2–15. IEEE, 2003.

[19] Mixminion's Official Website. http://mixminion.net/. [Accessed April 2015].

[20] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-Go MIXes: Providing Probabilistic Anonymity in an Open System. In *Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[21] Roger Dingledine, Andrei Serjantov, and Paul Syverson. Blending Different Latency Traffic with Alpha-Mixing. In *Workshop on Privacy Enhancing Technologies (PET 2006)*, pages 245–257. Springer, 2006.

[22] Marc Rennhard and Bernhard Plattner. Practical Anonymity for the Masses with MorphMix. In *Financial Cryptography and Data Security*, pages 233–250. Springer-Verlag, LNCS 3110, 2004.

[23] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer Based Anonymous Internet Usage with Collusion Detection. In *ACM Workshop on Privacy in the Electronic Society*, pages 91–102. ACM, 2002.

[24] Claudia Diaz and Andrei Serjantov. Generalising Mixes. In *Privacy Enhancing Technologies Workshop*, pages 18–31. Springer-Verlag, LNCS 2760, 2003.

[25] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous Batching: From Cascades to Free Routes. In *Privacy Enhancing Technologies*, pages 186–206. Springer, 2005.

[26] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21. USENIX Association, 2004.

[27] Bassam Zantout and Ramzi Haraty. I2P Data Communication System. In *The Tenth International Conference on Networks*, pages 401–409, 2011.

[28] The Invisible Internet Project. https://geti2p.net/, . [Accessed April 2015].

[29] U.S., British Intelligence Mining Data from Nine U.S. Internet Companies in Broad Secret Program. http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html. [Accessed April 2015].

[30] OpenNet Initiative: Internet Filtering and Surveillance. https://opennet.net/. [Accessed April 2015].

[31] Joss Wright, Susan Stepney, John A Clark, and Jeremy Jacob. Designing Anonymity: A Formal Basis for Identity Hiding. *Internal Yellow Report, York University, York, UK*, 2004.

[32] IEEE Xplore. http://ieeexplore.ieee.org. [Accessed April 2015].

[33] Freehaven Bibliography. http://freehaven.net/anonbib/full/date.html. [Accessed April 2015].

[34] Andreas Ptfizmann and Michael Waidner. Networks without User Observability — Design Options. In *Advances in Cryptology (EUROCRYPT'85)*, 1986.

[35] Michael Waidner. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. In *Advances in Cryptology (EUROCRYPT'89)*, pages 302–319. Springer, 1989.

[36] David A Cooper and Kenneth P Birman. Preserving Privacy in a Network of Mobile Computers. In *IEEE Symposium on Security and Privacy*, pages 26–38. IEEE, 1995.

[37] Pan Wang, Peng Ning, and Douglas S Reeves. A k-Anonymous Communication Protocol for Overlay Networks. In *ACM Symposium on Information, Computer and Communications Security*, pages 45–56. ACM, 2007.

[38] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05): 557–570, 2002.

[39] Pierangela Samarati. Protecting Respondents Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[40] Traian Marius Truta and Bindu Vinay. Privacy Protection: p-Sensitive k-Anonymity Property. In *ICDE Workshops*, page 94, 2006.

[41] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy beyond k-Anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[42] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-Closeness: Privacy beyond k-Anonymity and l-Diversity. In *IEEE International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[43] David Rebollo-Monedero, Jordi Forne, and Josep Domingo-Ferrer. From t-Closeness-Like Privacy to Postrandomization via Information Theory. *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1623–1636, 2010.

[44] Justin Brickell and Vitaly Shmatikov. The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 70–78. ACM, 2008.

[45] Cynthia Dwork. Differential Privacy. In *Encyclopedia of Cryptography and Security*, pages 338–340. Springer, 2011.

[46] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, 2000.

[47] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[48] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[49] Claude Elwood Shannon. A Mathematical Theory of Communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[50] Andrei Serjantov and George Danezis. Towards an Information Theoretic Metric for Anonymity. In *Privacy Enhancing Technologies*, pages 259–263. Springer, 2003.

[51] Claudia Dıaz, Joris Claessens, Stefaan Seys, and Bart Preneel. Information Theory and Anonymity. In *Symposium on Information Theory in the Benelux*, pages 179–186, 2002.

[52] Yong Guan, Xinwen Fu, Riccardo Bettati, and Wei Zhao. An Optimal Strategy for Anonymous Communication Protocols. In *International Conference on Distributed Computing Systems*, pages 257–266. IEEE, 2002.

[53] Sandra Steinbrecher and Stefan Köpsell. Modelling Unlinkability. In *Privacy Enhancing Technologies*, pages 32–47. Springer, 2003.

[54] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring Anonymity Revisited. In *Nordic Workshop on Secure IT Systems*, pages 85–90. Espoo, Finland, 2004.

[55] Gergely Tóth and Zoltán Hornák. Measuring Anonymity in a Non-Adaptive, Real-Time System. In *Privacy Enhancing Technologies*, pages 226–241. Springer, 2005.

[56] Vitaly Shmatikov and Ming-Hsiu Wang. Measuring Relationship Anonymity in Mix Networks. In *ACM Workshop on Privacy in Electronic Society*, pages 59–62. ACM, 2006.

[57] Sebastian Clauß and Stefan Schiffner. Structuring Anonymity Metrics. In *ACM Workshop on Digital Identity Management*, pages 55–62. ACM, 2006.

[58] Paul F Syverson and Stuart G Stubblebine. *Group Principals and the Formalization of Anonymity.* Springer, 1999.

[59] Sjouke Mauw, Jan HS Verschuren, and Erik P de Vink. A Formalization of Anonymity and Onion Routing. In *Computer Security (ESORICS 2004)*, pages 109–124. Springer, 2004.

[60] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. A Model of Onion Routing with Provable Anonymity. In *Financial Cryptography and Data Security*, pages 57–71. Springer, 2007.

[61] Matthew Edman, Fikret Sivrikaya, and Bülent Yener. A Combinatorial Approach to Measuring Anonymity. In *Intelligence and Security Informatics*, pages 356–363. IEEE, 2007.

[62] Benedikt Gierlichs, Carmela Troncoso, Claudia Diaz, Bart Preneel, and Ingrid Verbauwhede. Revisiting a Combinatorial Approach toward Measuring Anonymity. In *ACM Workshop on Privacy in the Electronic Society*, pages 111–116. ACM, 2008.

[63] David Rebollo-Monedero, Javier Parra-Arnau, Claudia Diaz, and Jordi Forné. On the Measurement of Privacy as an Attackers Estimation Error. *International Journal of Information Security*, 12(2):129–149, 2013.

[64] Alastair R Beresford and Frank Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[65] Jing Deng, Richard Han, and Shivakant Mishra. Intrusion Tolerance and Anti-Traffic Analysis Strategies for Wireless Sensor Networks. In *International Conference on Dependable Systems and Networks*, pages 637–646. IEEE, 2004.

[66] Marco Gruteser and Dirk Grunwald. Anonymous Usage of Location-Based Services through Spatial and Temporal Cloaking. In *International Conference on Mobile Systems, Applications and Services*, pages 31–42. ACM, 2003.

[67] Giuseppe Ateniese, Amir Herzberg, Hugo Krawczyk, and Gene Tsudik. Untraceable Mobility or How to Travel Incognito. *Computer Networks*, 31(8): 871–884, 1999.

[68] Qi He, Dapeng Wu, and Pradeep Khosla. The Quest for Personal Control over Mobile Location Privacy. *IEEE Communications Magazine*, 42(5):130–136, 2004.

[69] Didier Samfat, Refik Molva, and N Asokan. Untraceability in Mobile Networks. In *Annual International Conference on Mobile Computing and Networking*, pages 26–36. ACM, 1995.

[70] Jiejun Kong, Xiaoyan Hong, Medy Yahya Sanadidi, and Mario Gerla. Mobility Changes Anonymity: Mobile Ad-hoc Networks Need Efficient Anonymous Routing. In *IEEE Symposium on Computers and Communications (ISCC 2005), pages=57–62, year=2005, organization=IEEE.*

[71] Anonymizer. https://anonymizer.com/, . [Accessed April 2015].

[72] StrongVPN. https://www.strongvpn.com/. [Accessed April 2015].

[73] SwissVPN. http://www.swissvpn.net/. [Accessed April 2015].

[74] Jessica A Wood. The Darknet: A Digital Copyright Revolution. *Rich. JL & Tech.*, 16:1, 2009.

[75] Steve Mansfield-Devine. Darknets. *Computer Fraud & Security*, 2009(12):4–6, 2009.

[76] Peter Biddle, Paul England, Marcus Peinado, Bryan Willman, et al. The Darknet and the Future of Content Distribution. In *ACM Workshop on Digital Rights Management*, volume 6, page 54, 2002.

[77] BrightPlanet. Understanding the Deep Web in 10 Minutes. White paper, 2013.

[78] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Simplification and Analysis of Transitive Trust Networks. *Web Intelligence and Agent Systems*, 4(2):139–161, 2006.

[79] RetroShare: Friend-2-Friend and Secure Decentralised Communication Platform. http://retroshare.sourceforge.net/. [Accessed April 2015].

[80] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-Preserving P2P Data Sharing with OneSwarm. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 111–122. ACM, 2010.

[81] OneSwarm: Privacy Preserving Peer-to-Peer Data Sharing. http://www.oneswarm.org/. [Accessed April 2015].

[82] GigaTribe: Private File Sharing. http://www.gigatribe.com/. [Accessed April 2015].

[83] Infinit: Private File Sharing. https://infinit.io/. [Accessed April 2015].

[84] n2n: A Layer Two Peer-to-Peer VPN. http://www.ntop.org/products/n2n/. [Accessed April 2015].

[85] Anonet: A Decentralized IP Based Darknet. http://www.ntop.org/products/n2n/, . [Accessed April 2015].

[86] Lantern: Anti-censorship Peer-to-Peer Routing Network. https://getlantern.org/. [Accessed April 2015].

[87] Philippe Golle and Ari Juels. Dining Cryptographers Revisited. In *Advances in Cryptology-Eurocrypt*, pages 456–473. Springer, 2004.

[88] Yong Guan, Xinwen Fu, Riccardo Bettati, and Wei Zhao. A Quantitative Analysis of Anonymous Communications. *IEEE Transactions on Reliability*, 53(1):103–115, 2004.

[89] Claudia Diaz, George Danezis, Christian Grothoff, Andreas Pfitzmann, and Paul Syverson. Panel Discussion — Mix Cascades versus Peer-to-Peer: Is One Concept Superior? In *Privacy Enhancing Technologies*, pages 242–242. Springer, 2005.

[90] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The Disadvantages of Free MIX Routes and How to Overcome Them. In *Designing Privacy Enhancing Technologies*, pages 30–45. Springer, 2001.

[91] George Danezis. Mix-Networks with Restricted Routes. In *Privacy Enhancing Technologies*, pages 1–17. Springer, 2003.

[92] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a Trickle to a Flood: Active Attacks on Several Mix Types. In *Information Hiding*, pages 36–52. Springer, 2003.

[93] O. Berthold, H. Federrath, and M. Köhntopp. Project "Anonymity and Unobservability in the Internet". In *Conference on Computers, Freedom and Privacy: Challenging the Assumptions*, pages 57–65. ACM, 2000.

[94] Oliver Berthold and Heinrich Langos. Dummy Traffic against Long Term Intersection Attacks. In *Privacy Enhancing Technologies*, pages 110–128. Springer, 2003.

[95] Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between Two Practical Mix Designs. In *European Symposium On Research in Computer Security (ESORICS)*, pages 141–159. Springer, 2004.

[96] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Do Dummies Pay off? Limits of Dummy Traffic Protection in Anonymous Communications. In *Privacy Enhancing Technologies*, pages 204–223. Springer, 2014.

[97] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient Anonymous Channel and All/Nothing Election Scheme. In *Advances in Cryptology*, pages 248–259. Springer, 1994.

[98] Birgit Pfitzmann. Breaking an Efficient Anonymous Channel. In *Advances in Cryptology (EUROCRYPT'94)*, pages 332–340. Springer, 1995.

[99] Kazue Sako and Joe Kilian. Receipt-Free MixTtype Voting Scheme. In *Advances in Cryptology (EUROCRYPT95)*, pages 393–403. Springer, 1995.

[100] Markus Michels and Patrick Horster. Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme. In *Advances in Cryptology (ASIACRYPT'96)*, pages 125–132. Springer, 1996.

[101] Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault Tolerant Anonymous Channel. *Information and Communications Security*, pages 440–444, 1997.

[102] Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers, author=Abe, Masayuki, booktitle=Advances in Cryptology (EUROCRYPT'98), pages=437–447, year=1998, publisher=Springer.

[103] Markus Jakobsson. A Practical Mix. In *Advances in Cryptology (EUROCRYPT'98)*, pages 448–461. Springer, 1998.

[104] Yvo Desmedt and Kaoru Kurosawa. How to Break a Practical MIX and Design a New One. In *Advances in Cryptology (EUROCRYPT 2000)*, pages 557–572. Springer, 2000.

[105] Markus Jakobsson. Flash Mixing. In *Annual ACM Symposium on Principles of Distributed Computing*, pages 83–89. ACM, 1999.

[106] Jun Furukawa and Kazue Sako. An Efficient Scheme for Proving a Shuffle. In *Advances in Cryptology (CRYPTO 2001)*, pages 368–387. Springer, 2001.

[107] C Andrew Neff. A Verifiable Secret Shuffle and Its Application to E-Voting. In *ACM Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[108] David Mazieres and M Frans Kaashoek. The Design, Implementation and Operation of an Email Pseudonym Server. In *ACM Conference on Computer and Communications Security*, pages 27–36. ACM, 1998.

[109] Ceki Gulcu and Gene Tsudik. Mixing E-mail with Babel. In *Symposium on Network and Distributed System Security*, pages 2–16. IEEE, 1996.

[110] I2P-Bote: A Fully Decentralized and Distributed Email System. http://i2pbote.i2p.us/, . [Accessed April 2015].

[111] Pere Manils, Chaabane Abdelberri, Stevens Le Blond, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. Compromising Tor Anonymity Exploiting P2P Information Leakage. *arXiv preprint arXiv:1004.1461*, 2010.

[112] GNUnet: GNU's Framework for Secure Peer-to-Peer Networking. https://gnunet.org/. [Accessed April 2015].

[113] Perfect Dark: Secure Peer-to-Peer File Sharing. http://perfectdark.benri-tool.net/. [Accessed April 2015].

[114] StegoShare: Steganographic File Sharing System. http://stegoshare.sourceforge.net/. [Accessed April 2015].

[115] Ronald Deibert. *Access Denied: The Practice and Policy of Global Internet Filtering.* MIT Press, 2008.

[116] Ronald Deibert, John Palfrey, Rafal Rohozinski, Jonathan Zittrain, and Miklos Haraszti. *Access Controlled: The Shaping of Power, Rights, and Rule in Cyberspace.* MIT Press, 2010.

[117] Ronald Deibert, Jonathan L Zittrain, John Palfrey, and Rafal Rohozinski. *Access Contested: Security, Identity, and Resistance in Asian Cyberspace.* MIT Press, 2011.

[118] Hal Roberts, Ethan Zuckerman, and John Palfrey. Circumvention Tool Evaluation. *Berkman Center for Internet & Society at Harvard University*, 10: 2012, 2011.

[119] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689. ACM, 2014.

[120] Dillon Reisman, Steven Englehardt, Christian Eubank, Peter Zimmerman, and Arvind Narayanan. Cookies that Give You Away: Evaluating the Surveillance Implications of Web Tracking. Technical report, Working paper, 2014.

[121] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium (USENIX)*, 2014.

[122] Andrew Hintz. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies workshop (PET 2002), year = 2002, publisher = Springer-Verlag, LNCS 2482.*

[123] George Danezis and Richard Clayton. Route Fingerprinting in Anonymous Communications. In *IEEE International Conference on Peer-to-Peer Computing*, pages 69–72. IEEE, 2006.

[124] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A Critical Evaluation of Website Fingerprinting Attacks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 263–274. ACM, 2014.

[125] David Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology*, pages 199–203. Springer, 1983.

[126] A Chiesa, C Garman, I Miers, M Virza, E Ben-Sasson, M Green, and E Tromer. Zerocash: Practical Decentralized Anonymous E-Cash from Bitcoin. *IEEE Security and Privacy (S&P)*, 2014.

[127] Mixcoin: Anonymity for bitcoin with accountable mixes.

[128] Philip Koshy, Diana Koshy, and Patrick McDaniel. An Analysis of Anonymity in Bitcoin Using P2P Network Traffic. In *Financial Cryptography and Data Security (FC'14)*, March 2014.

[129] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.

[130] Osiris: Serverless Portal System. http://www.osiris-sps.org/. [Accessed April 2015].

[131] Syndie: A System for Operating Distributed Forums. https://www.syndie.de/. [Accessed April 2015].

[132] G. Danezis and C. Diaz. A Survey of Anonymous Communication Channels. Technical Report MSR-TR-2008-35, Microsoft Research, 2008.

[133] George Danezis and Len Sassaman. Heartbeat Traffic to Counter $(n-1)$ Attacks: Red-Green-Black Mixes. In *ACM Workshop on Privacy in the Electronic Society*, pages 89–93. ACM, 2003.

[134] Markus Jakobsson, Ari Juels, and Ronald L Rivest. Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. In *USENIX Security Symposium*, pages 339–353, 2002.

[135] Markus Jacobson and David MRaïhi. Mix-Based Electronic Payments. In *Selected Areas in Cryptography*, pages 157–173. Springer, 1999.

[136] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In *Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

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

[138] Paul F Syverson, Michael G Reed, and David M Goldschlag. Onion Routing Access Configurations. In *DARPA Information Survivability Conference and Exposition*, volume 1, pages 34–40. IEEE, 2000.

[139] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer, 2001.

[140] Steven Murdoch and Nick Mathewson. Top Changes in Tor since the 2004 Design Paper (Part 1). https://blog.torproject.org/blog/top-changes-tor-2004-design-paper-part-1, . [Accessed April 2015].

[141] Steven Murdoch and Nick Mathewson. Top Changes in Tor since the 2004 Design Paper (Part 2). https://blog.torproject.org/blog/top-changes-tor-2004-design-paper-part-2, . [Accessed April 2015].

[142] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. (Nothing else) MATor(s): Monitoring the Anonymity of Tor's Path Selection. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 513–524. ACM, 2014.

[143] John Geddes, Rob Jansen, and Nicholas Hopper. IMUX: Managing Tor Connections from Two to Infinity, and Beyond. In *Workshop on Privacy in the Electronic Society*, pages 181–190. ACM, 2014.

[144] Anne Edmundson, Simpson AKornfeld, Joshua A Kroll, and Edward W Felten. Security Audit of Safeplug "Tor in a Box". In *USENIX Workshop on Free and Open Communications on the Internet (FOCI 14)*. USENIX Association.

[145] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar Weippl. Spoiled Onions: Exposing Malicious Tor Exit Relays. In *Privacy Enhancing Technologies*, pages 304–331. Springer, 2014.

[146] Nicholas Hopper. Challenges in Protecting Tor Hidden Services from Botnet Abuse. In *Financial Cryptography and Data Security*, pages 316–325. Springer, 2014.

[147] Tao Wang and Ian Goldberg. Improved Website Fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society*, pages 201–212. ACM, 2013.

[148] Marco Valerio Barbera, Vasileios P Kemerlis, Vasilis Pappas, and Angelos D Keromytis. Cellflood: Attacking Tor Onion Routers on the Cheap. In *Computer Security (ESORICS 2013)*, pages 664–681. Springer, 2013.

[149] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-Resource Routing Attacks against Tor. In *ACM Workshop on Privacy in Electronic Society*, pages 11–20. ACM, 2007.

[150] Ian Goldberg and Adam Shostack. Freedom Network 1.0 Architecture and Protocols. White paper, Zero Knowledge Systems Inc., 1999.

[151] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom Systems 2.0 Architecture. White paper, Zero Knowledge Systems Inc., 2000.

[152] Ian Avrum Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, University of California at Berkeley, 2000.

[153] Adam Back, Ian Goldberg, and Adam Shostack. Freedom 2.1 Security Issues and Analysis. White paper, 2001.

[154] Roger R Dingledine. The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven. Master's thesis, Massachusetts Institute of Technology, 2000.

[155] Garlic Routing - I2P. https://geti2p.net/en/docs/how/garlic-routing. [Accessed April 2015].

[156] Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[157] J.P. Timpanaro, I Chrisment, and O. Festor. Group-Based Characterization for the I2P Anonymous File-Sharing Environment. In *International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2014.

[158] Christoph Egger, Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. Practical Attacks Against the I2P Network. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013)*, October 2013.

[159] Michael Herrmann and Christian Grothoff. Privacy-Implications of Performance-Based Peer Selection by Onion-Routers: A Real-World Case Study Using I2P. In *Privacy Enhancing Technologies*, pages 155–174. Springer, 2011.

[160] L Schimmer. Peer Profiling and Selection in the I2P Anonymous Network. In *extended abstracts of the Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1).–Dresden: TU, Fak. Informatik*, pages 59–70, 2009.

[161] Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. Improving Content Availability in the I2P Anonymous File-Sharing Environment. In *International Symposium on Cyberspace Safety and Security*, volume 4, pages 77–92. Springer, 2012.

[162] Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. Monitoring the I2P Network. Technical Report RR-7844, 2011.

[163] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *ACM Conference on Computer and Communications Security (CCS 2002)*, 2002.

[164] Michael J Freedman, Emil Sit, Josh Cates, and Robert Morris. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In *Peer-to-Peer Systems*, pages 121–129. Springer, 2002.

[165] Li Zhuang, Feng Zhou, Ben Y Zhao, and Antony Rowstron. Cashmere: Resilient Anonymous Routing. In *Symposium on Networked Systems Design & Implementation-Volume 2*, pages 301–314. USENIX Association, 2005.

[166] Parisa Tabriz and Nikita Borisov. Breaking the Collusion Detection Mechanism of MorphMix. In *Privacy Enhancing Technologies*, pages 368–383. Springer, 2006.

[167] Michael K Reiter and Aviel D Rubin. Anonymous Web Transactions with Crowds. *Communications of the ACM*, 42(2):32–48, 1999.

[168] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[169] Andy Jones. Anonymous Communication on the Internet. *Indiana University*, page 13, 2004.

[170] Brian Neil Levine and Clay Shields. Hordes: A Multicast Based Protocol for Anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[171] Tianbo Lu, Binxing Fang, Yuzhong Sun, and Xueqi Cheng. WonGoo: A Peer-to-Peer Protocol for Anonymous Communication. In *PDPTA*, pages 1102–1106, 2004.

[172] Tianbo Lu, Binxing Fang, Yuzhong Sun, and Xueqi Cheng. Performance Analysis of WonGoo System. In *International Conference on Computer and Information Technology (CIT 2005)*, pages 716–722. IEEE, 2005.

[173] Claudia Diaz. *Anonymity and Privacy in Electronic Services*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 2005.

[174] William Feller. *An Introduction to Probability Theory and Its Applications: Volume One*. John Wiley & Sons, 1950.

[175] Andrei Serjantov. A Fresh Look at the Generalised Mix Framework. In *Privacy Enhancing Technologies*, pages 17–29. Springer, 2007.

[176] Luke OConnor. On Blending Aattacks for Mixes with Memory. In *Information Hiding*, pages 39–52. Springer, 2005.

[177] Andrei Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, 2004.

[178] Amir Houmansadr, Giang T. K. Nguyen, Matthew Caesar, and Nikita Borisov. Cirripede: Circumvention Infrastructure using Router Redirection with Plausible Deniability. In *ACM Conference on Computer and Communications Security (CCS 2011)*, 2011.

[179] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. Telex: Anticensorship in the Network Infrastructure. In *USENIX Security Symposium*, 2011.

[180] Josh Karlin, Daniel Ellard, Alden W. Jackson, Christine E. Jones, Greg Lauer, David P. Mankins, and W. Timothy Strayer. Decoy Routing: Toward Unblockable Internet Communication. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI 2011)*, 2011.

[181] M. Bråding. Generic, Decentralized, Unstoppable Anonymity: The Phantom Protocol. White paper, 2011.

[182] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol Version 2. *Draft, July*, 2003.

Glossary

active attack	is an attack on the system where the attacker attempts to alter system resources or affect their operation.
active attacker	is an attacker that carries out an *active attack*.
anonymity set	is the usual suspects, that is, the set of subjects with potentially the same attributes who might cause an action. The *anonymity set* is relative with respect to the attacker and may vary over time.
batching strategy	is the algorithm in a *mix* which specifies how to store, mix and forward messages; and may consist of multiple *selection algorithms*.
bias function	is a mathematical function used as part of the *selection algorithm* of a *mix* to determine the bias of the probability distribution that is used to randomly select a subset of messages in the pool.
bitwise unlinkability	is the property of two encoded network messages where an observer is unable to link them by finding a meaningful relation between their bit patterns.
Chaumian mix	refers to the first design of *mix* that was proposed in 1981 by Chaum [8].
external attack	is an attack on the system where the attacker only controls communication links.
flushing time	is the moment a *mix* flushes a number of messages it contains according to its *batching strategy*.
global anonymity	is the degree of anonymity enjoyed by all of the users of an *ACS*.
global attacker	is an attacker that can observe all of network links including the edges of the network.
guest ACS	is an *ACS* hosted and operates within *GRF*.

individual anonymity	is the degree of anonymity enjoyed by an individual subject.
internal attack	is an attack on the system where the attacker controls one or several entities that are part of the system (e.g. some communication nodes or some elements inside a *mix*).
local attacker	is an attacker that can observe only a subset but not all of network links.
metadata	is what remains of a communication or document after its contents and substance is excluded (e.g. IP addresses, time of communication, volume of the communicated material).
mix	is a network router that mixes, and might alter, a number of incoming messages before relaying them, thereby offering anonymity by obfuscating the link between the incoming and outgoing messages.
mix system	is an ACS that leverages the concept of *mix* to achieve anonymous communication, and may consist of one or more mixes.
mixnet	is a network of interconnected *mixes*.
passive attack	is an attack on the system where the attacker attempts to learn or make use of information from the system but does not affect system resources.
passive attacker	is an attacker that carries out a *passive attack*.
reusability	refers to the use of existing assets (e.g. source code, software components, test suites, designs and documentation) within the software product development process.
selection algorithm	is an algorithm in a *mix* that specifies which subset of the incoming messages should be flushed.
threat model	is the description of the security issues considered in the context of this thesis.
traffic mixer	is the same as a *mix*.